

**OBJECT RECOGNITION MOBILE APP
FOR VISUALLY IMPAIRED USER**

WON WEI CHENG

**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Science
(Honours) Software Engineering**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

April 2021

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : Wei Cheng

Name : Won Wei Cheng

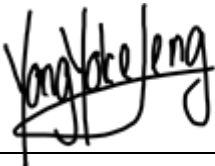
ID No. : 1803164

Date : 13-4-2021

APPROVAL FOR SUBMISSION

I certify that this project report entitled **OBJECT RECOGNITION MOBILE APP FOR VISUALLY IMPAIRED USER** was prepared by **WON WEI CHENG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Dr Yong Yoke Leng

Date : 14 April 2021

Signature : _____

Co-Supervisor : _____

Date : _____

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

©2021, Won Wei Cheng. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Yong Yoke Leng for her invaluable advice, guidance and her enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement and support during the development of the project.

ABSTRACT

Vision is one of the most important human senses and it plays a critical role in understanding the surrounding environment. However, millions of people in the world experience visual impairment. These people face difficulties in their daily navigation since they are unable to see the obstacles in their surroundings. Despite there are many options such as white canes and different advanced technologies to help visually impaired people when navigating, some of the options are unreliable, expensive, and hard to access. Hence, a mobile application is proposed to help visually impaired people to recognise objects in their surroundings using real-time object detection and object recognition techniques. This project also has applied transfer learning on multiple pre-trained models to train the models that are able to classify 40 classes of objects. The performance of the trained models is compared to select a suitable model to be implemented in the mobile application. The Evolutionary Prototyping Model is the development methodology adopted in this project. It involves developing the application in a series of iterations and refining the application based on feedback collected in each iteration. A literature review was conducted on similar existing mobile applications to understand the machine learning framework used for the implementation of object detection and recognition, and also identify the important features and workflow within the application. Finally, an Android-based mobile application was developed successfully and passed all testing. In conclusion, this project has helped visually impaired people to determine the objects in their surrounding in a more cost-effective, accessible and reliable way. They are being informed of the names and directions of the detected objects in the surroundings through voice feedback without requiring any network connection or photo capturing.

TABLE OF CONTENTS

DECLARATION		i
APPROVAL FOR SUBMISSION		ii
ACKNOWLEDGEMENTS		iv
ABSTRACT		v
TABLE OF CONTENTS		i
LIST OF TABLES		v
LIST OF FIGURES		vii
LIST OF SYMBOLS / ABBREVIATIONS		x
LIST OF APPENDICES		xi
CHAPTER		
1	INTRODUCTION	1
1.1	Introduction	1
1.2	Background	1
1.3	Problem Statement	2
1.3.1	Safety Issues When Navigating Using White Canes	2
1.3.2	Accessibility and Affordability Issues of Advanced Technologies	3
1.4	Goal and Sub-Objectives	3
1.4.1	Goal	3
1.4.2	Sub-Objectives	4
1.5	Final Solution	4
1.6	Final Approach	6
1.7	Project Scope	7
1.7.1	Delimitations	7
1.7.2	Limitations	8
1.7.3	Assumptions	9
2	LITERATURE REVIEW	10

2.1	Introduction	10
2.2	Object Detection and Object Recognition Concepts	10
2.2.1	Object Detection and Object Recognition	10
2.2.2	Intersect over Union	11
2.3	State-of-Art Methods for Object Detection and Object Recognition	12
2.3.1	Region-based Convolutional Neural Network (R-CNN)	12
2.3.2	Fast Region-based Convolutional Neural Network (Fast R-CNN)	14
2.3.3	Faster Region-based Convolutional Neural Network (Faster R-CNN)	15
2.3.4	You Only Look Once (YOLO)	16
2.3.5	Single-Shot Multibox Detector (SSD)	17
2.4	Transfer Learning	18
2.4.1	Transfer Learning and Traditional Machine Learning	18
2.4.2	Formal Definitions of Transfer Learning	20
2.4.3	Transfer Learning Settings	21
2.4.4	Transfer Learning in The Project	23
2.5	Existing Object Detection and Object Recognition Mobile Application	26
2.6	API Based Object Detection and Object Recognition Mobile Application for Visually Impaired User	29
2.6.1	Mode	29
2.6.2	Application Programming Interface (API)	30
2.6.3	Conclusion	33
3	PROJECT METHODOLOGY AND PLANNING	34
3.1	Introduction	34
3.2	Software Development Methodology	34
3.2.1	Requirements Gathering	34
3.2.2	Iteration Process	35
3.2.3	Testing	36
3.3	Project Plan	36

	3.3.1	Work Breakdown Structure	37
	3.3.2	Gantt Chart	37
3.4		Development Tools	37
	3.4.1	Programming Languages	37
	3.4.2	Framework	38
	3.4.3	Tools and Integrated Development Environment (IDE)	38
	3.4.4	Version Control System	40
4		PROJECT SPECIFICATION	42
	4.1	Introduction	42
	4.2	Requirements Discovery	42
	4.2.1	Features of Similar Existing Mobile Applications	42
	4.2.2	Comparison between Existing Mobile Application	51
	4.2.3	Conclusion	53
	4.3	Requirement Specification	54
	4.3.1	Functional Requirements	54
	4.3.2	Non Functional Requirements	55
	4.4	Use Case Modeling	56
	4.4.1	Use Case Diagram	56
	4.4.2	Use Case Description	56
	4.5	Proposed User Interface Design	63
5		PROJECT IMPLEMENTATION	65
	5.1	Introduction	65
	5.2	Pre-training	66
	5.2.1	Dataset Preparation	66
	5.2.2	Pre-trained Model Selection	67
	5.3	Training	68
	5.3.1	Preparation of Labelmap File	68
	5.3.2	Data Loading	68
	5.3.3	Model Training	69
	5.4	Export Trained Model	69
	5.5	Evaluation	69

5.6	Implementation of Mobile Application	70
5.6.1	Object Detection and Recognition on Mobile Application	70
5.6.2	Mobile Application User Interface	71
5.6.3	Reporting of Directions of Detected Objects	72
5.6.4	Adjusting Speech Rate	73
6	RESULTS AND DISCUSSIONS	75
6.1	Introduction	75
6.2	Training Settings	75
6.2.1	Batch Size	75
6.2.2	Training Time of Different Models	78
6.3	Evaluation	79
6.3.1	Comparison of Performance between Pre-trained and Fine-tuned Models	79
6.3.2	Comparison of Performance between Fine-tuned SSD and Faster R-CNN Models	86
6.4	Summary	97
7	SYSTEM TESTING	99
7.1	Introduction	99
7.2	Unit Testing	99
7.2.1	Object Detection and Recognition Module	99
7.2.2	Speech Feedback Module	101
7.2.3	Set Speech Rate Module	103
7.2.4	Play/Stop Speech Module	104
7.3	Integration Test	104
7.4	Usability Test	107
8	CONCLUSIONS	109
8.1	Introduction	109
8.2	Conclusions	109
8.3	Challenges	110
8.4	Future Enhancements	110
	REFERENCES	111
	APPENDICES	116

LIST OF TABLES

Table 2.1: Different Settings of Transfer Learning	22
Table 2.2: List of Object Classes for Pre-trained Models and Target Models	24
Table 2.3: Papers of Mobile Applications that Apply Object Detection and Object Recognition Techniques	27
Table 2.4: Comparison between Recognition Accuracy of Different APIs	31
Table 2.5: Comparison between Google Cloud Vision API and Tensorflow Object Detection API	32
Table 3.1: System Information of Tested Local Machine	39
Table 3.2: Comparison of Training Performance	39
Table 4.1: Comparison between Existing Object Recognition Mobile Applications for Visually Impaired People	51
Table 4.2: Scan Surroundings Use Case	56
Table 4.3: Detect and Recognise Objects Use Case	57
Table 4.4: View Information of Detected Objects Use Case	58
Table 4.5: Receive Speech Feedback Use Case	59
Table 4.6: Stop Speech Use Case	60
Table 4.7: Play Speech Use Case	61
Table 4.8: Set Speech Rate Use Case	62
Table 5.1: Object Classes to be Trained in Transfer Learning	65
Table 6.1: Comparison of Training Performance When Using Different Batch Size	75
Table 6.2: Comparison of Training Time Between SSD and Faster R-CNN Models	78

Table 6.3: mAP Values of Pre-trained and Fine-tuned SSD and Faster R-CNN Models	86
Table 6.4: True Positive, False Negative, and False Positive of Fine-tuned SSD and Faster R-CNN Models	89
Table 6.5: Precision and Recall Values for 40 Classes of Fine-tuned SSD and Faster R-CNN Models	92
Table 7.1: Unit Test for Object Detection and Recognition	99
Table 7.2: Unit Test for Speech Feedback	101
Table 7.3: Unit Test for Set Speech Rate	103
Table 7.4: Unit Test for Play and Stop Speech	104
Table 7.5: Integration Test	104
Table 7.6: Usability Test Results	107

LIST OF FIGURES

Figure 1.1: Proposed System Overview	5
Figure 1.2: Evolutionary Prototyping Model	7
Figure 2.1: Formula of Intersect over Union (Alsing, 2018)	11
Figure 2.2: Architecture in R-CNN (Argawal, 2018)	13
Figure 2.3: Architecture in Fast R-CNN (Argawal, 2018)	14
Figure 2.4: Architecture in Faster R-CNN (Liu et al., 2019)	16
Figure 2.5: Concept of YOLO in Object Detection and Recognition (Zhao et al., 2019)	17
Figure 2.6: Architecture in SSD (Liu et al., 2019)	18
Figure 2.7: Traditional Learning and Transfer Learning (Pan and Yang, 2009)	19
Figure 2.8: Off-The-Shelf Feature Extraction	26
Figure 3.1: Evolutionary Prototyping Model	34
Figure 3.2: Interface of LabelImg Annotation Tool	40
Figure 4.1: Options Available to Users on the Intelligence Eye Home Screen	43
Figure 4.2: Screenshot of Real-Time Object Recognition in Intelligence Eye	43
Figure 4.3: Screenshot of Object Recognition	45
Figure 4.4: Image and Text Recognition in the Applicaton	46
Figure 4.5: Object Recognition on the Android Application	47
Figure 4.6: Homescreen of Intelligence Eye	48
Figure 4.7: Image Capturing and Object Recognition in Intelligence Eye	48
Figure 4.8: Banknote Capturing and Recognition in Intelligence Eye	49

Figure 4.9: Object Recognition in the Application	50
Figure 4.10: Object Detection and Recognition in the Application	51
Figure 4.11: Use Case Diagram of the Application	56
Figure 4.12: UI Design for Object Recognition Screen	63
Figure 4.13: Recognition for Multiple Objects	64
Figure 5.1: Summary of the Training Process	66
Figure 5.2: Display of Certain Records in CSV File	67
Figure 5.3: Display of Some Classes in Labelmap File	68
Figure 5.4: Process Flow for Object Detection and Recognition on Mobile Application	70
Figure 5.5: Screen when Detecting Single Object	71
Figure 5.6: Screen when Detecting Multiple Objects	72
Figure 5.7: Splitting of Mobile Screen into Three Parts	73
Figure 5.8: Adjust Speech Rate Screen	74
Figure 6.1: Development of Total Loss During Training When Batch Size=4 (After 2 h 31 min of Training)	76
Figure 6.2: Development of Total Loss During Training When Batch Size=12 (After 5 h 17 min of Training)	76
Figure 6.3: Development of Total Loss During Training When Batch Size=24 (After 2 h 57 min of Training)	77
Figure 6.4: Development of Total Loss During Training When Batch Size=36 (After 5 h 41 min of Training)	77
Figure 6.5: Detections Results of Different Models in Toilet	80
Figure 6.6: Detections Results of Different Models in Bedroom	81

Figure 6.7: Detections Results of Different Models at Outside	82
Figure 6.8: AP Values of Pre-trained and Fine-tuned SSD Models	84
Figure 6.9: AP Values of Pre-trained and Fine-tuned Faster R-CNN Models	85
Figure 6.10: Multiclass Confusion Matrix of Fine-tuned SSD Model	87
Figure 6.11: Multiclass Confusion Matrix of Fine-tuned Faster R-CNN Model	88
Figure 6.12: Total Number of True Positives, False Negatives, and False Positives against Fine-tuned Models	90
Figure 6.13: Comparison of AP Values for 40 Classes between Fine-tuned SSD and Faster R-CNN Models	95
Figure 6.14: Comparison of mAP Values between Fine-tuned SSD and Faster R-CNN Models	96
Figure 6.15: Speed and Accuracy Comparison of Fine-tuned Models	97
Figure 6.16: Comparison of mAP Values between Different Models	98

LIST OF SYMBOLS / ABBREVIATIONS

<i>D</i>	Domain
<i>T</i>	Task
\mathcal{X}	Feature Space
<i>Y</i>	Label Space
<i>API</i>	Application Programming Interface
<i>IoU</i>	Intersect over Union
<i>mAP</i>	Mean Average Precision
<i>CNN</i>	Convolutional Neural Network
<i>R-CNN</i>	Region-based Convolutional Neural Network
<i>YOLO</i>	You Look Only Once
<i>FPS</i>	Frame Per Second
<i>SSD</i>	Single-Shot Multibox Detector
<i>HTML</i>	HyperText Markup Language
<i>XML</i>	Extensible Markup Language
<i>IDE</i>	Integrated Development Environment
<i>CPU</i>	Central Processing Unit
<i>GPU</i>	Graphics Processing Unit
<i>TPU</i>	Tensor Processing Unit
<i>CUDA</i>	Compute Unified Device Architecture
<i>PASCAL</i>	Pattern Analysis, Statistical Modelling and Computational Learning
<i>VOC</i>	Visual Object Classes
<i>GUI</i>	Graphical User Interface
<i>COCO</i>	Common Objects in Context
<i>CSV</i>	Comma-Separated Values

LIST OF APPENDICES

APPENDIX A: Work Breakdown Structure	116
APPENDIX B: Gantt Chart	118

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter proposes the background, problem statement, goal and sub-objectives, proposed solution, proposed approach, and project scope of the project.

1.2 Background

Smartphones have become a very significant device in our lives. They allow us to access various services and information more easily. Nevertheless, there are millions of people unable to see the environment in this world due to visual impairment. Visually impairment hinders the people from carrying out a lot of daily activities. It is a challenge for them to travel independently since they are unable to see the obstacles around them. They always require someone to guide them when navigating around to prevent injuries and accidents. Furthermore, they also face difficulties to complete their daily tasks such as reading and finding an object. They may need help from others to complete these tasks. However, these already increase the burden of family members and friends of a visually impaired person.

With the common use of smartphones, visually impaired people are able obtain benefits from smartphone applications. Several types of mobile applications have been invented to help the visually impaired people, such as text readers that read out text on books and documents, color readers that notify the visually impaired user regarding the colour information of an object, navigation assistance that helps visually impaired users to navigate around by telling the route, and other applications. These applications allow visually impaired people to do some simple tasks independently without having to seek help from others. The project is proposed to aid the visually impaired people using computer vision, object detection, and object recognition techniques by building a mobile application that detects and recognizes objects in the surroundings and gives audio feedback.

1.3 Problem Statement

Vision is critical in our daily lives. However, according to the World Health Organization (2019), there were more than 2.2 billion people worldwide suffering from visual impairment. These people are unable to view the surrounding objects unlike a person with normal vision. They face challenges in detecting obstacles when navigating (Rajwani et al., 2018; Thakare et al., 2017). Although there are several options available for visually impaired people to help them when navigating, such as white canes and advanced technologies, they still encounter a few problems when accessing or using the tools.

1.3.1 Safety Issues When Navigating Using White Canes

In Malaysia, the white cane is accepted as a symbol of blindness (National Council For The Blind Malaysia, 2020). White canes are widely used by visually impaired people in detecting obstacles since they are cheap and easy to get (Khan, Khusro and Ullah, 2018; Santos et al., 2020; Chanana et al., 2017). The canes are painted white to make others notice them easily, especially when navigating (Industries For The Blind And Visually Impaired, 2020).

But, white canes cannot help them to determine the type of objects in front of them. Therefore, visually impaired people usually identify the object in front of them based on their own experience (Parikh, Shah and Safvan Vahora, 2018). Unfortunately, Santos et al. (2020) and Chanana et al. (2017) states that they may make incorrect expectations, which can cause them injury.

Besides, white canes are also unable to detect the obstacles above the waist level (Khan, Khusro and Ullah, 2018; Santos et al., 2020; Chanana et al., 2017). According to Santos et al. (2020), 40% of visually impaired people experienced at least one head accidents every year. Santos et al. (2020) also reported that 23% of accidents had medical consequences. Therefore, white canes expose visually impaired people to the risks of colliding with the obstacles above the waist level, such as tree branches, windows, and floating shelves.

1.3.2 Accessibility and Affordability Issues of Advanced Technologies

Several types of technologies that involve special devices have been developed to help visually impaired people. One of the examples is smart glasses with a camera to capture the user's surroundings and send images to a smartphone for processing (Thakare et al., 2017). Smart stick with ultrasonic, infrared, or laser sensors also has been developed to inform user information of obstacles (Sharma et al., 2017). Moreover, Radio Frequency Identification (RFID) tag and RFID reader is used to assist visually impaired people. RFID tags are attached to the objects so that the user can identify and locate objects more easily (Abdul Malik Shaari and Nur Safwati, 2017). However, although these technologies are able to help the visually impaired users to do their tasks independently and safely, most of these technologies are expensive and hard to access (Anitha, Subalaxmi and Vijayalakshmi, 2019; Awad et al., 2018; Rajwani et al., 2018).

1.4 Goal and Sub-Objectives

This section discusses goal and sub-objectives.

1.4.1 Goal

The goal of this project is to implement an Android-based mobile application that detects and recognises multiple objects captured by the smartphone's camera in real-time and provides audio feedback to assist visually impaired people identifying surrounding objects more easily. The object detection and recognition are achieved using transfer learning from existing pre-trained object recognition models. The system also incorporates new classes for detection and recognition.

Since only a smartphone is needed to identify surrounding objects without any other hardware, this can solve the accessibility issues of advanced technologies because smartphones are easier to be accessed than these special devices. Furthermore, smartphones are more affordable than these advanced technologies. The price of smart glasses ranges from USD2,950 to USD5,950, while the price of smart sticks ranges between RM579 and RM699, which are more expensive than a low-end smartphone (BAWA, 2020; Wewalk, 2020; IrisVision Global, 2021).

1.4.2 Sub-Objectives

- i. To apply transfer learning to multiple pre-trained object detection models then train the models to detect and recognize 40 classes of objects. This also helps to fine-tune the pre-trained models and improve the models performance.
- ii. To develop a real-time application that is able to analyse the surrounding scene and detect a maximum of 10 objects within the camera's field-of-view.
- iii. To evaluate the training performance and recognition accuracy of different pre-trained object detection models before and after transfer learning.
- iv. To develop an application that uses voice feedback to notify the user of the names and directions of detected objects. This helps to reduce the safety issues encountered when navigating with the white cane.

1.5 Final Solution

With the rapid advancement of technology, smartphones have become a familiar and highly available device (Rajwani et al., 2018; Khan Shishir et al., 2019; Anitha, Subalaxmi and Vijayalakshmi, 2019). According to TheStar (2018), smartphone penetration in Malaysia stood at 70% in the third quarter of 2017. Hence, the final solution proposed an Android mobile application to assist visually impaired people to detect objects around them using real-time object detection and object recognition techniques. Since the smartphone is the only device needed, this solution is more cost-effective and easier to access rather than the technologies that need special devices. Furthermore, it is safer than white canes because the application is able to determine and inform the types of obstacles so that visually impaired people can avoid making wrong assumptions when navigating. The application also helps users to detect and identify obstacles above the waist level. An overall system architecture was designed to describe the solution, as illustrated below.

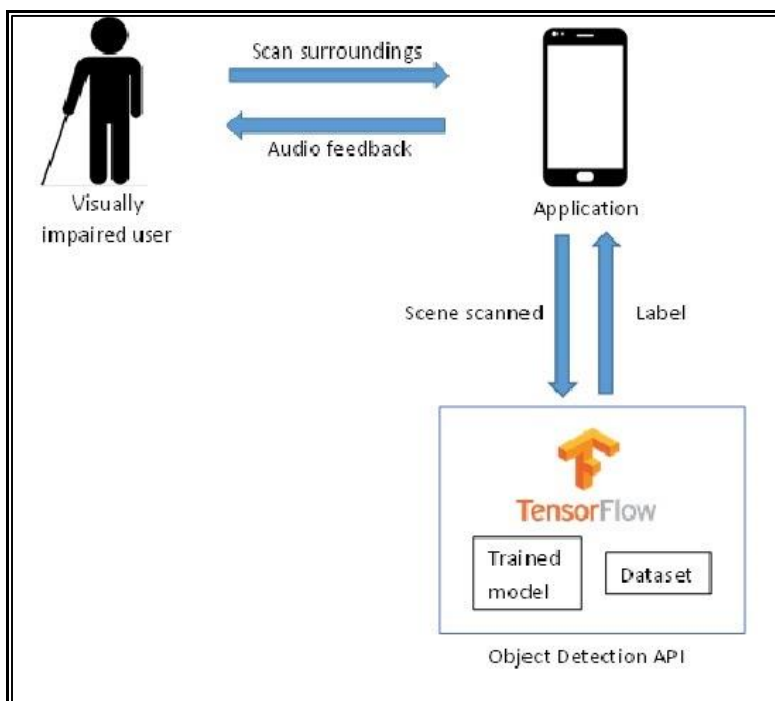


Figure 1.1: Proposed System Overview

In this system, a visually impaired user opens the mobile application and scans his or her surroundings using the smartphone camera. The scene captured through the continuous stream rather than taking a picture of a specific scene. Hence, the visually impaired user does not have difficulty in taking a good quality picture and does not need to capture a picture every time. Furthermore, the scene captured will not be stored in the smartphone memory so that the user does not have to delete the images from time to time.

The scene scanned is then sent to the Tensorflow Object Detection API (Application Programming Interface) for detecting and recognising objects in the scene. Due to time limitations, an API is utilised in developing the application since it provides a list of operations so that a developer does not need to write code from scratch. The Tensorflow Object Detection API is an open source machine learning framework. It has been developed by Google Brain Team in 2015. It prepares a collection of pre-trained object detection and recognition models for a developer to deploy directly into the application and a developer also can choose to train his or her own model using this framework.

Other than Tensorflow Object Detection API, there are other types of object detection API available, such as Google Cloud Vision API, Microsoft

Cognitive Toolkit, and Pytorch. Although Google Cloud Vision API provides more features such as colour recognition, landmark recognition, handwritten text recognition, and others, it is free for the first 1000 units per month only for each feature. Both Microsoft Cognitive Toolkit and Pytorch are open source. However, Microsoft Cognitive Toolkit does not support object detection models for mobile devices (Argawal, 2018). Pytorch provides more pretrained models than Tensorflow, but it has less community compared to Tensorflow so it will be harder to get the tutorials (Lobo, 2017). Moreover, Rane, Patil and Barse (2019) state that using Pytorch, the process is less efficient and is less reliable than Tensorflow. Hence, the main reason for choosing Tensorflow Object Detection API is that it is open source and has the largest community.

Instead of applying the existing pre-trained models into the application directly, transfer learning was applied to retrain the existing pre-trained models to recognise new classes and improve the accuracy of the recognition results of the object detection model. The final object detection model has been trained to detect and recognise the objects in environment context, especially indoor and outdoor obstacles. Its purpose is to make navigation of visually impaired people easier.

The object detection model within the application detects and recognises an object based on its knowledge trained on the dataset. After that, the object detection model will return the label, coordinate, and confidence score of each object detected. As the visually impaired user is difficult to see the name and the direction of the object displayed on the mobile screen, the name and direction are spoken out in audio feedback by using Android text-to-speech API. The audio feedback is provided to the user through smartphone speakers or earphones.

1.6 Final Approach

The methodology applied in the development of this application was the evolutionary prototyping model. The development of the complete system was done by undergoing a series of iterations until an acceptable prototype was built. The first phase of the model was requirements gathering. Next, design, prototyping, and user evaluation phases were performed repeatedly until the

prototype was accepted by users. The design and development of the prototype were incrementally improved based on the users' feedback in every iteration. After the users were satisfied with the prototype, the final prototype was developed into the complete system. Figure 1.2 below illustrates the overview of the evolutionary prototyping model.

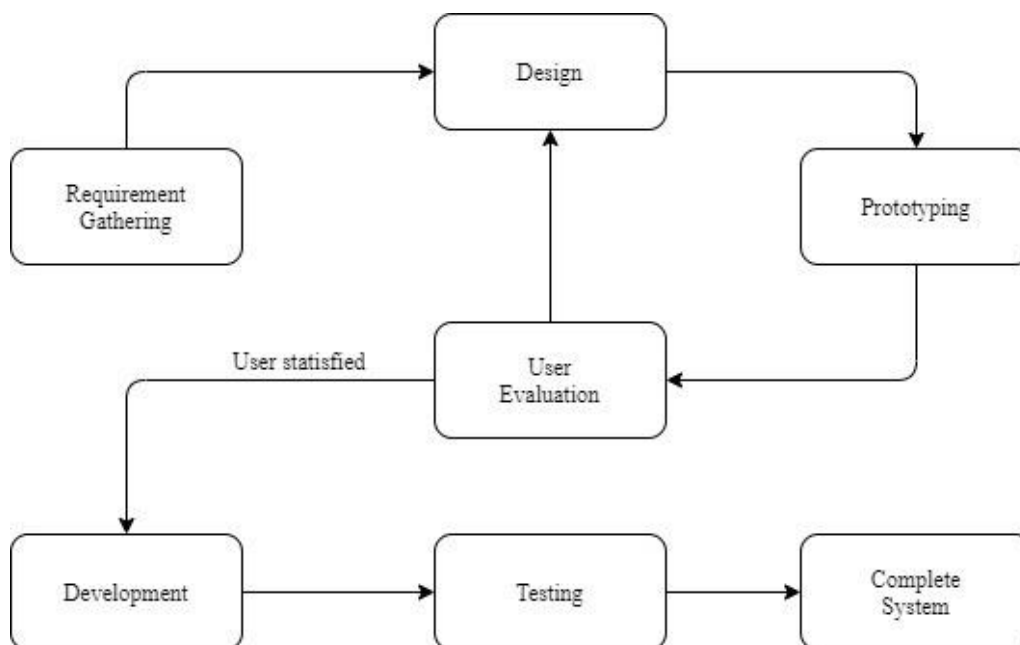


Figure 1.2: Evolutionary Prototyping Model

By applying this methodology, users are involved actively in the software development process. A developer can get feedback from users and find out the limitations, errors, and missing requirements in the prototype early. This is able to reduce the cost of rework since the cost of bugs tends to increase as the project proceeds. Furthermore, the complete system will be more user friendly and able to meet the users' requirements more accurately as the system is improved according to users' opinions.

1.7 Project Scope

The section discusses delimitations, limitations, and assumptions of the project.

1.7.1 Delimitations

This project is aimed to develop an Android-based mobile application that detects and recognizes objects in real-time.

Users can access this application by installing it on their Android smartphone. The main target users of this application are people who have a vision impairment or blindness. By using this application, visually impaired users can scan their surroundings using their smartphone's camera without taking a photo of the objects and saving the photo into the memory. This application is able to recognize at least 90 classes of objects. It will detect, recognize, and locate up to 10 objects within the scene captured by the camera. The location of each object detected is surrounded with a rectangular box (boundary box) and its name is labeled, along with a score representing the confidence of the accuracy of the detection. The detection results of an object are only displayed and informed when the confidence score is above 0.6 to maintain the accuracy of the application. Besides that, audio feedback is provided to inform users about the name of objects detected through headphones or the smartphone's speaker.

1.7.2 Limitations

Certain scope is not covered in this project due to time constraints and wide coverage of the existing scope. The following are the uncovered scope of this project:

i. The development of this application that does not support iOS-based platforms.

Although iOS smartphones have a large population, this project takes the Android-based application as initial prototype development due to limited time. The application on iOS-based platforms will be developed in the future.

ii. Color detection and recognition

This application is not able to recognize the colors of the detected objects due to the wide scope. Colour detection and recognition is useful because it can assist the visually impaired users to determine the colors of the detected objects instead of their name only. Besides, color detection and recognition also assists people with color blindness to identify colors.

iii. Object detection and recognition in the dark

The accuracy of object detection and recognition results in the dark is affected by the camera quality of the smartphone. The application may not work well if the smartphone's camera has a poor performance when shooting in the dark.

1.7.3 Assumptions

- i. Assume that users are not completely blind.
- ii. Assume that users are not deaf-blind people.
- iii. Assume that users have allowed the system to access their smartphone cameras.
- iv. Assume that the smartphone of users has enough battery for them to use this application for navigation.
- v. Assume that the minimum API level of the user's smartphone is 23 (Android 6.0).

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter studies on concepts of object detection and object recognition, state-of-art methods, transfer learning, existing object detection and object recognition mobile application, and API based object detection and object recognition mobile application for visually impaired user.

2.2 Object Detection and Object Recognition Concepts

2.2.1 Object Detection and Object Recognition

For the term “object detection”, researchers have put forward different views. Liu et al. (2019) and Argawal (2018) propose that object detection consists of recognising objects in an image and localising them by drawing bounding boxes around them. Several authors including Badave et al. (2020), Michelucci (2019, p.198), Zhao et al. (2019), and Alsing (2018) add that object detection involves object classification and object localisation. Object classification is a process of identifying and recognising the category of each object into the classes which were defined previously in a given image (Badave et al., 2020; Michelucci, 2019, p.198; Zhao et al., 2019). This process involves labelling the name of the objects detected in the image. Badave et al. (2020), Michelucci (2019, p.198), and Zhao et al. (2019) state that object localisation is the process of labelling the location of each object in the image by drawing a rectangular box around the object, known as a boundary box. However, the bounding box drawn is always rectangular in shape and object localisation does not draw the exact shape of the object.

However, Garrido and Joshi (2018, p.170) argue that object detection only includes localising objects rather than classifying them. They suggest that object detection means detecting whether there are any objects in an image. The object can be anything because the category of the object is not the concern in object detection. This means that in object detection, a computer does not need to know what the object is. When an object is detected, a

boundary box will be drawn around the detected object in the image to notify people that something is detected and indicate the location of the object.

Furthermore, Garrido and Joshi (2018, p.170) also introduce the term “object recognition”. According to Garrido and Joshi (2018, p.170), object recognition refers to the process of determining what the object is in the image based on previous knowledge or experience. Hence, to implement object recognition, a dataset is used to train a model so that the model can identify the category of different objects based on its knowledge. Opposite to object detection, object recognition does not inform the location of the object inside a given image. In addition to outputting the name of the object, Garrido and Joshi (2018, P. 170) also point out advanced object recognition can extract other information related to the object detected such as colour, species, and type.

2.2.2 Intersect over Union

Nonetheless, for localising the object in an image, IoU (Intersect over Union) is applied to identify the amount of overlap between predicted boundary box and the ground-truth (actual object boundary). Its formula is shown below:

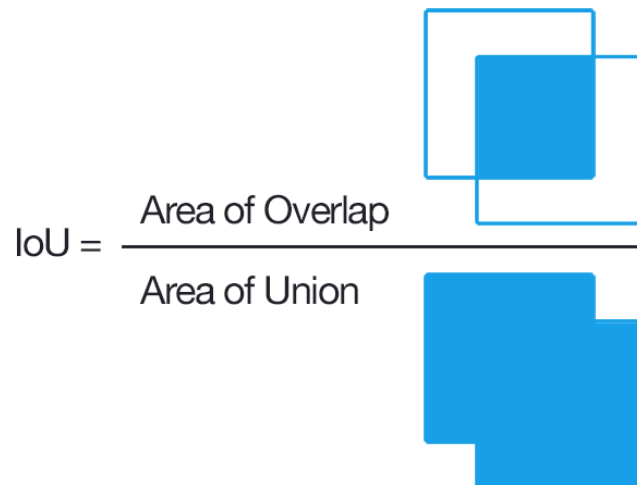
$$IoU = \frac{\text{Area of overlap}}{\text{Area of union}}$$


Figure 2.1: Formula of Intersect over Union (Alsing, 2018)

Michelucci (2019, p.198) notes that IoU is equal to 1 when there is an ideal case of perfect overlap, while IoU is equal to 0 when there is no

overlapping. Alsing (2018) points out that IoU must be equal or greater than 0.5 to consider the detection as true positive. False positive is a condition that there are duplicate boundary boxes or IoU is lesser than 0.5. False negative is returned when an object is not detected at all. The study (Alsing, 2018) shows that the predicted match will only be true positive if it is not used previously to eliminate the duplicate detections of objects.

Furthermore, Alsing (2018) also mentions that boundary boxes and classes of all objects detected are arranged in descending order of probability. A cut-off threshold can be used to discard the results below a specific score to increase the accuracy of the model.

2.3 State-of-Art Methods for Object Detection and Object Recognition

The state-of-art methods for object detection and object recognition can be categorised into two types that are Region Proposal Based Framework and Unified Based Framework (Zhao et al., 2019).

Region Proposal Based Framework is a two stages method. It produces proposed regions from the input image and classifies each proposed region into different classes (Zhao et al., 2019; Liu et al., 2019). The examples of Region Proposal Based Framework are R-CNN, Fast R-CNN, and Faster R-CNN.

However, computations in Region Proposal Based Framework are expensive for mobile devices since the devices have smaller storage and computational capability (Liu et al., 2019). Hence, Unified Based Framework, which is a one-stage method, has been developed. Unified Based Framework predicts boundary box coordinates and class probabilities directly without proposing regions from the input image (Zhao et al., 2019). This can reduce the time needed in object detection and recognition. Two common examples of Unified Based Framework are SSD and YOLO.

2.3.1 Region-based Convolutional Neural Network (R-CNN)

According to Argawal (2018), Zhao et al. (2019) and Liu et al. (2019), R-CNN involves three stages as listed below. Figure 2.2 illustrates the stages in R-CNN.

- i. Around 2000 proposed regions are extracted from an input image by scanning the image by using an algorithm named Selective Search.
- ii. Every proposed region is wrapped and resized into the fit size. At this stage, CNN is used to extract a high-level feature representation of each proposed region.
- iii. Object classification and localisation are performed. Linear SVMs (Support Vector Machines) are utilised to classify the proposed regions and assign the boundary boxes based on the features extracted by CNN. Each proposed region is given a score on the examples of positive and negative regions. Bounding box regressor is applied to the scored regions to generate the boundary box that surrounds the specific object in the image.

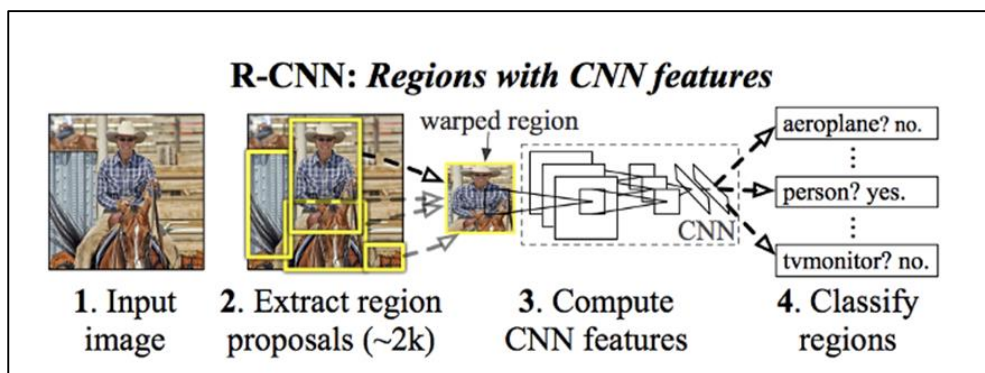


Figure 2.2: Architecture in R-CNN (Argawal, 2018)

However, there are certain limitations of R-CNN. Argawal (2018), Alsing (2018) Zhao et al. (2019) and Liu et al. (2019) agree that training an R-CNN is time-consuming since every stage is required to be trained separately. Besides, Zhao et al. (2019) and Liu et al. (2019) point out that the training process also consumes a large amount of disk space. It is because features extracted from the proposed regions are stored in the disk. Besides, the detection and recognition process of R-CNN is slow. According to Argawal (2018), it takes 47 seconds to detect and recognise objects in one image on a GPU. Hence, this causes R-CNN is not suitable to be applied in the real-time object detection and recognition applications.

2.3.2 Fast Region-based Convolutional Neural Network (Fast R-CNN)

To improve R-CNN, Fast R-CNN is proposed. According to Argawal (2018) and Liu et al. (2019), the processes involved in Fast R-CNN is stated below and illustrated in Figure 2.3:

- i. The image is input to a single CNN that contains several convolutional layers to generate a convolution feature map.
- ii. Same as R-CNN, the image is scanned by Selective Search to generate proposed regions. Each proposed region is then warped and fed into a RoI (Region of Interest) pooling layer.
- iii. A fixed-length feature for every proposed region is extracted by the RoI pooling layer and to be passed into a sequence of CNN fully connected layers.
- iv. The fully connected (FC) layers consist of two branches, which are softmax classifier and bounding box regressor. Softmax classifier identifies the class of the object while bounding box regressor adjusts the bounding boxes.

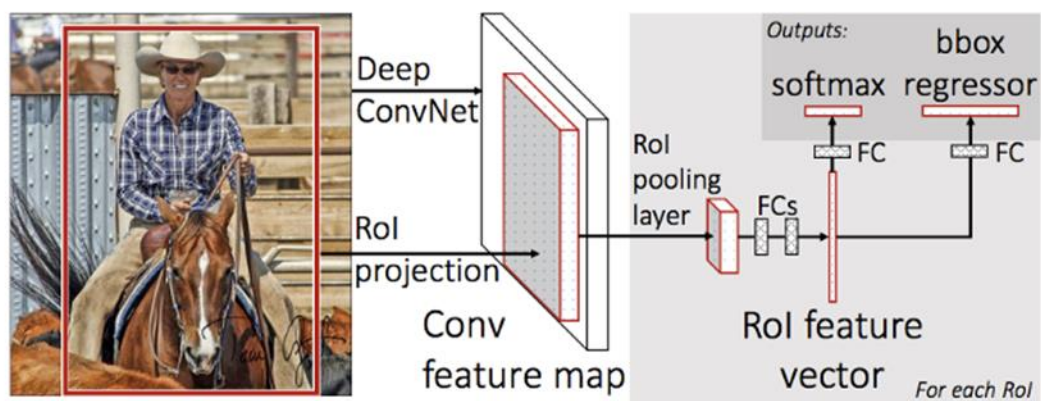


Figure 2.3: Architecture in Fast R-CNN (Argawal, 2018)

Alsing (2018) and Argawal (2018) mentions that Fast R-CNN is nine times faster than R-CNN in speed. Additionally, Alsing (2018) and Liu et al. (2019) also indicate that Fast-RCNN makes the training process more efficient because the classifier and bounding box regression can be trained simultaneously without separating them. Other than that, Argawal (2018) adds that one CNN is needed to be trained to extract the entire image in Fast R-CNN, rather than many CNNs are trained to extract each proposed region in

the image by using R-CNN. However, the Selective Search Algorithm that applied in Fast R-CNN is slow and expensive. It brings negative impacts to the performance of Fast R-CNN.

2.3.3 Faster Region-based Convolutional Neural Network (Faster R-CNN)

Faster R-CNN is an improvement of Fast R-CNN. Many papers (Argawal, 2018; Alsing, 2018; Liu et al., 2019; Zhao et al., 2019) state that Faster R-CNN improves the performance by replacing the Selective Search and Edge Box within Fast R-CNN by Region Proposal Network (RPN). RPN is faster than Selective Search and Edge Box as it shares the convolutional features of entire images with the detection network and this increases the efficiency and the accuracy in generating proposed regions from the input image (Zhao et al., 2019; Alsing, 2018).

According to the studies (Zhao et al., 2019; Liu et al., 2019; Argawal, 2018; Alsing, 2018), as shown in Figure 2.4, there are several processes in Faster R-CNN:

- i. The image is input into CNN to produce proposed regions.
- ii. RPN takes the image to produce k proposed regions that have different aspect ratios for every convolutional feature map location. Each of them is called an anchor.
- iii. Every anchor is fed into the fully connected layers that can be divided into two parts, which are classifier and bounding box regressor. An objectness score and four coordinates of the boundary box is assigned to each anchor.
- iv. The proposed regions are chosen by comparing the objectness score with the threshold. If a region has an objectness score that is higher than the threshold, the region and the convolutional feature map will be chosen and passed to the detector.

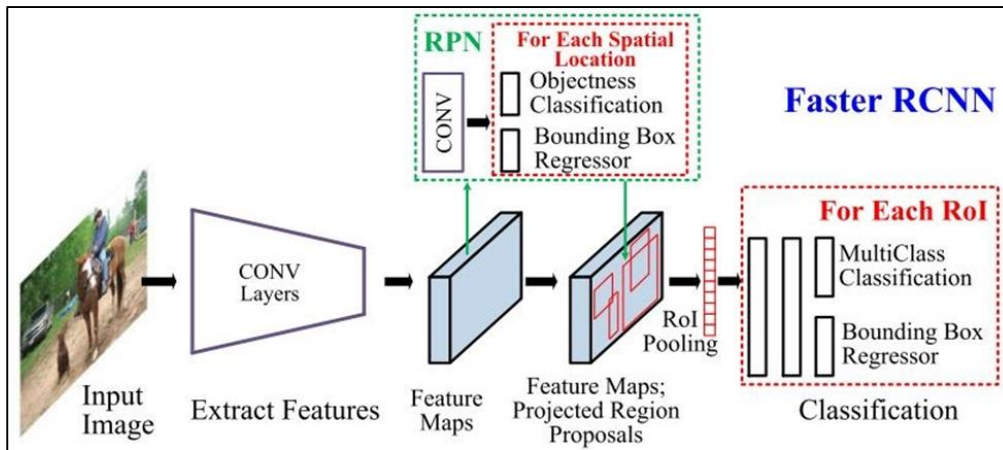


Figure 2.4: Architecture in Faster R-CNN (Liu et al., 2019)

Since RPN shares the convolutional features with Fast R-CNN, the cost of region proposal is almost free (Argawal, 2018; Zhao et al., 2019; Liu et al., 2019). Therefore, the proposal for Faster R-CNN makes real-time object detection and object recognition possible. Nevertheless, Zhao et al. (2019) say that performance bottleneck is still a problem since some time is needed to deal with different components in Faster R-CNN.

2.3.4 You Only Look Once (YOLO)

As YOLO is a method that treats object detection and recognition as a regression problem from image pixels to spatial location of the boundary boxes, it is able to unify classification and localisation tasks (Alsing, 2018; Zhao et al., 2019). Several of papers (Liu et al., 2019; Zhao et al., 2019; Alsing, 2018) explain that YOLO will divide the input image into a grid that contains $S \times S$ cells, as shown in Figure 2.5. Every grid cell is limited to predict only one class of object within the cell. Each of them will predict B boundary box, C class probabilities, and confidence scores. The entire process in YOLO does not involve generating proposed regions from the input. The features from the whole image are used in a global way (Liu et al., 2019).

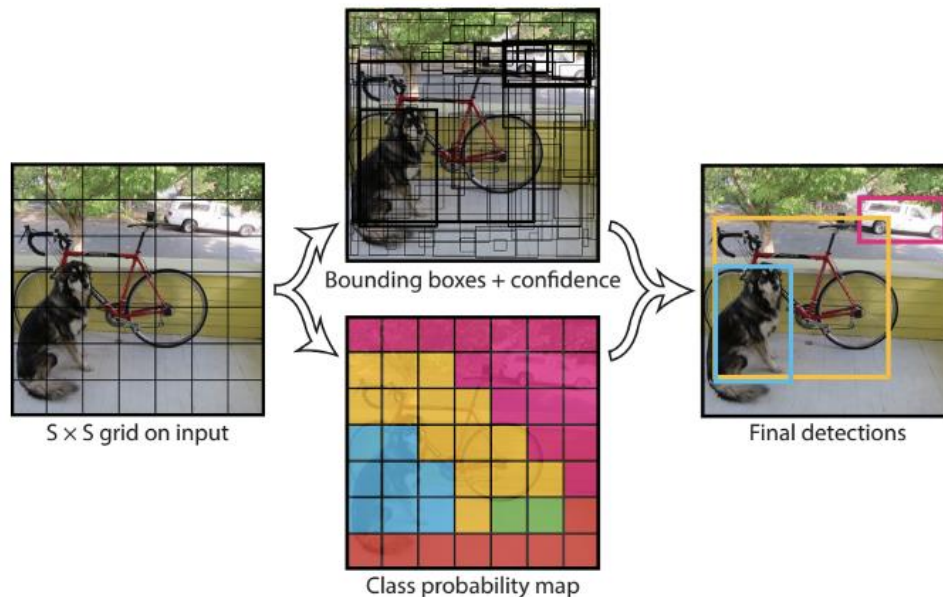


Figure 2.5: Concept of YOLO in Object Detection and Recognition (Zhao et al., 2019)

YOLO performs fast after eliminating the region proposals. It can process images in real-time at 45 Frames per second (FPS) while Fast YOLO at 155 FPS. Moreover, YOLO has a lesser chance of predicting false-positives in the background because it looks at the entire image when processing the image (Liu et al., 2019; Zhao et al., 2019). However, Liu et al. (2019) add that YOLO tends to have more localisation errors than Fast R-CNN due to its coarse division of grid. In addition to coarse division of grid, YOLO faces difficulties in handling small objects since each cell grid is allowed to classify one object only.

2.3.5 Single-Shot Multibox Detector (SSD)

Argawal (2018) defined SSD as a method that can predict the boundary box and the class of objects in a single shot simultaneously. An input image is required to go through all convolutional layers to create convolutional feature maps with different sizes. Convolutional filters are utilised to produce anchors for each feature map and predict class probabilities and coordinates of the boundary box. Instead of using fixed-size grids like YOLO, SSD uses a set of anchors that have different sizes and scales (Zhao et al., 2019). Zhao et al. (2019) state that objects with different sizes can be handled by merging

detection results from multiple feature maps with different resolutions. Since it is a Unified Based Framework, region proposals are not implemented. The architecture in SSD is illustrated in Figure 2.6.

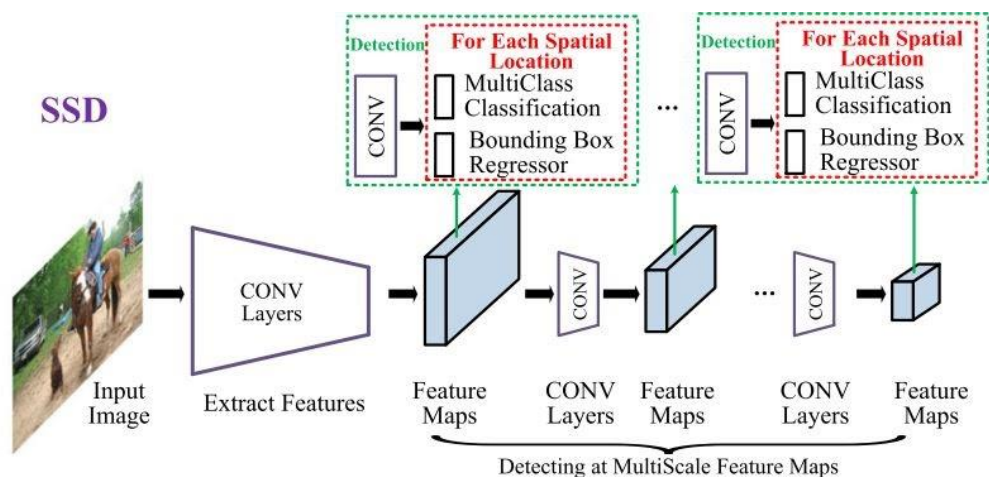


Figure 2.6: Architecture in SSD (Liu et al., 2019)

SSD has an outstanding performance. According to Liu et al. (2019), SSD operates at 59 FPS, so it is faster than Faster R-CNN (7 FPS) and YOLO (45 FPS). Although SSD has excellent speed performance, its accuracy is not degraded severely, but has an excellent performance too. Zhao et al. (2019) and Liu et al. (2019) found that the accuracy of SSD can compete with Region Proposal Framework such as Faster R-CNN. Furthermore, varying scales of feature maps in SSD results in higher detection accuracy than YOLO, especially when detecting and recognising small objects. In conclusion, SSD is a method that allows real-time speed and maintaining high-quality detection at the same time.

2.4 Transfer Learning

2.4.1 Transfer Learning and Traditional Machine Learning

Transfer learning is a famous method used in machine learning nowadays. Transfer learning allows the knowledge to be transferred from the previously trained model to a new model. Hence, the new model can be built and trained in a time-saving way and with fewer data. On the other hand, traditional machine learning learns the task from scratch. Figure 2.7 illustrates the difference between the learning process of traditional machine learning and transfer learning.

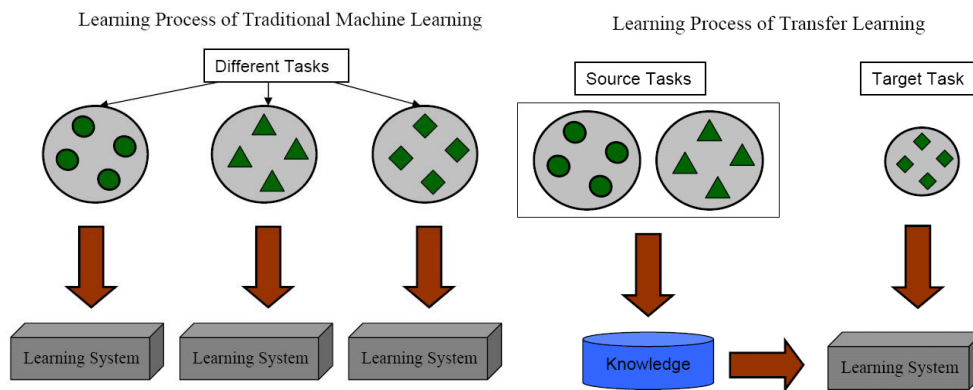


Figure 2.7: Traditional Learning and Transfer Learning (Pan and Yang, 2009)

Sarkar (2018) mentions that traditional transfer learning is isolated. No knowledge is transferred from a model to another model. According to Weiss, Khoshgoftaar, and Wang (2016), in traditional machine learning, training data and testing data are assumed to be chosen from the same domain. Hence, the characteristics of feature space and data distribution should be the same. When there are differences between data distribution of training data and testing data, the performance of the model will be degraded. For example, we are going to train a model that detects objects in the kitchen. If the training dataset is related to object detection at the kitchen, traditional machine learning allows the target model to achieve good detection results. Nevertheless, if the training dataset is obtained from object detection at the park, the differences in domain data will cause the detection results to be degraded. However, Weiss, Khoshgoftaar, and Wang (2016) mention that in real-world scenarios, that are cases where it is hard to collect the training data that corresponds to the feature space and data distribution of testing data.

Thus, transfer learning is needed to improve a learner from the target domain by transferring the knowledge from a related domain (Weiss, Khoshgoftaar, and Wang, 2016). In the example regarding object detection at the kitchen, object detection at the kitchen and the park still have some common characteristics. Both of them detects objects, so certain low-level features such as edges, intensity, corners, and shapes can be shared among them (Sarkar, 2018). Furthermore, several papers (Argawal, 2018; Patil and Gaikwad, 2018; Sarkar, 2018; Weiss, Khoshgoftaar, and Wang, 2016) mention

that transfer learning requires less target training data compared to traditional machine learning. Moreover, Argawal (2018) and Zhang, Yang and Sinnott (2019) point out that the benefits of performing transfer learning on a pre-trained model are to shorten the training time and improve the performance of the model.

2.4.2 Formal Definitions of Transfer Learning

Pan and Yang (2009) introduce some notations and definitions to explain transfer learning. This section will list and discuss the notations and definitions used for the rest of the paper.

According to Pan and Yang (2009) and Weiss, Khoshgoftaar, and Wang, (2016), a domain, D , contains two parts. The first part is a feature space \mathcal{X} while the second part is a marginal probability distribution $P(X)$, where $X = \{x_1, \dots, x_n\} \in \mathcal{X}$. According to the example provided by Weiss, Khoshgoftaar, and Wang (2016), if the machine learning task is software module defect classification and every software metric is treated as a feature, and then \mathcal{X} is the space of all possible feature vectors, x_i is i^{th} feature vector corresponding to some software modules, while X is a learning sample.

Given a particular domain, a task, T , consists of two elements. One of the element is a label space Y while another element is a predictive function $f(\cdot)$. The function $f(\cdot)$ is learned from the training data that consists of feature and label pairs $\{x_i, y_i\}$ where $x_i \in X$ and $y_i \in Y$ (Sarkar, 2018; Weiss, Khoshgoftaar, and Wang, 2016; Pan and Yang, 2009). The function $f(\cdot)$ is used to predict the corresponding label $f(x)$, of an instance x . Refer to the example of software module defect classification, Y is a set contains all labels, which are True and False, while y_i is the value of True or False.

From the definitions discussed above, Pan and Yang (2009) denote domain by $D = \{\mathcal{X}, P(X)\}$ and denote a task by $T = \{Y, f(\cdot)\}$. Relate back to transfer learning, source domain data is denoted as $D_S = \{(x_{S_1}, y_{S_1}), \dots, (x_{S_n}, y_{S_n})\}$, where $x_{S_i} \in \mathcal{X}_S$ is the data instance of D_S while $y_{S_i} \in Y_S$ is the corresponding class label. Also, Pan and Yang (2006) denote target domain data as $D_T = \{(x_{T_1}, y_{T_1}), \dots, (x_{T_n}, y_{T_n})\}$, which the input $x_{T_i} \in \mathcal{X}_T$ while $y_{T_i} \in Y_T$ is the corresponding class label for x_{T_i} .

Given a source domain D_S with corresponding source task T_S and a target domain D_T with corresponding target task T_T , the goal of transfer learning is to improve the target predictive function $f_T(\cdot)$ in D_T by utilising the knowledge in D_S and T_S , by which $D_S \neq D_T$ or $T_S \neq T_T$ (Sarkar, 2018; Weiss, Khoshgoftaar, and Wang, 2016; Pan and Yang, 2009).

Pan and Yang (2009) and Weiss, Khoshgoftaar, and Wang (2016) discuss that since $D = \{\mathcal{X}, P(X)\}$, $D_S \neq D_T$ represents two conditions, which are illustrated in the following using the software module defect example:

- i. $\mathcal{X}_S \neq \mathcal{X}_T$, where the feature space of the source domain and target domain are different. In the example, the metrics of the source software project are different from that of the target software project.
- ii. $P(X_S) \neq P(X_T)$, where the marginal probability of the source domain and target domain are different. For example, the source software is an application software while the target software is a driver software.

Similarly, Pan and Yang (2009) and Weiss, Khoshgoftaar, and Wang (2016) state that since $T = \{Y, P(Y | X)\}$, $T_S \neq T_T$ implies the two conditions below:

- i. $Y_S \neq Y_T$, where the label spaces between the source task and target task are different. For example, the source software has binary classes to detect module defect, such as True for defect module while False for non-defect module. The target software uses three classes to classify three levels of defect software module.
- ii. $P(Y_S | X_S) \neq P(Y_T | X_T)$, where the conditional probability distributions of source and target domains are different. It is corresponding to the condition where the source software modules and the target software modules are very unbalanced in terms of classes defined.

2.4.3 Transfer Learning Settings

Transfer learning can be classified into three settings, those are transductive transfer learning, inductive transfer learning, and unsupervised transfer learning, according to the nature of the source and target domains and tasks

(Sarkar, 2018; Pan and Yang, 2009). Table 2.1 summarises the different settings of transfer learning.

Table 2.1: Different Settings of Transfer Learning

Transfer Learning Settings	Related Areas	Source Domain Labels	Target Domain Labels	Tasks
Inductive Transfer Learning	Multi-task Learning	Available	Available	Classification, Regression
	Self-taught Learning	Unavailable	Available	Classification, Regression
Transductive Transfer Learning	Sample Selection Bias, Domain Adaptation, Co-variate Shift	Available	Unavailable	Classification, Regression
Unsupervised Transfer Learning		Unavailable	Unavailable	Dimensionality Reduction, Clustering

2.4.3.1 Inductive Transfer Learning

Pan and Yang (2009) propose that in inductive transfer learning, the target task is different from the source task, whether the source and target domains are same or not. Besides, inductive transfer learning requires some labelled data in the target domain to induce an object predictive model $f_T(\cdot)$ for the target domain.

Inductive transfer learning can be further classified into multi-task learning and self-taught learning according to the labelling situations in the source domain. According to Pan and Yang (2009), inductive transfer learning multi-task learning are similar if the source domain has a lot of labelled data.

If labelled data in the source domain are not available, inductive transfer learning will be similar to self-taught learning (Pan and Yang, 2009).

2.4.3.2 Transductive Transfer Learning

The source and target tasks in transductive transfer learning setting are the same, only their domains are different. Within the transductive transfer learning setting, there is a lot of labelled data in the source domain while the target domain does not have available labelled data (Sarkar, 2018; Pan and Yang, 2009). Depends on whether feature spaces among both source and target domains are different or their marginal probabilities are different, transductive transfer learning is able to be further classified into domain adaptation, sample selection bias, and co-variate shift (Sarkar, 2018; Pan and Yang, 2009).

2.4.3.3 Unsupervised Transfer Learning

In unsupervised transfer learning, Pan and Yang (2009) propose that the target task is different to the source task, but both of them are related to each other. Hence, solving unsupervised tasks in the target domain is the main focus of this setting (Sarkar, 2018; Pan and Yang, 2009). Some examples of tasks include dimensionality reduction, clustering, and density estimation (Pan and Yang, 2009). In this setting, labelled data are not available in both source domain and target domain.

2.4.4 Transfer Learning in The Project

The above research found that transfer learning provides many benefits for model training. Due to time constraints and computational resource limitations, the project has applied transfer learning to train object detection models.

2.4.4.1 Pre-trained Models Selected

Tensorflow has provided a collection of open source object detection models pre-trained on the Kitti dataset, COCO dataset, Open Images dataset, and other datasets. Since the problem solved in the COCO dataset is similar to this project, the pre-trained models which were previously trained on COCO dataset was chosen to implement transfer learning. The COCO dataset is a

large and high-quality dataset for computer vision. It includes 80 object classes and provides more than 200,000 labelled images (COCO, 2021).

The pre-trained models used for transfer learning are SSD Mobilenet V1 COCO and Faster R-CNN Inception V2 COCO. The transfer learning performance between these two models was compared.

2.4.4.2 Transfer Learning Settings Applied

In this project, inductive transfer learning was applied. Both source and target models were trained with annotated data. Besides, the models trained in this project and the pre-trained models chosen have different tasks due to their difference in label spaces. Table 2.2 shows the label spaces of both pre-trained models and target models.

Table 2.2: List of Object Classes for Pre-trained Models and Target Models

	Pre-trained Model		Target Model
Common Objects Classes	<ul style="list-style-type: none"> • Person • Bicycle • Car • Motorcycle • Bus • Train • Truck • Bench • Cat • Dog 	<ul style="list-style-type: none"> • Umbrella • Ball • Skateboard • Bottle • Knife • Chair • Couch • Potted plant • Bed • Table 	<ul style="list-style-type: none"> • Toilet • Computer • Mouse • Cell phone • Sink • Refrigerator • Clock • Vase • Scissors
Different Objects Classes	<ul style="list-style-type: none"> • Book • Hair drier • Toothbrush • Television • Remote • Microwave • Oven • Teddy bear • Toaster • Keyboard • Airplane • Boat • Traffic light 	<ul style="list-style-type: none"> • Tie • Suitcase • Frisbee • Skis • Snowboard • Kite • Baseball bat • Baseball glove • Surfboard • Tennis racket • Wine glass • Cup 	<ul style="list-style-type: none"> • Window • Door • Plush toy • Bin • Tree • Bag • Street sign • Fan • Street lamp • Shelf • Staircase

<ul style="list-style-type: none"> • Fire hydrant • Stop sign • Parking meter • Bird • Horse • Sheep • Cow • Elephant • Bear • Zebra • Giraffe • Backpack • Handbag 	<ul style="list-style-type: none"> • Fork • Spoon • Bowl • Banana • Apple • Sandwich • Orange • Broccoli • Carrot • Hotdog • Pizza • Donut • Cake 	
--	--	--

According to Table 2.2, the target models detect and recognise certain object classes that are not included in the COCO dataset, such as bin, window, and tree. However, they also detect and recognise certain common classes, such as mouse, table, train, bus, and truck. Differences in object classes recognised cause $Y_S \neq Y_T$ and then $T_S \neq T_D$.

Moreover, the target domains are different but related to the domains of the pre-trained models. The target models are designed to detect and recognise objects in the environment context, especially indoor and outdoor obstacles. However, the COCO dataset focuses more on the general context. Hence, the target domain can be considered as a subdomain of the COCO dataset domain.

2.4.4.3 Transfer Learning Strategies Applied

This project reused the feature extractor parameters from the existing object detection checkpoint of the pre-trained models and used these features to detect and recognise the custom classes. According to Alsing (2018), Gao and Mosalam (2018), Marcelino (2018), and Sarkar (2018), as shown in Figure 2.8, this process is known as off-the-shelf feature extraction and it involves a series of steps below.

- i. Remove the last fully connected layer in the base CNN, which provides the final class label for the pre-trained model.

- ii. Freeze the convolutional base in the pre-trained model.
- iii. The pre-trained model is then treated as a fixed feature extractor. Use these features to train a new classifier that detects and recognises the target classes.

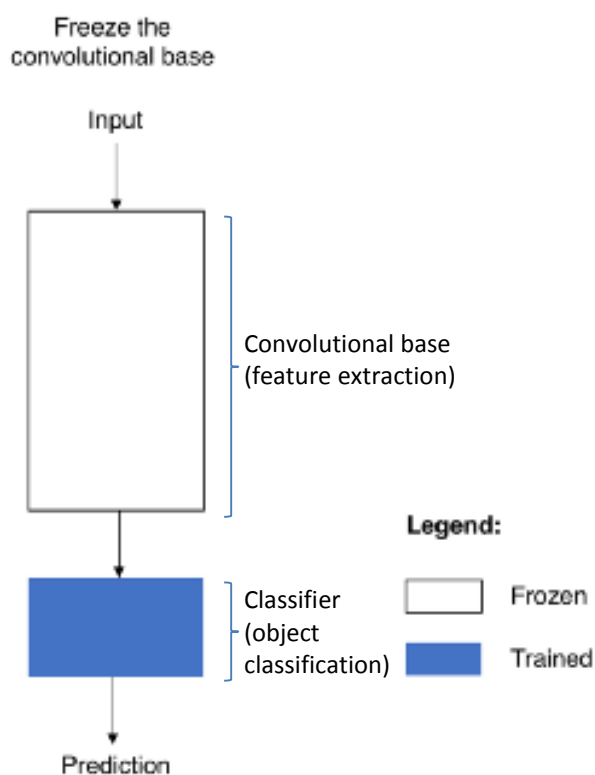


Figure 2.8: Off-The-Shelf Feature Extraction

This strategy is useful when the computational power is limited and the target dataset is small (Marcelino, 2018; Gao and Mosalam, 2018). Other than that, Gao and Mosalam (2018) mention that this strategy is able to reduce the training time greatly.

2.5 Existing Object Detection and Object Recognition Mobile Application

With the popularity of smartphones and the invention of various object detection have brought forth the launching of various mobile applications. Each of them has different purposes and usages. They tend to detect and recognise objects in different contexts. The applications take the scene captured by the smartphone camera as input and display the results in text and

boundary boxes. Table 2.3 below shows some papers of mobile applications that apply object detection and object recognition techniques.

Table 2.3: Papers of Mobile Applications that Apply Object Detection and Object Recognition Techniques

Paper	Target User	Detection and Recognition Category	Purpose
Object Detection in Refrigerators using Tensorflow (Argawal, 2018)	Everyone	Grocery food	Recognise objects kept in a refrigerator and tell the user whether the object could be present in a refrigerator
FoodTracker: A Real-time Food Detection Mobile Application by Deep Convolutional Neural Networks (Sun, Radecka, and Zilic, 2019)	Everyone who cares about health	Food	Recognise food and return nutrition facts to the user
Object Detection and Its Implementation on Android Devices (Li and Zhang, 2017)	Road user	Person and car	Detect and recognise objects on the road such as car, pedestrian, and cyclist
Classification of Vegetables using TensorFlow (Patil and Gaikwad, 2018)	Supermarket cashier	Vegetable	To help cashier staff in classifying the species of vegetables purchased by customers

TensorFlow: A Vegetable Classification System and Its Performance Evaluation (Ruedeeniramana, Ikeda and Barolli, 2017)	Farmer	Vegetable	To help farmers in classifying the species of vegetable
A Mobile Application for Cat Detection and Breed Recognition Based on Deep Learning (Zhang, Yang and Sinnott, 2019)	Everyone	Cat	Classify different species of cats

Although the pre-trained model is sufficient to classify different classes of objects, a number of authors (Argawal, 2018; Patil and Gaikwad, 2018; Zhang, Yang and Sinnott, 2019) justify that they retrain the model with the dataset of the targeted context to improve the accuracy of the object recognition in that context. For instance, Zhang, Yang and Sinnott (2019) have trained the pre-trained model with an additional 14 classes of cat species. The authors (Argawal, 2018; Patil and Gaikwad, 2018; Zhang, Yang and Sinnott, 2019) claim that the method to fine-tune the model is called transfer learning.

Li and Zhang (2017) found that the accuracy of their application degrade when the scene becomes darker or blurry. For example, the scene becomes blurry when it is rainy or foggy, while the scene gets darker on a cloudy day or at night. Besides, Zhang, Yang and Sinnott (2019) also test their application in complex scenarios. The scenarios include one cat, parts of cats, multiple cats, and many objects not related to cats appear in the scene. Moreover, some papers (Sun, Radecka and Zilic, 2019; Li and Zhang, 2017; Ruedeeniramana, Ikeda, and Barolli, 2017; Zhang, Yang and Sinnott, 2019)

found that the uniqueness of the characteristics of the object can increase the accuracy of the recognition results. However, small variations among different objects and varying instances of a specific type of an object such as colours, sizes, and shapes will decrease the quality of the results too. All of these factors should be considered when developing an object detection and recognition application.

2.6 API Based Object Detection and Object Recognition Mobile Application for Visually Impaired User

In addition to object detection and object recognition mobile applications invented for people with normal vision, some researchers have not forgotten the visually impaired community. They have developed object detection and object recognition mobile applications to help visually impaired people to identify objects in their surroundings. These applications are able to give a lot of benefits to the visually impaired people since it does not need any other device, easy to access, and cost-effective (Anitha, Subalaxmi and Vijayalakshmi, 2019; Awad et al., 2018; Rajwani et al., 2018).

Many related papers (Badave et al., 2020; Vaidya et al., 2020; Khan Shishir et al., 2019; Anitha, Subalaxmi and Vijayalakshmi, 2019; Jakhete et al., 2019; Felix, Kumar, and Veeramuthu, 2018; Awad et al., 2018; Rajwani et al., 2018) state that their applications scan or capture images of the objects to make object detection and recognition and use a text-to-speech engine to provide voice feedback to the visually impaired users through smartphone's speaker or earphones.

2.6.1 Mode

Some of these applications (Vaidya et al., 2020; Awad et al., 2018; Rajwani et al., 2018; Parikh, Shah and Vahora, 2018; Felix, Kumar, and Veeramuthu, 2018) require the users to capture the image of the objects every time they want to detect and recognise the objects. The image captured is then sent to a deep learning model in the cloud or the server through the Internet for image processing. Nevertheless, Khan Shishir et al. (2019) argue that it may be difficult for visually impaired people to take a proper image since they are not able to see. Furthermore, Khan Shishir et al. (2019) also add that the images

captured will be stored in the smartphone memory, so the visually impaired users require to delete the images frequently.

Therefore, in order to solve the problems mentioned above, some papers (Badave et al., 2020; Khan Shishir et al., 2019; Anitha, Subalaxmi and Vijayalakshmi, 2019; Jakhete et al., 2019) propose mobile applications that do not require the visually impaired people to take pictures of their surroundings. The visually impaired people just need to scan their surroundings using their smartphone's camera without taking a picture of the objects. These applications will detect and recognise the objects in a frame using real-time object detection (Badave et al., 2020; Khan Shishir et al., 2019; Anitha, Subalaxmi and Vijayalakshmi, 2019). According to Anitha, Subalaxmi and Vijayalakshmi (2019), this approach allows the visually impaired users to identify objects without pressing the capture image button every time. Moreover, Khan Shishir et al. (2019) state that visually impaired users do not need to worry about taking a high-quality picture.

2.6.2 Application Programming Interface (API)

There are several object detection API that can be used to implement object detection and object recognition in mobile applications. Tensorflow Object Detection API is utilised by the majority of authors (Badave et al., 2020; Khan Shishir et al., 2019; Anitha, Subalaxmi and Vijayalakshmi, 2019; Jakhete et al., 2019) to develop their applications. A few authors (Felix, Kumar, and Veeramuthu, 2018; Rajwani et al., 2018) use Google Cloud Vision API while some authors (Vaidya et al., 2020; Awad et al., 2018) do not use any APIs in the implementation of object detection and object recognition. For those who do not use API, they generally use many libraries such as OpenCV, Tensorflow, CNNdroid, or other libraries for image processing. The accuracy of each API is stated in Table 2.4.

Table 2.4: Comparison between Recognition Accuracy of Different APIs

Paper	API used	Accuracy
Eye Assistant : Using mobile application to help the visually impaired (Khan Shishir et al., 2019)	Tensorflow Object Detection API	More than 80%
Android Based Object Detection System for Visually Impaired (Badave et al., 2020)	Tensorflow Object Detection API	Around 87%
A Smart Personal AI Assistant for Visually Impaired People (Felix, Kumar and Veeramuthu, 2018)	Google Cloud Vision API	Around 85%
Real-Time Object Detection for Visually Challenged People (Vaidya et al., 2020)	None	More than 75%

According to Table 2.4, the object detection and recognition application that does not use API have the lowest accuracy. The applications that employ Tensorflow's Object Detection API and Google Cloud Vision API have a similarity in their accuracy. Thus, performance of the APIs are better than the proprietary algorithms in terms of recognition accuracy.

The papers (Felix, Kumar, and Veeramuthu, 2018; Rajwani et al., 2018) that apply Google Vision API in their application state that the image captured is uploaded to Google Cloud through the network connection. Google Cloud Vision API will then analyse the image by comparing the features of the input image with the images in the dataset to obtain the label. After the analysis of the image is done, the label is sent back to the application. Hence, the disadvantage of utilising Google Cloud Vision API is it requires a network connection for image processing. Moreover, the applications also need the users to capture an image of the object every time. Furthermore, privacy issues arise when the image is uploaded to the Google Cloud. Besides, another downside of Google Cloud Vision API is it is not open-source. It is just free for the first 1000 units per month for each service. However, both papers (Felix, Kumar, and Veeramuthu, 2018; Rajwani et al., 2018) mention that

Google Cloud Vision API provides a larger variety of services, including face detection, logo detection, landmark detection, and image properties detection. Google Cloud Vision API is able to classify more classes of objects.

According to Khan Shishir et al. (2019), in the applications that use Tensorflow Object Detection API, the image captured will be sent to the trained object detection model. The model will then extract the features in the image and perform object recognition by assigning labels. The whole process does not require any network connection or remote server and it is happening in real-time (Khan Shishir et al., 2019). Another strengths of Tensorflow Object Detection API are it is open-source and it prepares a collection of pre-trained object detection and recognition models for a developer to adopt the model directly into the application. Pre-trained models that employ different kind of state-of-art methods are available, such as SSD and Faster R-CNN. However, according to Badave et al. (2020), Tensorflow Object Detection API is required to be retained to recognise various dishes, colours, and landmarks. Both Google Cloud Vision API and Tensorflow Object Detection API have their strengths and weaknesses. Table 2.5 summarises the comparison between Google Cloud Vision API and Tensorflow Object Detection API.

Table 2.5: Comparison between Google Cloud Vision API and Tensorflow Object Detection API

API	Google Cloud Vision API	Tensorflow Object Detection API
Open-source	No	Yes
Features	Object detection, face detection, logo detection, landmark detection, and image properties detection	Object detection
Internet connection	Required	Does not required
Pre-trained models available	Less	More
Community	Smaller	Larger

2.6.3 Conclusion

From the study of existing object detection and object recognition mobile application for visually impaired user, the application can be developed into two ways. The first way is the application requires the users to capture photo of the object for recognition. The second way is the application scans and recognises the users' surroundings automatically without capturing images. Many researchers found that the second way is better in helping the visually impaired people. However, existing applications that use Google Cloud Vision API are implemented in the first way since the users need to capture an image for uploading to Google Cloud for image processing. This also causes the application cannot recognize objects when there is no Internet connection. On the other hand, Tensorflow Object Detection API allows the application to be implemented in the first way or second way. Most of the existing applications that implement Tensorflow Object Detection API do not require Internet connection to run. In this project, the second way was applied together with Tensorflow Object Detection API to improve the usability of the application.

CHAPTER 3

PROJECT METHODOLOGY AND PLANNING

3.1 Introduction

The chapter discusses the software development methodology, work breakdown structure, Gantt chart, and development tools applied in this project.

3.2 Software Development Methodology

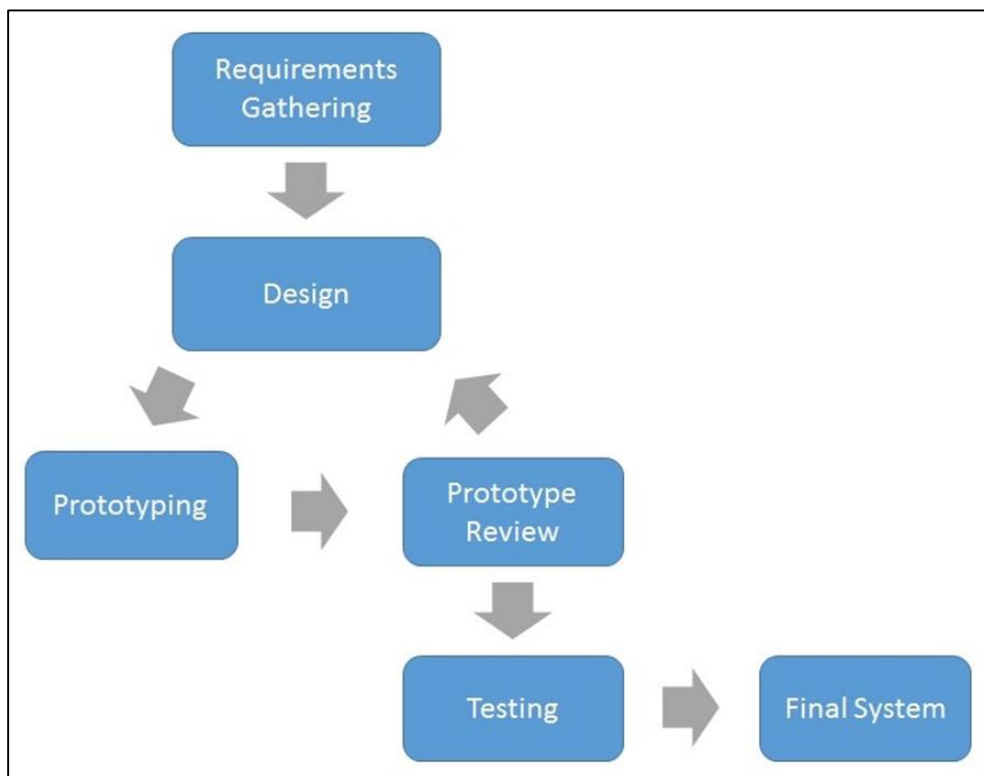


Figure 3.1: Evolutionary Prototyping Model

This project adopted the Evolutionary Prototyping Model as the development methodology. Evolutionary Prototyping Model involves developing the application incrementally until the final system meeting the entire scope is delivered.

3.2.1 Requirements Gathering

During this phase, the requirements of this project were collected through conducting research. The focus areas of the study included machine learning

framework applied and functionalities provided by some existing object detection and object recognition mobile applications. The requirement gathering was carried out via a literature review.

The literature review was conducted on several existing object detection and recognition mobile applications designed for visually impaired people. The purpose of the literature review was to understand the machine learning framework or API used by similar existing applications. The speed and accuracy of the frameworks or APIs were studied to find out the suitable tool that provides quality assistance to the visually impaired people.

Furthermore, another goal of the literature review was to isolate out the important functionalities provided by those similar existing applications. From the research, the standard workflow and features available in those related applications could be identified. Some strengths and weaknesses of the applications also could be found and reviewed to improve this project.

3.2.2 Iteration Process

This project has three iterations of prototype development. The application would be refined and improved after each iteration. In the first iteration, real-time object detection and recognition was the feature to be developed. The application will detect and recognise the objects scanned by the smartphone's camera and return the name and confidence score of the object in textual form. Boundary boxes are designed to label the location for each detected object. The second iteration involved training multiple models to recognise 40 classes of objects using transfer learning and select the most suitable model for real-time implementation. In the last iteration, it involved the development of the feature that provides voice feedback to inform users the names of objects detected. In this iteration, the features that allow user to stop and play the voice feedback and adjust speech rate of voice feedback were developed. There are three phases in each iteration, which are design, prototyping, and prototype review.

3.2.2.1 Design

For the design phase, the system architecture, workflow and interface of the application was designed. The design of the application should follow the requirements and feedbacks collected. This phase was repeated after each iteration of prototyping review.

3.2.2.2 Prototyping

The prototype system was developed based on the design done in the previous phase. The prototype was responsible for presenting the user interface and features in the application. It demonstrated the ways of application works and how it looks like.

3.2.2.3 Prototype Review

After the prototype had been developed, users would test and evaluate the prototype. Their feedback and comments were recorded and reviewed. These feedbacks and comments were essential to this project for developing a better prototype in the next iteration and final system. The prototype was then refined based on the feedback and comments collected from users and continued with the next iteration.

3.2.3 Testing

During the testing phase, different types of testing had been performed on the application to ensure that the final system meet the desired quality and requirements. Bugs found in the testing were fixed. The types of testing that used in this project are unit testing, integration testing, and usability testing. After all tests passed, the application was ready for deployment.

3.3 Project Plan

Project plan was developed to manage the schedule of the project. This aided in ensuring that a proper schedule is adhered to and the project can be completed by the deadline. Work Breakdown Structure and Gantt chart were utilized in project planning.

3.3.1 Work Breakdown Structure

A work breakdown structure breaks the scope of the project into smaller activities. The work breakdown structure of this project is attached as “Appendix A: Work Breakdown Structure”.

3.3.2 Gantt Chart

Gantt chart is used to display project schedule information by listing the project activities in the work breakdown structure and their start and finish dates. The Gantt chart of this project refers to “Appendix B: Gantt Chart”.

3.4 Development Tools

3.4.1 Programming Languages

3.4.1.1 Java

Java is one of the official languages for Android applications development. Besides, addition to the Android platform, Java can run on several platforms such as Windows, Mac OS, and Unix. Object-oriented properties of Java allow extension of the application simply and effectively.

3.4.1.2 Extensible Markup Language (XML)

XML is a markup language like HTML. In Android, XML is often used in designing the layout of the application. It is easy to be understood by humans and computers and compatible with Java completely (IBM, 2020). The structure of XML makes it easier to do modifications.

3.4.1.3 Python

Python is an interpreted programming language. It is famous because its syntax is similar to English, which makes it easier to read and understand. Besides, its syntax is clear and less complex (W3 Schools, 2021). In this project, Python was used in machine learning to train an object detection model.

3.4.2 Framework

3.4.2.1 Tensorflow Lite

Tensorflow Lite is an open-source framework for deep learning. It is a machine learning model explicitly designed for mobile devices to meet the limitations of mobile phone processing and memory resources. Tensorflow Lite already provided a collection of pre-trained models that solve different types of machine learning problems. In this project, Tensorflow Lite was used for object detection and recognition.

3.4.2.2 Camera 2 API

Camera 2 API provides a framework for accessing and configuring cameras on the devices. It allows the user to capture images and videos in the application developed. Other than capturing images and videos, a developer can customise the camera feature through this API. In this application, Camera 2 API was applied to capture the real-time scene.

3.4.3 Tools and Integrated Development Environment (IDE)

3.4.3.1 Android Studio

Android Studio is an official IDE for development of Android application. It contains a variety of features that helps to increase development speed and improve application quality. These features include emulator, testing tools, drag and drop interfaces editor, and real-time CPU and memory statistics (Developers, 2020).

3.4.3.2 Google Colaboratory

Google Colaboratory is a free cloud environment built based on Jupyter Notebook. It allows Python code to be executed in a web browser. Most importantly, it provides free access to computing resources such as GPU and TPU for users to train machine learning models. It also offers commands to access data in Google Drive. The Google Colaboratory notebooks are stored in Google Drive in Jupyter Notebook format (.ipynb).

A test was carried to compare the Tensorflow training performance in various environments, which were local machine CPU and Google

Colaboratory CPU, GPU, and TPU. GPU offered by Google Colaboratory is 12GB NVIDIA Tesla K80 GPU. GPU on the local machine was not tested because it did not meet the minimum CUDA architecture requirements supported by Tensorflow, that was CUDA architecture must be higher than 3.5. Table 3.1 displays the system information of the local machine.

Table 3.1: System Information of Tested Local Machine

Operating System	Windows 7 Home Premium
System Type	64-bit operating system
Processor	Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz
RAM	8GB

Table 3.2 summarises the training performance in different environments. The model to be retrained in this testing is the SSD Mobilenet V1 COCO pre-trained model. According to the results, the training speed on Google Colaboratory CPU is twice as fast as the training speed on the local machine CPU. On the other hand, the training speed on Google Colaboratory GPU and TPU are several times faster than the training speed on CPU of both local machine and Google Colaboratory. Therefore, in order to train the object detection model more efficiently, free GPU and TPU on Google Colaboratory were utilised in this project.

Table 3.2: Comparison of Training Performance

Environment	Average time needed for per step (seconds)
CPU on Local Machine	23
CPU on Google Colaboratory	12
GPU on Google Colaboratory	0.5
TPU on Google Colaboratory	1.3

3.4.3.3 LabelImg

LabelImg is an open source image annotation tool written in Python. It provides users with an easy way to label boundary boxes of various object

classes in images. Labelling allows annotations to be saved as XML files, either in PASCAL VOC format or YOLO format.

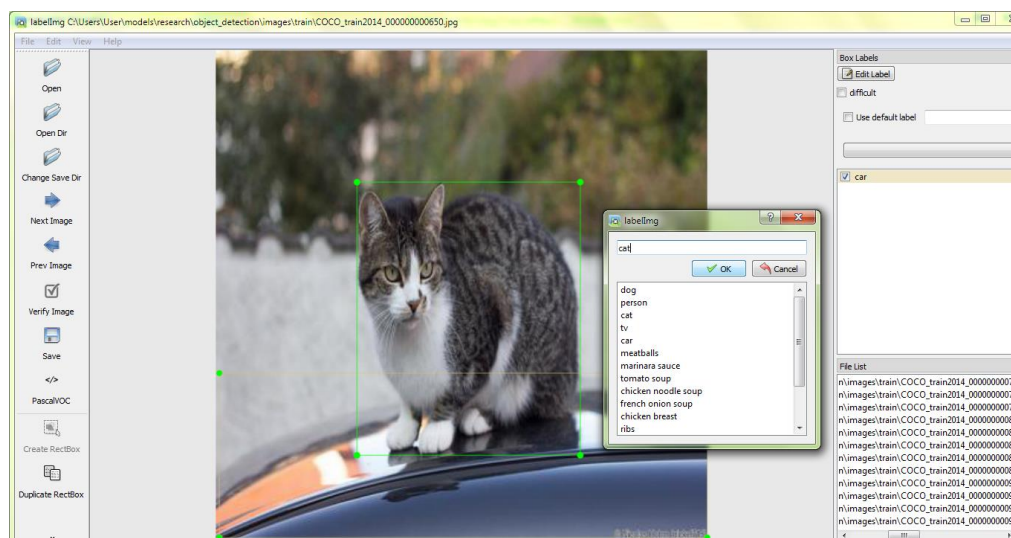


Figure 3.2: Interface of Labelling Annotation Tool

3.4.3.4 Tensorboard

Tensorboard is a built-in visualisation tool of Tensorflow. It visualises the information in the event files output by Tensorflow program. This assists users to track and visualise experiment metrics such as accuracy and loss and also visualise model graph.

3.4.3.5 Anaconda Navigator

Anaconda Navigator is available for Windows, macOS, and Linux. It is a desktop GUI covered in the Anaconda distribution. It allows users to launch applications and manage various conda packages and environments more easily without using command-line commands (Anaconda, 2021).

3.4.4 Version Control System

3.4.4.1 Github

Github is a version control system that manages multiple versions of the source code and other documentation through recording every commit to a file or a set of files. It enables developers to view project history and restore the

required files to a specific previous version quickly. This means that if you break the code or lose the file, you can recover it easily with minimal overhead.

CHAPTER 4

PROJECT SPECIFICATION

4.1 Introduction

This chapter discusses the requirements discovery, requirements specification, use case modelling, and preliminary user interface design.

4.2 Requirements Discovery

In fact-finding, comparison between similar existing object recognition mobile applications for visually impaired people was performed to find out the basic requirements. A total amount of seven similar existing applications proposed by researchers were studied to identify the requirements and features. All of these applications capture surroundings through the smartphone's back camera. The final output is then provided in audio form through the smartphones' speaker or earphones.

4.2.1 Features of Similar Existing Mobile Applications

4.2.1.1 Eye Assistant: Using Mobile Application to Help the Visually Impaired (Khan Shirshir et al., 2019)

Eye Assistant is an Android application proposed by Khan Shishir, Rashid Fahim, Habib, and Farah in 2019 in order to provide assistance to visually impaired people. The main features of the mobile application are recognising objects and recognising text in real-time environments. All features in the application are accessible without requiring any Internet connection.

To use the application, the user first opens the application and two options are displayed on the screen. As shown in Figure 4.1, the first option is "Detect Text" and the second option is "Detect Object". A user can choose whether to read text or recognise objects. If the user chooses the "Detect Text" option, the camera will open and the application will read out the text detected to the user. If the user chooses the "Detect Object" option, the smartphone's camera will also open and the application recognises the object detected in the scene then informs the user the name of each object. Besides, the name of the

object detected and its confidence score are displayed on the screen. The accuracy of the application in recognising every object is larger than 80%.

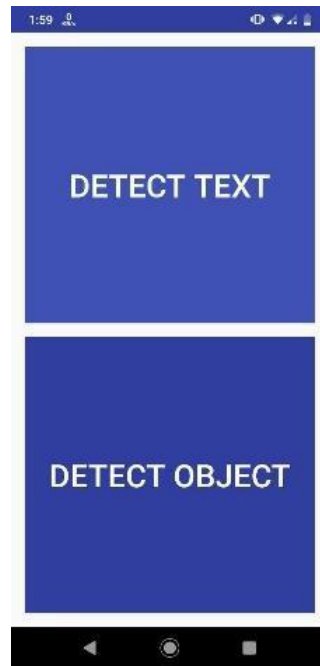


Figure 4.1: Options Available to Users on the Intelligence Eye Home Screen

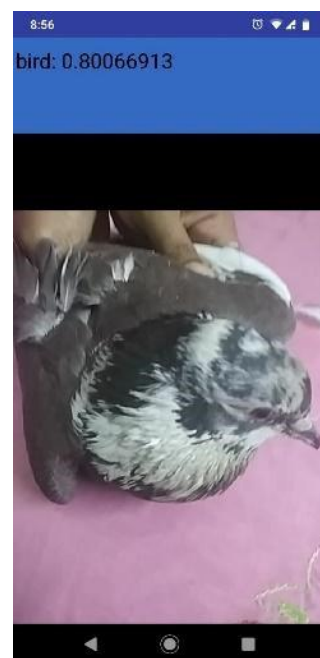


Figure 4.2: Screenshot of Real-Time Object Recognition in Intelligence Eye

4.2.1.2 Android Based Object Detection System for Visually Impaired (Badave et al., 2020)

The paper has been proposed by Badave, Jagtap, Kaovasia, Rahatwad, and Kulkarni in 2020. The name of the application is not mentioned in the paper. The application is developed in Android platforms. The main feature of the application is to locate and recognise objects in surroundings.

The application opens the camera and scans objects immediately without requiring the user to input any additional commands or options once it is opened. The application will recognise the objects, calculate their distance from the user, and determine the direction of the objects. Three directions are used in the application, that are left, right, and center. After scene processing is done, the application will tell the user about the name and direction of each object and the distance between the user and the object. For instance, the audio feedback is “A cat is at 2 metres at the right”. As shown in Figure 4.3, on the mobile screen, the name of the object and the confidence score of the recognition results are shown. The boundary box is used to indicate the location of the object in the scene.

Moreover, if there are multiple objects in the scene, the application does not speak out all objects that appear in the scene. It will assign priorities to different detected objects and speak out the object with the highest priority only. The object that causes potential danger to the user such as a car, is given the highest priority. Furthermore, to improve the application, the recognition results that are below than 70% will be discarded, meaning that the objects will not be informed to the user. Besides, the application can recognise objects with 87% accuracy. There are three languages available in the application, that are English, Hindi, and Marathi.

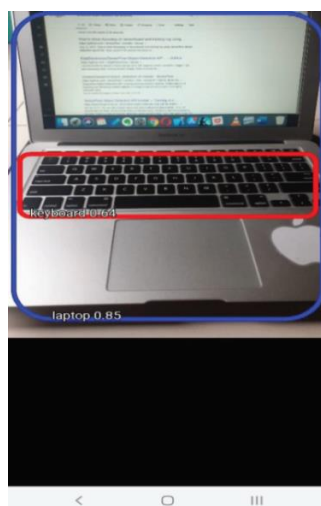


Figure 4.3: Screenshot of Object Recognition

4.2.1.3 A Smart Personal AI Assistant for Visually Impaired People (Felix, Kumar and Veeramuthu, 2018)

Felix, Kumar, and Veeramuthu have proposed a paper regarding an Android assistant application for visually impaired people in 2018. There are several features available in the application, those are object recognition, landmark recognition, text recognition, currency recognition, and chat bot.

A chat bot accepts voice commands from the user then returns suitable audio responses. A user can ask the chat bot about his or her current location and guidance to reach a destination. Furthermore, when the user asks the application for surrounding information, the application will reply to the user about the names of object detected in the images and their distance from the user. Other than that, the application also can alert the user about the obstacles in the surrounding through speech. Additionally, a user also can ask the application to read out the printed or handwritten text through text recognition.

However, the application must work under Internet connection, It is also developed in English only.

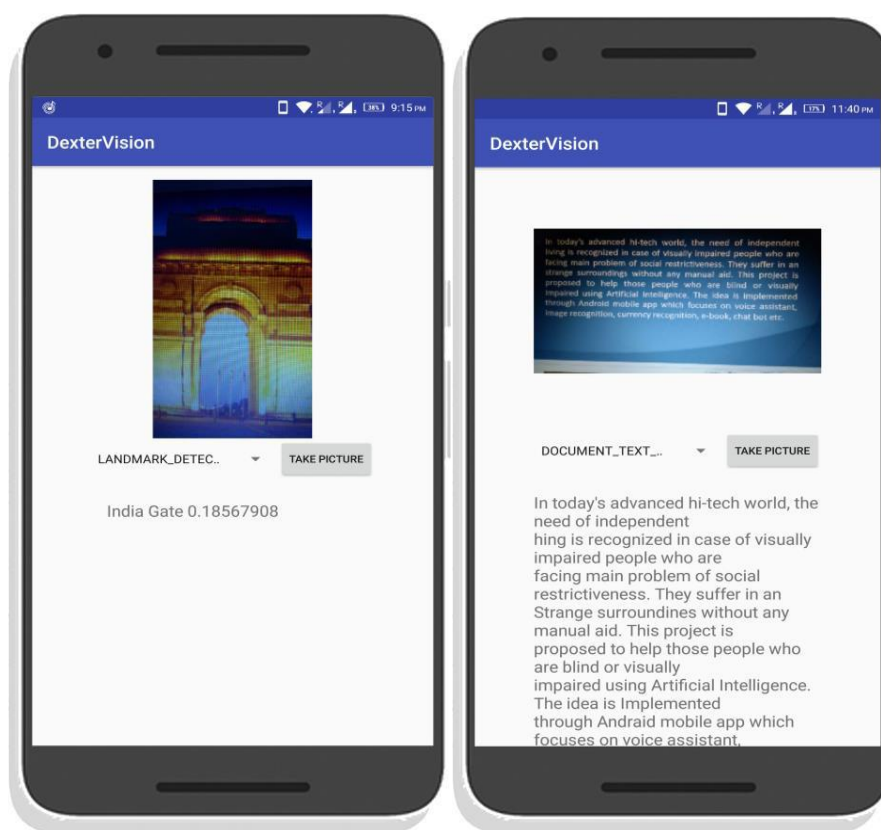


Figure 4.4: Image and Text Recognition in the Application

4.2.1.4 Real-Time Object Detection for Visually Challenged People (Vaidya et al, 2020)

This is another paper regarding object recognition application for visually impaired users. It is proposed by Vaidya, Shah, Shah, and Shankarmani in 2020. The application is developed in web and Android platforms. The only feature of the application is recognising objects in real-time and the application can run without requiring Internet connection.

Once the application is opened, the camera will be launched automatically and it will capture the surroundings view. User needs to press on the button "Start/Stop Yolo " to recognise the objects in the view. After the view captured is processed successfully, the application informs the name of the objects detected and the absolute location of the objects through voice output. The application also displays the name and the confidence score of each object and the boundary box that encloses the object.

The average computational time required for recognition is 2000ms. The application can detect and recognise more than 75% of the objects accurately at a time.

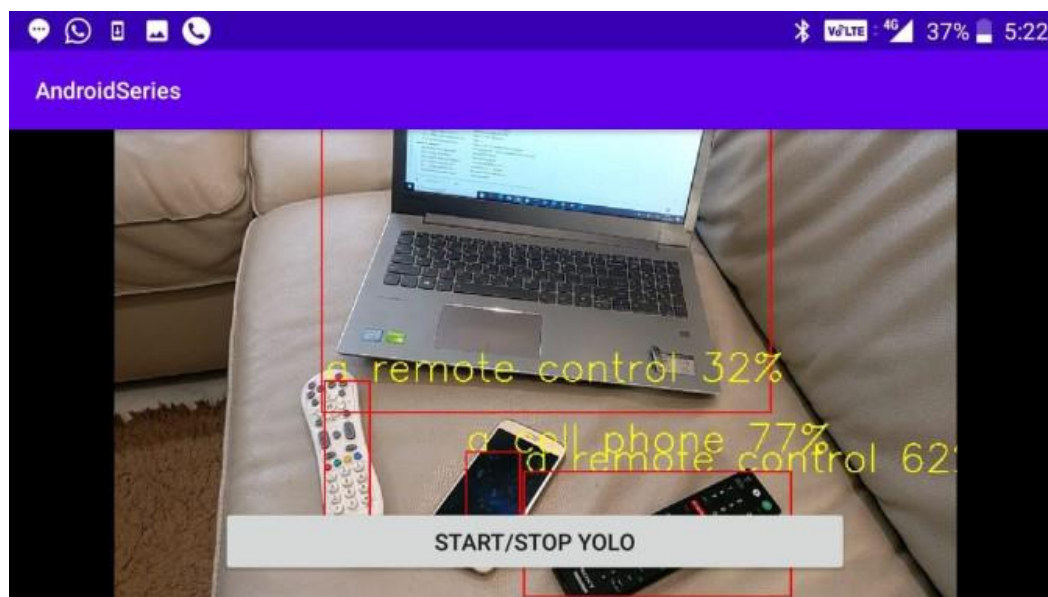


Figure 4.5: Object Recognition on the Android Application

4.2.1.5 Intelligent Eye: A Mobile Application for Assisting Blind People (Awad et al., 2018)

Intelligent Eye is also an Android application developed to perform object recognition to assist visually impaired people. It has been proposed by Awad, Haddad, Khneisser, Mahmoud, Yaacoub, and Malli in 2018. The application provides a set of features, including light detection, colour recognition, object recognition, and banknote recognition. All features can be accessed without Internet connection.

When opening the application, a homescreen with several options in Figure 4.6 is displayed. A user needs to choose an option by clicking a particular button. Light detection is performed by using the embedded light sensor in a smartphone. The values of light intensity are read. Then, the application plays sound beeps with different pitches based on the light intensity. The user can stop the beeps by clicking the “Stop” button. To make the application provide an audio description regarding the nature of the light, the user is required to click the “Discover button”. On the other hand, colour detection is performed by recognising the colour of the object in the image

captured. The colour name of the object is then notified to the user through speech. For object recognition, the user needs to take an image of the object. The application will then display and read out the first three relevant detection results together with their relevance scores. Adding the relevance scores of the three detection results brings a sum of 100%. For banknote recognition, the user also needs to capture an image of the banknote. The value of the banknote is then detected and read out.

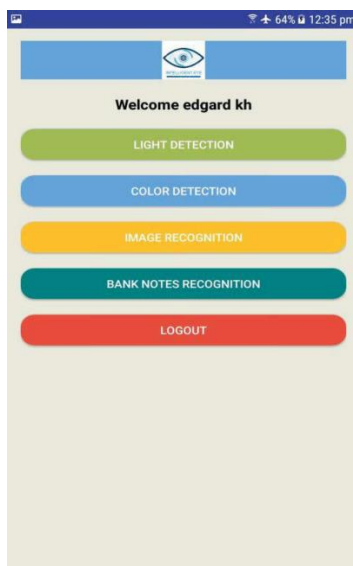


Figure 4.6: Homescreen of Intelligence Eye

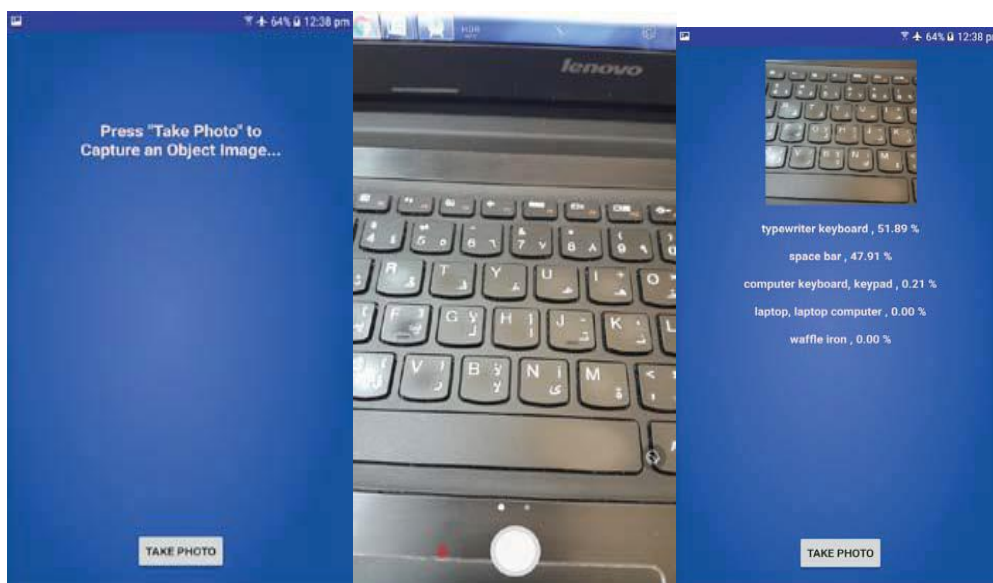


Figure 4.7: Image Capturing and Object Recognition in Intelligence Eye

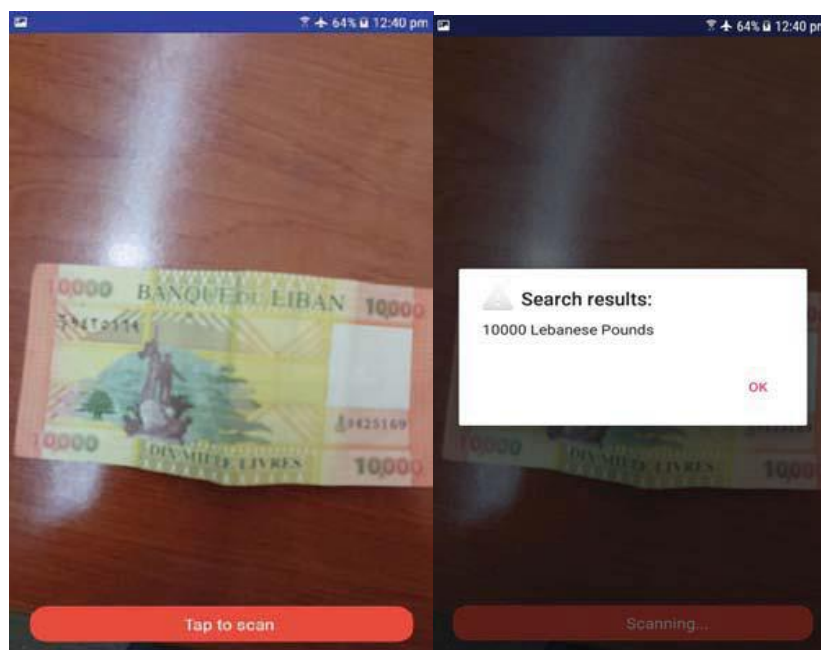


Figure 4.8: Banknote Capturing and Recognition in Intelligence Eye

4.2.1.6 Real Time Object Detection for Visually Challenged Persons (Anitha, Subalaxmi and Vijayalakshmi, 2019)

Anitha, Subalaxmi, and Vijayalakshmi have developed a real-time object recognition mobile application on Android platform. Object recognition is the only feature provided by the application. The application also does not require Internet connection to work.

After users open the application, the smartphone's camera is launched automatically and the application starts to detect and recognise the objects in the scene without requiring any commands or input from the user. As illustrated in Figure 4.9, the first three relevant detection results are displayed on the screen together with the probability. Sum of probabilities for all detection results is 100%. At the same time, the application will give a voice output to inform the detected object with the highest probability.

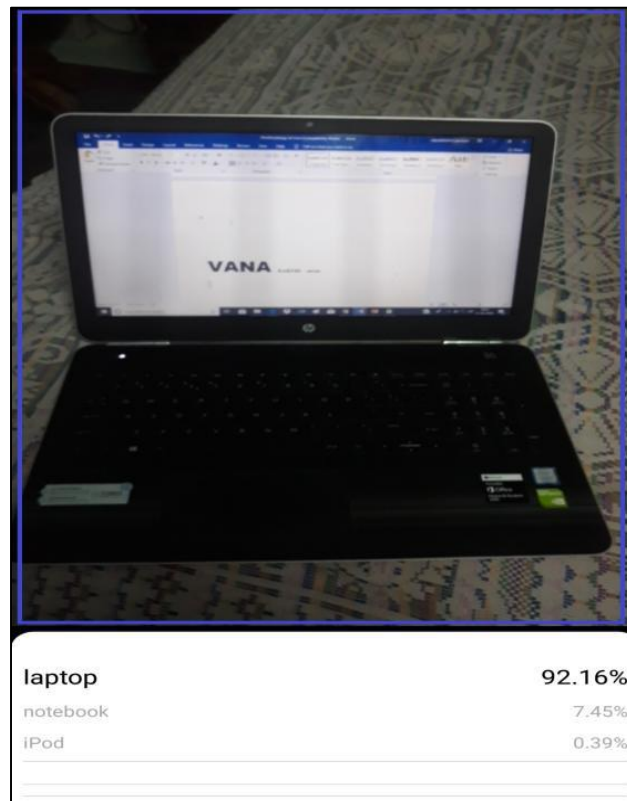


Figure 4.9: Object Recognition in the Application

4.2.1.7 Object Recognition App for Visually Impaired (Jakhete et al., 2019)

The paper has been proposed by Jakhete, Bagmar, Dorle, Rajurkar, and Pimplikar in 2019. This paper proposes an Android application that assists visually impaired people to recognise objects in real-time. The main features of the application is to locate the objects and provide the names of objects.

The application will detect and recognise objects in the scene automatically without requiring any user input after the application is opened. The application will then generate an audio output to inform about the object detected that has the maximum confidence in the scene captured. The scenes are chosen at a particular time interval to prevent the hindrance of the audio output. At the same time, the label, boundary boxes, and confidence score of all objects detected are shown on the screen.



Figure 4.10: Object Detection and Recognition in the Application

4.2.2 Comparison between Existing Mobile Application

Table 4.1: Comparison between Existing Object Recognition Mobile Applications for Visually Impaired People

Paper/ Application *	A	B	C	D	E	F	G
Platform	Android	Android	Android	Android, web	Android	Android	Android
Number of objects can be recognised in a scene	One	Multiple	One	Multiple	One	One	Multiple

Information provided in object recognition audio feedback	Names of object detected	Name, direction, and distance of only one object in the scene with the highest priority	Names and distances of objects detected	Names and absolute locations of objects detected	First three relevant detection results together with their relevance scores	Name of only one object in the scene with the highest confidence score	Name of only one object in the scene with the highest confidence score
Additional functions	Text recognition	No	Landmark recognition , text recognition , currency recognition , and chat bot	No	Light detection, colour detection, and banknote recognition	No	No
Interactivity	Yes (Click on buttons to choose recognising text or object)	No	Yes (in voice command form)	Yes (Click on a button to start or stop processing image)	Many	No	No
Interface navigability	Simple	Very simple	More interfaces	Very simple	A lot of interfaces	Very simple	Very simple
Internet connection	No	No	Yes	No	No	No	No

Languages	English	English, Hindi, and Marathi	English	English	English	English	English
-----------	---------	-----------------------------	---------	---------	---------	---------	---------

* A: Eye Assistant : Using Mobile Application to Help the Visually Impaired (Khan Shirshir et al., 2019); B: Android Based Object Detection System for Visually Impaired (Badave et al., 2020); C: A Smart Personal AI Assistant for Visually Impaired People (Felix, Kumar and Veeramuthu, 2018); D: Real-Time Object Detection for Visually Challenged People (Vaidya et al, 2020); E: Intelligent Eye: A Mobile Application for Assisting Blind People (Awad et al., 2018); F: Real Time Object Detection for Visually Challenged Persons (Anitha, Subalaxmi and Vijayalakshmi, 2019); G: Object Recognition App for Visually Impaired (Jakhete et al., 2019)

4.2.3 Conclusion

Object recognition is the most important function to be included in the object recognition mobile application for visually impaired people. All applications in Table 4.1 include this function. Due to time constraints, this project focused on this important function only.

However, each application can recognise a different number of objects in the scene. In this project, an application that can recognise up to 10 objects in a scene was developed. Furthermore, there are some differences in information provided in audio feedback among the applications compared. In this project, the application developed will notify the visually impaired user about names of the objects detected within a scene. The names are reported following the descending order of accuracy scores. The reason to recognise and report multiple objects in the scene is to increase the safety of the users when navigating. This tries to prevent the users from colliding with any obstacles in their path.

Furthermore, the level of interactivity between the application and the user of each existing application is different. Some existing applications capture scenes and recognise objects in the surrounding without requiring any input from the user. On the other hand, some applications require the user to

click on buttons or give commands before capturing scenes and recognising objects. Nevertheless, it is not a good idea to include too many visual interactions and interfaces since the visually impaired users are not able to see the screen. In this project, the application does not need the user to capture an image of the objects for object recognition. This can prevent the problem of inaccurate recognition results due to poor image quality.

The application to be developed in this project does not require Internet connection to run. It enables the application to be used when the Internet signal is not good or the user does not have Internet access.

Although object recognition is the only service to be developed, this project also developed some other features to improve user experience. Rather than just scanning surroundings and telling the name, location, and confidence score of each detected object like the applications studied, this application allows the user to stop or play speech and set speech rate of voice feedback. The user gains more control over audio feedback settings through implementation of these features. Besides, this project developed the application that supports English only since English is the international language.

4.3 Requirement Specification

4.3.1 Functional Requirements

The functional requirements for the application are:

- i. The application shall be able to detect and recognise at least 90 classes of objects.
- ii. The application shall be able to detect and recognise up to 10 objects in a scene within 2 metres away from the camera's field-of-view.
- iii. The application shall be able to launch the camera once getting camera access permission from the user.
- iv. The application shall be able to scan and capture the surrounding objects automatically once the camera is launched without requiring the user to click anything to capture the image.
- v. The application shall be able to indicate the location of a detected object using a boundary box.

- vi. The application shall be able to display the predicted name and the confidence score of each object detected.
- vii. The application shall be able to provide voice feedback to alert the user of the detected objects.
- viii. The application shall be able to discard incorrect recognition results with a confidence score of less than 60%.
- ix. The application shall be able to enable users to stop the feedback speech.
- x. The application shall be able to enable users to play the speech after stopping the speech.
- xi. The application shall be able to enable users to set the speech rate of the voice feedback.

4.3.2 Non Functional Requirements

4.3.2.1 Availability

- i. The application shall be available for 24/7.
- ii. The application shall be available to work normally offline.

4.3.2.2 Usability

- i. The application shall be user friendly by providing a simple user interface and system flow which are able to be used by the visually impaired user easily and conveniently although they have problems in watching the screen.

4.3.2.3 Performance

The application shall be able to detect and recognise objects and give voice feedback to the users within five seconds.

4.4 Use Case Modeling

4.4.1 Use Case Diagram

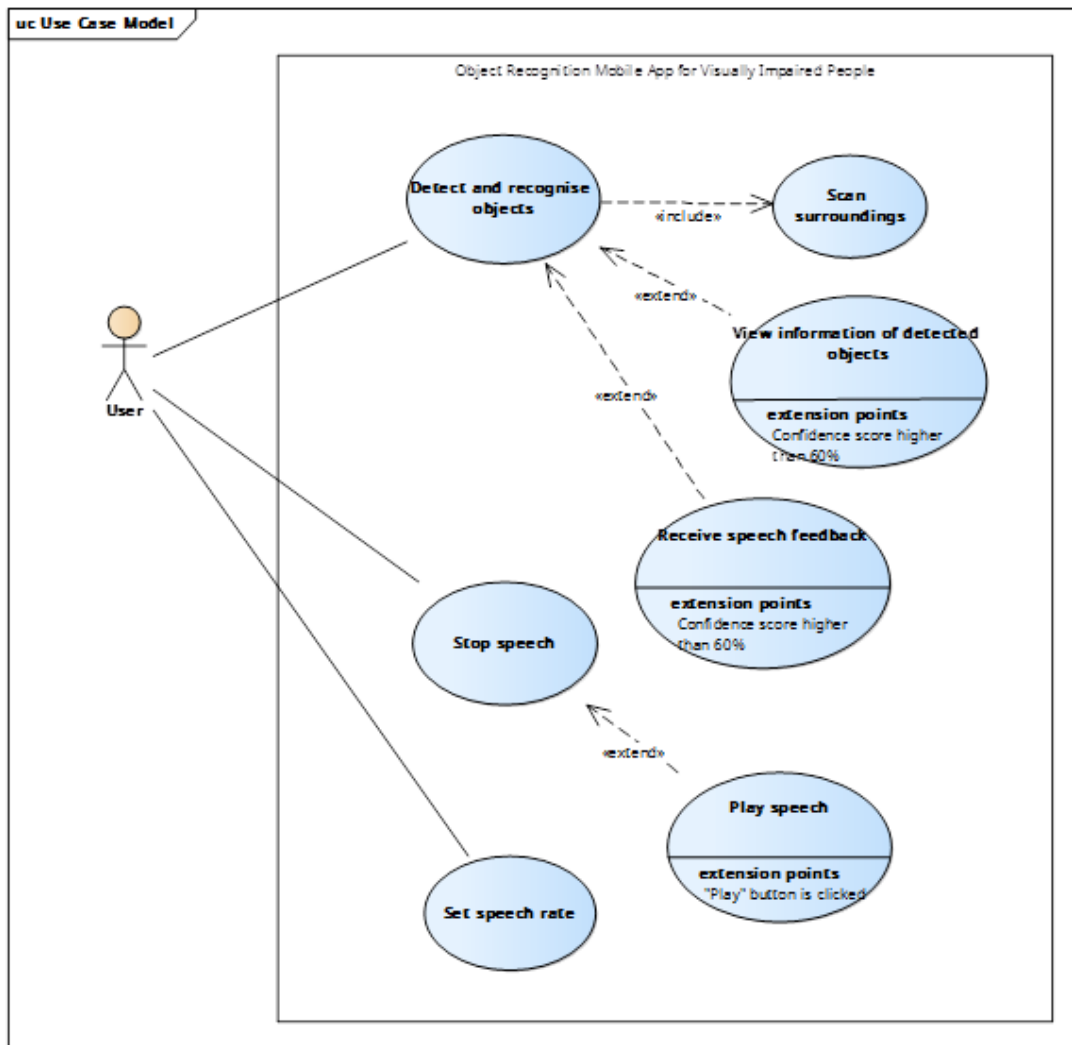


Figure 4.11: Use Case Diagram of the Application

4.4.2 Use Case Description

Table 4.2: Scan Surroundings Use Case

Use Case ID	1
Use Case Name	Scan surroundings
Actors	User
Description	User wants to scan his or her surroundings using the smartphone's camera.

Pre-condition	User opens the application.
Post-condition	-
<p>Relationships:</p> <p>Association: n/a</p> <p>Include: n/a</p> <p>Extend: n/a</p> <p>Generalization: n/a</p>	
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. User clicks on the icon of the application. 2. System launches the camera automatically and captures the surrounding view. 	
<p>Alternative flow of events:</p> <p>2.1 If the application is launched on the device for the first time, system requests camera access permission from user.</p> <p style="padding-left: 40px;">2.1.1 User choose to allow the camera access by clicking the “Allow” button.</p> <p style="padding-left: 80px;">2.1.2.1 Use case continues.</p> <p style="padding-left: 40px;">2.1.2 User choose to deny the camera access request by clicking the “Deny” button.</p> <p style="padding-left: 80px;">2.1.2.1 System exits.</p> <p style="padding-left: 80px;">2.1.2.2 Use case terminates.</p>	

Table 4.3: Detect and Recognise Objects Use Case

Use Case ID	2
Use Case Name	Detect and recognise objects
Actors	User
Description	User wants to detect and recognise objects in the surroundings.

Pre-condition	User scans the surroundings using the smartphone's camera.
Post-condition	-
<p>Relationships:</p> <p>Association: User</p> <p>Include: Scan surroundings</p> <p>Extend: Receive speech feedback</p> <p>Generalization: n/a</p>	
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. Perform Scan surroundings use case. 2. User holds the camera against the objects wanted to be recognised. 3. System detects and recognises objects in the scene captured. 4. System assigns a label, boundary box coordinates, and confidence score for each object detected. 	
<p>Alternative flow of events:</p> <p>2.1 The camera is covered.</p> <p>2.1.1 The system cannot detect and recognise any objects.</p> <p>2.1.2 Use case terminates.</p>	

Table 4.4: View Information of Detected Objects Use Case

Use Case ID	3
Use Case Name	View information of detected objects
Actors	User
Description	User wants to view the information of detected objects displayed on the smartphone screen.
Pre-condition	System detects and recognises an object in the scene captured and its confidence score is larger than 60%.
Post-condition	-

<p>Relationships:</p> <p>Association: n/a</p> <p>Include: n/a</p> <p>Extend: n/a</p> <p>Generalization: n/a</p>
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. Perform Detect and recognise objects use case. 2. System indicates the location of each object detected by using a boundary box to surround the object. 3. System displays the name and confidence score of each object detected. 4. User views the name, boundary box, and confidence score of each object detected.
<p>Alternative flow of events:</p> <ol style="list-style-type: none"> 2.1 The recognition result of an object is lower than 60%. <ol style="list-style-type: none"> 2.1.1 System discards the recognition result. 2.1.2 System does not display the information of the object on the screen.

Table 4.5: Receive Speech Feedback Use Case

Use Case ID	4
Use Case Name	Receive speech feedback
Actors	User
Description	User wants to get speech feedback about the objects in the surroundings through the smartphone's speaker or earphones.
Pre-condition	System detects and recognises an object in the scene captured and its confidence score is larger than 60%.
Post-condition	-

<p>Relationships:</p> <p>Association: n/a</p> <p>Include: n/a</p> <p>Extend: n/a</p> <p>Generalization: n/a</p>
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. Perform Detect and recognise objects use case. 2. System sorts the recognition results in descending order of confidence score. 3. System speaks out the names of the objects in the recognition results according to the order. 4. User receives the speech feedback about the information of objects in the surroundings.
<p>Alternative flow of events:</p> <ol style="list-style-type: none"> 2.1 The recognition result of an object is lower than 60%. <ol style="list-style-type: none"> 2.1.1 System discards the recognition result. 2.1.2 System does not inform user about the name of the object.

Table 4.6: Stop Speech Use Case

Use Case ID	5
Use Case Name	Stop speech
Actors	User
Description	User wants to stop speech feedback from system.
Pre-condition	User opens the application.
Post-condition	-

<p>Relationships:</p> <p>Association: User</p> <p>Include: n/a</p> <p>Extend: Play speech</p> <p>Generalization: n/a</p>
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. User clicks on the “Stop/Play Speech” button. 2. System stops giving the speech feedback. 3. User does not receive speech feedback from system anymore.
<p>Alternative flow of events: -</p>

Table 4.7: Play Speech Use Case

Use Case ID	6
Use Case Name	Play speech
Actors	User
Description	User wants to play the stopped speech feedback.
Pre-condition	User stops the speech feedback.
Post-condition	-
<p>Relationships:</p> <p>Association: n/a</p> <p>Include: n/a</p> <p>Extend: Play speech</p> <p>Generalization: n/a</p>	
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. User clicks on the “Stop/Play Speech” button. 2. Perform Receive speech feedback use case. 	

Alternative flow of events: -

Table 4.8: Set Speech Rate Use Case

Use Case ID	7
Use Case Name	Set speech rate
Actors	User
Description	User wants to set the rate of the speech feedback provided by system.
Pre-condition	User opens the application.
Post-condition	-
<p>Relationships:</p> <p>Association: User</p> <p>Include: n/a</p> <p>Extend: n/a</p> <p>Generalization: n/a</p>	
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. User clicks on the “Set speech rate” button. 2. System displays a list of speech rate. 3. User selects the speech rate he or she wants. 4. System saves the speech rate chosen by user. 5. System updates the speech rate of the voice feedback. 	
<p>Alternative flow of events:</p> <ol style="list-style-type: none"> 3.1 User does not select a speech rate. <ol style="list-style-type: none"> 3.1.1 Use case terminates 	

4.5 Proposed User Interface Design

The figures below illustrate the proposed user interface design. The design was finished in the first iteration of the project development. The design is able to work and recognise objects.

This is the screen displayed when the user opens the application. The application will launch the camera automatically then detect and recognise objects in the scene. The name, location, and confidence score of each object are returned.



Figure 4.12: UI Design for Object Recognition Screen



Figure 4.13: Recognition for Multiple Objects

CHAPTER 5

PROJECT IMPLEMENTATION

5.1 Introduction

As mentioned in the earlier chapters, SSD Mobilenet V1 and Faster R-CNN Inception V2 are the models pre-trained on the COCO dataset. The COCO dataset focuses more on the objects within the general context, while the fine-tuned models focus on the objects within the environmental context, especially indoor and outdoor obstacles. Transfer learning has been applied to the pre-trained models to train the models to detect and recognise 40 classes of objects stated in Table 5.1. Also, the fine-tuned models were trained to detect and recognise certain object classes that are not included in the COCO dataset. The detailed comparison of object classes in the pre-trained and fine-tuned models are available in Table 2.2 in Chapter 2.

Table 5.1: Object Classes to be Trained in Transfer Learning

Person	Cat	Couch	Plush toy
Bicycle	Dog	Potted plant	Bin
Car	Umbrella	Bed	Bag
Motorcycle	Ball	Table	Street sign
Bus	Skateboard	Scissors	Fan
Train	Bottle	Toilet	Tree
Truck	Knife	Window	Street lamp
Bench	Chair	Door	Shelf
Staircase	Mouse	Sink	Clock
Computer	Cell phone	Refrigerator	Vase

This chapter discusses the activities carried before, during, and after the model training. It also describes the implementation of the fine-tuned model on the mobile application. Figure 5.1 shows a summary of the training processes.

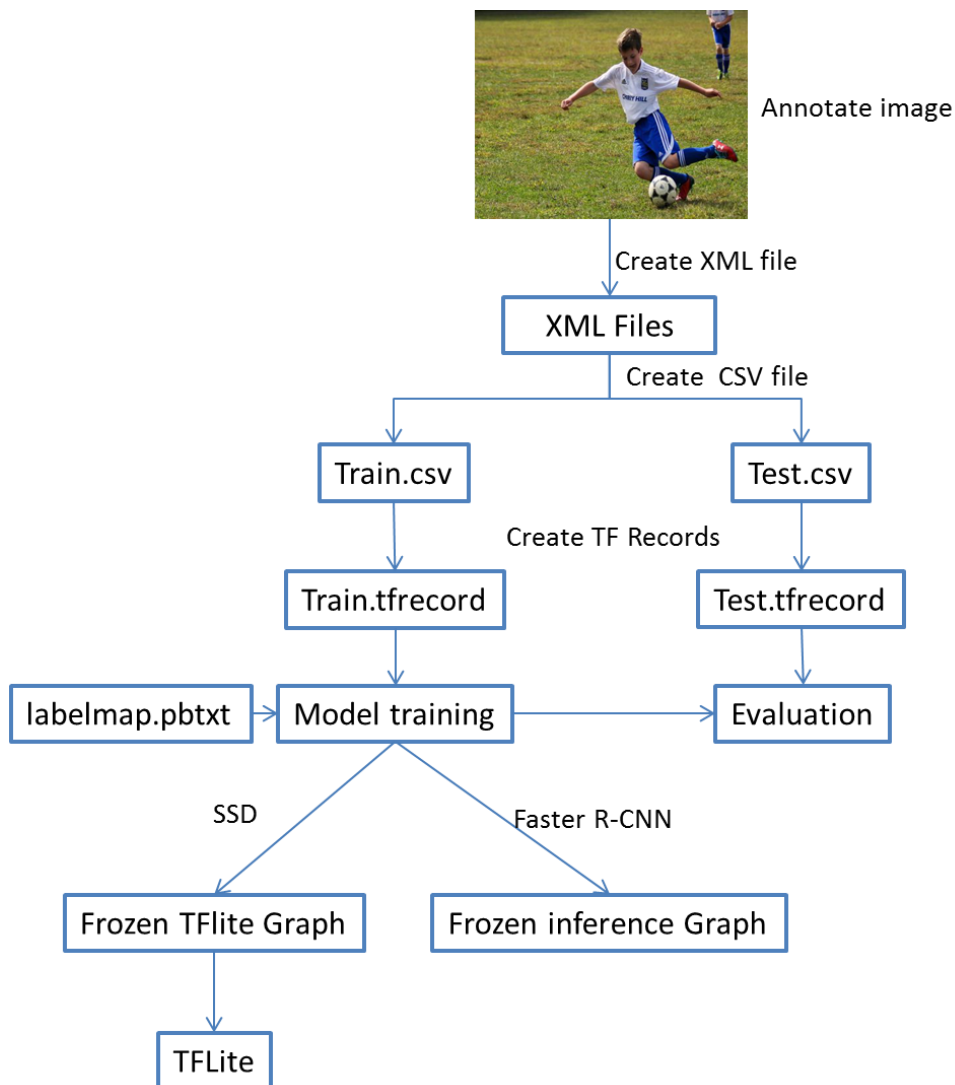


Figure 5.1: Summary of the Training Process

5.2 Pre-training

Before starting the transfer learning, the train and test datasets were prepared and the pre-trained models were chosen.

5.2.1 Dataset Preparation

First and foremost, a train dataset and test dataset must be prepared before starting to train a model. Since the pre-trained models are trained on the COCO dataset, most of the images in the train dataset are chosen from the COCO dataset. The dataset was then processed to ensure that every object class has at least 200 examples in the train dataset to improve the performance

of the trained model. The final train dataset contains 8004 training images, while the test dataset consists of 674 testing images.

Then, the train and test dataset were annotated manually. The annotated classes and coordinates of boundary boxes of all objects in an image are stored in an XML file. After that, for each train dataset and test dataset, all XML files of the dataset were converted into a single CSV file, which was further converted into a TF Record file. TF Record is a standard format accepted by Tensorflow and they store the data into binary strings and serve as input for machine learning framework.

	A	B	C	D	E	F	G	H	I
1	filename	width	height	class	xmin	ymin	xmax	ymax	
2	145443423	1500	1000	cat	497	309	1407	833	
3	145443423	1500	1000	dog	62	353	1499	844	
4	159188042	1000	667	cat	463	246	720	485	
5	159188042	1000	667	dog	284	124	539	589	
6	172240464	542	371	motorcycl	183	106	400	351	
7	172240464	542	371	person	232	31	390	189	
8	20210117_	232	412	staircase	16	80	173	412	
9	20210117_	232	412	bin	208	236	232	274	
10	20210117_	232	412	scissors	75	202	154	357	
11	20210117_	232	412	mouse	190	166	232	225	
12	20210118_	232	412	scissors	96	105	228	233	
13	20210118_	232	412	bin	18	124	187	290	
14	20210118_	232	412	chair	1	1	232	377	
15	20210118_	232	412	shelf	39	1	232	114	
16	20210118_	232	412	shelf	8	1	57	85	
17	20210118_	412	232	scissors	154	122	293	178	
18	20210118_	412	232	bin	170	33	323	180	
19	20210118_	412	232	bottle	137	61	180	138	
20	20210118_	412	232	chair	109	1	412	232	
21	20210118_	412	232	table	1	143	110	232	
22	20210118_	232	412	bin	65	213	159	324	
23	20210118_	232	412	chair	1	100	232	412	
24	20210118_	232	412	fan	137	1	210	107	
25	20210118_	232	412	table	1	68	56	140	

Figure 5.2: Display of Certain Records in CSV File

5.2.2 Pre-trained Model Selection

Afterwards, the pre-trained models were selected to conduct transfer learning. As mentioned in the earlier chapters, the pre-trained models to be used in this project are SSD Mobilenet V1 COCO Model and Faster R-CNN Inception V2 COCO Model from the Tensorflow Zoo. These models do not perform well at

detecting and recognising the 40 classes required for this project. Hence, transfer learning was done to fine-tune the pre-trained models with the required datasets so they are able to predict the 40 classes more effectively.

5.3 Training

There are several steps involved in model training, which are preparing a labelmap file, loading data into memory, and then running the training.

5.3.1 Preparation of Labelmap File

A labelmap file that contains string class names with their corresponding integer IDs is created and saved in ptxt format. This labelmap file and the TF records are requisite for the model training.

```
item {  
  id: 1  
  name: 'bag'  
}  
item {  
  id: 2  
  name: 'ball'  
}  
item {  
  id: 3  
  name: 'bed'  
}  
item {  
  id: 4  
  name: 'bench'  
}  
item {  
  id: 5  
  name: 'bicycle'  
}
```

Figure 5.3: Display of Some Classes in Labelmap File

5.3.2 Data Loading

Since the volume of the training data is too large, it requires a large amount of memory. It is difficult to load all images into the local memory. This causes the training to be killed at the beginning. Hence, the data is loaded by batches during the training. Nevertheless, this solution still consumes a large amount of memory and decreases the efficiency of the training significantly. Thus, Google Colaboratory that provides 35 GB of memory was used to train the

model. All files required for training were uploaded to Google Drive to be accessed by Google Colaboratory. GPU on Google Colaboratory also has been used to speed up the training process.

5.3.3 Model Training

The checkpoint of the pre-trained model was used as the starting point for the transfer learning process and was used to initialise the weights of the CNN. Then, the last fully connected layer which is a classifier that returns the predicted class label for the pre-trained model was removed from the pre-trained model. The rest of the layers in the pre-trained model were frozen and used to train a new classifier that detects and recognises the required classes.

During the training, every input image was resized to a fixed size of 300x300 pixels to improve the training efficiency. Furthermore, dropout and data augmentation were applied to prevent the model from overfitting. Dropout is a mechanism where a subset of neurons was removed in each training iteration to construct many networks from the same single CNN. Data augmentation is a mechanism to create artificial images by making modifications such as transformations and rotations on the existing images. Horizontal flips were used during the training process. In this project, the model was trained for 200000 steps before being evaluated.

5.4 Export Trained Model

After the model training was finished, the model was converted into a graph for running the detections. For the SSD model, it was exported to frozen SSD Tensorflow Lite Graph and then further converted to Tensorflow Lite file. Tensorflow Lite is an optimised version of the trained model for lightweight mobile use. As Faster R-CNN was not supported by Tensorflow Lite, the Faster R-CNN model was exported into Frozen Inference Graph. These graphs allow the trained model to run on the mobile application.

5.5 Evaluation

The models were evaluated after the training was completed. The evaluation results will be discussed in the next chapter.

5.6 Implementation of Mobile Application

An Android-based object detection and recognition mobile application was developed to help the visually impaired people to identify the objects in their surroundings. This mobile application was written in Java.

5.6.1 Object Detection and Recognition on Mobile Application

The application will start to detect and recognise objects in the scene once the application is launched without requiring the user to click any button. After the application is launched successfully, the application will retrieve the scene captured by the smartphone's camera. Next, the scene is passed to the trained Tensorflow model for detecting and recognising the objects in the scene. After scene processing, the class, coordinates of boundary box, and confidence score of each detected object are returned by the model. Android Graphics library is used to draw the rectangle boundary box and text on the screen. Figure 5.4 illustrates the overall process flow of object detection and recognition on the mobile application.

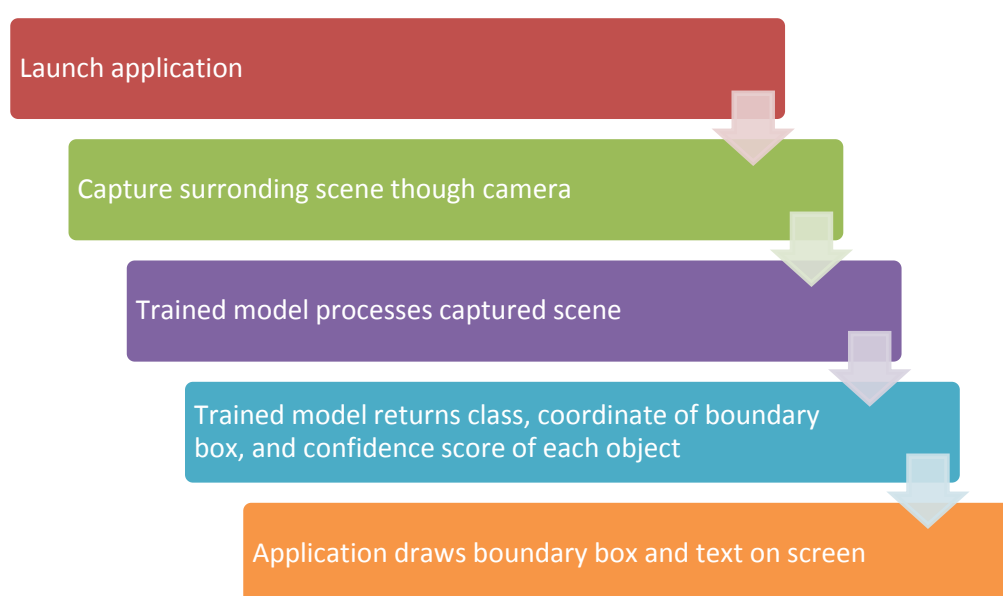


Figure 5.4: Process Flow for Object Detection and Recognition on Mobile Application

5.6.2 Mobile Application User Interface

Figure 5.5 and Figure 5.6 shows the home screen of the application. Since this application is developed for visually the impaired people, the application is designed in a simple way and have few buttons and screens to reduce the inconvenience when they use the application. The user is allowed to stop the voice feedback and play the voice feedback after being stopped. Furthermore, the user also can adjust the speech rate of the voice feedback.

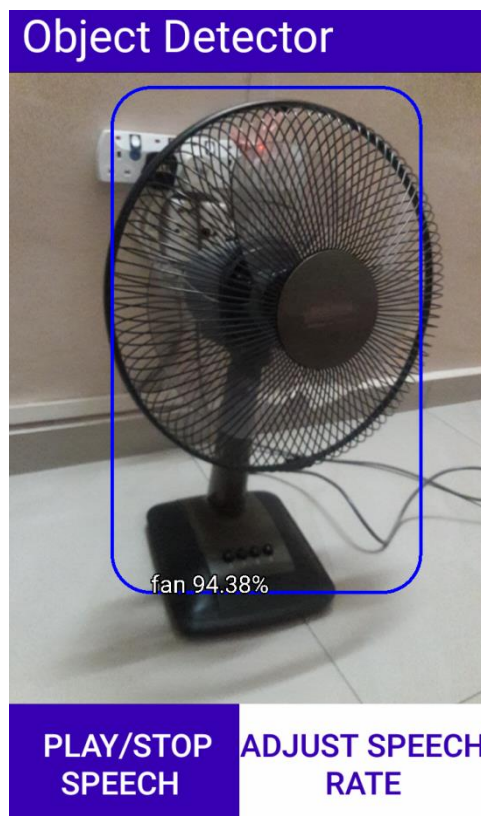


Figure 5.5: Screen when Detecting Single Object



Figure 5.6: Screen when Detecting Multiple Objects

5.6.3 Reporting of Directions of Detected Objects

In order to inform the user about the directions of the objects detected, the screen of the mobile screen is split into three parts, those are left, middle, and right. The direction of each object is decided by identifying which part of the screen it falls into. Figure 5.7 displays how the screen is split into three parts. The object which lies in the first 1/3 of screen width is considered on the left, while the object which lies in the last 1/3 of screen width is considered on the right. The object is considered in the middle if it lies in the rest of the area. The names and directions of the detected objects are spoken out every three seconds.

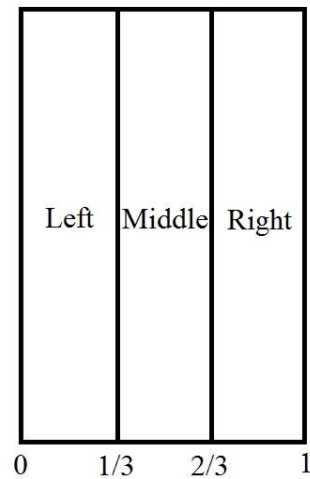


Figure 5.7: Splitting of Mobile Screen into Three Parts

5.6.4 Adjusting Speech Rate

The user is allowed to adjust the speech rate to the speed they are comfortable with. The application allows the user to choose three types of speech rate, those are slow, normal, and fast. The normal speech rate is 1.0. Slow speech rate has a speed of 0.5, while the fast speech rate has a speed of 1.5. The selected speech rate then will be saved into Shared Preferences, so the user does not need to readjust the speech rate again every time after closing the application. Figure 5.8 displays the screen when the user adjusts the speech rate.

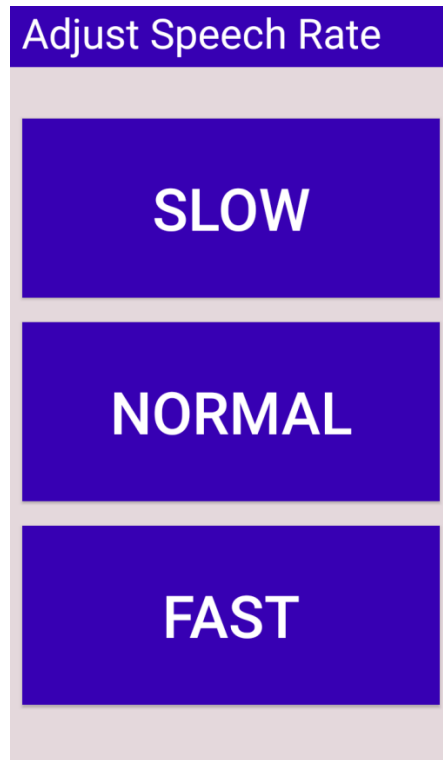


Figure 5.8: Adjust Speech Rate Screen

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 Introduction

This chapter will explain the effects of training settings on the performance of the model training. Also, the performance of the trained model was evaluated and discussed after the model training was done.

6.2 Training Settings

The performance of the model training can be affected by the training batch size used and the types of model to be trained. This sections will explore how the batch size and model types can affect the speed and effectiveness of the training.

6.2.1 Batch Size

Batch size defines the number of training samples used in one iteration. An experiment regarding the configuration of batch size was conducted on CPU on the local machine. The observations are recorded in Table 6.1. Figure 6.1 to Figure 6.4 visualises the development of the loss function during the model training when using four different batch size settings.

Table 6.1: Comparison of Training Performance When Using Different Batch Size

Batch size	Average time needed for per step (seconds)	Total training time	Number of steps trained
4	4	2 h 31 min	2456
12	12	5 h 17 min	1517
24	23	2 h 57 min	1034
36	40	5 h 41 min	493

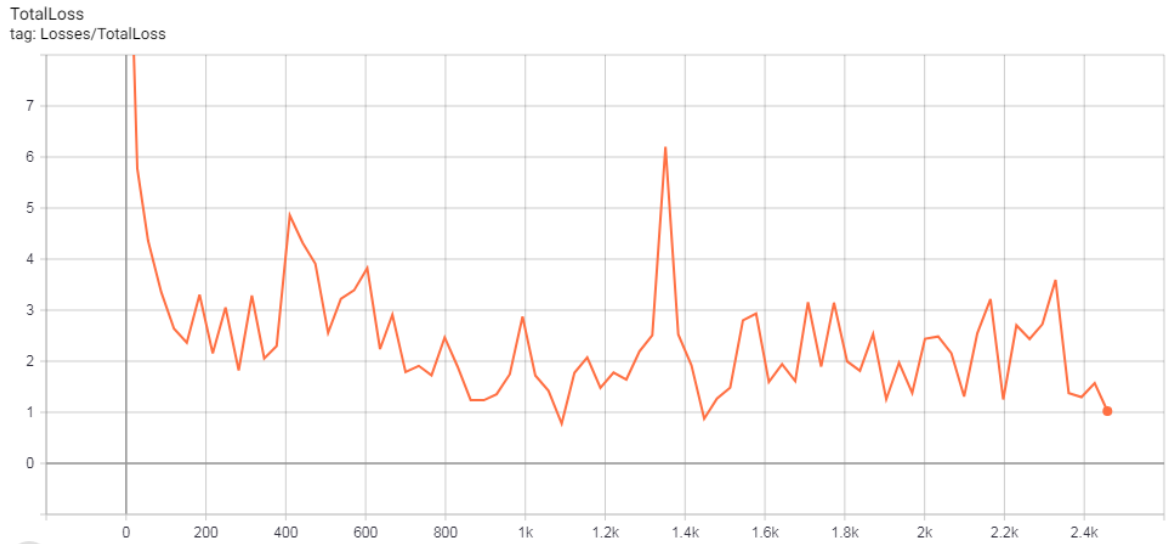


Figure 6.1: Development of Total Loss During Training When Batch Size=4
(After 2 h 31 min of Training)

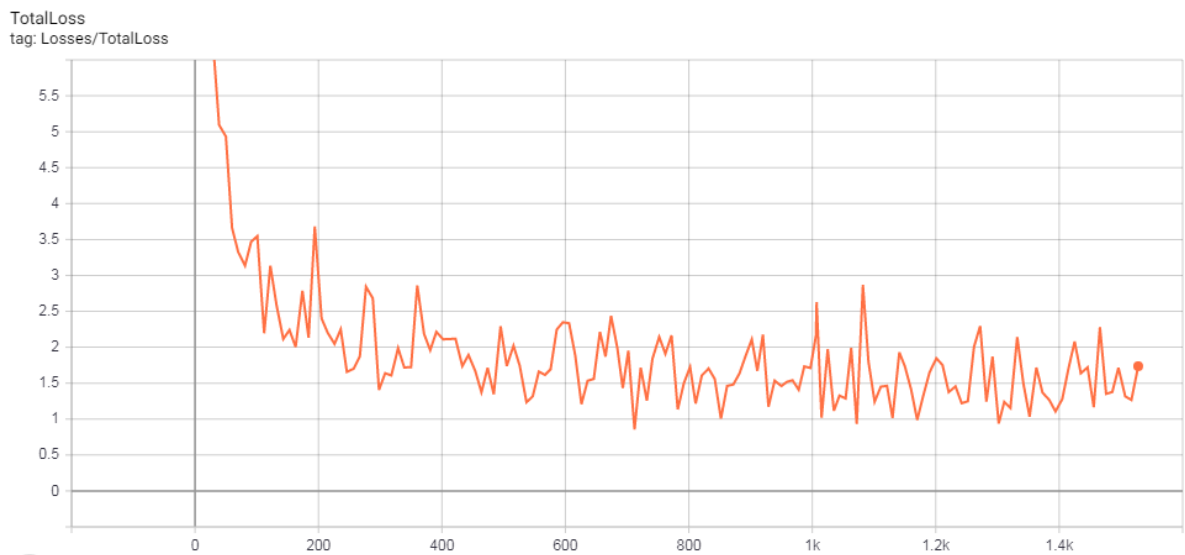


Figure 6.2: Development of Total Loss During Training When Batch Size=12
(After 5 h 17 min of Training)

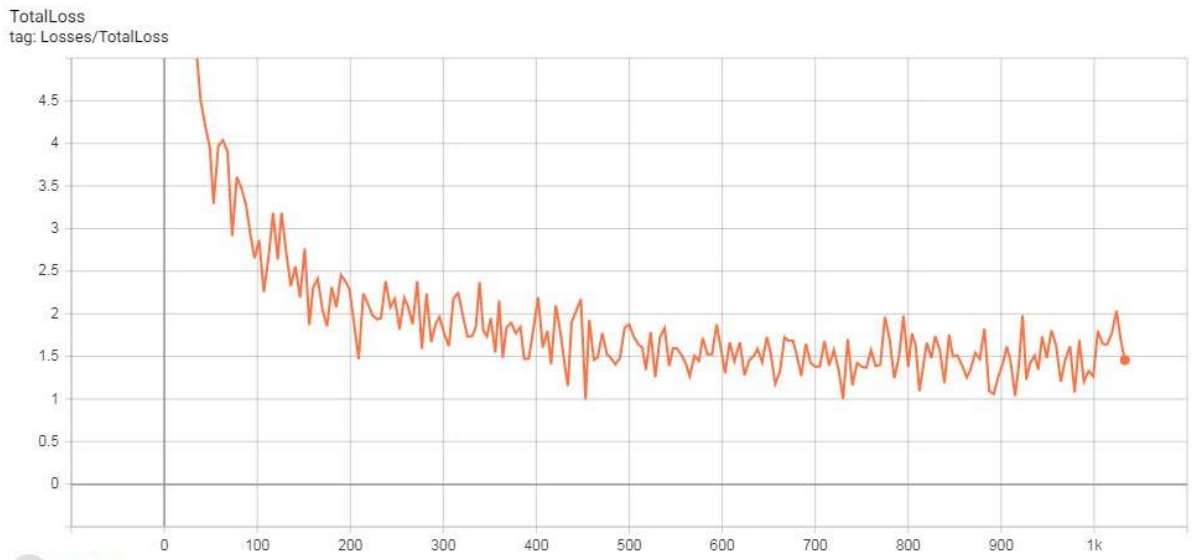


Figure 6.3: Development of Total Loss During Training When Batch Size=24
(After 2 h 57 min of Training)

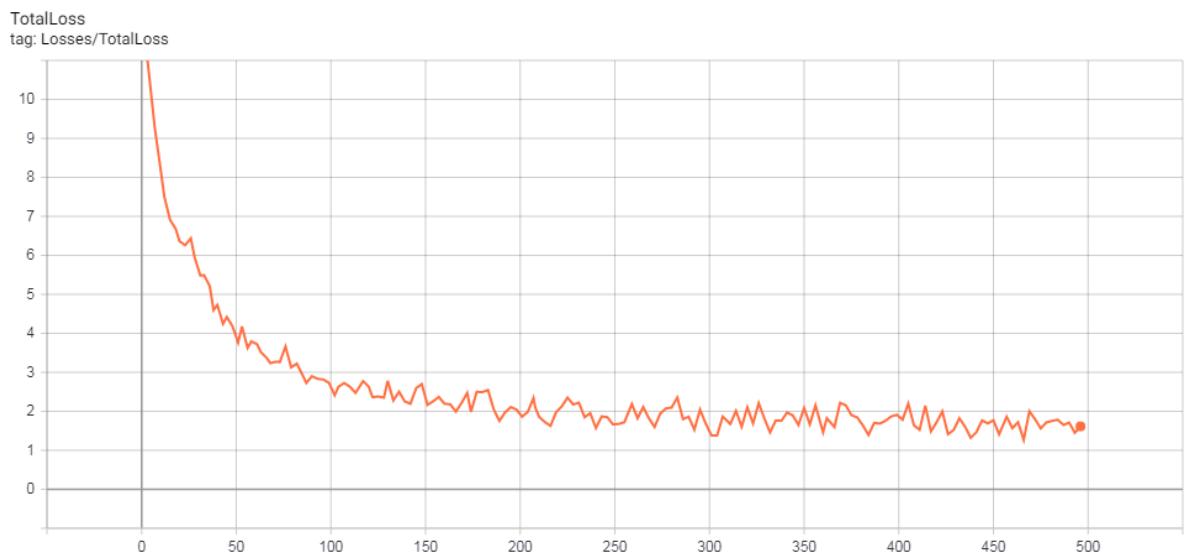


Figure 6.4: Development of Total Loss During Training When Batch Size=36
(After 5 h 41 min of Training)

A few issues arise when the batch size increases. Firstly, as shown in Table 6.1, the average time needed to run per step increases as the batch size increases. Besides, large batch size also consumes more system memory in the training. Furthermore, it will require more training time for the model to achieve the same accuracy compared to a batch size of 24 due to the decrease in iteration time. From Figure 6.3 and Figure 6.4, the model is able to achieve

loss below 2 consistently within 2 hours 57 minutes by using the batch size of 24, while the model that uses a batch size of 36 needs more than five hours of training to achieve loss below 2.

However, problems also arise when the batch size is too small. Although the average time needed per step decreases, the model may not be able to converge within the expected steps. As observed in Figure 6.3, the model that uses the batch size of 24 can achieve loss below 2 consistently after training for almost 450 steps. However, in Figure 6.1, loss of the model that uses a batch size of 4 is still very inconsistent after the model is trained for more than 2400 steps. Also, in Figure 6.2, the model that applies a batch size of 12 requires more than 1450 steps to reach loss below 2 consistently.

Hence, the batch size applied in this project is 24, which is the default value set by Tensorflow.

6.2.2 Training Time of Different Models

The training time required for different models to reach 200000 steps are recorded in Table 6.2.

Table 6.2: Comparison of Training Time Between SSD and Faster R-CNN Models

Model	Average time needed for per step (seconds)	Total Training Time to Reach 200k Steps
SSD	1.235	130 h 23 min
Faster R-CNN	0.293	20 hours

According to Table 6.2, the pre-trained Faster R-CNN model is almost six times faster than the pre-trained SSD to run each training step. Furthermore, the SSD pre-trained model consumes 130 hours 23 minutes (almost five days) to fine-tune until 200000 steps. On the other hand, Faster R-CNN just needs about 20 hours to complete the 200000 steps. This shows that Faster R-CNN has a much faster training speed compared to SSD.

6.3 Evaluation

The evaluation was carried out to compare the performance of both SSD Mobilenet V1 COCO Model and Faster R-CNN Inception V2 COCO Model before and after transfer learning. Certain types of evaluation metrics such as mAP, precision, recall, and confusion matrix are used to calculate the performance of the overall models and each class on the test dataset which contains 674 examples of 40 classes. All the evaluation results displayed in the following sections are calculated on this test dataset only.

6.3.1 Comparison of Performance between Pre-trained and Fine-tuned Models

The performance between both pre-trained and fine-tuned models were compared to prove how transfer learning can help in improving the accuracy of the model. Figure 6.5, 6.6, and 6.7 displays some examples of detection results of different models.



(i) Pre-trained SSD Model

(ii) Fine-tuned SSD Model

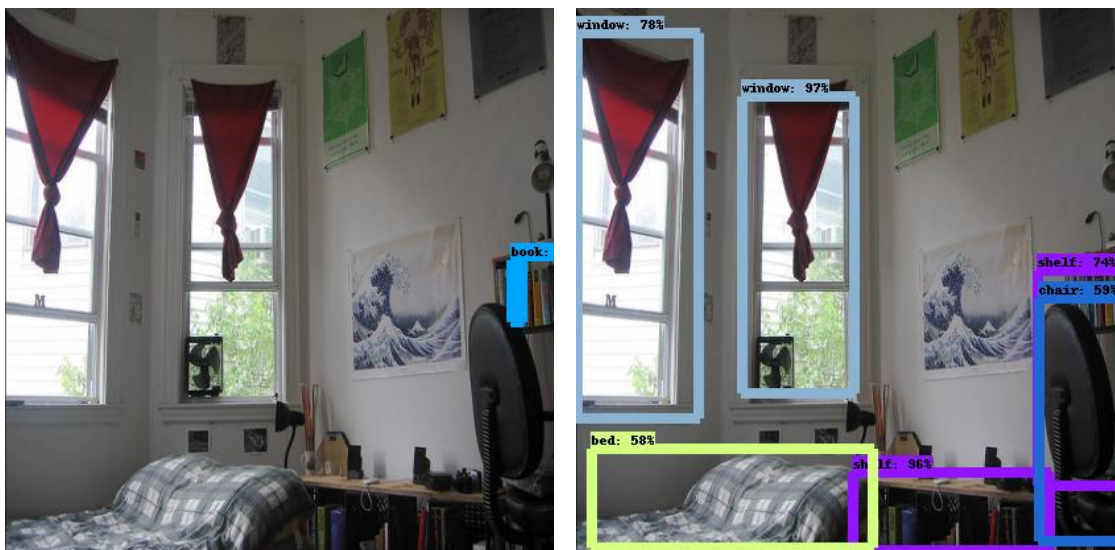


(iii) Pre-trained Faster R-CNN Model

(iv) Fine-tuned Faster-RCNN Model

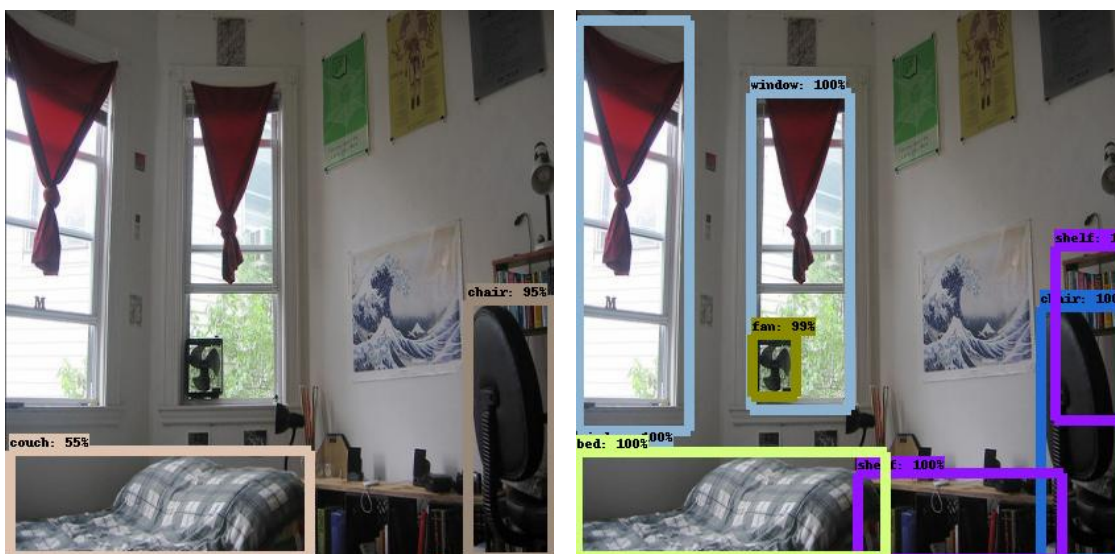
Figure 6.5: Detections Results of Different Models in Toilet

In Figure 6.5, the pre-trained and fine-tuned models are able to detect and recognise “toilet” and “sink” correctly. Since “window” is not present in the COCO dataset, so the pre-trained models are not able to detect and recognise the objects “window” in the scene. Also, the fine-tuned SSD and Faster R-CNN models have higher confidence score for their detected objects compared to that of their pre-trained models. Moreover, the pre-trained Faster R-CNN model detects two sinks instead of one sink.



(i) Pre-trained SSD Model

(ii) Fine-tuned SSD Model



(iii) Pre-trained Faster R-CNN Model

(iv) Fine-tuned Faster-RCNN Model

Figure 6.6: Detections Results of Different Models in Bedroom

Since “window”, “shelf”, and “fan” are not existing in the COCO dataset, the pre-trained models should not detect and recognise these objects in Figure 6.6. However, the pre-trained SSD model fails at detecting and recognising “bed” and “chair” successfully, while the pre-trained Faster R-CNN model misclassifies “bed” as “couch”. It can be observed that the fine-tuned models are able to detect and recognise more objects in the scene at a higher confidence score compared to the pre-trained models.



Figure 6.7: Detections Results of Different Models at Outside

In Figure 6.7, “tree” should not be detected and recognised by the pre-trained models because the class is not present in the COCO dataset. All models can detect and recognise “bus” and “car” successfully, except the pre-trained SSD model.

From these examples, it can be observed that the fine-tuned models have a better performance than the pre-trained models most of the time, regardless of the state-of-art methods applied. In this section, the performance between the pre-trained models and their fine-tuned models is compared according to the mAP (Mean Average Precision).

6.3.1.1 Mean Average Precision

The performance of the machine learning models can be compared using mAP values. This project used the PASCAL VOC 2010 detection metric, which takes an IoU of 0.5 when evaluating the quality of the object detection models. This means that the metric only considers a predicted object as true positive when IoU is equal or larger than 0.5 with respect to the ground-truth boundary

box. Figure 6.8 and 6.9 present the AP values for the 40 object classes of the pre-trained and fine-tuned SSD and Faster R-CNN models.

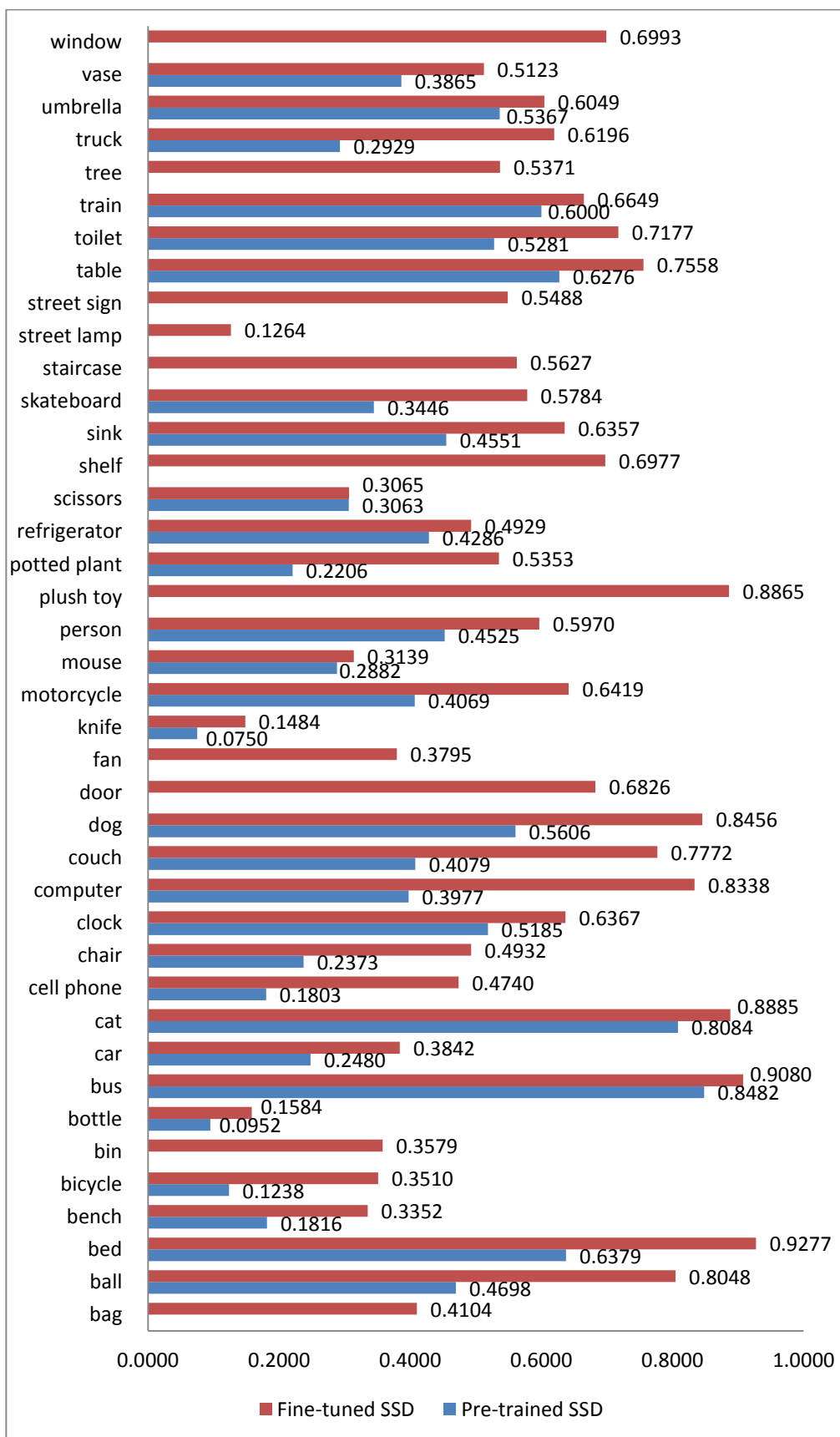


Figure 6.8: AP Values of Pre-trained and Fine-tuned SSD Models

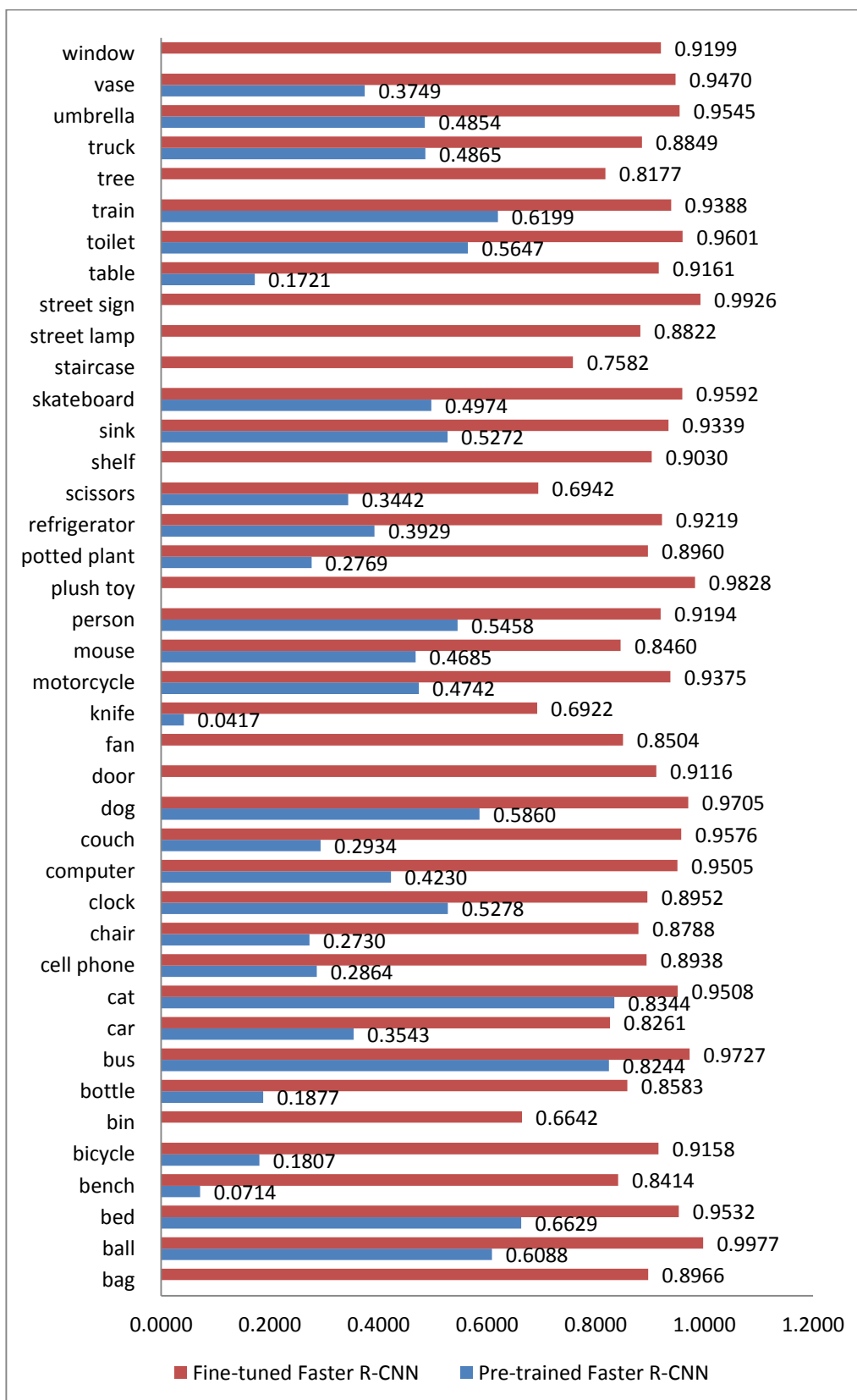


Figure 6.9: AP Values of Pre-trained and Fine-tuned Faster R-CNN Models

Table 6.3: mAP Values of Pre-trained and Fine-tuned SSD and Faster R-CNN Models

Object Detection Model	mAP
Pre-trained SSD	0.3942
Fine-tuned SSD	0.5708
Pre-trained Faster R-CNN	0.4271
Fine-tuned Faster R-CNN	0.8961

As shown in Figure 6.8 and Figure 6.9, the AP value of each object class increases significantly after the model undergoes transfer learning, especially the growth rate of the AP value of the Faster R-CNN model is very dramatic.

According to Table 6.3, the mAP score of the fine-tuned SSD model is more than 0.17 higher than its pre-trained model. From Figure 6.8, the AP scores of the pre-trained SSD model fall within a range of 0.075 (“knife”) and 0.8482 (“bus”). All object classes have AP scores that are lower than 0.64 except “cat” (0.8084) and “bus” (0.8482). On the other hand, the lowest AP score of the fine-tuned SSD model is 0.1264 (“street lamp”) and its highest AP score is 0.9277 (“bed”).

The fine-tuned Faster R-CNN model has a mAP value which is higher than that of its pre-trained model by almost 0.47. Before the Faster R-CNN model undergoes transfer learning, the AP scores of the model lie within the range between 0.0417 (“knife”) and 0.8344 (“cat”). Only two classes achieve AP values above 0.8. Those are “bus”(0.8244) and “cat”(0.8344). After the Faster R-CNN model undergoes transfer learning, most of the classes of the fine-tuned model can reach AP values above 0.8, except “bin” (0.6642), “knife” (0.6922), “scissors” (0.6942), and “staircase” (0.7582), thereby achieving a high mAP value that is close to 0.9.

6.3.2 Comparison of Performance between Fine-tuned SSD and Faster R-CNN Models

The performance of the fine-tuned SSD and Faster R-CNN models are compared to find out which state-of-art methods have better accuracy in object

Figure 6.12 visualises the comparison of the total amount of true positives, false negatives, and false positives between the fine-tuned SSD and R-CNN models.

Table 6.4: True Positive, False Negative, and False Positive of Fine-tuned SSD and Faster R-CNN Models

Class	Fine-tuned SSD			Fine-tuned Faster R-CNN		
	TP	FN	FP	TP	FN	FP
bag	26	94	0	108	11	5
ball	26	15	1	41	0	1
bed	54	15	7	67	4	18
bench	15	40	3	44	12	7
bicycle	20	59	3	71	10	6
bin	2	25	0	15	12	2
bus	24	5	1	27	0	10
bottle	31	132	2	133	30	9
car	87	94	17	148	33	13
cat	48	6	8	50	0	3
cell phone	33	54	0	75	12	8
chair	87	150	14	202	30	22
clock	29	25	1	48	6	2
computer	99	37	12	129	6	9
couch	42	13	16	53	2	13
dog	51	14	5	66	2	6
door	19	23	0	38	2	11
fan	2	7	0	9	2	2
knife	6	34	0	26	14	4
motorcycle	34	24	8	53	7	6
mouse	21	43	1	50	12	2
person	602	560	55	1072	102	98
plush toy	49	8	9	55	0	0
potted plant	34	47	3	74	5	6
refrigerator	7	6	0	13	0	3
street lamp	3	46	0	41	7	9
scissors	8	34	2	33	7	9
shelf	107	65	15	153	18	28
sink	21	21	1	39	2	1
staircase	8	13	2	14	4	1
skateboard	37	43	0	76	3	1
street sign	18	25	2	43	0	2

table	101	62	8	153	8	42
toilet	24	15	3	36	2	2
tree	49	66	3	94	21	19
truck	23	24	4	43	2	5
train	18	7	4	23	1	7
umbrella	21	23	4	42	2	1
vase	39	43	1	78	4	8
window	57	50	5	102	6	17

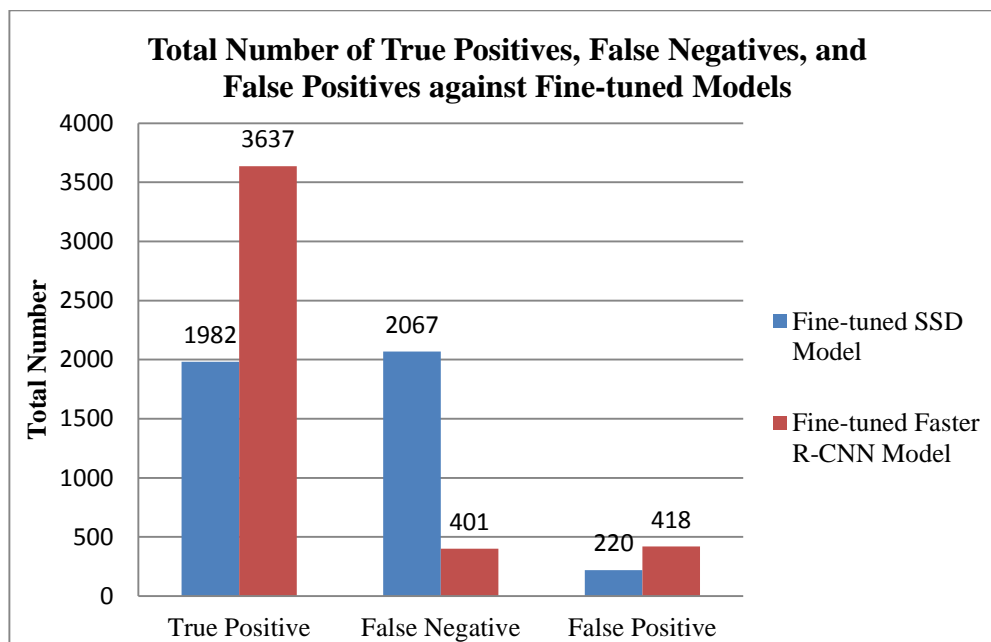


Figure 6.12: Total Number of True Positives, False Negatives, and False Positives against Fine-tuned Models

According to Figure 6.12, the total number of true positives of the fine-tuned Faster R-CNN model is more than 2600 higher than that of the fine-tuned SSD model. True positive is the condition in which the model predicts the class of an object correctly. The higher the total number of true positives, the better the model performance. This means that the fine-tuned Faster R-CNN model is able to classify the objects in the scene more accurately compared to the fine-tuned SSD model.

Apart from that, Figure 6.12 also shows that the total number of false negatives of the fine-tuned SSD model is almost five times higher than that of the fine-tuned Faster R-CNN model. False negative indicates a condition in which the ground-truth exists in a scene but the model fails to detect and

recognise this object. The number of false negatives should be as few as possible to achieve good accuracy. This shows that the fine-tuned SSD model is more likely to fail to detect and recognise objects present in the scene compared with the fine-tuned Faster R-CNN model.

Compared with the fine-tuned SSD model, the fine-tuned Faster R-CNN model has a better performance in the total number of true positives and false negatives. However, when comparing the total number of false positives, the performance of the fine-tuned Faster R-CNN model is worse than the fine-tuned SSD model. The total number of false positives of the fine-tuned Faster R-CNN model is almost twice more than that of the fine-tuned SSD model. False positive is the condition that a model labels an object in the scene when the object really does not exist in that scene, so the occurrence of this condition should be as few as possible. Figure 6.12 shows that the fine-tuned Faster R-CNN model tends to make this kind of errors more often compared to the fine-tuned SSD model.

6.3.2.3 Precision and Recall

After completing the transfer learning, the precision and recall of the fine-tuned SSD and Faster R-CNN models are calculated to measure the quality of the models. Precision measures the ratio of correct recognitions and detections (Alsing, 2018). Recall measures how good is a model in finding true positive (Alsing, 2018). Their formulas are shown below.

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

The precision and recall values for 40 classes of each fine-tuned model are recorded in Table 6.5 below.

Table 6.5: Precision and Recall Values for 40 Classes of Fine-tuned SSD and Faster R-CNN Models

Class	Precision		Recall	
	Fine-tuned SSD	Fine-tuned Faster R-CNN	Fine-tuned SSD	Fine-tuned Faster R-CNN
bag	1.0000	0.9558	0.2167	0.9000
ball	0.9630	0.9762	0.6341	1.0000
bed	0.8852	0.7882	0.7606	0.9437
bench	0.8333	0.8627	0.2679	0.7857
bicycle	0.8696	0.9221	0.2469	0.8765
bin	1.0000	0.8824	0.0714	0.5357
bus	0.9600	0.7297	0.8276	0.9310
bottle	0.9394	0.9366	0.1890	0.8110
car	0.8365	0.9193	0.4754	0.8087
cat	0.8571	0.9434	0.8889	0.9259
cell phone	1.0000	0.9036	0.3793	0.8621
chair	0.8614	0.9018	0.3580	0.8313
clock	0.9667	0.9600	0.5370	0.8889
computer	0.8919	0.9348	0.7279	0.9485
couch	0.7241	0.8030	0.7500	0.9464
dog	0.9107	0.9167	0.7500	0.9706
door	1.0000	0.7755	0.4524	0.9048
fan	1.0000	0.8182	0.1818	0.8182
knife	1.0000	0.8667	0.1500	0.6500
motorcycle	0.8095	0.8983	0.5667	0.8833
mouse	0.9545	0.9615	0.3281	0.7813
person	0.9163	0.9162	0.5123	0.9123
plush toy	0.8448	1.0000	0.8596	0.9649
potted plant	0.9189	0.9250	0.4198	0.9136
refrigerator	1.0000	0.8125	0.5000	0.9286
street lamp	1.0000	0.8200	0.0612	0.8367
scissors	0.8000	0.7857	0.1905	0.7857
shelf	0.8770	0.8453	0.6221	0.8895
sink	0.9545	0.9750	0.5000	0.9286
staircase	0.8000	0.9333	0.3810	0.6667
skateboard	1.0000	0.9870	0.4625	0.9500
street sign	0.9000	0.9556	0.4186	1.0000
table	0.9266	0.7846	0.5941	0.9000
toilet	0.8889	0.9474	0.6154	0.9231
tree	0.9423	0.8319	0.4224	0.8103
truck	0.8519	0.8958	0.4340	0.8113

train	0.8182	0.7667	0.7200	0.9200
umbrella	0.8400	0.9767	0.4773	0.9545
vase	0.9750	0.9070	0.4756	0.9512
window	0.9194	0.8571	0.5182	0.9273
Average	0.9109	0.8895	0.4736	0.8745

As evident from Table 6.5, the average precision of the fine-tuned SSD model is slightly higher than the average precision of the fine-tuned Faster R-CNN model, which are 0.9109 and 0.8895 respectively. It is because the fine-tuned SSD model returns a smaller amount of false positives. This means that the fine-tuned SSD model is more able to make correct detection compared to the Faster R-CNN model.

However, the average recall of the fine-tuned SSD model is nearly 0.4 lower than that of the fine-tuned Faster R-CNN model, which are 0.4736 and 0.8745 respectively. It is caused by large number of false negatives returned by the fine-tuned SSD models. Therefore, the fine-tuned SSD model perform poorly when finding the true positive in the scene compared with the fine-tuned Faster R-CNN model.

Many studies (Liu et al., 2019; Zhao et al., 2019; Alsing, 2018; Argawal, 2018) mention that there is an inverse relationship between precision and recall. As the precision increases, the recall decreases or vice versa. So, when a model returns lesser false positives, it also returns lesser true positives. In this case, the higher precision and lower recall of the fine-tuned SSD model can be used to represent this situation. On the contrary, if a model can detect and recognise many true positives, however it will return many false positives too. This is why the fine-tuned Faster R-CNN model has lower precision than the fine-tuned SSD model, but has a higher recall. Thus, striking a balance between precision and recall is very vital. According to the results, the fine-tuned Faster R-CNN performs better than the fine-tuned SSD model in finding a balance between precision and recall.

6.3.2.4 Mean Average Precision

The mean average precision of the fine-tuned SSD and Faster R-CNN model are calculated and compared. As mentioned in the earlier sections, this project

used the PASCAL VOC 2010 detection metric, which takes an IoU of 0.5 when calculating the mAP. Figure 6.13 and Figure 6.14 present the AP and mAP values for the 40 object classes of the fine-tuned SSD and Faster R-CNN models respectively.

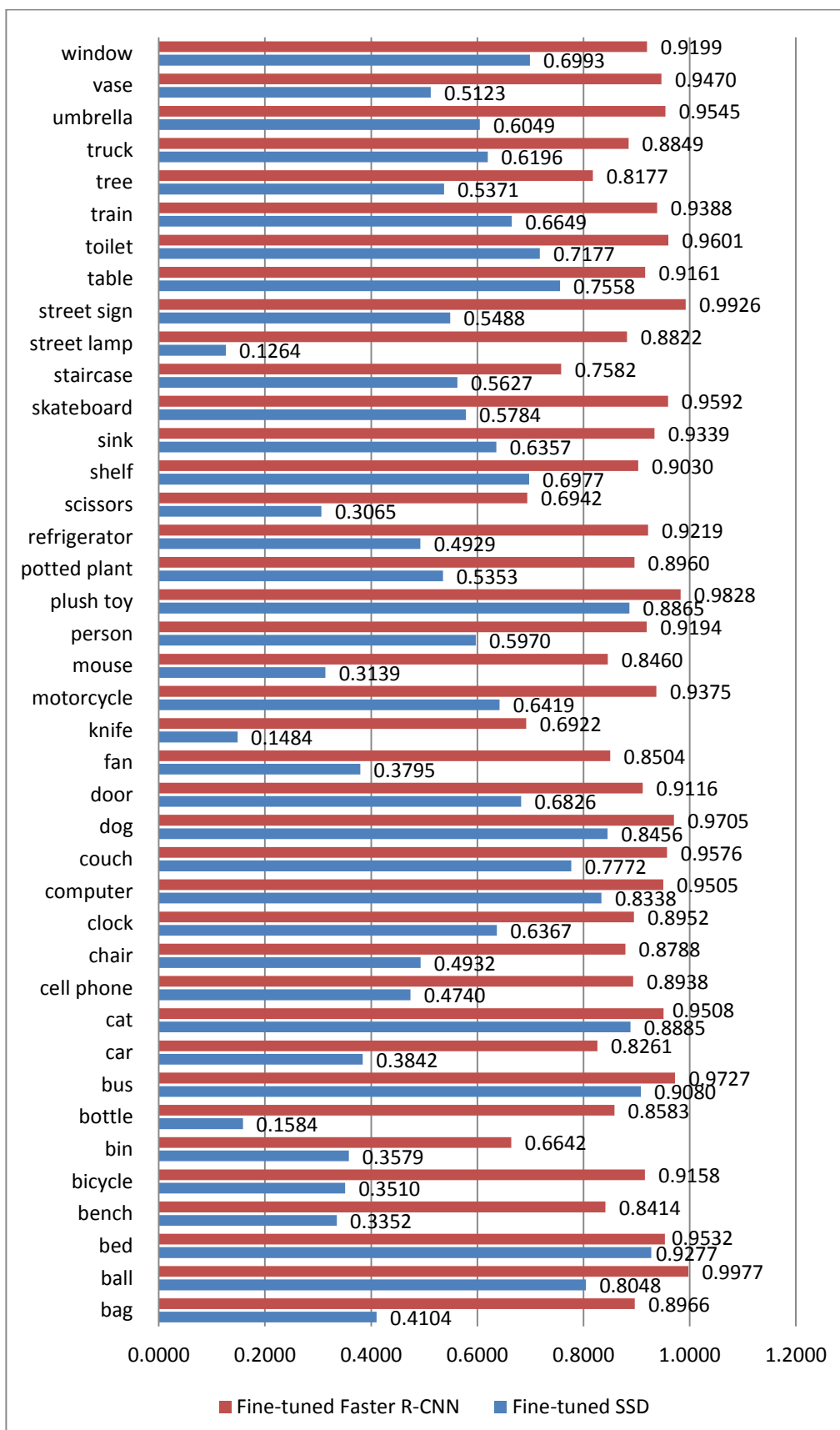


Figure 6.13: Comparison of AP Values for 40 Classes between Fine-tuned SSD and Faster R-CNN Models

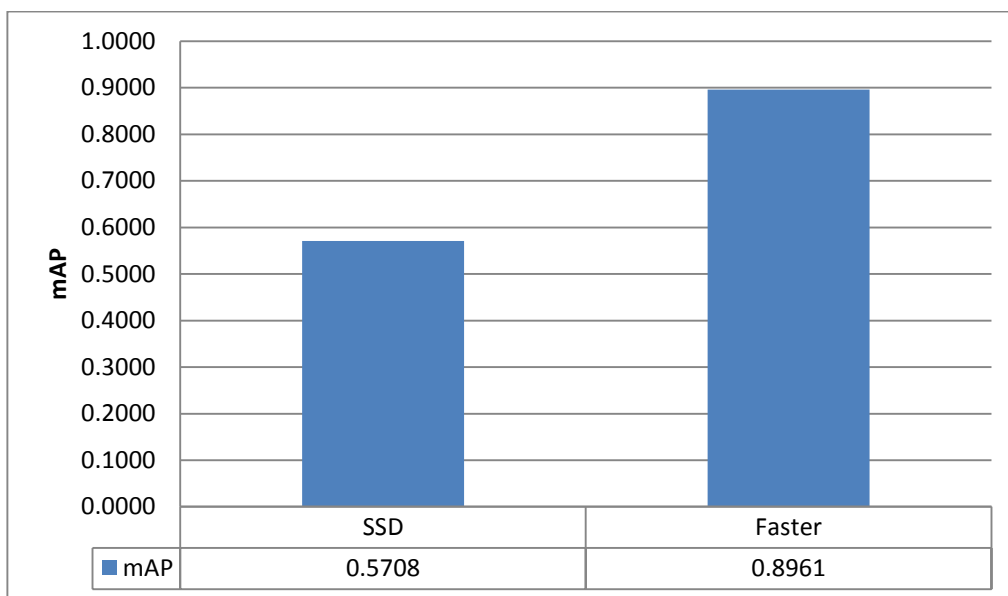


Figure 6.14: Comparison of mAP Values between Fine-tuned SSD and Faster R-CNN Models

It can be seen from Figure 6.13 that the AP values for all 40 object classes of the fine-tuned Faster R-CNN model are higher than that of the fine-tuned SSD model. The AP values of each object class of the fine-tuned Faster R-CNN model fall within the range between 0.6642 and 0.9977. The classes with the lowest and highest AP values are the classes “bin” and “ball” respectively. For the fine-tuned SSD model, it has the lowest AP value of 0.1264 and highest value of 0.9277, which are the object classes “street lamp” and “bed” respectively. Moreover, as displayed in Figure 6.14, the mAP values of the Faster R-CNN model is more than 0.3 higher than that of the fine-tuned SSD models. It can prove that the Faster R-CNN model always has better accuracy than the SSD model.

6.3.2.5 Speed and Accuracy Tradeoff of Different Object Detection and Recognition Models

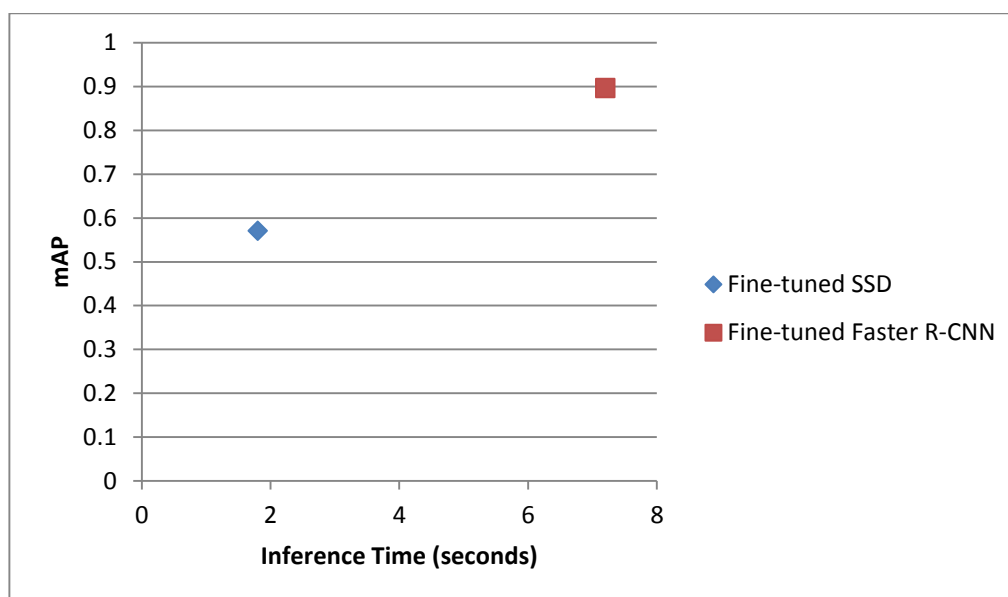


Figure 6.15: Speed and Accuracy Comparison of Fine-tuned Models

The inference time and accuracy of the fine-tuned SSD and Faster R-CNN models are compared in Figure 6.15. The number of seconds required for object detection and recognition of the fine-tuned SSD model is almost five seconds shorter than the fine-tuned Faster R-CNN model, with the number of seconds of 1.8 seconds. Furthermore, the fine-tuned Faster R-CNN model has a better mAP performance than the fine-tuned SSD model, with a mAP score of 0.8961. However, as the mAP values increases, the number of seconds required for inference also increases.

6.4 Summary

As a summary of the chapter, it can be seen that the batch size and types of model are able to affect the training performance. A large batch size consumes a larger amount of memory and requires more time to achieve certain accuracy, while a small batch size causes the model unable to converge within the expected steps. And then, the pre-trained Faster R-CNN model requires a shorter training time than that of the pre-trained SSD model.

In this chapter, the performance between pre-trained and the fine-tuned models was evaluated and compared. Figure 6.16 visualises the mAP values of different pre-trained and fine-tuned models. It can be observed that after transfer learning, the mAP values of the models increase significantly, regardless of their state-of-art method applied.

Other than that, the performance of the fine-tuned SSD and Faster R-CNN model also was compared. As displayed in Figure 6.16, the accuracy of the fine-tuned Faster R-CNN model is much higher than that of the fine-tuned SSD model by almost 0.33. However, due to the trade-off between accuracy and inference time, the fine-tuned Faster R-CNN requires 7.2 seconds of inference time, which is four times as long as that of the fine-tuned SSD model with an inference time of 1.8 seconds.

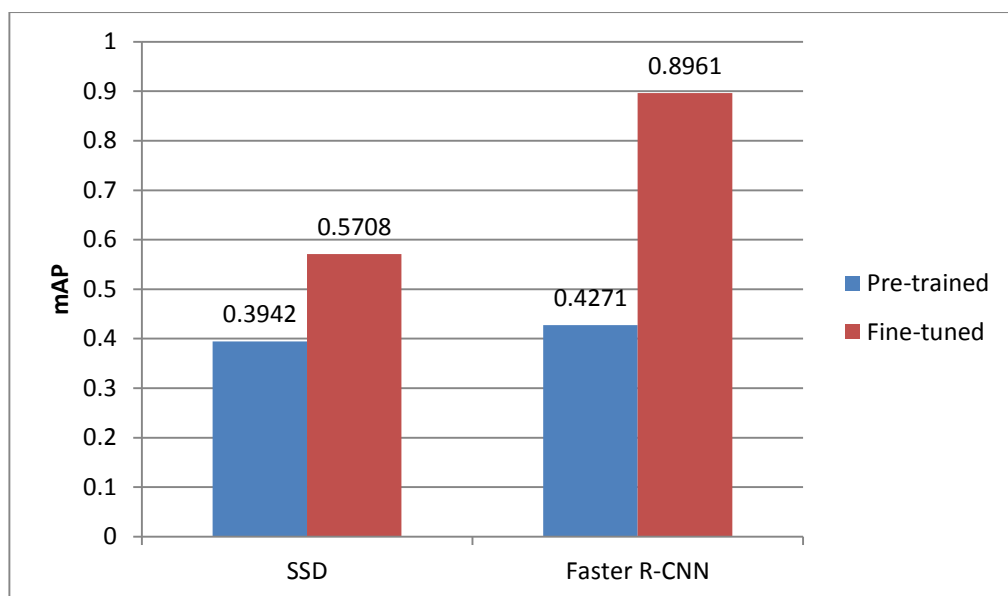


Figure 6.16: Comparison of mAP Values between Different Models

In this project, the application developed is expected to have good accuracy and shorter inference time. Hence, the fine-tuned SSD model was chosen to be implemented in the developed Android application.

CHAPTER 7

SYSTEM TESTING

7.1 Introduction

Testing was carried to ensure the application meets the specifications and works properly. This chapter discusses unit testing and integration testing conducted on the application developed.

7.2 Unit Testing

Before a component is combined with other components in the application, unit testing was carried out on each unit of the component to find out and resolve the bugs. In this project, the application is classified into several modules and unit testing was conducted on each module.

7.2.1 Object Detection and Recognition Module

Table 7.1: Unit Test for Object Detection and Recognition

Test Case	Test Steps	Test Data	Expected Result	Status
Scan user's surroundings	1. Open application	Surrounding scene.	Surroundings scenes are captured and displayed a in continuous stream.	Pass
An object is detected and recognised at a confidence score higher than threshold	1. Capture a scene with one object	A scene with a computer, which returns a confidence score higher than 60%.	The name, boundary box, and confidence score of the detected computer are displayed.	Pass

More than one object is detected and recognised at confidence score higher than threshold	1. Capture a scene with multiple objects	A scene with a cellphone and scissors, which both objects return confidence scores higher than 60%.	The names, boundary boxes, and confidence scores of the detected cellphone and scissors are displayed.	Pass
An object is detected and recognised at a confidence score lower than threshold	1. Capture a scene with an object	A scene with a computer, which returns a confidence score lower than 60%.	No object names, bounding boxes, and confidence scores are displayed.	Pass
Multiple objects are detected and recognised at confidence scores below the threshold, and some are above the threshold	1. Capture a scene with multiple objects	A scene with a cellphone and scissors, which returns a confidence score lower than 60% and higher than 60% respectively.	Only the name, boundary box, and confidence score of the detected scissors are displayed.	Pass
No objects are detected and recognised	1. Capture a scene without objects	A scene with a blank wall.	No object names, bounding boxes, and confidence scores are displayed.	Pass

7.2.2 Speech Feedback Module

Table 7.2: Unit Test for Speech Feedback

Test Case	Test Steps	Test Data	Expected Result	Status
An object on the left is detected and recognised at a confidence score higher than threshold	<ol style="list-style-type: none"> 1. Capture a scene 2. Process the scene captured 	Detection results return a mouse with a midpoint located in the first left 1/3 area of the screen and a confidence score higher than 60%.	“Mouse on the left” is spoken out.	Pass
An object at the middle is detected and recognised at a confidence score higher than threshold	<ol style="list-style-type: none"> 1. Capture a scene 2. Process the scene captured 	Detection results return a mouse with a midpoint located between first left 1/3 and first right 1/3 area of the screen and a confidence score higher than 60%.	“Mouse is straight forward” is spoken out.	Pass
An object on the right is detected and recognised at a confidence score higher than threshold	<ol style="list-style-type: none"> 1. Capture a scene 2. Process the scene captured 	Detection results return a mouse with a midpoint located in the first right 1/3 area of the screen and a confidence score higher than 60%.	“Mouse on the right” is spoken out.	Pass
An object is detected and recognised at a confidence score lower than threshold	<ol style="list-style-type: none"> 1. Capture a scene 2. Process the scene captured 	Detection results return a mouse with a midpoint located in the first left 1/3 area of the screen and a confidence score lower than 60%.	No object name and direction is spoken out.	Pass

More than one object is detected and recognised at confidence scores higher than threshold	1. Capture a scene 2. Process the scene captured	Detection results return a mouse and scissors with midpoints located in the first left 1/3 and the first right 1/3 area of the screen respectively, and both confidence scores higher than 60%.	“Mouse on the left and scissors on the right” is spoken out.	Pass
Multiple objects are detected and recognised at confidence scores below the threshold, and some are above the threshold	1. Capture a scene 2. Process the scene captured	Detection results return a mouse and scissors with midpoints located in the first left 1/3 and the first right 1/3 area of the screen respectively, and their confidence scores are lower and higher than 60% respectively.	“Scissors on the right” is spoken out.	Pass
No objects are detected and recognised	1. Capture a scene 2. Process the scene captured	Detection results return null.	No object name and direction is spoken out.	Pass

Identical scenes are detected and recognised for a few seconds	<ol style="list-style-type: none"> 1. Capture multiple similar scenes 2. Process several similar scenes 	Detection results return a mouse with a midpoint located in the first left 1/3 area of the screen and a confidence score higher than 60%.	“Mouse on the left” is spoken out every 3 seconds.	Pass
--	---	---	--	------

7.2.3 Set Speech Rate Module

Table 7.3: Unit Test for Set Speech Rate

Test Case	Test Steps	Test Data	Expected Result	Status
Adjust speech rate to slow	<ol style="list-style-type: none"> 1. Click the “Adjust Speech Rate” button 2. Click the “Slow” button 		The speech rate is adjusted to 0.5 (slow).	Pass
Adjust speech rate to normal	<ol style="list-style-type: none"> 1. Click the “Adjust Speech Rate” button 2. Click the “Normal” button 		The speech rate is adjusted to 1.0 (normal).	Pass
Adjust speech rate to fast	<ol style="list-style-type: none"> 1. Click the “Adjust Speech Rate” button 2. Click the “Fast” button 		The speech rate is adjusted to 1.5 (fast).	Pass
Do not select a speech rate	<ol style="list-style-type: none"> 1. Click the “Adjust Speech Rate” button 2. Click the phone’s exit button without selecting a speech rate 	The existing speech rate is normal.	The speech rate remains at 1.0 (normal).	Pass

7.2.4 Play/Stop Speech Module

Table 7.4: Unit Test for Play and Stop Speech

Test Case	Test Steps	Test Data	Expected Result	Status
Stop Speech	1. Click the “Play/Stop Speech” button		Voice feedback is stopped.	
Play Speech	1. Click the “Play/Stop Speech” button 2. Click the “Play/Stop Speech” button one more time		Voice feedback is available.	

7.3 Integration Test

Integration testing was performed after unit testing was completed. The main goal of integration testing is to identify the existence of bugs after integrating the modules tested in unit testing.

Table 7.5: Integration Test

Test Case	Test Steps	Expected Result	Status
Detect and recognise objects in the scene	1. Open application 2. Scan the surrounding using the phone’s camera	<ul style="list-style-type: none"> The names, boundary boxes, and confidence scores of each object are displayed on the screen. The names and the directions of the objects are spoken out every 3 seconds. 	Pass
No object is detected and recognised in the scene	1. Open application 2. Scan the surrounding using the phone’s camera	<ul style="list-style-type: none"> No object name, boundary box, and confidence score are displayed on the screen. No object names and direction are spoken out. 	Pass

Stop speech feedback	<ol style="list-style-type: none"> 1. Open application 2. Scan the surrounding using the phone's camera 3. Click the "Play/Stop Speech" button 	<ul style="list-style-type: none"> • The names, boundary boxes, and confidence scores of each object is displayed on the screen. • The names and the directions of the objects are not spoken out. 	Pass
Play speech feedback	<ol style="list-style-type: none"> 1. Open application 2. Scan the surrounding using the phone's camera 3. Click the "Play/Stop Speech" button 4. Click the "Play/Stop Speech" button again 	<ul style="list-style-type: none"> • The names, boundary boxes, and confidence scores of each object are displayed on the screen. • The names and the directions of the objects are spoken out after pressing the "Play/Stop Speech" button twice. • The speech rate is the same as the speech rate before clicking the "Play/Stop Speech" button. 	Pass
Adjust speech rate to slow	<ol style="list-style-type: none"> 1. Open application 2. Click the "Adjust Speech Rate" button 3. Click the "Slow" button 4. Scan the surrounding using the phone's camera 	<ul style="list-style-type: none"> • The speech rate in the shared preferences is updated to 0.5. • The message "speech rate is slow" is spoken out verbally. • The names and the directions of the objects are spoken out every 3 seconds with a speech rate of 0.5 (slow). 	Pass

Adjust speech rate to normal	<ol style="list-style-type: none"> 1. Open application 2. Click the “Adjust Speech Rate” button 3. Click the “Normal” button 4. Scan the surrounding using the phone’s camera 	<ul style="list-style-type: none"> • The speech rate in the shared preferences is updated to 1.0. • The message “speech rate is normal” is spoken out verbally. • The names and the directions of the objects are spoken out every 3 seconds with a speech rate of 1.0 (normal). 	Pass
Adjust speech rate to fast	<ol style="list-style-type: none"> 1. Open application 2. Click the “Adjust Speech Rate” button 3. Click the “Fast” button 4. Scan the surrounding using the phone’s camera 	<ul style="list-style-type: none"> • The speech rate in the shared preferences is updated to 1.5. • The message “speech rate is fast” is spoken out verbally. • The names and the directions of the objects are spoken out every 3 seconds with a speech rate of 1.5 (fast). 	Pass
Do not select a speech rate on the Adjust Speech Rate page	<ol style="list-style-type: none"> 1. Open application 2. Click the “Adjust Speech Rate” button 3. Click the phone’s exit button without selecting a speech rate 4. Scan the surrounding using the phone’s camera 	<ul style="list-style-type: none"> • The speech rate in the shared preferences is not updated. • The speech rate remains unchanged and the names and the directions of the objects are spoken out every 3 seconds with this speech rate. 	Pass

7.4 Usability Test

Usability test involves real-world users to test and evaluate the application. Its goal is to identify whether the system is suited to real-world scenarios. Due to the Cov-19 pandemic and standard operating procedures enforced by the Malaysian government to control the spread of the Cov-19, the difficulty of reaching out to visually impaired people has increased. Hence, there is a limited number of testers. In this project, a usability test has been conducted by two users including the author's grandparents, who are suffering from vision loss.

The users require to answer a list of questions after testing the application. For every question, the users need to give a score between 1 and 5, which 1 means strongly disagree while 5 means strongly agree.

Table 7.6: Usability Test Results

Questions	User 1	User 2
The application is helpful in identifying objects in surrounding.	3	4
The application identifies objects in surrounding accurately.	2	3
The application identifies objects in surrounding at a fast speed.	4	3
The voice feedback is clear and easy to understand.	3	4
The application is unnecessarily complex.	1	2
The application has a familiar user interface.	4	3
The application is easy to use.	4	4
I can learn to use this application quickly.	4	5
I do not need to remember many things to use the application.	3	4

I feel more confident when navigating using the application.	3	4
What did you like best about the application?	Easy to use, simple user interface	Good concept to help person with vision problem. No need Wifi.
What did you like least about the application?	Voice feedback is robotic. The application detects objects that are very far away sometimes, lead to confusion.	Hard to identify objects when the camera is shaking. Sometime detect things wrongly.
How would you describe this application to a colleague?	It is okay, but still need improvement before a person with low vision can fully rely on the application.	Nice application to help people in need.
Final comments	Hope it can identify all objects in daily lives and provide more detailed descriptions such as object colour and its distance.	The application should support other languages because not all people learn English. Hope that the application can use less time to detect thing and the time taken for detection is not affected by camera shaking.

CHAPTER 8

CONCLUSIONS

8.1 Introduction

This chapter discusses the conclusions, challenges and future enhancements of the project.

8.2 Conclusions

After six months, the project has accomplished the planned objectives successfully. Two different pre-trained models, which are SSD Mobilenet V1 COCO and Faster R-CNN Inception V2 COCO were chosen to implement transfer learning. By using transfer learning, this project has retrained the pre-trained models to detect and recognise 40 classes. Most of the trained object classes are the obstacles that visually impaired people may encounter when navigating. The train and test datasets that include these 40 classes were collected and labelled for model training. After transfer learning, the performance of models before and after transfer learning was compared based on mAP, confusion matrix, recall, and precision. Moreover, the speed-accuracy trade-off of the models trained was studied. Finally, the SSD fine-tuned model was selected to be implemented on the mobile application due to its good accuracy and short inference time.

An object detection and recognition mobile application has been developed using Tensorflow Object Detection API, which provides an interface for communication between the application and the trained SSD model. This mobile application provides contributions to the visually impaired community. It provides a cost-effective way to recognise the types of objects in the surroundings. Visually impaired people just need a smartphone to use this application, without a need to buy other special hardware. A smartphone is cheaper and more accessible compared to other advanced technologies. And then, they are allowed to use the application anywhere, anytime, without an Internet connection.

Furthermore, this application assists visually impaired people to carry out daily activities and navigate more freely and safely. Visually impaired people can be informed of the names and directions of the detected objects in the surroundings through voice feedback of the smartphone's speaker or earphone. The direction of the object is classified into left, middle, or right. Hence, they do not need to guess the type of objects in the surrounding and this prevents injuries due to making wrong assumptions. Apart from that, visually impaired people are able to stop and play the voice feedback if necessary. Besides, they can also adjust the frequency of speech to a comfortable level, so they can understand the feedback more clearly.

8.3 Challenges

There are few challenges encountered when developing this project. The challenges are listed below:

- i. Required a lot of self-learning to pick up Tensorflow and setup the Tensorflow environment.
- ii. Lack of powerful computing resources to train the object detection model.
- iii. Required a lot of time to label the images manually for transfer learning.

8.4 Future Enhancements

A future enhancement of the project is planned. Although the additional features are not available in the current application, they are useful for visually impaired people and will be added to the application in the future. The future enhancements are listed as following:

- i. Recognise text and currency.
- ii. Provide multiple languages such as Chinese, Germany and French.
- iii. Develop an iOS version of the application.

REFERENCES

- Abdul Malik Shaari and Nur Safwati, 2017. Position and Orientation Detection of Stored Object Using RFID Tags. *Procedia Engineering*, [e-journal] 184, pp.708-715. <http://dx.doi.org/10.1016/j.proeng.2017.04.146>.
- Alsing, O., 2018. *Mobile Object Detection using TensorFlow Lite and Transfer Learning*. Degree, Kth Royal Institute of Technology School of Electrical Engineering and Computer Science. Available at: <https://www.diva-portal.org/smash/get/diva2:1242627/FULLTEXT01.pdf> [Accessed 14 July 2020].
- Anitha, J., Subalaxmi, A. and Vijayalakshmi, G., 2019. Real Time Object Detection for Visually Challenged Persons. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, [online] Available at: <https://www.ijitee.org/wp-content/uploads/papers/v8i8/H6339068819.pdf> [Accessed 23 July 2020].
- Argawal, K., 2018. *Object Detection in Refrigerators using Tensorflow*. B. Tech, Uttar Pradesh Technical University. Available at: <http://venus.library.uvic.ca/handle/1828/10464> [Accessed 3 July 2020].
- Awad, M., Haddad, J.E., Khneisser, E., Mahmoud, T., Yaacoub, E. and Malli, M., 2018. Intelligent eye: A mobile application for assisting blind people. *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, [e-journal] pp.1 - 6. <http://dx.doi.org/10.1109/menacomm.2018.8371005>.
- Badave, A., Jagtap, R., Kaovasia, R., Rahatwad, S. and Kulkarni, S., 2020. Android Based Object Detection System for Visually Impaired. *2020 International Conference on Industry 4.0 Technology (I4Tech)*, [e-journal] pp. 34-38. <http://dx.doi.org/10.1109/i4tech48345.2020.9102694>.
- BAWA, 2020. *A New Era of Accessibility*. [online] Available at: <https://www.bawa.tech/> [Accessed 23 February 2021].
- Chanana, P., Paul, R., Balakrishnan, M. and Raol, P.V.M., 2017. Assistive technology solutions for aiding travel of pedestrians with visual impairment. *Journal of Rehabilitation and Assistive Technologies Engineering*, [e-journal] 4, pp.1 - 16. <http://dx.doi.org/10.1177/2055668317725993>.
- COCO, 2020. *COCO: Common Objects in Context*. [online] Available at: <https://cocodataset.org/#home> [Accessed 12 December 2020].

Developers, 2020a. *Android Studio*. [online] Available at: <<https://developer.android.com/studio>> [Accessed 16 August 2020].

Developers, 2020b. *Camera API*. [online] Available at: <<https://developer.android.com/guide/topics/media/camera>> [Accessed 16 August 2020].

dos Santos, A.D.P., Medola, F.O., Cinelli, M.J., Garcia Ramirez, A.R. and Sandnes, F.E., 2020. Are electronic white canes better than traditional canes? A comparative study with blind and blindfolded participants. *Universal Access in the Information Society* (2020), [e-journal] <http://dx.doi.org/10.1007/s10209-020-00712-z>.

Felix, S.M., Kumar, S. and Veeramuthu, A., 2018. A Smart Personal AI Assistant for Visually Impaired People. *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, [e-journal] pp. 1245-1250. <http://dx.doi.org/10.1109/ICOEI.2018.8553750>.

Gao, Y. and Mosalam, K.M., 2018. Deep Transfer Learning for Image-Based Structural Damage Recognition. *Computer-Aided Civil and Infrastructure Engineering*, [e-journal] 33(9), pp.1 - 21. <https://doi.org/10.1111/mice.12363>.

Garrido, G. and Joshi, P., 2018. *OpenCV 3.x with Python By Example*. 2nd ed. Birmingham: Packt Publishing Ltd.

Google Cloud, 2020. *Detect multiple objects*. [online] Available at: <https://cloud.google.com/vision/docs/object-localizer#vision_localize_objects-java> [Accessed 17 July 2020].

IBM, 2020. *Advantages of XML*. [online] Available at: <https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_73/rzamj/rzajmintroadvantages.htm> [Accessed 18 August 2020].

Industries For The Blind And Visually Impaired, 2020. *The White Cane and its Meaning*. [online] Available at: <<https://ibvi.org/blog/the-white-cane-and-its-meaning/#:~:text=Use%20of%20the%20white%20cane,easier%20for%20others%20to%20see>> [Accessed 13 July 2020].

IrisVision Global, 2021. *Top 5 Electronic Glasses for the Blind and Visually Impaired*. [online] Available at: <<https://irisvision.com/electronic-glasses-for-the-blind-and-visually-impaired/>> [Accessed 23 February 2021].

Jakhete, S.A., Bagmar, P., Dorle, A., Rajurkar, A. and Pimplikar, P., 2019. Object Recognition App for Visually Impaired. *2019 IEEE Pune Section International Conference (PuneCon)*, [e-journal] pp.1 - 4. <http://dx.doi.org/10.1109/PuneCon46936.2019.9105670>.

Khalid, H., 2018. *Difference Between Evolutionary Prototyping and Throw-away Prototyping*. *Prototype Info Sharing Blog*, [blog] 4 January. Available at: <<https://prototypeinfo.com/evolutionary-prototyping-and-throw-away-prototyping/>> [Accessed 25 July 2020].

Khan, I., Khusro, S. and Ullah, I., 2018. Technology-assisted white cane: evaluation and future directions. *PeerJ*, [e-journal] pp.1- 27. <http://dx.doi.org/10.7717/peerj.6058>.

Khan Shishir, M.A., Rashid Fahim, S., Habib, F.M. and Farah, T., 2019. Eye Assistant: Using mobile application to help the visually impaired. *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, [e-journal]. <http://dx.doi.org/10.1109/icasert.2019.8934448>.

Li, Z. and Zhang, R., 2017. *Object Detection and Its Implementation on Android Devices*. [online] Available at: <http://cs231n.stanford.edu/reports/2017/pdfs/627.pdf> [Accessed 17 July 2020].

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X. and Pietikäinen, M., 2019. Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, [e-journal] 128, pp. 261-318. <http://dx.doi.org/10.1007/s11263-019-01247-4>.

Lobo, S., 2017. *Is Facebook-backed PyTorch better than Google's TensorFlow?* [online] Available at: <<https://hub.packtpub.com/dl-wars-pytorch-vs-tensorflow/>> [Accessed 13 July 2020].

Marcelino, P., 2018. *Transfer learning from pre-trained models*. [online] Available at: <<https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751>> [Accessed 26 January 2021].

Michelucci, U., 2019. *Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection*. Dübendorf: Apress. Available at: <<https://doi.org/10.1007/978-1-4842-4976-5>> [Accessed 13 July 2020].

National Council For The Blind Malaysia, 2020. *What is Blindness?* [online] Available at: <<http://ncbm.org.my/index/understanding-the-blind/>> [Accessed 13 July 2020].

Pan, S.J. and Yang, Q., 2009. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, [e-journal] 22(10), pp.1345 – 1359. <http://dx.doi.org/10.1109/TKDE.2009.191>.

Parikh, N., Shah, I. and Vahora, S., 2018. Android Smartphone Based Visual Object Recognition for Visually Impaired Using Deep Learning. *2018 International Conference on Communication and Signal Processing (ICCSP)*, [e-journal] pp.420 - 425. <http://dx.doi.org/10.1109/ICCSP.2018.8524493>.

Patil, O., and Gaikwad, V., 2018. Classification of Vegetables using TensorFlow. *International Journal for Research in Applied Science & Engineering Technology*, [e-journal] 6(4), pp. 2926-2934. <http://doi.org/10.22214/ijraset.2018.4488>.

PyTorch, 2020. *Torchvision.models*. [online] Available at: <<https://pytorch.org/docs/stable/torchvision/models.html#object-detection-instance-segmentation-and-person-keypoint-detection>> [Accessed 16 July 2020].

Rajwani, R., Purswani, D., Kalinani, P., Ramchandani, D. and Dokare, I., 2018. Proposed System on Object Detection for Visually Impaired. *International Journal of Information Technology (IJIT)*, [online] Available at: <<http://www.ijitjournal.org/volume-4/issue-2/IJIT-V4I2P1.pdf>> [Accessed 23 July 2020].

Rane, M., Patil, A. and Barse, B., 2019. Real Object Detection Using TensorFlow. *International Conference on Communications and Cyber-Physical Engineering (ICCCE)*, [e-journal] pp.39 - 45. http://dx.doi.org/10.1007/978-981-13-8715-9_5.

Ruedeenniraman, N., Ikeda, M. and Barolli, L., 2017. TensorFlow: A Vegetable Classification System and Its Performance Evaluation. *Innovative Mobile and Internet Services in Ubiquitous Computing*, [e-journal] 994, pp. 132-141. <https://doi.org/10.1007/978-3-030-22263-5>.

Sarkar, D., 2018. *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning*. [online] Available at: <<https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>> [Accessed 26 January 2021].

Sharma, S., Gupta, M., Kumar, A., Tripathi, M. and Gaur, M.S., 2017. Multiple distance sensors based smart stick for visually impaired people. 2017 *IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, [e-journal]. <http://dx.doi.org/10.1109/CCWC.2017.7868407>.

Sun, J., Radecka, K. and Zilic, Z., 2019. FoodTracker: A Real-time Food Detection Mobile Application by Deep Convolutional Neural Networks. *The 16th International Conference on Machine Vision Applications*, [online] Available at: <https://arxiv.org/pdf/1909.05994.pdf> [Accessed 7 July 2020].

Tensorflow, 2020. *Deploy machine learning models on mobile and IoT devices*. [online] Available at: <<https://www.tensorflow.org/lite/>> [Accessed 19 August 2020].

Thakare, P.U., Shubham, K., Ankit, P., Ajinkya, R. and Om, S., 2017. Smart Assistance System for the Visually Impaired. *International Journal of Scientific and Research Publications*, [online] Available at: <<http://www.ijsrp.org/research-paper-1217/ijsrp-p7254.pdf>> [Accessed 24 July 2020].

The Star, 2018. Mobile cellular penetration reaches 131.8%. *The Star*, [online] 14 February. Available at: <<https://www.thestar.com.my/business/business-news/2018/02/14/mobile-cellular-penetration-reaches-1318/>> [Accessed 1 July 2020].

Tutorials Point, 2020. *Java Tutorial*. [online] Available at: <<https://www.tutorialspoint.com/java/index.htm>> [Accessed 18 August 2020].

Vaidya, S., Shah, N., Shah, N. and Shankarmani, R., 2020. Real-Time Object Detection for Visually Challenged People. *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, [e-journal] pp.311 - 316. <http://dx.doi.org/10.1109/ICICCS48265.2020.9121085>.

Weiss, K., Khoshgoftaar, T.M. and Wang, D.D., 2016. A survey of transfer learning. *Journal of Big Data*, [e-journal] 3(9), pp. 1 - 40. <https://doi.org/10.1186/s40537-016-0043-6>.

Wewalk, 2020. *Revolutionary Smart Cane and Smartphone App*. [online] Available at: <<https://wewalk.io/en/>> [Accessed 23 February 2021].

World Health Organization, 2019. *Blindness and vision impairment*. [online] Available at: <<https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>> [Accessed 30 June 2020].

Zhang, X., Yang, L., and Sinnott, R., 2019. A Mobile Application for Cat Detection and Breed Recognition Based on Deep Learning. *2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile)*, [e-journal] pp. 7-12. <http://doi.org/10.1109/AI4Mobile.2019.8672684>.

Zhao, Z.Q., Zheng, P., Xu, S.T. and Wu, X., 2019. Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, [e-journal] 30(11), pp. 1-21. <http://dx.doi.org/10.1109/TNNLS.2018.2876865>.

APPENDICES

APPENDIX A: Work Breakdown Structure

Object Detection and Recognition Mobile Application

1.0 Project Planning		0%		Start	Due	Assigned
1.1	Create gantt chart	0%	<input type="text"/>	9 Jul 20	9 Jul 20	
1.2	Gather requirements	0%	<input type="text"/>	22 Jun 20	15 Jul 20	
1.3	Define problem statement	0%	<input type="text"/>	16 Jul 20	21 Jul 20	
1.4	Define goal and objectives	0%	<input type="text"/>	22 Jul 20	24 Jul 20	
1.5	Propose solution	0%	<input type="text"/>	25 Jul 20	28 Jul 20	
1.6	Propose approach	0%	<input type="text"/>	29 Jul 20	30 Jul 20	
1.7	Prepare preliminary report	0%	<input type="text"/>	31 Jul 20	3 Aug 20	
2.0 Literature Review		0%		Start	Due	Assigned
2.1	Object detection and object recognition	0%	<input type="text"/>	3 Aug 20	Yesterday	
2.2	Existing object detection and recognition	0%	<input type="text"/>	3 Aug 20	14 Aug 20	
2.3	API-based object detection and recognition	0%	<input type="text"/>	13 Aug 20	17 Aug 20	
3.0 Development		0%		Start	Due	Assigned
3.1 Iteration 1		0%				
3.1.1	Design	0%	<input type="text"/>	27 Jul 20	4 Aug 20	
3.1.2	Prototyping	0%	<input type="text"/>	5 Aug 20	27 Aug 20	
3.1.3	Evaluation and review	0%	<input type="text"/>	27 Aug 20	5 Sep 20	
Iteration 2		0%				
3.2.1	Design	0%	<input type="text"/>	11 Jan 21	13 Jan 21	
3.2.2	Prototyping	0%	<input type="text"/>	14 Jan 21	1 Feb 21	
3.2.3	Evaluation and review	0%	<input type="text"/>	2 Feb 21	9 Feb 21	
Iteration 3		0%				
3.3.1	Design	0%	<input type="text"/>	10 Feb 21	12 Feb 21	
3.3.2	Prototyping	0%	<input type="text"/>	13 Feb 21	3 Mar 21	
3.3.3	Evaluation and review	0%	<input type="text"/>	4 Mar 21	8 Mar 21	
4.0 Testing		0%		Start	Due	Assigned
4.1	Unit testing	0%	<input type="text"/>	9 Mar 21	13 Mar 21	
4.2	Integration testing	0%	<input type="text"/>	15 Mar 21	19 Mar 21	
4.3	System testing	0%	<input type="text"/>	20 Mar 21	24 Mar 21	
5.0 Deployment		0%		Start	Due	Assigned
5.1	Prepare final report	0%	<input type="text"/>	24 Mar 21	2 Apr 21	

[View in Gantt](#)

APPENDIX B: Gantt Chart

Object Detection and Recog...

