

DEVELOPMENT OF IMAGE CLASSIFICATION APPS USING ANDROID STUDIO

ONG CHEE TENG



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DEVELOPMENT OF IMAGE CLASSIFICATION APPS USING ANDROID STUDIO

ONG CHEE TENG

**This report is submitted in partial fulfilment of the requirements for
the degree of Bachelor of Computer Engineering Technology
(Computer Systems) with Honours**

**Faculty of Electronics and Computer Technology and Engineering
Universiti Teknikal Malaysia Melaka**

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2025

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : Development of Image Classification Apps using Android Studio
Sesi Pengajian : 2024/2025

Saya ONG CHEE TENG mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐

SULIT*

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

☐

TERHAD*

(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.

☒

TIDAK TERHAD

Disahkan oleh:

Alamat Tetap:

DR. NOR HAFIZAH BINTI HUSSIN
Pensyarah
Fakulti Teknologi Dan Kejuruteraan Elektronik
Dan Komputer (FTKEK)
Universiti Teknikal Malaysia Melaka (UTeM)

Tarikh : 16 Januari 2025

Tarikh : 16 Januari 2025

DECLARATION

I declare that this project report entitled “Development of Image Classification Apps using Android Studio” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature

:

Student Name

:

ONG CHEE TENG

Date

:

14-01-2025



اونیورسیتی تکنیکل مالیزیا ملاک

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Engineering Technology (Computer Systems) with Honours.

Signature :

Supervisor Name :

Dr. Nor Hafizah binti Hussin

Date :

16 January 2025

Signature :

Co-Supervisor :

Name (if any)

Date :

DEDICATION

To my beloved mother, Long Boon Hui.



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ABSTRACT

Image classification has emerged as a crucial task in the field of computer vision, facilitating numerous applications across various domains. It known in this area of study includes the training of sound and reliable machine learning models to classify the images accordingly into the categories defined. The problem is image classification application require many resources to be used and some challenge about reliability of currently available image classification apps. Those applications cannot classify the object very detailed. For the objectives of the project, the application needs to provide easy to use UI, detect more specific items and improve the accuracy of result for the image classification. To complete the project, the android studio was used to create an android application and TensorFlow Lite were trained using machine learning that import into the android app. After that, the application can recognize the image and give the result. The apps should be a simple UI and easy to use. Since this project is related to image classification, the analysis part will use accuracy testing to evaluate this app. The results show that developed apps have successfully detected the correct image up to 90% accuracy. The higher the accuracy, the greater the performance of model implement in the image classification app. This project presents how mobile apps can use machine learning to do something that can bring advanced features like real time classification. The image classification app can be used in many domains like education to help enthusiasts recognise species or serve as an educational tool for student and environmental conservation. In business, it could be adapted into specialized software for industries requiring automated identification.

ABSTRAK

Pengelasan imej telah muncul sebagai tugas penting dalam bidang penglihatan komputer, memudahkan banyak aplikasi merentas pelbagai domain. Ia dikenali dalam bidang kajian ini termasuk latihan model pembelajaran mesin yang baik dan boleh dipercayai untuk mengklasifikasikan imej mengikut kategori yang ditentukan. Masalah ialah aplikasi pengelasan imej memerlukan banyak sumber yang boleh digunakan dan beberapa cabaran tentang kebolehpercayaan aplikasi pengelasan imej yang tersedia pada masa ini. Aplikasi tersebut tidak boleh mengklasifikasikan objek dengan sangat terperinci. Untuk objektif projek, aplikasi perlu menyediakan UI yang mudah digunakan, mengesan item yang lebih khusus dan meningkatkan ketepatan keputusan untuk klasifikasi imej. Untuk melengkapkan projek, Android Studio digunakan untuk mencipta aplikasi android dan TensorFlow Lite dilatih menggunakan pembelajaran mesin yang mengimport ke dalam aplikasi android. Selepas itu, aplikasi boleh mengenali imej dan memberikan Keputusan. Aplikasi mesti UI yang ringkas dan mudah digunakan. Memandangkan projek ini berkaitan dengan klasifikasi imej, bahagian analisis akan menggunakan ujian ketepatan untuk menilai aplikasi ini. Lebih tinggi ketepatan, lebih tinggi prestasi model dalam aplikasi klasifikasi imej. Keputusan menunjukkan bahawa aplikasi yang dibangunkan telah berjaya mengesan imej yang betul sehingga 90% ketepatan. Projek ini membentangkan cara aplikasi menggunakan pembelajaran mesin untuk melakukan sesuatu yang boleh membawa ciri lanjutan seperti pengelasan masa nyata. Aplikasi klasifikasi imej boleh digunakan dalam banyak bidang seperti pendidikan untuk membantu peminat mengenali spesies atau berfungsi sebagai alat pendidikan untuk pelajar dan pemuliharaan alam sekitar. Dalam perniagaan, ia boleh disesuaikan menjadi perisian khusus untuk industri yang memerlukan pengenalan automatik.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, DR. NOR HAFIZAH HUSSIN for precious guidance, words of wisdom and patient throughout this project.

I am also indebted to Universiti Teknikal Malaysia Melaka (UTeM) and family for the financial support through my journey which enables me to accomplish the project. Not forgetting my fellow colleague, for the willingness of sharing his thoughts and ideas regarding the project.

My highest appreciation goes to my parents, parents in-law, and family members for their love and prayer during the period of my study.

Finally, I would like to thank all the staffs at the UTeM, fellow colleagues and classmates, the faculty members, as well as other individuals who are not listed here for being co-operative and helpful.

TABLE OF CONTENTS

	PAGE
DECLARATION	
APPROVAL	
DEDICATIONS	
ABSTRACT	i
ABSTRAK	ii
ACKNOWLEDGEMENTS	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	vi
LIST OF FIGURES	vii
LIST OF SYMBOLS	viii
LIST OF ABBREVIATIONS	ix
LIST OF APPENDICES	x
 CHAPTER 1 INTRODUCTION	 11
1.1 Background	11
1.2 Addressing Object Recognition and Detection through Image Classification	12
1.3 Problem Statement	13
1.4 Project Objective	13
1.5 Scope of Project	14
 CHAPTER 2 LITERATURE REVIEW	 15
2.1 Introduction	15
2.2 Image Classification Problem	15
2.3 Traditional Image Classification	16
2.3.1 Traditional Neural Network	17
2.4 Deep Learning	19
2.4.1 Convolutional Neural Networks (CNNs)	20
2.5 Similar Project	22
 CHAPTER 3 METHODOLOGY	 24
3.1 Introduction	24
3.2 Selecting and Evaluating Tools for Image Classification App Development	24
3.3 Methodology	25
3.3.1 Project Software flowchart	25
3.3.1.1 Phase 1: Design App	27
3.3.1.2 Phase 2: Import TensorFlow Lite into App	30

3.3.1.3	Phase 3: Testing Performance of the App	32
3.4	Equipment	33
3.4.1	Android Studio	33
3.4.2	Teachable Machine	34
3.4.3	TensorFlow Lite	35
3.5	Limitation of proposed methodology	36
3.6	Summary	37
CHAPTER 4	RESULTS AND DISCUSSIONS	38
4.1	Introduction	38
4.2	Preliminary Result	38
4.3	Results and Analysis	41
4.3.1	Result	43
4.3.2	Analysis	44
4.3.2.1	Parrot Model	45
4.3.2.2	Owl Model	51
4.3.2.3	Goose Model	57
4.4	Summary	63
CHAPTER 5	CONCLUSION AND RECOMMENDATIONS	65
5.1	Conclusion	65
5.2	Future Works	66
5.3	Project Potential	66
REFERENCES		67
APPENDICES		71

LIST OF TABLES

TABLE	TITLE	PAGE
Table 4.1	Table of Accuracy Result for Accuracy Testing of Parrot Model	45
Table 4.2	Table of Accuracy Result for Accuracy Testing of Owl Model	51
Table 4.3	Table of Accuracy Result for Accuracy Testing of Goose Model	57



اونيورسيتي تېكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 2.1:	Artificial Neural Network (ANN)	18
Figure 2.2	Deep Neural Networks (DNNs)	19
Figure 2.3:	Convolutional Neural Networks (CNNs)	21
Figure 3.1:	Flowchart of project structure	26
Figure 3.2:	Flowchart of App design	28
Figure 3.3:	Flowchart of model training	30
Figure 3.4:	Bug of the App	32
Figure 3.5:	Android Studio	33
Figure 3.6:	Teachable Machine	34
Figure 2.5:	TensorFlow Lite	36
Figure 4.1:	Preliminary Result of Amazon Parrot	38
Figure 4.2:	Preliminary Result of Caique Parrot	39
Figure 4.3:	Preliminary Result of Cockatoo	40
Figure 4.4:	Preliminary Result of Conure Parrot	41
Figure 4.5:	Example of Training Image	42
Figure 4.6:	Example of Testing Image	42
Figure 4.7:	UI design for Main Activity	43
Figure 4.8:	Model Selection	44
Figure 4.9:	Amazon Parrot	51
Figure 4.10:	Parrotlet	51

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
Appendix A	Coding for MainActivity.kt	71
Appendix B	Task Schedule of PSM1	76
Appendix C	Task Schedule of PSM2	77



اونيورسيتي تېكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 1

INTRODUCTION

1.1 Background

Image classification has emerged as a crucial task in the field of computer vision, facilitating numerous applications across various domains. Given the recent developments in deep learning especially for the types of convolutional neural networks, the accuracy and the speed for image classification has increased. What is currently known in this area of study includes the training of sound and reliable machine learning models to classify the images accordingly into the categories defined. Furthermore, the exponential rise of portable devices with high-performance GPUs and software tools such as TensorFlow Lite has creatively enabled the functionality of image classification on mobiles.

However, there is a significant research gap concerning the construction of efficient, accurate, and easy to use image classification applications for Android devices given the improvements in image classification and advancements in the mobile technology. Many existing solutions have complex interfaces or are computationally intensive and thus compatible with mobile devices. Furthermore, the diversification and incorporation of machine learning models into Android applications pose some issues to developers, especially when it comes to efficiency and compatibility with the different technical specifications of the devices. These are the areas that need to be filled in order to make image classification available and easily deployable on mobile devices and for use by users from different backgrounds.

This research seeks to fill this gap by proposing the development of an Image Classification App developed using Android Studio that will enhance the user experience in image classification and at the same time ensure high accuracy in classification tasks. It will entail researching and identifying proper machine learning models that can be implemented on a mobile device, further, fine-tuning these models for mobile devices, and developing an easy-to-navigate Android app front-end. Furthermore, some added functionalities in the application will include live image capturing and presentation of results to the users. This research aims to assist users to carry out image classification activities with ease using their Android cellular phones to promote equal use of this capable technology in the development of mobile applications.

1.2 Addressing Object Recognition and Detection through Image Classification

Object recognition and detection involving image classification are key breakthroughs in technologies in the field of computer vision, which has significantly affected numerous industries and uses. Image classification is the process whereby an object or a set of objects in an image is recognized and classified by using advanced machine learning algorithms hence providing a host of practical solutions across various fields.

Within the context of visual search engines, Image classification extends the capabilities of image search by allowing users to search for products, landmarks, or artworks, directly from the image, thereby improving the quality of the search and providing more natural means of navigating digital content. Further, in the context of automated surveillance systems, image classification is central in identifying the objects of interest, including persons of interest or illicit automobiles, thereby boosting up security and allowing the detection of threats in advance.

In conclusion, image classification is a very flexible and necessary method for solving the problems of object recognition and detection across various industries and applications. Through correct naming and categorizing of objects in images, image classification eases work automation, improves decision making and fosters creative approaches to solved problems.

1.3 Problem Statement

This issue is rather surprising, given the current trends in machine learning and the development of mobile applications for Android. However, there are no useful software applications available for image classification for early. Existing options that are available are therefore difficult to use or require so many resources to be able to be used. Further, there is some doubt to the reliability of currently available image classification apps, and it is said to the information regarding what they are able to classify correctly is not very detailed. This lack of available alternatives has created a need for a new, efficient Android application with which preliminary issues can be addressed, while also including accuracy improvements in areas that existing applications fail to excel.

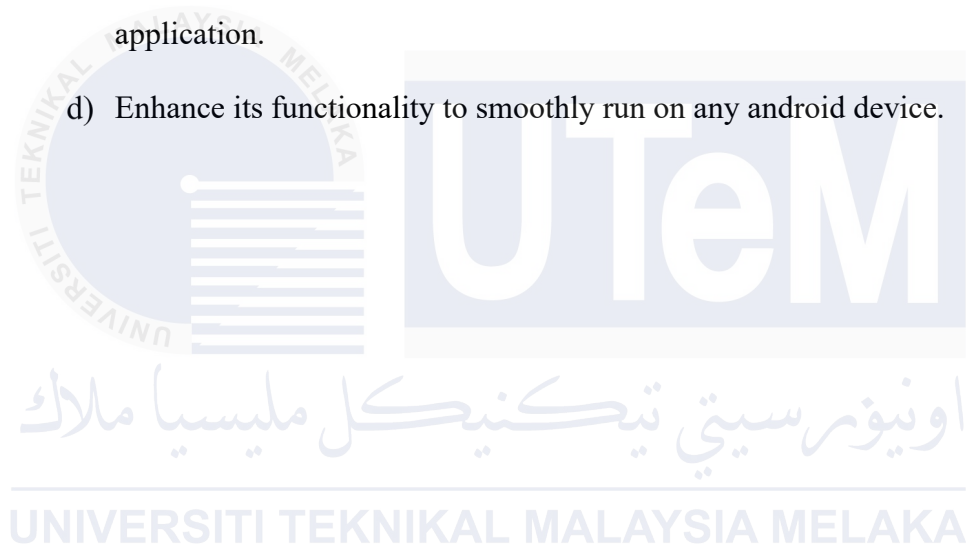
1.4 Project Objective

The project objectives are as follows:

- a) To provide a simple and easy-to-use Android application interface for image classification.
- b) To detect more specific item according to customer's needed, for example the type of cat.
- c) To improve the accuracy of result for the image classification.

1.5 Scope of Project

- a) Implement an Android application that should be built using Android Studio that involves freedom to capture an image and choose an image from the gallery and user can interact especially when involving classification tasks.
- b) Design a GUI layout for the Android application, which ensures easy handling of the application and the captured images for viewing the results.
- c) Implement machine learning models for the recognition of images in the Android application.
- d) Enhance its functionality to smoothly run on any android device.



CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Image classification is a part of computer vision that could let machines correctly identify and understand the images that they see. The process of dividing the images into the classes that are predefined or into certain categories based on the features of the images is known as image classification. This basic function has extensive uses for multiple industries including healthcare and self-driving cars as well as retail and entertainment. When using modern approaches to artificial intelligence, such as deep learning techniques like convolutional neural networks, image classification algorithms may efficiently recognize many details and easily differentiate between various objects and scenes. The feature of image classification is that it can automate the process of visual recognition with the help of artificial intelligence to improve the performance of industries and to unlock new possibilities for interaction between humans and machines in the modern world.

2.2 Image Classification Problem

Some of the difficulties that are met during the teaching process with class or images are known as image classification. It is still possible to raise several questions by considering machine learning decision making, for example, bias and impartiality. At times, the data used to train these systems may be 'Historically biased' which implies that during the data gathering process, the data bias can be intentionally made or can be a result of oversight to favor the minority. It can lead to biased decisions to either decrease or increase the benefit of the individuals/ groups because they are not well represented in the data. For example, if

a model was worked out with mostly one type of person, then it is likely to be inaccurate when dealing with persons of a different type.

Another challenge is robustness. It must, however, be noted that such systems are not always complete in one way, manner or another under different situations. They may not do well in anything that involves fluctuations in light intensity, casual orientation or anything that is not straightforward when it comes to pictures. This proves to have real drawbacks in the extent to which they can be translated into real life scenarios and hence real problems that they have to face difficulties in terms of accuracy and efficiency in how they are supposed to work.

Privacy and security are also other considerations of concern. I need to be very careful at this point lest these systems when not well managed, may put the individuals at risk of exposing some sensitive information, or cause some form of harm to the people involved. Namely, if the system has wrong classes for images where people are included, then it may lead to wrong accusations as well as invasion of privacy.

2.3 Traditional Image Classification

Image classification in fact is an integral part of many computer vision applications in the related fields as well as in image depth processing. Standard image classification operates in several phases which are pre-processing of the image, feature extraction, classifier building and training learning. Traditional concept of image classification mostly depends on the characteristics that are extracted from the image in order to arrive at the classification and this can be the base by which the rest of the computer-based extraction of the semantic details of the image can be taken. Traditional image classification typically calculates image

characteristics using color, texture, and other data, then applies logistic regression and support vector machines to achieve image classification. In addition to the retrieved characteristics, which are a significant factor, the results of picture classification are impacted by knowledge and experience in related fields [4].

Applying the manually obtained features to image classification is not only challenging, but it also takes a long time to analyze the feature data. However, traditional machine learning cannot handle processing large datasets, and optimizing feature design, feature selection, and model training is a difficult task that yields poor model classification. Therefore, image classification methods based on traditional machine learning have an impact on many application fields. Studies indicate that texture, shape, and color attributes are useful for both image recognition and classification. Traditional image classification techniques typically extract a single feature or a mixture of features, then use the extracted features as the support vector machine's input value.

2.3.1 Traditional Neural Network

One of the main components of contemporary machine learning and artificial intelligence is known as an artificial neural network (ANN). An artificial neural network is made up of interconnected nodes, or neurons, arranged into layers and modeled after the structure of the human brain [9].

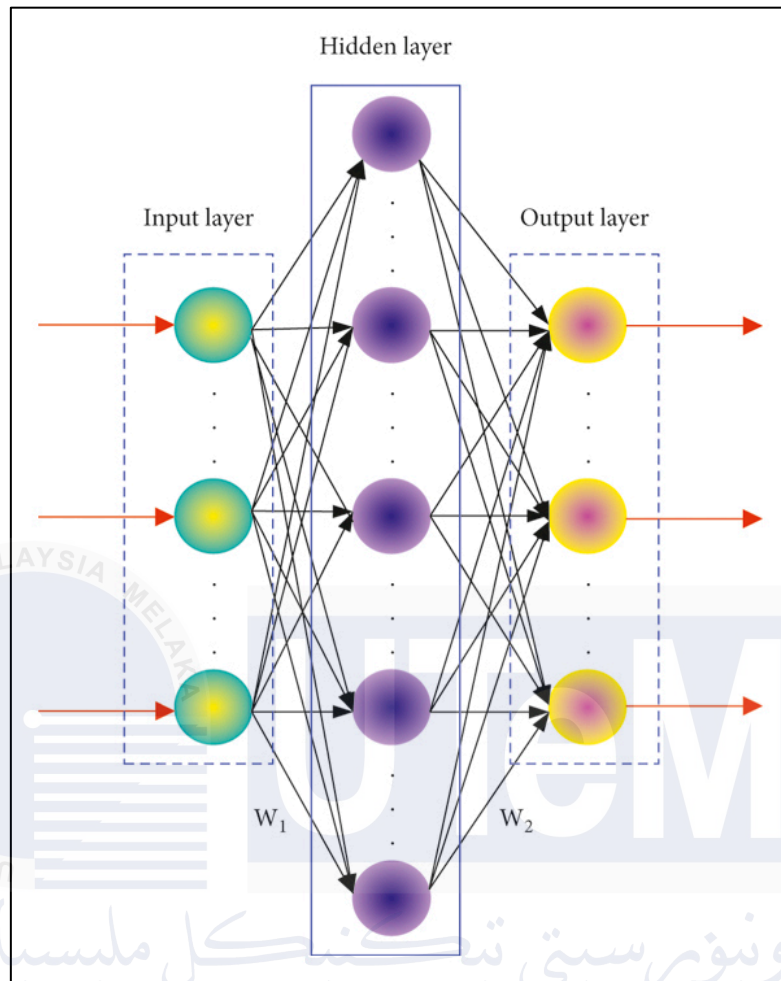


Figure 2.1: Artificial Neural Network (ANN)

Three primary types of layers are shown in Figure 2.1: input, hidden, and output layers. The original data is received by the input layer and is subsequently processed through the hidden layers with activation functions and mathematical operations. The output layer generates the network's computed result at the end [10].

A large dataset of input-output pairs is fed into an artificial neural network (ANN) during training, and the weights of the connections between neurons are adjusted to minimize the discrepancy between the desired and actual outputs. The backpropagation can be referred to as the direction for this process as gradient descent and various other optimization algorithms are used.

Natural language processing, pattern recognition, image and speech recognition have all been solved using ANNs. However, some disadvantages of traditional ANNs are known, for instance, the models' computational cost and their poor performance at processing large high-dimensional data.

2.4 Deep Learning

This is a branch of machine learning that deals with deep artificial neural networks or deep neural networks. Compared with other machine learning algorithms, deep learning algorithms learn these representations from the raw input itself [4].

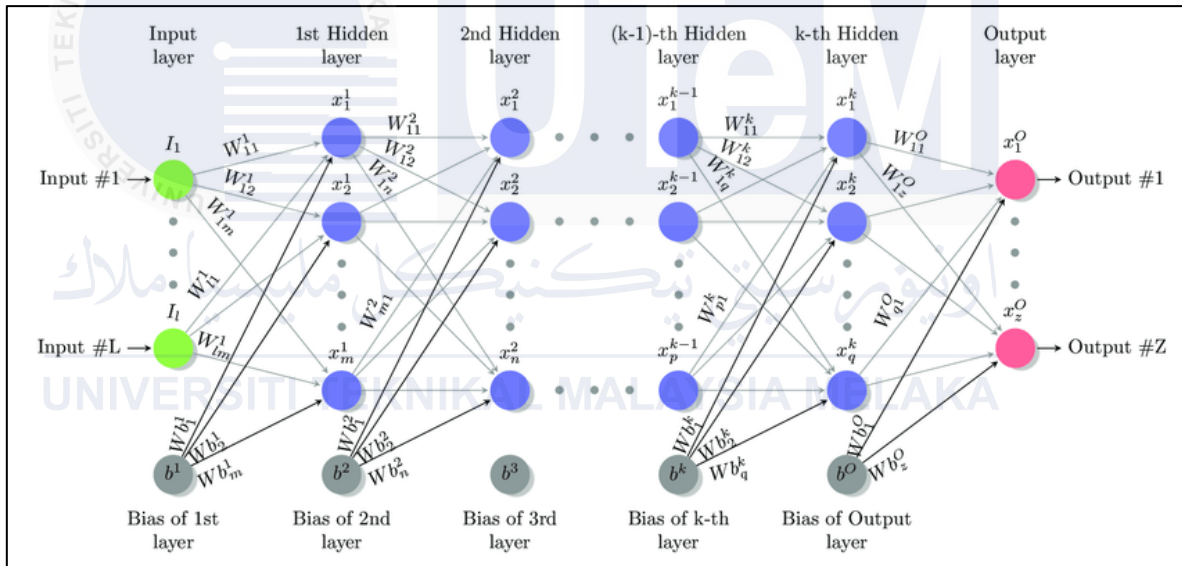


Figure 2.2 Deep Neural Networks (DNNs)

Another advantage of deep learning is that, compared with other machine learning methods, deep learning can automatically locate features and other subtle patterns to be learned from big data. The deep neural network is shown in Figure 2.2. In this research, the deep neural network is as shown below. 2, they enforce very complex dependencies and in addition they do this with high accuracy while at the same time learning lower and lower level and very abstract representations of the input data.

As seen, deep learning has shown high potential in various applications including; computer vision, natural language processing, speech recognition, and reinforcement learning. For example, there are special types of network like convolutional neural networks (CNNs) which are inherently in a position to learn some of the feature like edges, textures, shapes etc for image classification. As a result of this characteristic, it is more suited for managing data that amass chronologically or successively such as the language translation and the voice to text conversion [1].

The ability to use large scale Deep Neural Networks through the use of powerful computing equipment such as GPU and TPU, and open-source deep learning systems like TensorFlow and Pytorch have helped in increasing the usage of Deep Learning. These frameworks ease the access of this deep learning technology and take innovation to different fields through strong and feasible tools and abstractions to develop train and deploy deep learning models[11].

Therefore, deep learning is a relatively new branch of machine learning, which has changed how certain ML problems are solved and has even expanded the potential of AI technologies.

2.4.1 Convolutional Neural Networks (CNNs)

To solve tasks related to images and other types of data that are initially visual, people use a particular type of deep neural networks called convolutional neural networks or CNNs. They have transformed the field of computer vision as the learning systems can learn representations of images and the features in the images and make accurate predictions on their own [3].

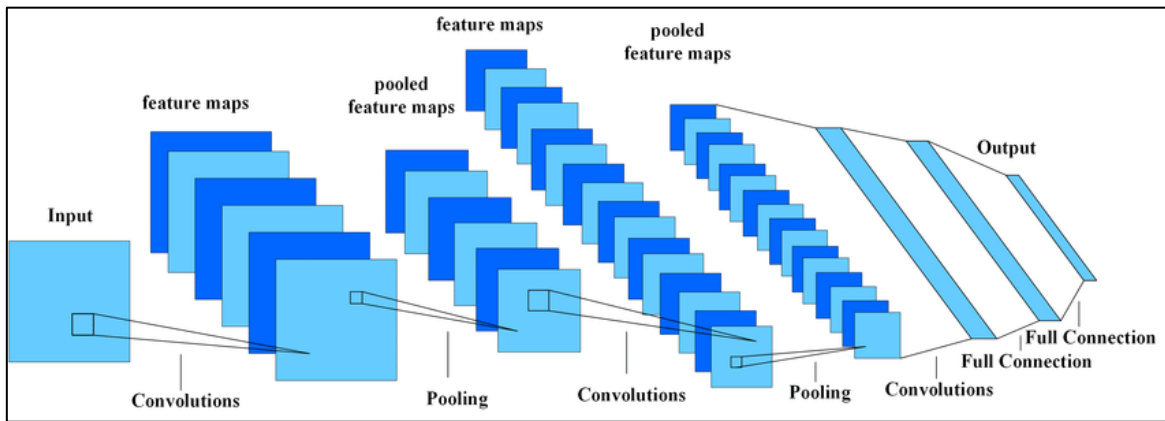


Figure 2.3: Convolutional Neural Networks (CNNs)

In Figure 2. 3, CNNs have different structures through the fully connected layers, pooling layers, and convolutional layers. In convolutional layers of the network, filters or also known as kernels are used to come up with local feature of the input image like edges, texture and pattern. The convolution layers produce feature maps that are reduced in size by the pooling layers; they preserve as much data as can be useful in the reduction process by shrinking the size of the data. The final feature map undergoes prediction on the features using fully connected layers [3].

There are several advantages associated with CNNs, and one of these is the capability of learning spatial feature hierarchies. In this manner the network is capable of recognizing objects and patterns at multiple resolutions and abstraction levels since through the convolutional and pooling layers, the input image is dissected and higher and higher levels of features are extracted from it.

CNNs are applied in most computer vision applications including but not limited to object detection, image segmentation, image generation and classification. CNNs give the optimal results in the picture recognition tasks such as the task of sorting the pictures under certain

classes such as the recognition of handwritten digits in pictures or recognizing different objects in the photographic images. In object detection, CNN can point out the locations and also recognize multiple objects within an image, thus it has potential application areas including auto-mobile, video, and medical imaging. In image segmentation, CNNs can segment an image through partitioning images into regions of interest as it involves the analysis of medical images or understanding of a scene.

2.5 Similar Project

“A review of the application of deep learning in medical image classification and segmentation” written by Lei Cai, Jingyang Gao, Di Zhao in Jun 2020. In this project, they have 2 methods to create machine learning models [21].

First is TensorFlow. TensorFlow is an open-source software with coding tool by Google used for computational basic on data flow. In early 2015, TensorFlow was released to the public, and it has been used as the fundamental tool for machine learning and deep learning ever since. It implements several deep learning frameworks including CNNs, RNNs, LSTM as well as other common machine learning algorithms. TensorFlow also possesses a very good visualization tool known as TensorBoard through which developers can easily produce very good visual representation of its model in terms of its performance and structure. The flexibility encompasses distributed heterogeneous computing to complete model training across different GPUs or systems. Developed using C++, TensorFlow enables efficient operation to make it a platform of choice for big-data machine learning [21].

Second is PyTorch. PyTorch is another deep learning toolkit, created by the FAIR team at Facebook, and worked on a dynamic neural network expressed on the GPU. Beginning with

its inception on GitHub in January 2017, PyTorch proved popular among researchers principally because of its dynamic computation graph, which can be modified in real-time depending on computational requirements. Compared with TensorFlow which uses static computation graph, PyTorch offers more flexibility for use, which is more suitable for research purpose and testing. Due to its smooth compatibility with Python language and simplicity, it is suitable for use by researchers as well as developers [21].



CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter presents the research methodology and the steps that will be followed in this project. It will be centered on the software components that a beginner needs to master in order to build an image classification application using Android Studio. The app is also going to use TensorFlow Lite for on-device machine learning to classify pictures quickly. The following plan will outline how the app will be created with detailed explanations for how the user interface will be designed, how the machine learning model will be implemented, and how the app will be tested for efficacy. The methodologies discussed above are intended to guarantee that the image classification app is properly developed and implemented on Android devices.

3.2 Selecting and Evaluating Tools for Image Classification App Development

There are a few tools and technologies that have to be selected and reviewed for design and deployment of the app that is created out of Android Studio to be used in the classification of images [13]. This entails a few methodological factors, such as measuring accuracy of the machine learning models, studying the interaction between different libraries/ frameworks and ensuring the app responsiveness for mobile gadgets [14]. There is also another consideration that concerns the app accessibility and interface, always considering who would find it easily and comfortable to work with it and to find the necessary view or option. On compatibility for development environment, TensorFlow Lite shall therefore support compatibility with Android Studio. This involves checking what is compatible with the latest

android SDKs, and that the performance of the app is tested across a range of android devices [15]. Again, this can be done with Android Studio's layout editor and prototyping tools to make an app with an appealing interface. Gathering beta testers and making necessary changes to the design incorporating the opinion of the target market to create a simple user interface of the app.

3.3 Methodology

The project provides the reader with a broad framework on how to design a simple image classification application using Android Studio with TensorFlow Lite. To this end, the proposed methodology is based on convolutional neural networks (CNNs) as the primary tool for image classification and works efficiently on mobile platforms. The operations remain both qualitative and quantitative oriented so as to create a viable and easy to use application.

3.3.1 Project Software flowchart

Figure 3.1 shows the flowchart of project structure. An early stage of this project will involve a mobile application design, followed by importing the TensorFlow lite and finally will proceed with performance testing.

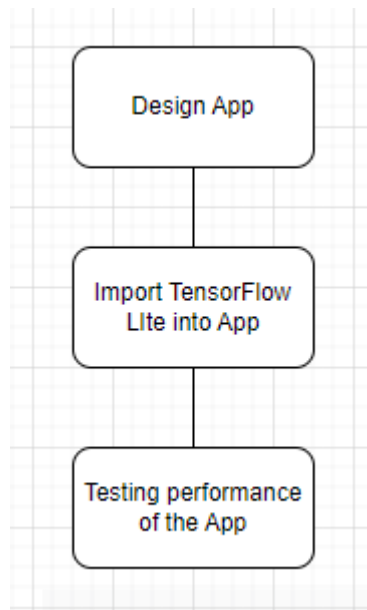


Figure 3.1: Flowchart of project structure

Phase 1: Design App

The initial phase focuses on designing the app, ensuring it has a user-friendly interface and seamless functionality. This step needs to identify the app's objectives and user needs, such as the types of images to be classified and the expected accuracy and create a new project in Android Studio, configure project settings, and organize the project structure. Develop the app's user interface using XML layout files and the layout editor to design screens for capturing or selecting images and displaying classification results.

Phase 2: Import TensorFlow Lite into App

In this phase, the trained machine learning model is integrated into the Android application to enable image classification. This step needs to develop and train a convolutional neural network (CNN). Once the model achieves satisfactory accuracy, convert it to TensorFlow Lite format using the TensorFlow Lite Converter. It include TensorFlow Lite dependencies in the project's Gradle files and Import the TensorFlow Lite model (.tflite file) into the Android Studio project. Then, implement the TensorFlow Lite Interpreter API to load and

run the model within the app. Write code to preprocess input images and pass them through the model for classification. Implement methods to handle the capture or selection of images, preprocess them, and interpret the model's results.

Phase 3: Testing Performance of the App

The final phase involves thorough testing to ensure the app's functionality, performance, and user experience meet the desired standards. This step tests the app on various Android devices to ensure compatibility across different screen sizes, resolutions, and hardware configurations and measure the app's performance, focusing on the speed and accuracy of image classification. Optimize the model and app code to enhance performance.

3.3.1.1 Phase 1: Design App

Figure 3.2 shows the flowchart of the App design. In the phase 1, the app was created and the image input was added in the display area. This step is shown in the flowchart in figure 3.2 below. Next step, users can choose image from gallery or upload image by camera. After input image, the input image will display on display area. Lastly, the App will predict image and show the result.

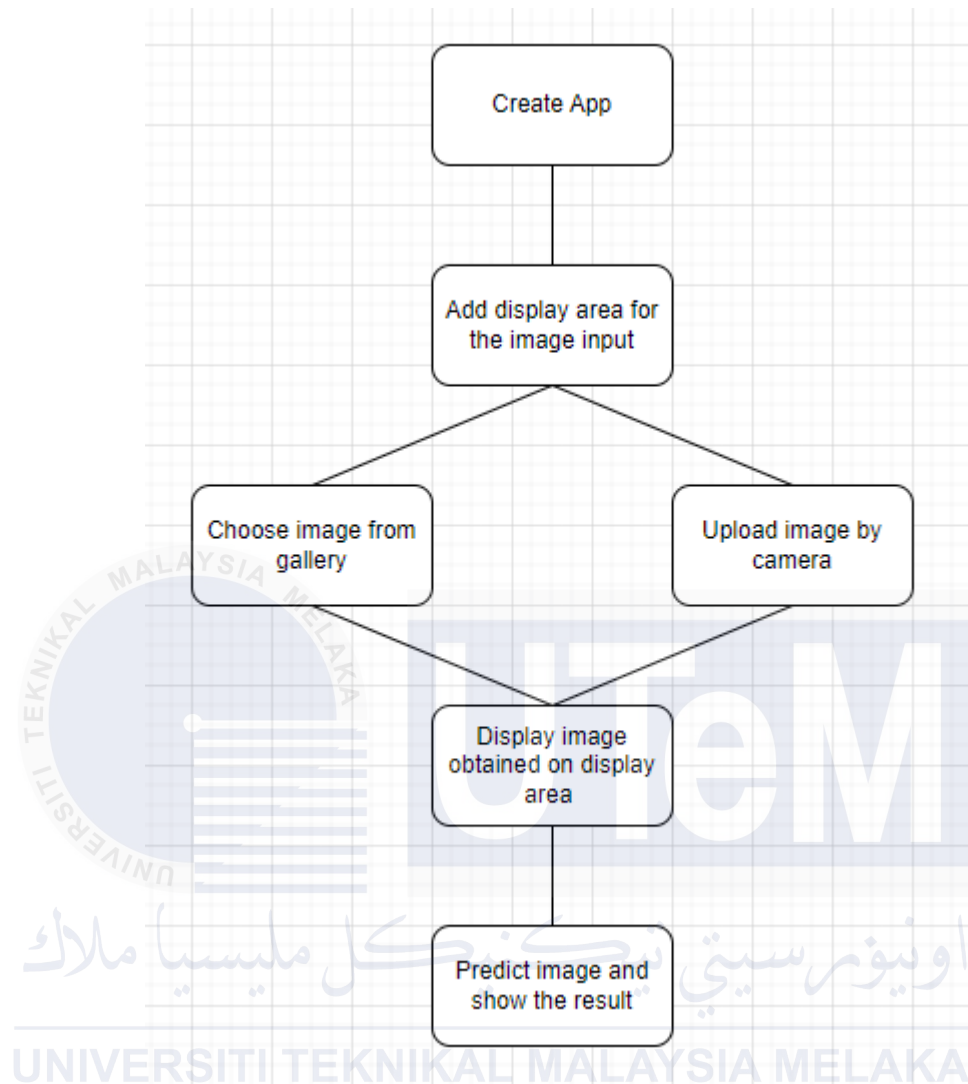


Figure 3.2: Flowchart of App design

Step 1: Create App

The initial phase focuses on setting up the Android project and preparing the basic framework for the application. Open Android Studio and create a new project with a suitable name and package name. Configure the project settings, including the minimum SDK level and required dependencies. After that, organize the project structure by creating appropriate packages for activities, fragments, adapters, and utilities. Set up the necessary libraries, including TensorFlow Lite.

Step 2: Add display area for the image input.

In this phase, design and implement the user interface element where the image will be displayed. This step creates an XML layout file for the main activity. Add a `ImageView` widget to serve as the display area for the image input. Use Android Studio's layout editor to position the `ImageView` appropriately within the UI. Ensure it has the necessary properties to display images clearly.

Step 3: Choose Image from Gallery/Upload Image by Camera

This phase involves adding functionality for users to select an image from the gallery or capture a new image using the camera. Implement an `Intent` to open the device's gallery and allow the user to select an image. Handle the result in the `onActivityResult` method. Image Capture via Camera: Implement functionality to launch the device's camera using an `Intent`, capture an image, and return it to the app. Handle the result in the `onActivityResult` method. Ensure the app requests and handles the necessary permissions for accessing the gallery and camera.

Step 4: Display image obtained on display area.

After obtaining the image, display it in the designated area within the app. Write code to take the image URI or bitmap obtained from the gallery or camera and set it to the `ImageView` for display.

Step 5: Predict image and show the result.

Integrate the TensorFlow Lite model to classify the displayed image and show the result to the user. Display the classification result in the UI, such as in a `TextView` or dialog box.

3.3.1.2 Phase 2: Import TensorFlow Lite into App

Figure 3.3 shows the flowchart of model training. Firstly, the data was collected from internet like picture. After that, Teachable Machine website was used to train model using data collected. Next step, the accuracy of the model will be tested. Lastly, I will export model as TensorFlow Lite.

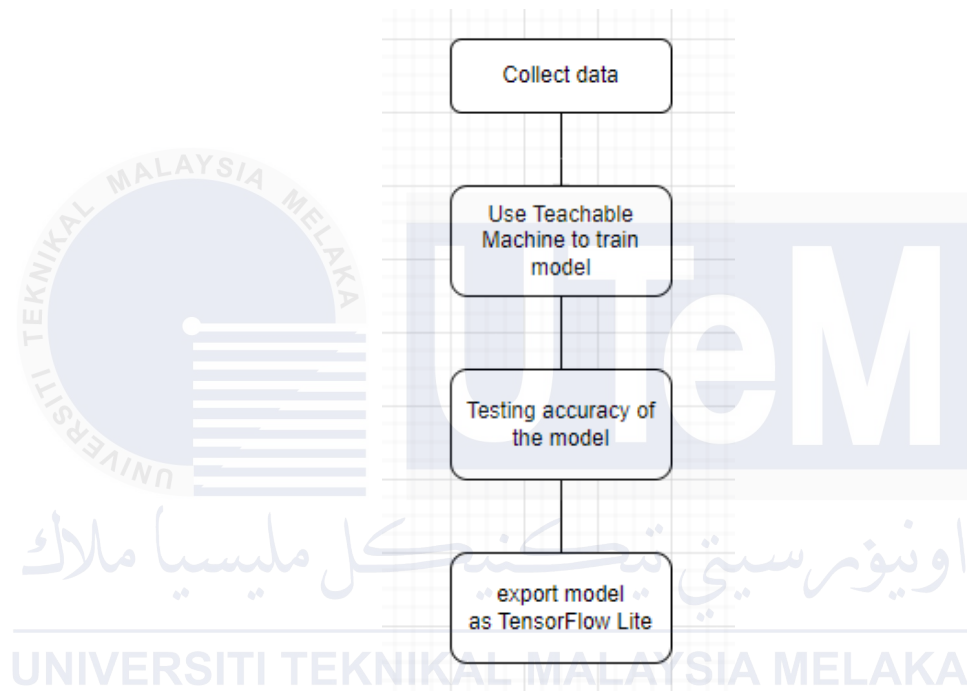


Figure 3.3: Flowchart of model training

Step 1: Collect data.

The first phase focuses on gathering and preparing the dataset required for training the image classification model. Gather a diverse set of images that represent the different classes you want the model to classify. Ensure the dataset is comprehensive enough to cover various scenarios the app might encounter. Label the images accurately to ensure the model learns from correct examples. Organize the dataset into folders based on their respective classes.

Step 2: Use Teachable Machine to train model.

In this phase, leverage Teachable Machine to create and train an image classification model without writing any code. Import the collected and labelled images into Teachable Machine's interface. Set up the training parameters, including the number of epochs and batch size, as per the complexity of the dataset and desired accuracy. Use Teachable Machine's intuitive interface to train the model. Monitor the training process and adjust parameters if necessary to improve performance.

Step 3: Testing accuracy of the model

This phase involves evaluating the trained model's accuracy and ensuring it meets the desired performance criteria. Use a separate set of validation images to test the model's accuracy. This helps in understanding how well the model generalizes to unseen data. Based on the performance metrics, refine the model by retraining with adjusted parameters or augmented data if necessary.

Step 4: export model as TensorFlow Lite

In the final phase, convert the trained model into TensorFlow Lite format for integration into the Android app. Use Teachable Machine's export functionality to save the trained model as a TensorFlow Lite model (.tflite file). Optionally apply optimization techniques such as quantization to reduce the model size and improve inference speed on mobile devices. Download the TensorFlow Lite model file and prepare it for integration into the Android app.

3.3.1.3 Phase 3: Testing Performance of the App

In the final phase, the performance of the app will test to ensure user can use the app smoothly.

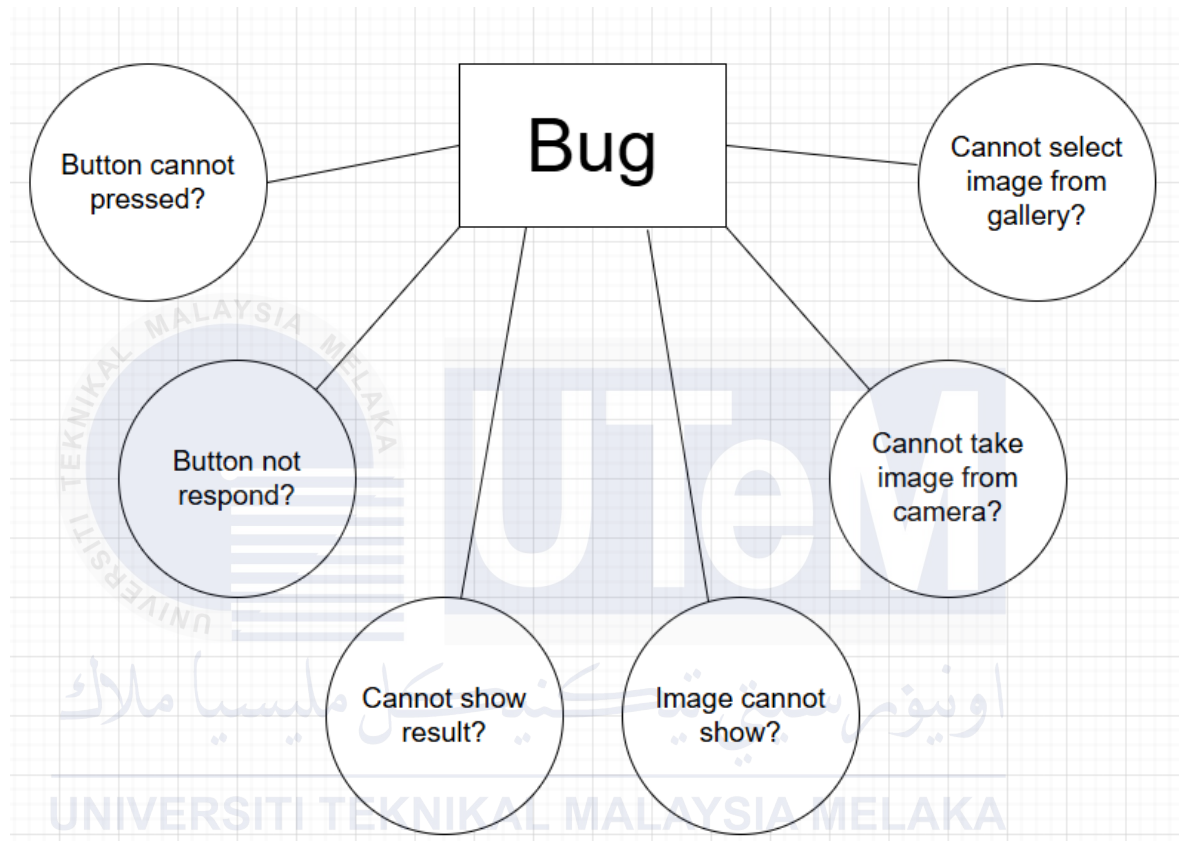


Figure 3.4: Bug of the App

Figure 3.4 shows the possible variable bug of the app. Button cannot pressed or not respond. Image cannot show on the display area. Users cannot select image from gallery or take image form camera. Result will not show after press button. In PSM2, the app may have other bugs other than those. Those bugs happen can has several reasons like problem of coding, UI design problem, version of android studio and other. Those bugs can affect the performance of the app and need to fix those.

3.4 Equipment

3.4.1 Android Studio



Figure 3.5: Android Studio

Generically, the tool used for writing Android applications is referred to as Android Studio, which is the official Integrated Development Environment (IDE) for writing Android applications. It was developed by Google and it is one of the most versatile platforms designed to ease the generation of applications; be it the interface or writing codes, or even the testing phase [16][17].

One of the significant advantages of Android Studio is the presence of a set of bright and convenient GUI means for designing the application layout, including the set of graphical editors that allow the layout to be constructed visually, their drag-and-drop functionality, and the preview pane which helps the developer to observe the application layout as it is seen on various devices and with different screen resolution.

In addition to the above, the IDE that is used by Android Studio has incorporate a powerful code editor that has characteristics such as the highlighting of syntax, completion of codes and the manipulation of code combining functions. One of it enables developers to choose a programming language that is most advisable for a certain application from a range of favorable languages including Java, Kotlin among others [18].

3.4.2 Teachable Machine



Figure 3.6: Teachable Machine

We can see the Teachable Machine in Figure 3.6. Google created Teachable Machine, an innovative web-based tool that lets users quickly and easily create custom machine learning models without requiring any coding knowledge. Designed to be user-friendly, it allows users to train models for image, sound, and pose classification tasks through a simple, interactive interface [19][20].

3.4.3 TensorFlow Lite

Google came up with TensorFlow Lite (TFLite), which is an open-source machine learning model made for installation on devices with headers. By extending the same architecture, it enables developers to directly deploy and execute ML models on smart phones, tablets, IoT devices and all other edge devices for inferencing on the said devices with least latency and best optimization in resource usage [7].

Hence, since TFLite is more efficient in terms of performance, speed and hence readiness to tackle machine learning models on device-like gadgets with less storage and computing power. This is done through the following techniques such as model quantization in which the float values of model parameters are trimmed to fewer bits in a bid to use relatively smaller memory and increase on the inference speed.

Liquid DNNS can be executed on mobile and embedded applications because TensorFlow Lite (TFLite) can be used to perform the conversion of machine learning models on the device. TFLite is a quantized version of TensorFlow that is specifically designed to produce models optimized for mobile and edge use-cases.

The developers can use TFLite to convert the CNN models that they have developed using TensorFlow or other related tools into the format suitable for performing inference on embedded and mobile hardware. This conversion process optimizes the model for inference on apps, smartphones and tablets, IoT particles and edge devices.

TFLite is compatible with various layers of CNN structures like MobileNet, Inception, EfficientNet and several other structures for specific applications, as well as customized

structures. Some of the computer vision applications that could be solved with these models include semantic analysis, object recognition, image recognition and classification.



Figure 2.5: TensorFlow Lite

Once converted to the TFLite model, the CNN models can be used within mobile applications using the TFLite runtime library. This library comes with interfaces for loading, compiling and running TFLite models across operating systems on mobile devices, making it easy to integrate the machine learning functionality into mobile apps.

In summary, TensorFlow Lite allows developers to deploy CNN models on mobile and edge platforms and speeds up the development of on-device machine learning applications with improved privacy, reduced latency, and offline capabilities. TFLite has the potential of extending the usage of machine learning in smart camera and augmented reality, recommendation system, Health-tech products, and more.

3.5 Limitation of proposed methodology

- a) Data Quality and Quantity

The performance of the machine learning model heavily depends on the quality and quantity of the training data. Insufficient or poor-quality data can lead to inaccurate classifications and reduced model performance. Moreover, collecting and labelling a large dataset can be time-consuming and resource intensive.

b) Device Performance

Running machine learning models on mobile devices using TensorFlow Lite can be constrained by the limitations of these devices. While TensorFlow Lite is optimized for mobile inference, models with high computational complexity may still experience latency issues or impact the device's performance and battery life.

3.6 Summary

The procedure for designing an image classification application using Android Studio and TensorFlow Lite sub-disciplines a well-ordered and systematic process. First of all, there is an app design with the graphic interface and configuration of a given project in Android Studio. After that TensorFlow Lite is included into the app; it is the process of training the convolutional neural network (CNN) using Teachable Machine. After training the model, the complete model is quantized for the TensorFlow Lite format and integrated into the Android application framework. The last stage is mainly devoted to the testing that shows compatibility, efficiency and satisfactory results to the users. These are like ensuring that the app has been tested on several devices and the performance has been improved. As such, this means that there will be an end-to-end app that seeks to offer the best solution as a viable image classification app through the utilization of TensorFlow Lite on Android.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

The conclusion and discussion section of any research report is crucial since it contains the project's goals and analysis. This section enables the data to be interpreted and provides evidence of the project's success. This part can also be used to discuss the project's shortcomings and present the findings of the outcome.

4.2 Preliminary Result

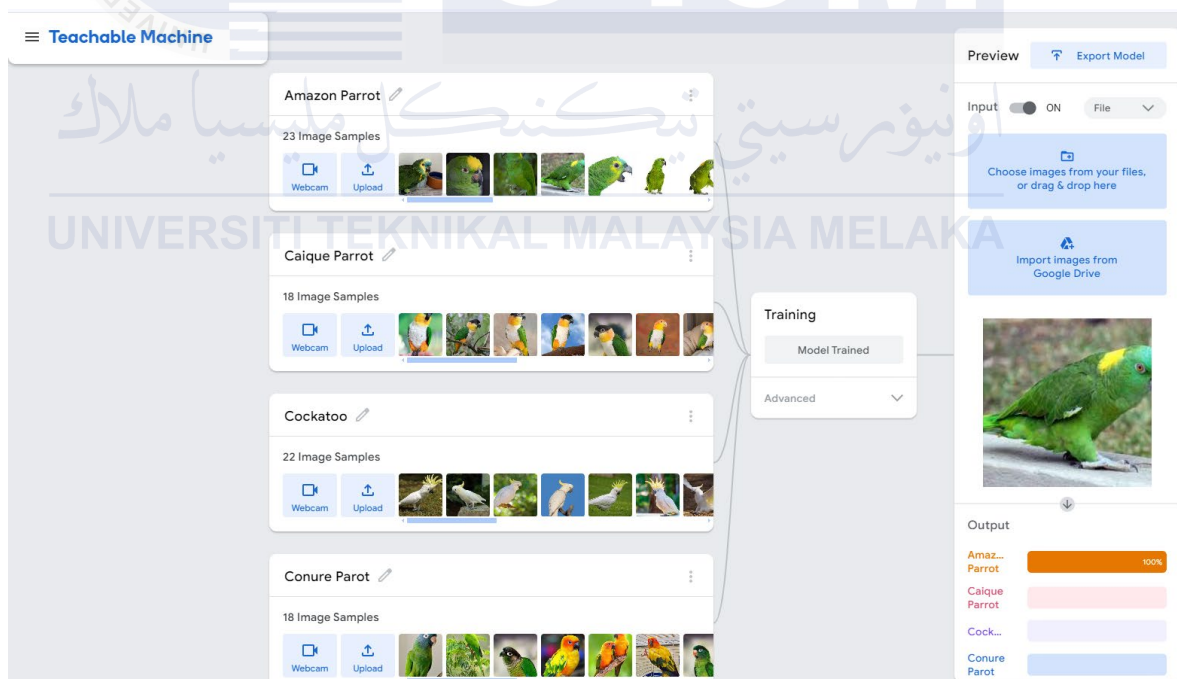


Figure 4.1: Preliminary Result of Amazon Parrot

Figure 4.1 shows the trained model predicting that input image is 100% amazon parrot. The result is correct.

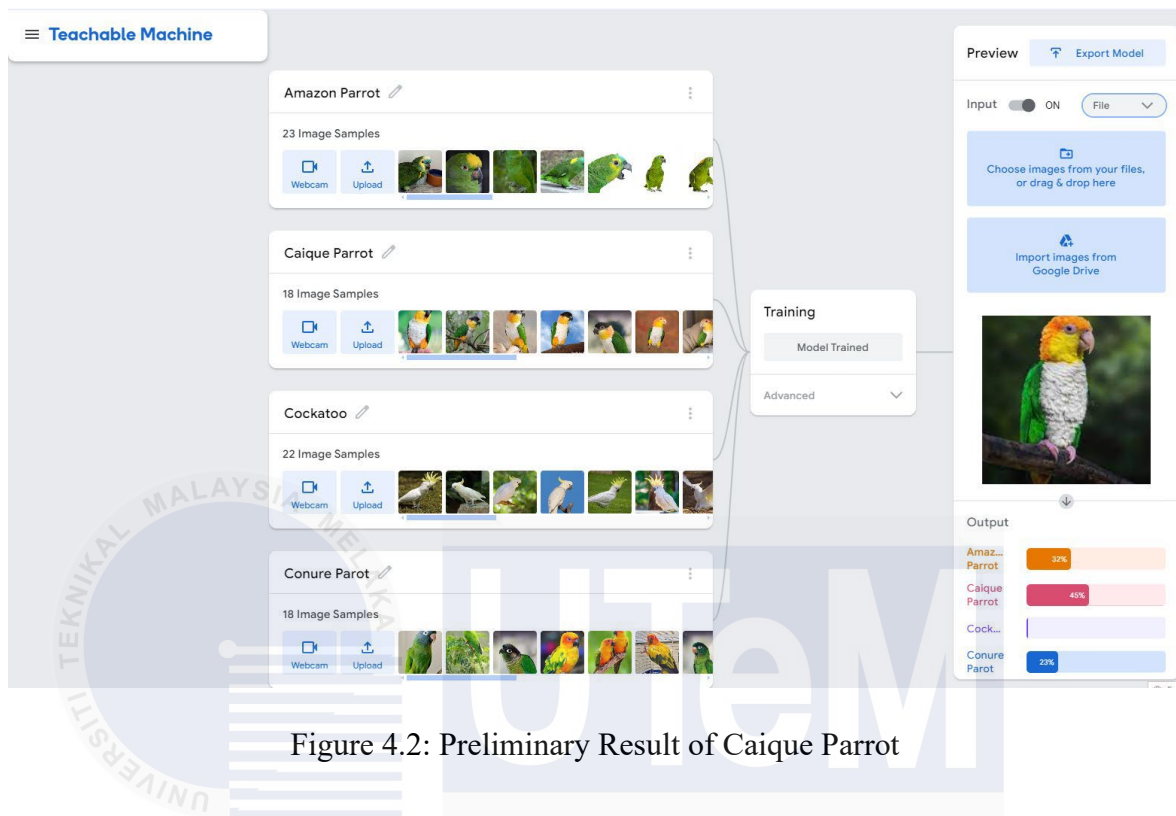


Figure 4.2: Preliminary Result of Caique Parrot

Figure 4.2 shows the trained model predicting that input image is 32% is amazon parrot, 45% is caique parrot and 23% is Conure Parrot. The result is correct as the highest percentage is 45% caique parrot but the accuracy is lower.

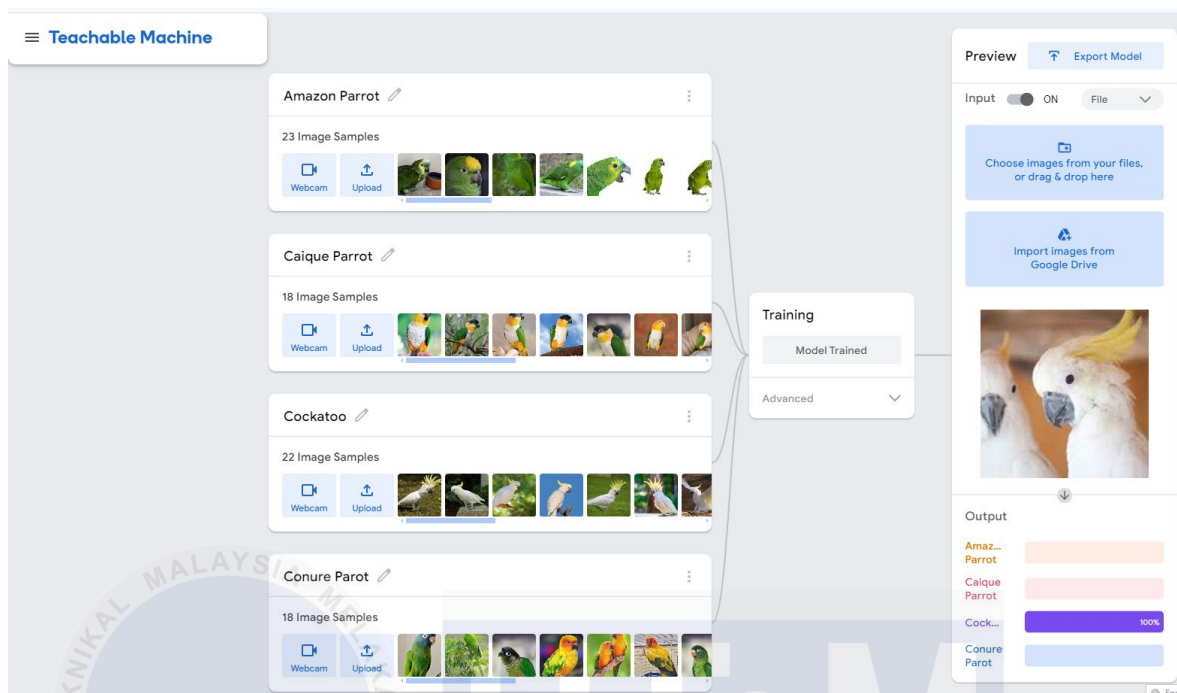


Figure 4.3: Preliminary Result of Cockatoo

Figure 4.3 shows the trained model predicting that input image is 100% is cockatoo. The result is correct.



Figure 4.4: Preliminary Result of Conure Parrot

Figure 4.4 shows the trained model predicting that input image is 89% is conure parrot. The result is correct.

Those results are only preliminary results. In PSM2, more types of parrots will be added inside it and improve the accuracy. The image taken into Teachable Machine has cropped into fixed size. This means some bigger images cannot be taken fully. For example, only the body of parrot can be learned by machine learning and the head of parrot cannot. This can cause a decrease in accuracy of model training.

4.3 Results and Analysis

This section will show the results such as UI design and analysis using accuracy testing.

For image used, there are two sets which are training image set and testing image set. The training image set used the PNG format which does not include the background, only the object. The testing image set used the JPG format which include the background. The training image set was used when training model on the Teachable Machine website. The testing image set was used when doing the accuracy testing. The testing image set will not used together with training image set when training model on the Teachable Machine website.



Figure 4.5: Example of Training Image

Figure 4.5 shows the image in the PNG format. This figure only has object only, not background. This format image was used on training image set only.



Figure 4.6: Example of Testing Image

Figure 4.6 shows the image in the JPG format. This figure has object include background. This format image was used on testing image set only.

4.3.1 Result

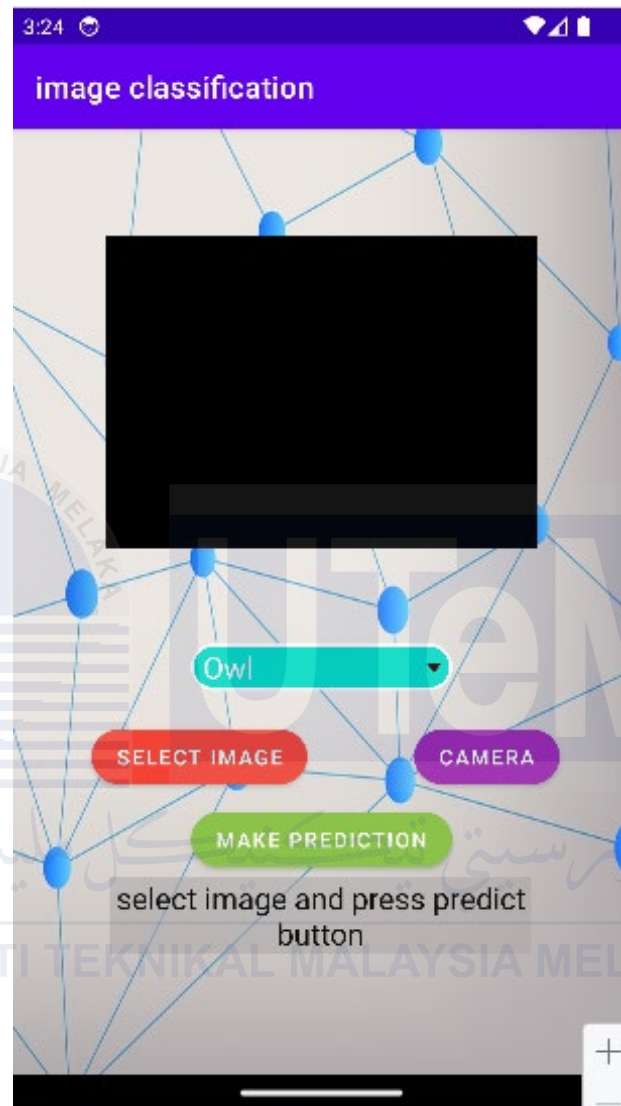


Figure 4.7: UI design for Main Activity

Based on figure 4.5, the UI includes the image display area, spinner, button “select image”, button “camera”, button “make prediction” and text display area. The image display area shows the image taken from gallery or camera. The spinner can select the model wanted. The button “select image” selects image from gallery. The button “camera” opens the camera and take one image. The button “make prediction” is give result according to model. The text display area will show the result after making predictions.

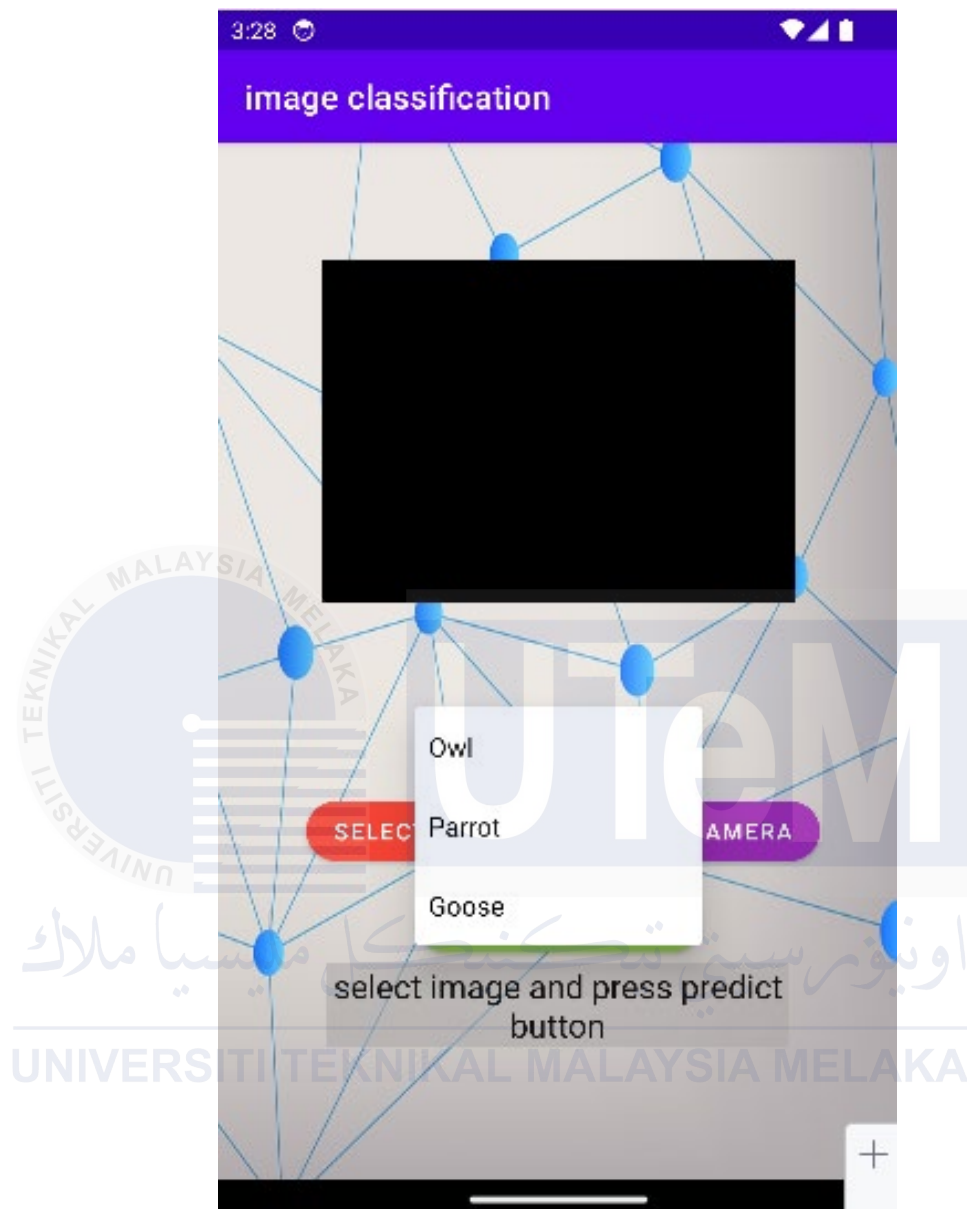


Figure 4.8: Model Selection

Based on figure 4.6, there are 3 models that can choose after clicking the spinner which are owl, parrot and goose.

4.3.2 Analysis

In this section, each model will run accuracy testing using 10 testing images and produce accuracy result using formula:

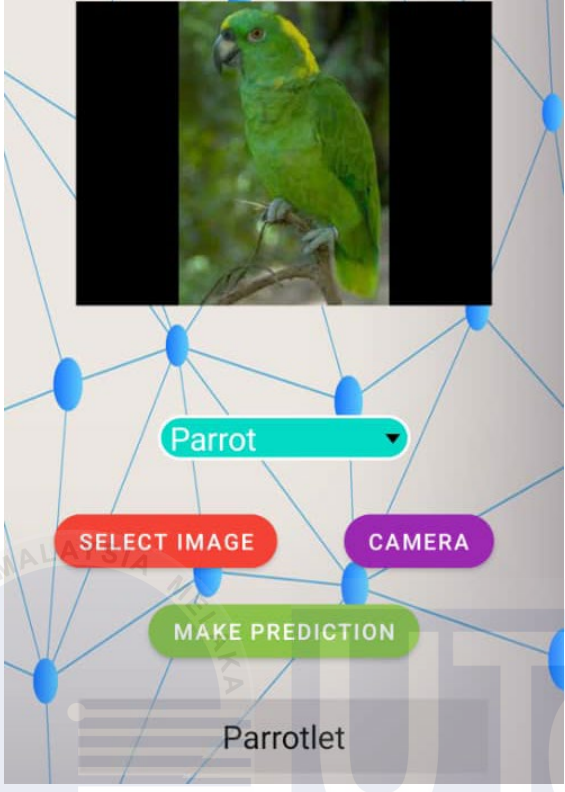
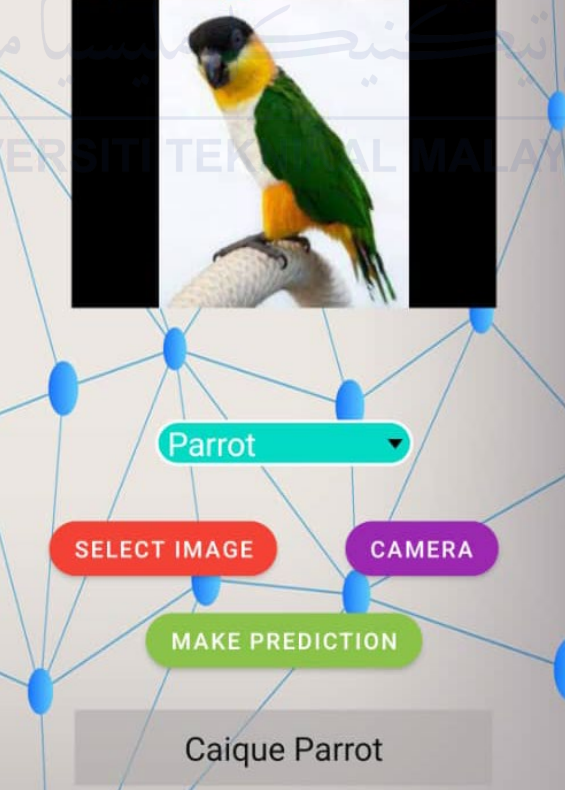
$$Accuracy = \frac{\text{sucess prediction count}}{10} \times 100\%$$

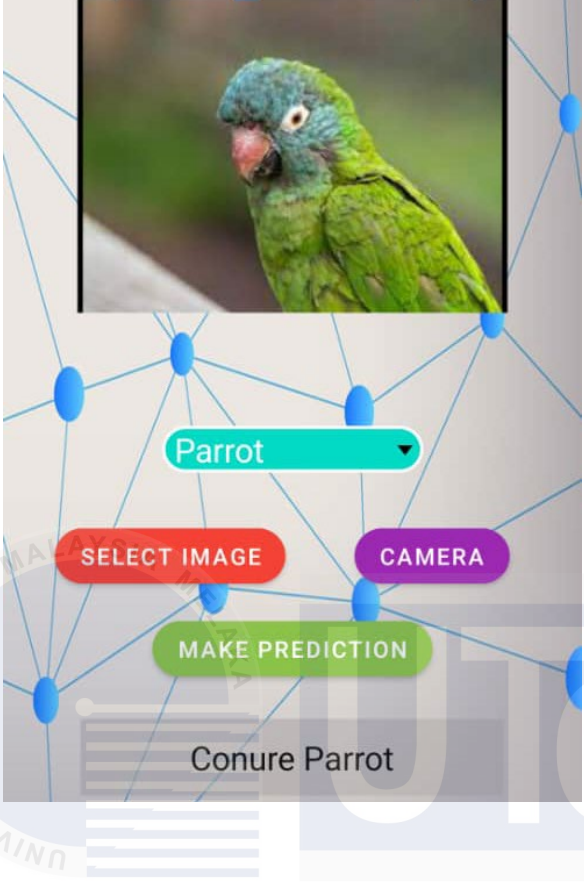

4.3.2.1 Parrot Model

In this model, there are Amazon parrot, Caique parrot, Cockatoo, Conure parrot, Eclectus parrot, Lovebird, Macaw, Parrotlet, Quaker Parakeet and Senegal parrot.

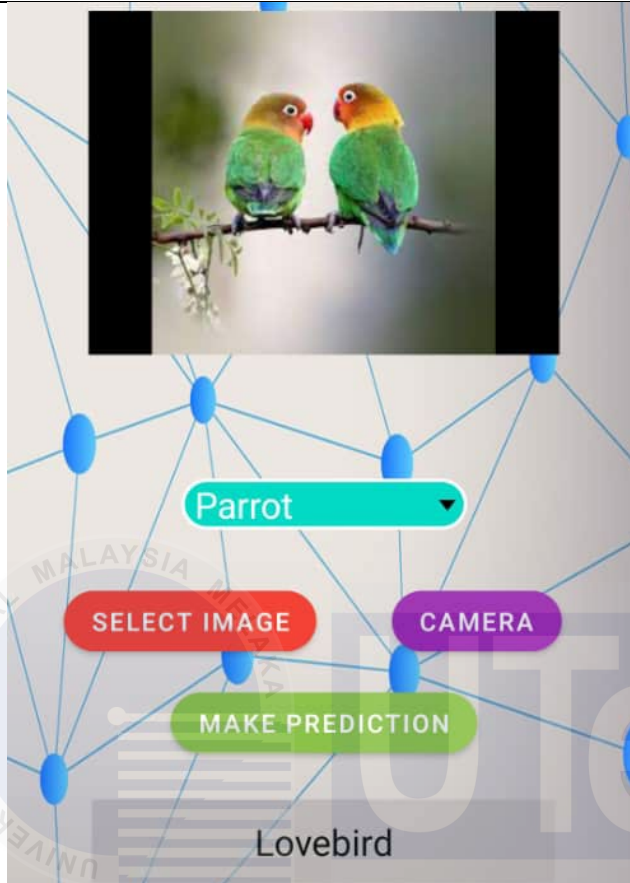
Table 4.1 Table of Accuracy Result for Accuracy Testing of Parrot Model

No	Output	Result
1		Correct.

2	 <p>The screenshot shows a mobile application interface for parrot identification. At the top, there is a photo of a green Amazon parrot. Below the photo is a teal dropdown menu currently set to 'Parrot'. Underneath the dropdown are two buttons: a red 'SELECT IMAGE' button and a purple 'CAMERA' button. Below these is a green 'MAKE PREDICTION' button. At the bottom of the interface, the text 'Parrotlet' is displayed as the prediction result. The background of the app has a network-like pattern of blue nodes and lines.</p>	<p>Wrong. The result should be amazon parrot.</p>
3	 <p>This screenshot shows the same application interface as the previous one, but with a different image. The photo at the top is of a Caique parrot, which has green, yellow, and orange plumage. The dropdown menu is still set to 'Parrot'. The 'SELECT IMAGE', 'CAMERA', and 'MAKE PREDICTION' buttons are present. At the bottom, the prediction result is 'Caique Parrot'.</p>	<p>Correct.</p>

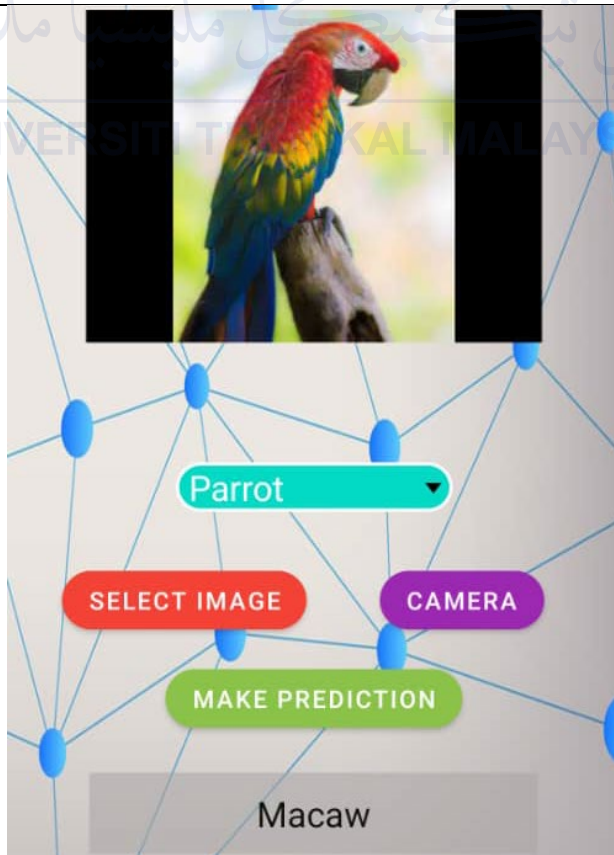
4		Correct.
5		Correct.

6



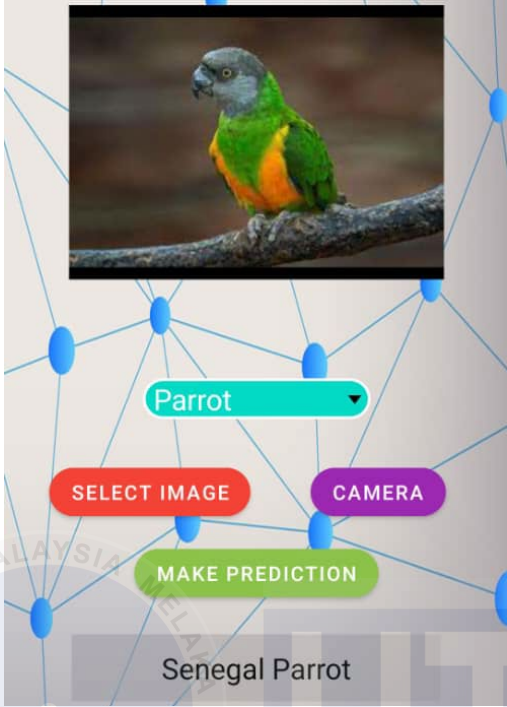
Correct.

7



Correct.

8		Correct.
9		Correct.

10		Correct.
----	---	----------

The accuracy of the parrot model:

$$Accuracy = \frac{9}{10} \times 100\%$$

$$Accuracy = 90\%$$



Figure 4.9: Amazon Parrot



Figure 4.10: Parrotlet

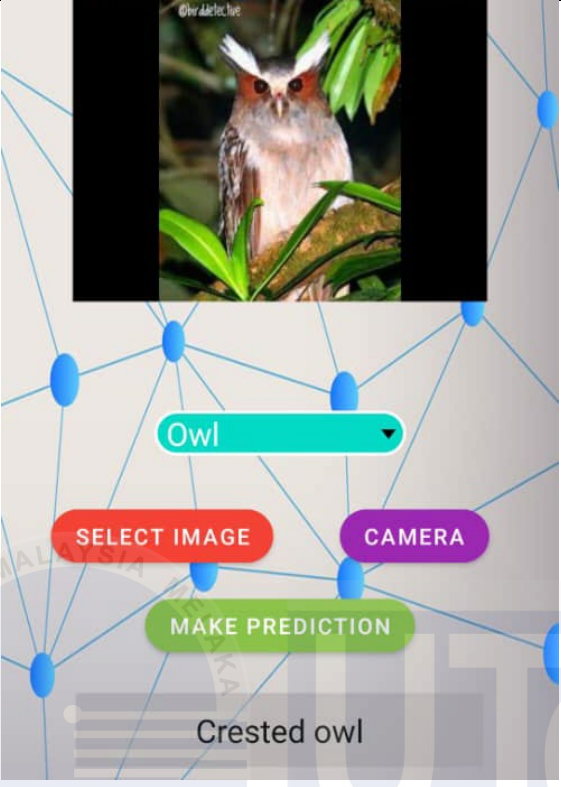
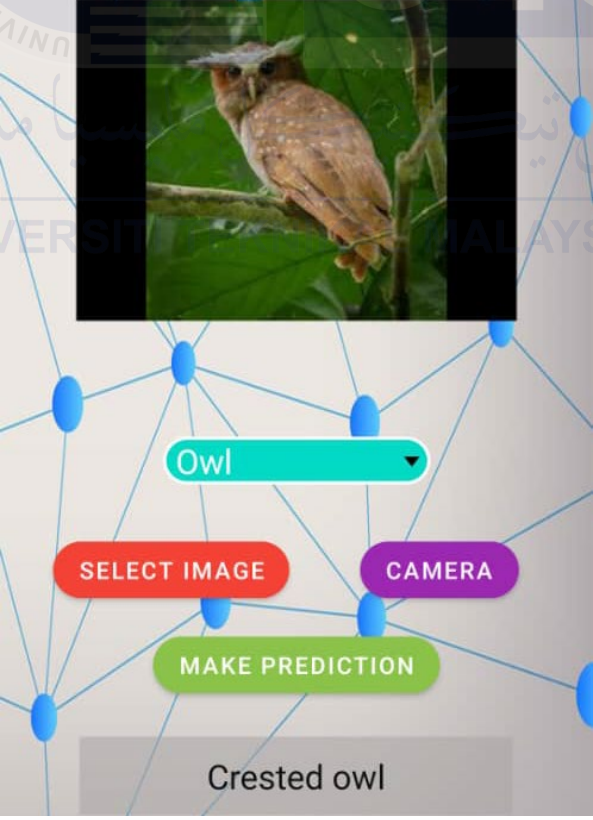
The accuracy of parrot model is 90%. From the figure 4.7 and 4.8, the amazon parrot and parrotlet have several similar characteristics like shape of body and body colour. It causes the app wrongly to detect it.

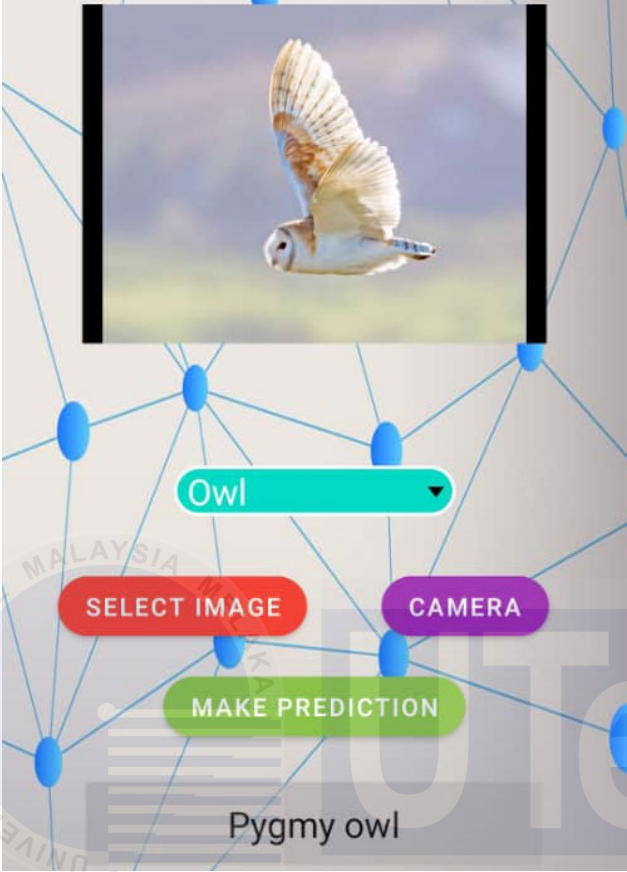
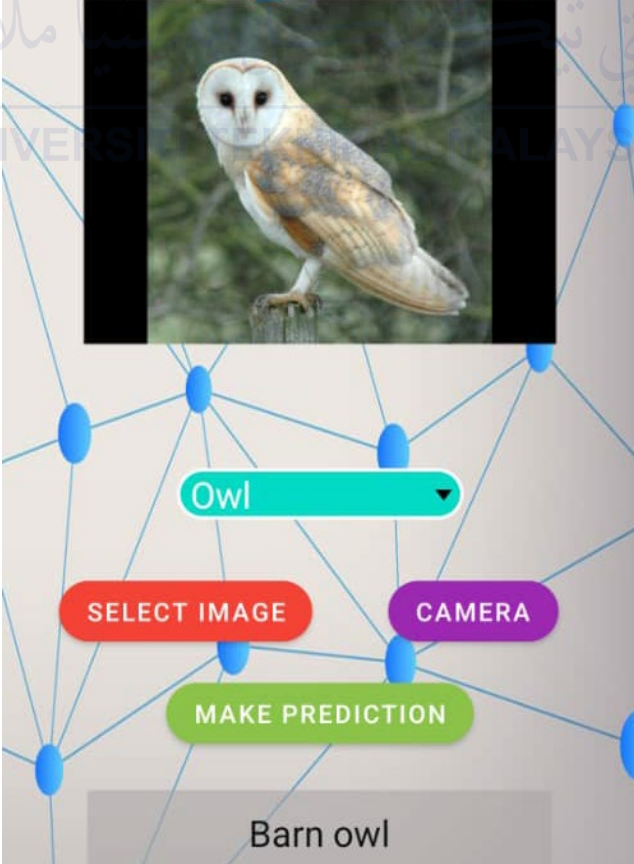
4.3.2.2 Owl Model

In this model, there are Barn owl, Crested owl, Pygmy owl, Snowy owl, and True owl.

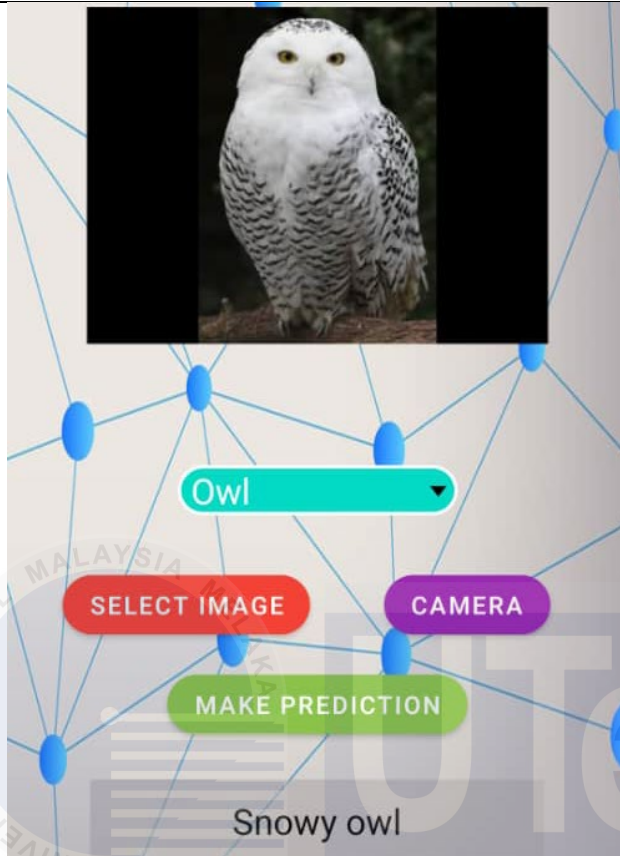
Table 4.2 Table of Accuracy Result for Accuracy Testing of Owl Model

No	Output	Result
----	--------	--------

1		Correct.
2		Correct.

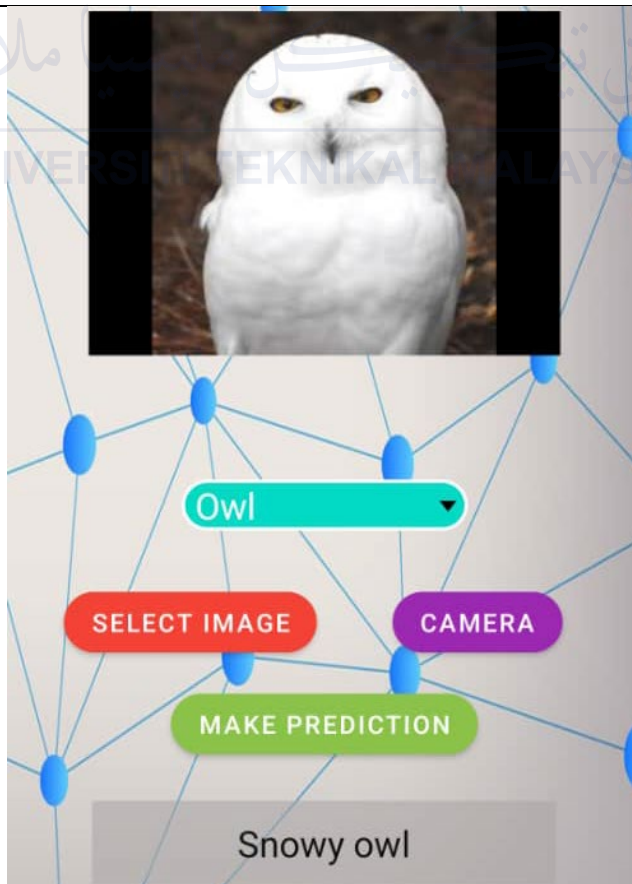
3		<p>Wrong. The result should be barn owl.</p>
4		<p>Correct.</p>

5

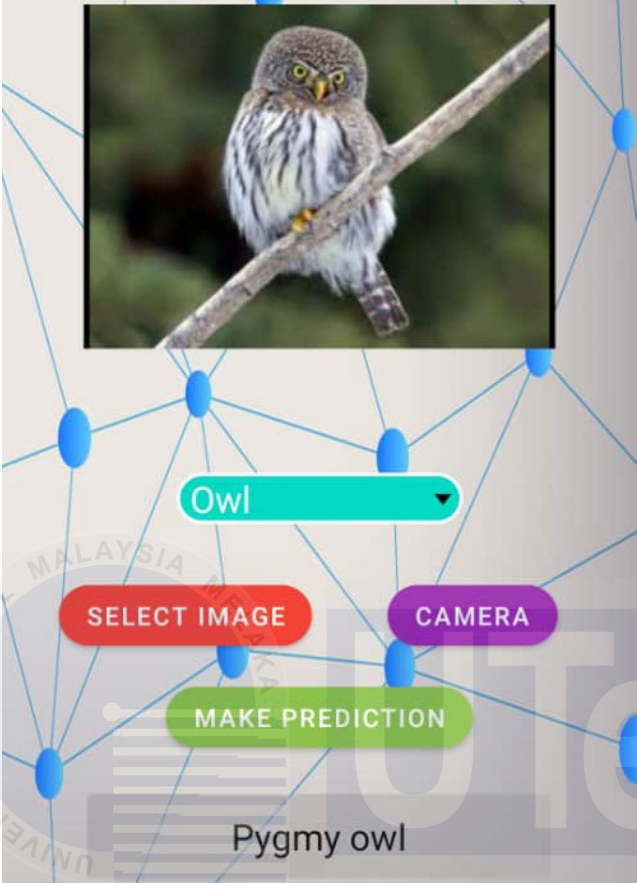
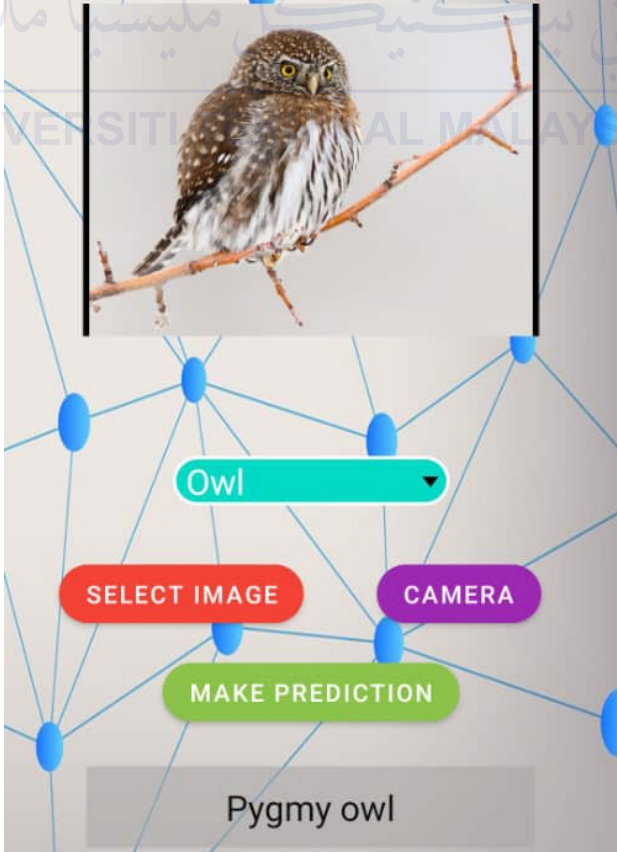


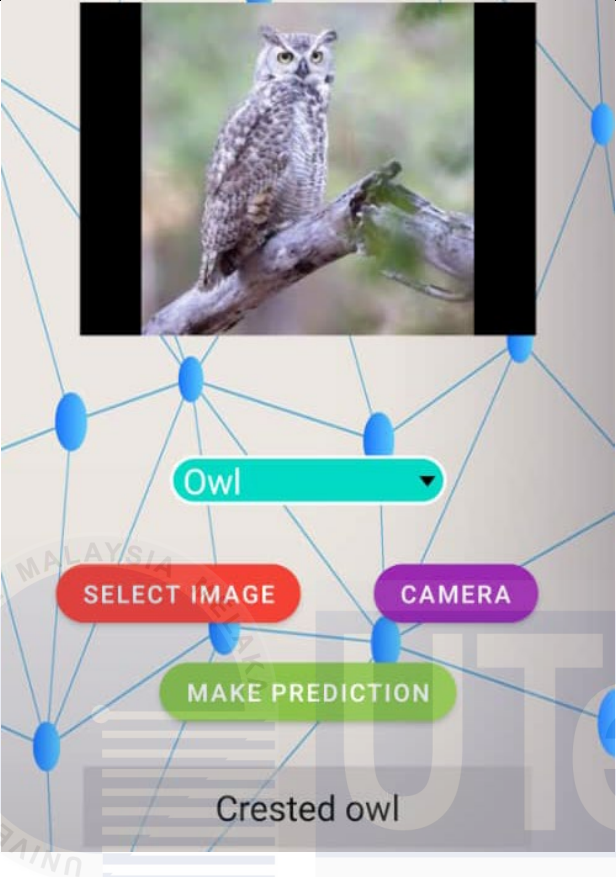

Correct.

6



Correct.

7		Correct.
8		Correct.

9		<p>Wrong. The result should be true owl.</p>
10		<p>Correct.</p>

The accuracy of the owl model:

$$Accuracy = \frac{8}{10} \times 100\%$$

$$Accuracy = 80\%$$

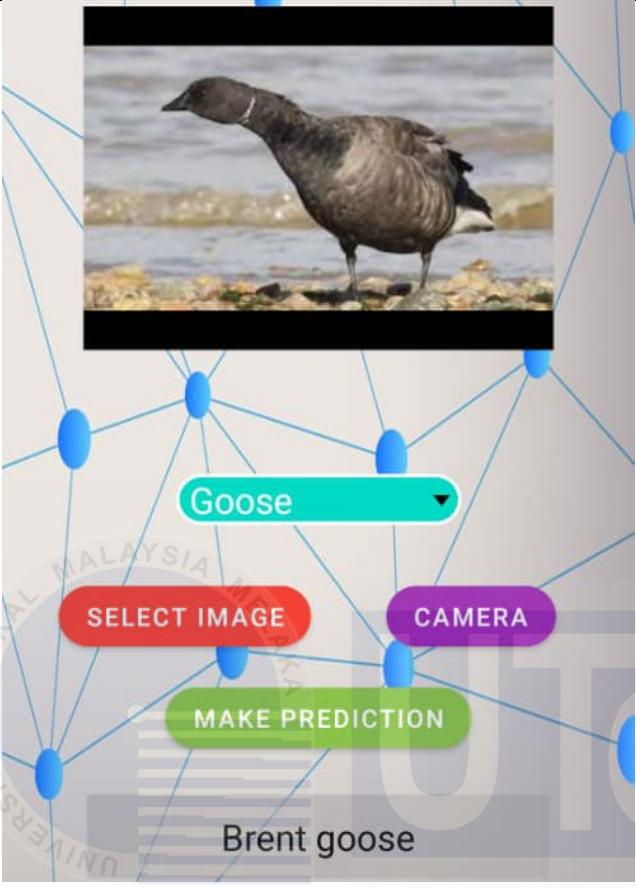
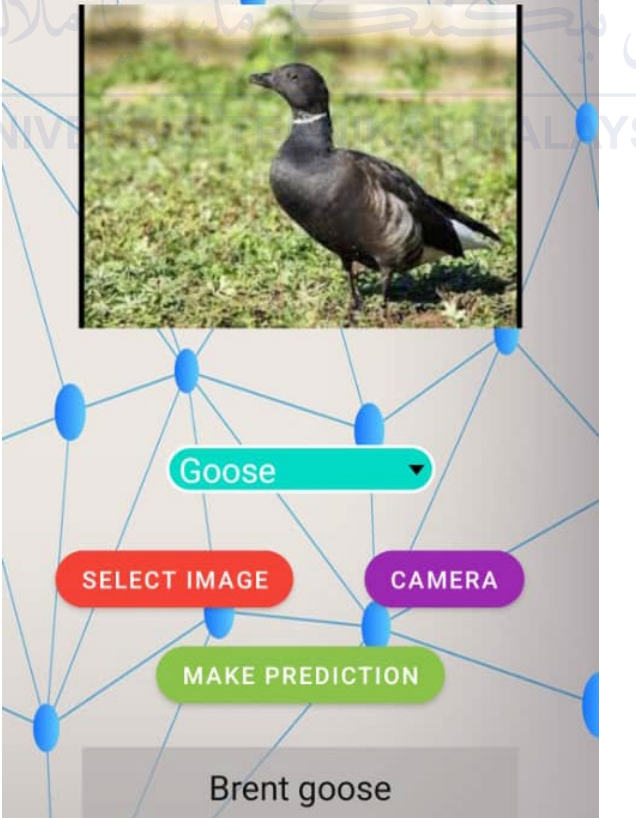
The accuracy of owl model is 80%. The one of the pygmy owls cannot be detected because the pattern or the feather are similar with barn owl. The one of the true owls also cannot be detected because the body shape and color are similar with crested owl.

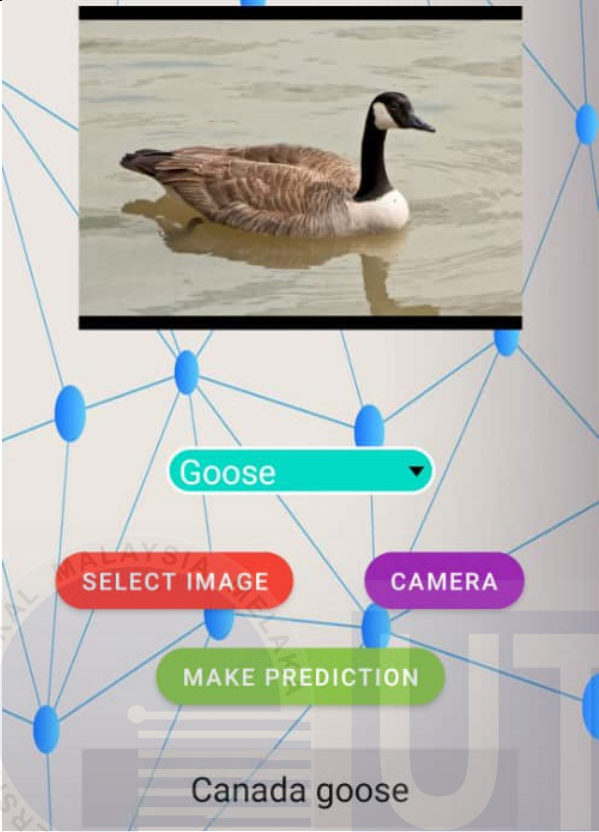
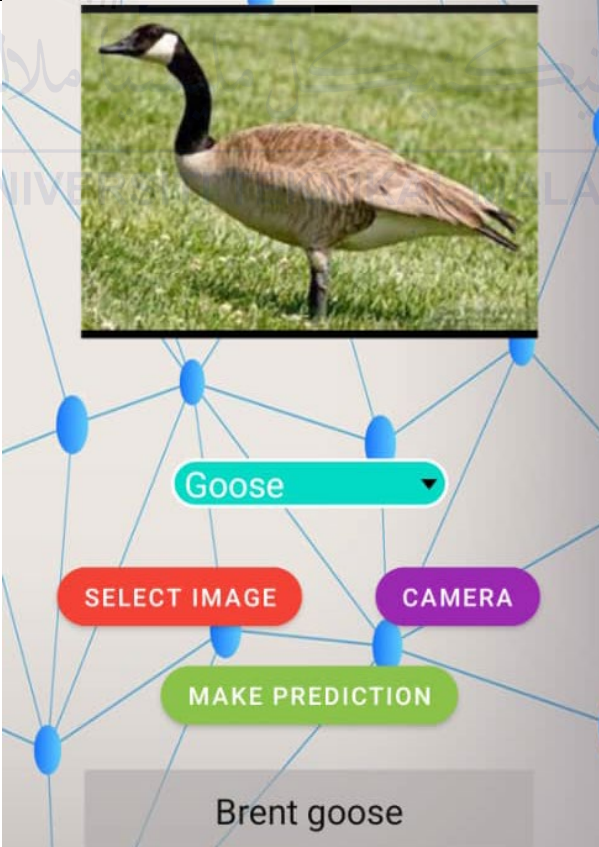
4.3.2.3 Goose Model

In this model, there are Brent goose, Canada goose, Emperor goose, Red-breasted goose and Snow goose.

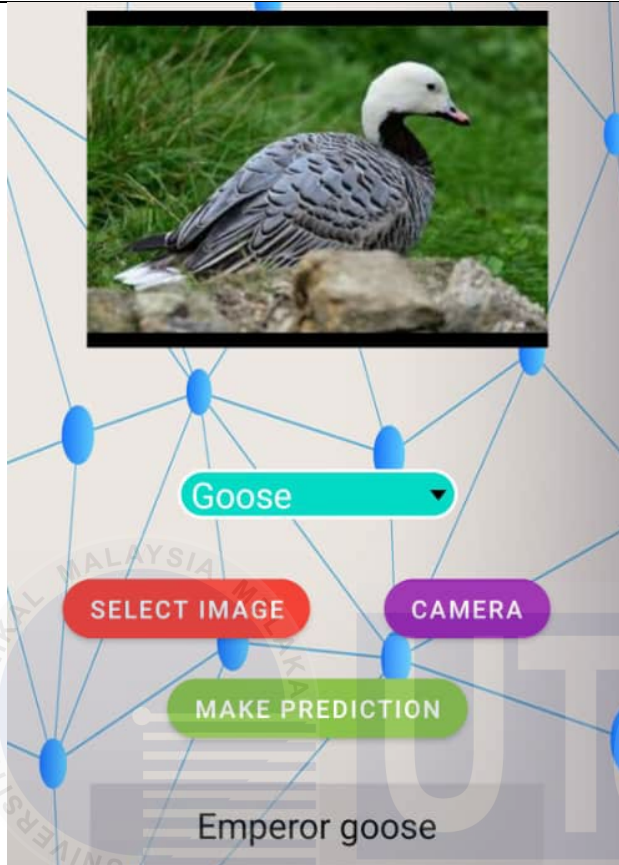
Table 4.3 Table of Accuracy Result for Accuracy Testing of Goose Model

No	Output	Result
----	--------	--------

1		Correct.
2		Correct.

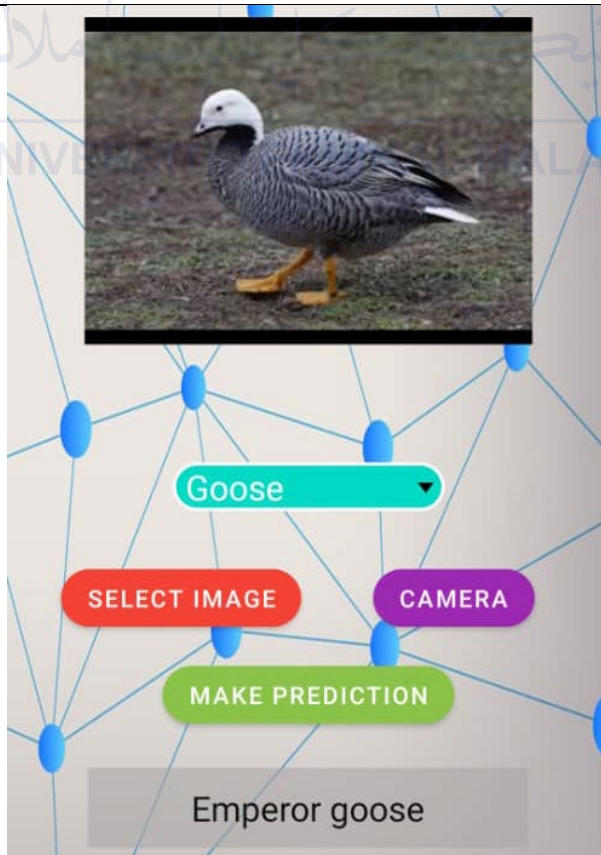
3		Correct.
4		Wrong. The result should be Canada goose.

5

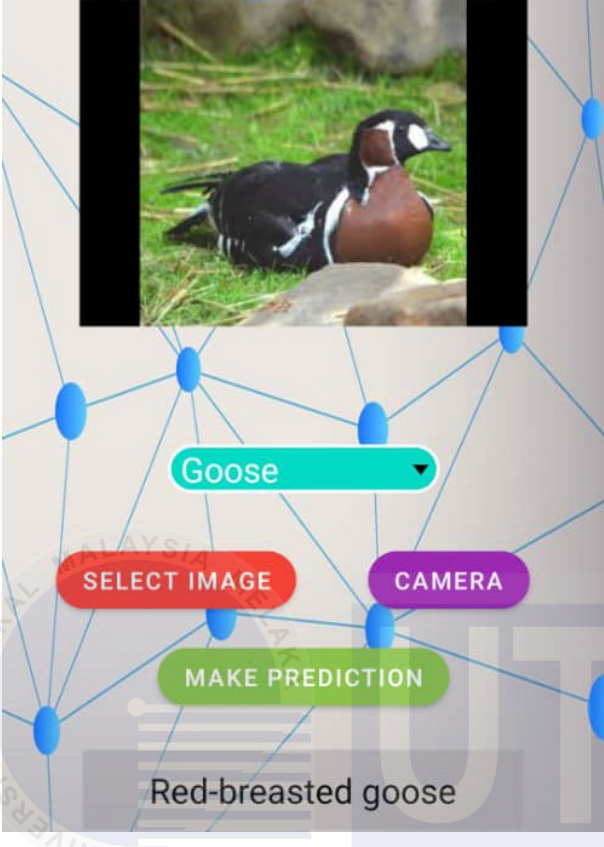
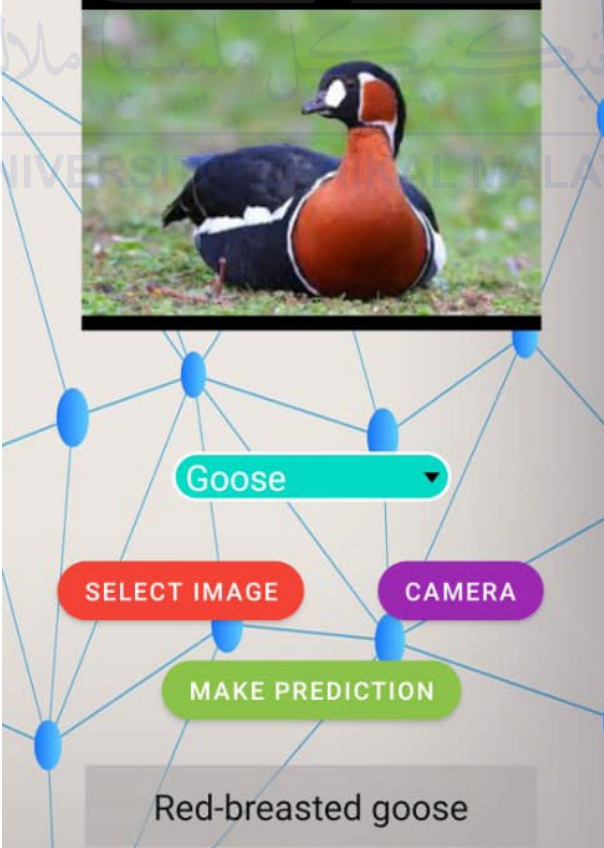


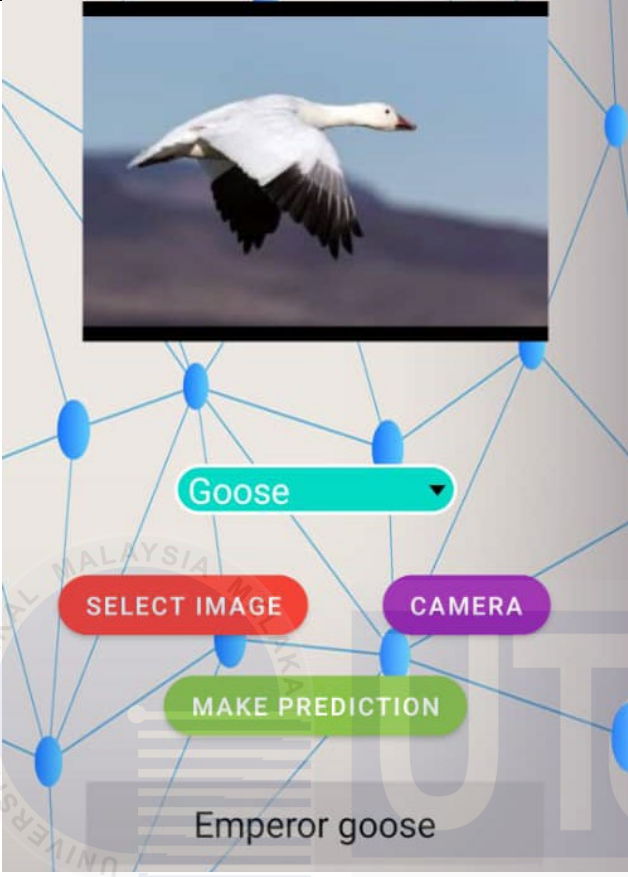
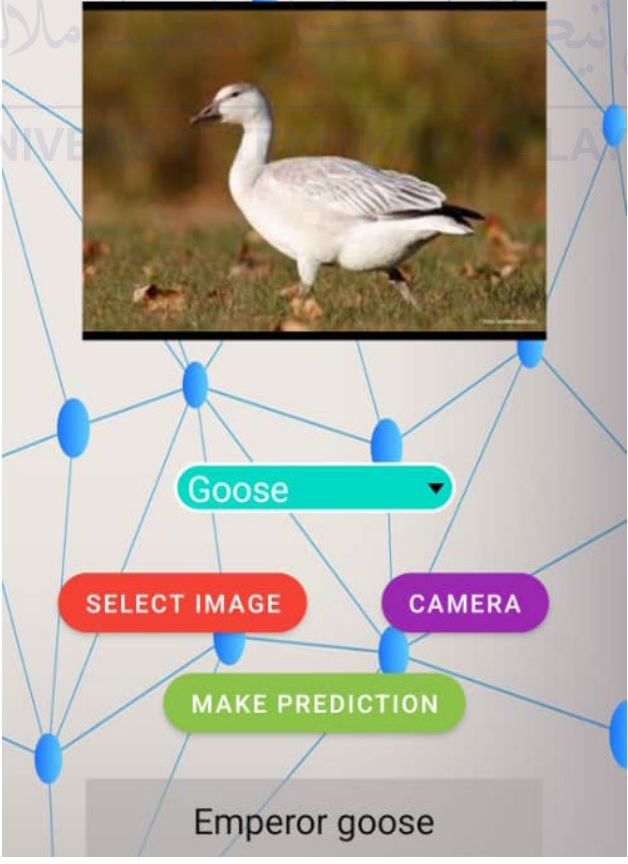
Correct.

6



Correct.

7		Correct.
8		Correct.

9		<p>Wrong. The result should be snowy goose.</p>
10		<p>Wrong. The result should be snowy goose.</p>

The accuracy of the goose model:

$$Accuracy = \frac{7}{10} \times 100\%$$

$$Accuracy = 70\%$$

The accuracy of the goose model is 70%. The app cannot detect one of the Canada geese and detect it as brent goose because color of the head and feather similar with brent goose. Beside Canada goose, two snowy geese cannot detect successfully by app and detected as emperor goose as the color of feather similar with emperor goose.

4.4 Summary

This chapter discusses the results and provides the performance evaluation on the image classification app. The user interfaces of the app have been designed successfully to integrate buttons and an image display area to facilitate users' usability. These UI elements are easy to interact with, that users could easily choose models and be able to make prediction and view the output.

The analysis focused on evaluating the performance of the models trained to classify images into 3 categories: owl, parrot, and goose. For each model, use 10 testing images to calculate their accuracy percentage. The results showed that the app could recognize pictures well highlighting how effective the chosen method was and how TensorFlow Lite worked smoothly for making predictions on the device itself.

The accuracy of each model is different. It can have many causes. The app detects the image by characteristics like shape, colour, texture, pattern and others. If two objects have high similarity for the characteristics, the app can detect it wrongly. For parrot model, it has

slightly similarity and obtain 90% accuracy. For owl model, it has low similarity and obtain 80% accuracy. For goose model, it has moderate similarity and obtain 70% accuracy. From the 3 models, we can discuss that the higher similarity, the lower the accuracy.



CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project successfully developed an image classification app using Android Studio and TensorFlow Lite, capable of classifying images into three categories: owl, parrot, and goose. The app also incorporates necessary user-friendly features like buttons and a specific image display area for a better usability and to make the interaction with the app smoother for the user. This involved building models with Teachable Machine, converting them into TensorFlow Lite format, and including them in the app for lightning fast on device prediction. The models were evaluated through testing and analysis using 10 testing images per category to find the accuracy percentage each model produced. The results showed that the proposed methodology succeeded in improving image classification ability with good accuracy.

Using this case study, we present how mobile based machine learning applications can bring advanced features like real time classification directly to the users' devices. With TensorFlow Lite, the app can provide low latency predictions as well as get the best performance for while on the go image classification tasks.

Overall, this project successfully developed a functional and productive image classification tool by integrating the machine learning and mobile app development and creates a broad basis for future innovations in this field.

5.2 Future Works

For future improvements, accuracy of the image classification could be enhanced as follows:

- i) Increasing data training size and data training diversity.
- ii) Rotation, flipping, cropping, and brightness adjustments applied as data augmentation techniques.
- iii) It is possible to expand the app, to classify other objects or animals.

5.3 Project Potential

The image classification app developed in this project can be utilized in various domains, including wildlife monitoring, education, and environmental conservation. For example, by classifying birds such as owls, parrots, and geese, the app could assist in ecological studies, help enthusiasts identify species, or serve as an educational tool for students learning about biodiversity. Additionally, the app can be extended to classify other objects, making it applicable in fields like retail, healthcare, and security.

The app's core functionality of real-time image classification offers a scalable business opportunity. It could be adapted into specialized software for industries requiring automated identification, such as agriculture (for identifying crops and pests) or manufacturing (for quality control). Its integration with mobile devices ensures accessibility, and with further development, it could be marketed as a standalone application or integrated into larger systems.

REFERENCES

- [1] Sharma, N., Jain, V., & Mishra, A. (2018). An analysis of convolutional neural networks for image classification. *Procedia Computer Science*, 132, 377–384. <https://doi.org/10.1016/j.procs.2018.05.198>
- [2] Luo, L. (2021). Research on image classification algorithm based on convolutional neural network. *Journal of Physics. Conference Series*, 2083(3), 032054. <https://doi.org/10.1088/1742-6596/2083/3/032054>
- [3] Chen, L., Li, S., Bai, Q., Yang, J., Jiang, S., & Miao, Y. (2021). Review of image classification algorithms based on convolutional neural networks. *Remote Sensing*, 13(22), 4712. <https://doi.org/10.3390/rs13224712>
- [4] Lv, Q., Zhang, S., & Wang, Y. (2022). Deep Learning model of image classification using Machine learning. *Advances in Multimedia*, 2022, 1–12. <https://doi.org/10.1155/2022/3351256>
- [5] Lu, D., & Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 28(5), 823–870. <https://doi.org/10.1080/01431160600746456>
- [6] Olimov, B., Subramanian, B., Ugli, R. a. A., Kim, J. S., & Kim, J. (2023). Consecutive multiscale feature learning-based image classification model. *Scientific Reports*, 13(1). <https://doi.org/10.1038/s41598-023-30480-8>
- [7] Özçevik, Y., & Sönmez, F. (2024). An embedded TensorFlow lite model for classification of chip images with respect to chip morphology depending on varying feed. *Journal of Intelligent Manufacturing*. <https://doi.org/10.1007/s10845-023-02320-z>

- [8] Shah, Vishal & Sajnani, Neha. (2020). Multi-Class Image Classification using CNN and Tflite. *International Journal of Research in Engineering, Science and Management*. 3. 65-68. 10.47607/ijresm.2020.375.
- [9] Senthilkumar, M. (2010). Use of artificial neural networks (ANNs) in colour measurement. In *Elsevier eBooks* (pp. 125–146). <https://doi.org/10.1533/9780857090195.1.125>
- [10] Madhiarasan, M., & Louzazni, M. (2022). Analysis of artificial Neural network: architecture, types, and forecasting applications. *Journal of Electrical and Computer Engineering*, 2022, 1–23. <https://doi.org/10.1155/2022/5416722>
- [11] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A. Q., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1). <https://doi.org/10.1186/s40537-021-00444-8>
- [12] Aggarwal, C. C. (2023). Neural networks and deep learning. In *Springer eBooks*. <https://doi.org/10.1007/978-3-031-29642-0>
- [13] R. K. Deshmukh, S. Markandey, and P. Sahu, “Mobile Application Development with Android,” *International Journal of Advances in Applied Sciences*, vol. 7, no. 4, p. 317, Dec. 2018, doi: 10.11591/ijaas.v7.i4.pp317-321.
- [14] G. B. Hertantyo, W. E. Putra, and M. W. B. Fahrizal, “DEVELOPMENT OF AN ANDROID APPLICATION-BASED ACADEMIC INFORMATION SYSTEM AT IMMIGRATION POLYTECHNIC,” *Technology Management and Informatics Research Journals*, vol. 3, no. 2, pp. 23–53, Dec. 2021, doi: 10.52617/tematics.v3i2.331.

[15] “TensorFlow: A system for large-scale machine learning.”
<https://research.google/pubs/tensorflow-a-system-for-large-scale-machine-learning/>

[16] L. Ma, L. Gu, and J. Wang, “Research and development of mobile application for Android platform,” 2014. <https://www.semanticscholar.org/paper/Research-and-Development-of-Mobile-Application-for-Ma-Gu/f859f00ed7bf29f3660c1b0474bc2d2639311150>

[17] “What is Android Studio IDE (Integrated Development Environment) | IGI Global.”
<https://www.igi-global.com/dictionary/an-empirical-study-of-mobilehandheld-app-development-using-android-platforms/60617>

[18] P. Liu, L. Li, K. Liu, S. McIntosh, and J. Grundy, “Understanding the quality and evolution of Android app build systems,” *Journal of Software*, Aug. 2023, doi: 10.1002/smr.2602.

[19] P. M. Andreae and J. H. Andreae, “A teachable machine in the real world,” *International Journal of Man-machine Studies*, vol. 10, no. 3, pp. 301–312, May 1978, doi: 10.1016/s0020-7373(78)80048-0.

[20] H. Jeong, “Feasibility Study of Google’s Teachable Machine in Diagnosis of Tooth-Marked Tongue,” *Chiwi’saeng’gwa Haghoeji/Chiwisaeng Gwahakoeji*, vol. 20, no. 4, pp. 206–212, Dec. 2020, doi: 10.17135/jdhs.2020.20.4.206.

[21] Cai, L., Gao, J., & Zhao, D. (2020). A review of the application of deep learning in medical image classification and segmentation. *Annals of Translational Medicine*, 8(11), 713. <https://doi.org/10.21037/atm.2020.02.44>



APPENDICES

Appendix A Coding for MainActivity.kt

```
package com.example.imageclassification

import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.Bitmap
import android.graphics.Color
import android.os.Bundle
import android.provider.MediaStore
import android.util.Log
import android.view.View
import android.widget.AdapterView
import android.widget.AdapterView.OnItemClickListener
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.ImageView
import android.widget.Spinner
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.imageclassification.ml.Goosemodel
import com.example.imageclassification.ml.Owlmodel
import com.example.imageclassification.ml.Parrotmodel
import org.tensorflow.lite.DataType
import org.tensorflow.lite.support.image.TensorImage
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer

class MainActivity : AppCompatActivity() {

    lateinit var select_image_button : Button
    lateinit var make_prediction : Button
    lateinit var img_view : ImageView
    lateinit var text_view : TextView
    lateinit var bitmap: Bitmap
    lateinit var camerabtn : Button

    public fun checkandGetpermissions(){
        if(checkSelfPermission(android.Manifest.permission.CAMERA) ==
        PackageManager.PERMISSION_DENIED){
            requestPermissions(arrayOf(android.Manifest.permission.CAMERA), 100)
        }
        else{
            Toast.makeText(this, "Camera permission granted",
            Toast.LENGTH_SHORT).show()
        }
    }
}
```



```

    }

    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<out String>,
        grantResults: IntArray
    ) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if(requestCode == 100){
            if(grantResults[0] == PackageManager.PERMISSION_GRANTED)
            {
                Toast.makeText(this, "Camera permission granted",
                Toast.LENGTH_SHORT).show()
            }
            else{
                Toast.makeText(this, "Permission Denied", Toast.LENGTH_SHORT).show()
            }
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        select_image_button = findViewById(R.id.button)
        make_prediction = findViewById(R.id.button2)
        img_view = findViewById(R.id.imageView2)
        text_view = findViewById(R.id.textView)
        camerabtn = findViewById<Button>(R.id.camerabtn)

        val spinner = findViewById<Spinner>(R.id.select)

        // handling permissions
        checkandGetpermissions()

        val labels = application.assets.open("parrot_labels.txt").bufferedReader().use
        { it.readText() }.split("\n")

        select_image_button.setOnClickListener(View.OnClickListener {
            Log.d("mssg", "button pressed")
            var intent : Intent = Intent(Intent.ACTION_GET_CONTENT)
            intent.type = "image/*"

            startActivityForResult(intent, 250)
        })

        // Spinner setup
        val options = listOf("Owl", "Parrot", "Goose")
        val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, options)

```



```

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
spinner.adapter = adapter

spinner.onItemSelectedListener = object : AdapterView.OnItemSelectedListener {
    override fun onItemSelected(parent: AdapterView<*>, view: View, position: Int,
id: Long) {
        spinner.post {
            (spinner.selectedView as? TextView)?.apply {
                textSize = 20f
                setTextColor(Color.WHITE)
            }
        }
    }
    override fun onNothingSelected(parent: AdapterView<*>) {
        Log.d("Spinner", "Nothing selected")
    }
}

make_prediction.setOnClickListener(View.OnClickListener {
    if (::bitmap.isInitialized) {
        Log.d("Error", "No picture submitted")
        Toast.makeText(this, "Please submit a picture first",
Toast.LENGTH_SHORT).show()
        return@OnClickListener
    }

    // Resize the bitmap
    val resized = Bitmap.createScaledBitmap(bitmap, 224, 224, true)
    val tbuffer = TensorImage.fromBitmap(resized)
    val byteBuffer = tbuffer.buffer

    // Load the appropriate model based on the selected item in the spinner
    val selectedModel = spinner.selectedItem.toString()
    val model: Any
    val labels: List<String>
    try {
        when (selectedModel) {
            "Owl" -> {
                model = Owlmodel.newInstance(this)
                labels = application.assets.open("owl_labels.txt").bufferedReader().use
{ it.readText() }.split("\n")
            }
            "Parrot" -> {
                model = Parrotmodel.newInstance(this)
                labels = application.assets.open("parrot_labels.txt").bufferedReader().use
{ it.readText() }.split("\n")
            }
        }
    }
}

```



```

        "Goose" -> {
            model = Goosemodel.newInstance(this)
            labels = application.assets.open("goose_labels.txt").bufferedReader().use
{ it.readText() }.split("\n")
        }
        else -> {
            Log.d("Error", "Invalid model selected")
            Toast.makeText(this, "Invalid model selected",
Toast.LENGTH_SHORT).show()
            return@OnClickListener
        }
    }
} catch (e: Exception) {
    Log.e("ModelError", "Failed to load model: ${e.message}")
    Toast.makeText(this, "Error loading model",
Toast.LENGTH_SHORT).show()
    return@OnClickListener
}

// Create input feature and run model inference
val inputFeature0 = TensorBuffer.createFixedSize(intArrayOf(1, 224, 224, 3),
DataType.UINT8)
inputFeature0.loadBuffer(byteBuffer)

val outputs = when (model) {
    is Owlmodel -> model.process(inputFeature0).outputFeature0AsTensorBuffer
    is Parrotmodel ->
model.process(inputFeature0).outputFeature0AsTensorBuffer
    is Goosemodel ->
model.process(inputFeature0).outputFeature0AsTensorBuffer
    else -> null
}

if (outputs != null) {
    val max = getMax(outputs.floatArray)
    text_view.text = labels[max]
    Log.d("Prediction", "Predicted: ${labels[max]}")
} else {
    Log.e("PredictionError", "Failed to process model outputs")
}

// Close the model to release resources
when (model) {
    is Owlmodel -> model.close()
    is Parrotmodel -> model.close()
    is Goosemodel -> model.close()
}
})

```



```

camerabtn.setOnClickListener(View.OnClickListener {
    var camera : Intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
    startActivityForResult(camera, 200)
})

}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    if (requestCode == 250 && resultCode == RESULT_OK) {
        data?.data?.let { uri ->
            img_view.setImageURI(uri)
            bitmap = MediaStore.Images.Media.getBitmap(this.contentResolver, uri)
        } ?: run {
            Log.d("Error", "No image selected")
            Toast.makeText(this, "No image selected", Toast.LENGTH_SHORT).show()
        }
    } else if (requestCode == 200 && resultCode == RESULT_OK) {
        data?.extras?.get("data")?.let { imageData ->
            bitmap = imageData as Bitmap
            img_view.setImageBitmap(bitmap)
        } ?: run {
            Log.d("Error", "No image captured")
            Toast.makeText(this, "No image captured", Toast.LENGTH_SHORT).show()
        }
    }
}

fun getMax(arr: FloatArray): Int {
    var ind = 0
    var max = arr[0]

    for (i in arr.indices) {
        if (arr[i] > max) {
            max = arr[i]
            ind = i
        }
    }
    return ind
}
}

```


Appendix B Task Schedule of PSM1

No	Task	PSM1											
	Weeks	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
1	Create project title												
2	Do research about project title												
3	Briefing with supervisor												
4	Fill name and title to thesis template												
5	Research about Chapter 1 Introduction												
6	Drafting Chapter 1: Introduction												
7	Checking Chapter 1: Introduction												
8	Drafting Chapter 2: Literature Review												
9	Checking Chapter 2: Literature Review												
10	Drafting Chapter 3: Methodology												
11	Checking Chapter 3: Methodology												

Appendix C Task Schedule of PSM2

No	Task		PSM2											
	Weeks		W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
1	Collect training image and testing image data													
2	Design UI and write code.													
3	Train model using Teachable Machine													
4	Testing accuracy of model													
5	Implement model into Android Studio													
6	Drafting Chapter 4: Results and Discussions													
7	Checking Chapter 4: Results and Discussions													
8	Drafting Chapter 4: Conclusion and Recommendations													
9	Checking Chapter 4: Conclusion and Recommendations													



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA