SKIN LESION DETECTION USING DEEP NEURAL NETWORK BY SMART HANDHELD DEVICES

TAN HOU REN

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

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

September 2020

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	:	[illegible]
Name	:	TAN HOU REN
ID No.	:	17UEB00832
Date	:	4th October 2020

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"SKIN LESION DETECTION USING DEEP NEURAL NETWORK BY SMART HANDHELD DEVICES"** was prepared by **TAN HOU REN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Biomedical Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Ir. Dr. Hum Yan Chai
Date	:	4/10/2020

Signature	:	[illegible]
Co-Supervisor	:	Dr. Tee Yee Kai
Date	:	4/10/2020

ABSTRACT

Early detection of malignant skin lesions improves patient survival rates. Conventional self-detection method for public possess subjectivity, inaccuracy, and require experience. The goal of this project is to develop an Android based mobile application with object detection deep learning integration that allows global users to perform malignant skin lesions self-detection easily using a smartphone, for overcoming the limitations of the conventional method. Transfer Learning has been performed on various object detection models using ISIC skin lesions dataset with TensorFlow Object Detection API. The selected object detection model is SSD MobileNet V2 with 93.9% of evaluation accuracy after training due to its lightweight architecture therefore suitable for smartphone integration. The selected model has surpassed existing classification model in terms of accuracy after validation with a new dataset. A mobile application has been developed successfully with Android Studio. The trained object detection model successfully integrated into the mobile application using Firebase ML Kit and has achieved low detection time on smartphones. The mobile application has been proven to be compatible with various Android versions and screen sizes after tested with 7 different smartphones using Firebase Test Lab.

TABLE OF CONTENTS

DECLARATION **i**
APPROVAL FOR SUBMISSION **ii**
ABSTRACT **iv**
TABLE OF CONTENTS **v**
LIST OF TABLES **vii**
LIST OF FIGURES **ix**
LIST OF SYMBOLS / ABBREVIATIONS **xii**
LIST OF APPENDICES **xiii**

CHAPTER

1 **INTRODUCTION** **1**
1.1 General Introduction 1
1.2 Importance of the Study 1
1.3 Problem Statement 2
1.4 Aim and Objectives 2
1.5 Scope and Limitation of the Study 3

2 **LITERATURE REVIEW** **4**
2.1 Skin Cancer 4
2.1.1 Importance of Skin Cancer Early Detection 6
2.1.2 Self-Skin Examination Method and Efficacy 7
2.1.3 Summary 9
2.2 Object Detection with Deep Learning 10
2.2.1 Overview of CNN Architecture 11
2.2.2 Backbone Network of Object Detection 13
2.2.3 Object Detection Framework 19
2.2.4 Transfer Learning and Fine-Tuning 27
2.2.5 Object Detection Evaluation Metrics 28
2.2.6 Summary 33
2.3 Application of Deep Learning in Skin Lesions Classification and Detection 33
2.4 Deep Learning in Mobile Application 36
2.5 Summary 37

3	**METHODOLOGY AND WORK PLAN**		**39**
	3.1	Introduction	39
	3.2	Selection of Object Detection Model	40
	3.3	Selection of Skin Lesions Dataset	41
	3.4	Selection of Evaluation Metrics	42
	3.5	TensorFlow Object Detection API	42
		3.5.1 Dataset Preparation	43
		3.5.2 Configure Pipeline and Model Preparation	45
		3.5.3 Model Training	46
		3.5.4 Model Evaluation	46
		3.5.5 Tensorboard	47
		3.5.6 Convert TensorFlow Lite Model	47
	3.6	Mobile Application Development	48
		3.6.1 Integration of TensorFlow Lite Model	49
		3.6.2 Mobile Application Functionalities	51
		3.6.3 Mobile Application Compatibility Test and Inference Time Tracing	52
	3.7	Summary	53
4	**RESULTS AND DISCUSSION**		**54**
	4.1	Introduction	54
	4.2	Object Detection Models Comparison	54
		4.2.1 Model Validation	57
	4.3	Mobile Application Result	59
		4.3.1 Main, Setting, Document Activities	60
		4.3.2 Crop Image and Camera Activities	61
		4.3.3 Application Compatibility Test	63
		4.3.4 Inference Time and Application Size	64
	4.4	Summary	65
5	**CONCLUSIONS AND RECOMMENDATIONS**		**66**
	5.1	Conclusions	66
	5.2	Recommendations for future work	66
REFERENCES			**68**
APPENDICES			**76**

LIST OF TABLES

TABLE	TITLE	PAGE
2.1	Melanoma Stages (The Skin Cancer Foundation, 2020).	5
2.2	Dissemination of Disease at 5-Year Survival and Presentation (Doben and MacGillivray, 2009).	7
2.3	Error Rates of Different Model on ImageNet Validation (He et al., 2016).	15
2.4	Depthwise Separable vs Full Convolution MobileNet (Howard et al., 2017).	17
2.5	MobileNet vs MobileNetV2 (Sandler et al., 2018).	18
2.6	Comparison of The Number of Parameters and Operation Between CNN Models.	18
2.7	Comparison of Detectors Trained on PASCAL VOC Dataset in Terms Of Accuracy (Wu, Sahoo and Hoi, 2020).	26
2.8	Comparison of Testing Consumption on VOC 2007 Dataset (Zhao et al., 2019).	27
2.9	TP, FP, and FN (Padilla, R., 2020)	30
2.10	Summarized Works for Malignant Vs Benign Skin Lesion Classification.	34
2.11	Benchmark Image Classification Model Performance on iPhone 7 (Deng, 2019).	37
3.1	Pre-Trained Object Detection Model Online (GitHub., 2020)	40
4.1	Object Detection Models Comparison in terms of mAP, Inference time, and Model Size.	54
4.2	Area Under PR-curve (Average Precision) for Benign and Malignant Class of Three Models.	57
4.3	Accuracy of SSD MobileNet V2 object detection model and ResNet50 classification model using a new validation dataset.	59

4.4 Firebase Test Lab Results. 64

LIST OF FIGURES

FIGURE	TITLE	PAGE
2.1	Skin Anatomy (CDC, 2020).	5
2.2	ABCDE Criteria for Skin Cancer Diagnosis (Tsao et al., 2015).	8
2.3	General Flow of Object Detection (Pathak, Pandey and Rautaray, 2018).	10
2.4	CNN General Architecture (Rawat and Wang, 2017).	11
2.5	VGG16 Layers Definition.	14
2.6	Residual Learning: Residual Block (He et al., 2016).	15
2.7	Depthwise Convolution Filter (Howard et al., 2017).	16
2.8	1x1 Pointwise Convolution Filter (Howard et al., 2017).	16
2.9	General Architecture of Two-Stage Object Detector (Jiao et al., 2019).	20
2.10	Architecture of RCNN (Wu, Sahoo and Hoi, 2020).	21
2.11	Fast-RCNN Architecture (Wu, Sahoo and Hoi, 2020).	21
2.12	Faster-RCNN with RPN Module (Wu, Sahoo and Hoi, 2020).	22
2.13	Main Idea of YOLO (Redmon et al., 2016).	23
2.14	Architecture of YOLO Object Detection (Wu et al., 2020).	24
2.15	Main Idea of SSD Generate Multiple Size Anchor Boxed In Grid Cell (Liu et al., 2016).	25
2.16	Complete Architecture of SSD Detector With VGG16 Backbone (Liu et al., 2016).	25

2.17 Ground Truth and Predicted Bounding Box (DeepAI., 2020). 29

2.18 PR-curve (Manal El Aidouni., 2020). 31

2.19 Area Under PR-curve (Padilla, R., 2020). 32

2.20 General Flow of Using TensorFlow Object Detection API (Taqi et al., 2019). 35

3.1 Project Flowchart. 39

3.2 Pre-trained Model Files. 41

3.3 Benign Skin Lesion. 41

3.4 Malignant Skin Lesion. 42

3.5 Workspace Example. 43

3.6 Labelimg Software. 44

3.7 Label Detail Saved as ‘.xml’ Format. 44

3.8 Label Map. 45

3.9 Necessary Files in Annotation Folder. 45

3.10 Model Training Command Line. 46

3.11 Configure Evaluation Metrics in The Pipeline Configuration File. 46

3.12 Model Evaluation Command Line. 47

3.13 Tensorboard Activation Command Line. 47

3.14 Tensorboard Interface. 47

3.15 Export Inference Graph Command Line. 48

3.16 Convert Model into Tensorflow Lite Command Line (Tanner, 2020). 48

3.17 General Process to Build an Android Application (Verma, Kansal and Malvi, 2018). 49

3.18 Code of Adding Firebase ML Kit Library into Android Dependencies. 49

3.19	Normal Dermatoscope in Market.	51
3.20	Camera Activity Auto Crop Concept to Simulate Dermatoscope.	51
4.1	(a) PR-curve of three models for malignant skin lesion class. (b) PR-curve of three models for benign skin lesion class.	55
4.2	Confusion Matrix for (a) SSD MobileNet V2 object detection model, (b) ResNet50 classification model.	58
4.3	Main, Setting, and Document Activities.	60
4.4	Action Flow for Camera and Crop Image Activity.	61
4.5	Action Flow of Inference (a) if Image Captured from Camera (b) if Image Add from Gallery.	63
4.6	Inference Time Distribution of 1300 samples collected for the past 30 days.	64

LIST OF SYMBOLS / ABBREVIATIONS

mAP	Mean Average Precision
RoI	Region of Interest
IoU	Intersection of Union
TP	True Positive
FP	False Negative
FN	False Negative
AP	Average Precision

LIST OF APPENDICES

APPENDIX A: Coding 76

A-1 Android Studio Coding for Creating TensorFlow Lite Interpreter. 76

A-2 Android Studio Code for Inference. 76

A-3 Android Studio Code for Image Downscaling and Normalization. 77

CHAPTER 1

INTRODUCTION

1.1 General Introduction

According to the World Health Organisation (WHO), over the past ten years, the cases of malignant skin cancer have increased. Some researchers allege that early detection of skin cancer is required to classify skin lesions symptoms so that dermatologists and clinicians can provide ways to avert it (Abuzaghleh, Barkana and Faezipour, 2015). But it has been proven that the diagnose process of skin cancer is likely to result in misdiagnosis and inaccuracy because of doctor’s subjectivisms. However, due to deep learning has become trendy in medical imaging field, the classification and detection of skin lesions could be achieved by training object detection deep learning neural network models (Taqi et al., 2019). Besides, smartphones already have been widely used recently for object recognition, classification, and more due to smartphones provides flexibility and convenience. As a result, the combination of smartphones and deep learning could help in detecting and classifying malignant skin lesions, as well as eradicate the subjectivity in skin cancer diagnosis. Last but not least the public could perform self-diagnosis and detection on skin cancer by using a smartphone.

1.2 Importance of the Study

The purpose of this project is to develop a mobile application that can detect and classify malignant and benign skin lesions. This mobile application enables anyone to perform self-diagnosis on skin lesions. It also able to overcome the limitations of conventional self-detection method, therefore, reduce the probability of misdiagnosis. In short, this project and studies could contribute to skin cancer diagnosis with the development of mobile application that integrates trained object detection deep learning model which able to detect malignant and benign skin lesions.

1.3 Problem Statement

Early detection of skin cancer which causes by malignant lesions is crucial for treatment as it would increase the survival rate of patients. However, conventional detection method such as ABCDE criteria possesses various limitations such as subjectivity and inaccuracy, due to different experience level of dermatologist and irregular characteristics of malignant skin lesions (Abuzaghleh, Barkana and Faezipour, 2015).

Besides, the current state-of-the-art in detecting skin lesions using deep neural networks mainly focuses on the classification and segmentation of skin lesions. Also, deep learning model architectures such as 'ResNet' used to perform these tasks often complex, heavy in size, slow, and difficult to implement. Therefore, decrease the accessibility of this technology to the public. Also, self-detection method of skin cancer for the public currently still using the ABCDE criteria (Farberg and Rigel, 2017), which possess some other limitations in terms of public usage, such as layperson might difficult to understand the criteria itself which may lead to misdiagnosis (Tsao et al., 2015).

1.4 Aim and Objectives

This project aims to replace the conventional skin cancer detection method (ABCDE criteria) with mobile application that integrates object detection and deep learning technology. State-of-the-art skin lesions classification using deep learning only achieves accuracy below 90 %. Therefore, this project aims to transfer learning on a pre-trained object detection model with ISIC dataset and achieve an evaluation accuracy higher than 90 % in detecting malignant and benign skin lesions. As smartphones became popular nowadays, a mobile application can be developed to perform the detection task without having users to memorize ABCDE criteria and increase the accessibility of this technology to the public. To integrate the object detection model in a smartphone application, the selected model requires to be lightweight in terms of number of parameters to avoid high computational cost. The overall mobile application requires to achieve:

(i) Compatible with different Android smartphones.

(ii) Integration of object detection model (>90% evaluation accuracy) in detecting malignant and benign skin lesions.

(iii) Small application size (< 50 MB).

(iv) Fast inference time (< 1 sec).

1.5 Scope and Limitation of the Study

This project will be focusing on the development of mobile applications, integration of object detection deep learning model in mobile application and perform object detection deep learning model training on malignant and benign skin lesion datasets. In the project study, skin lesion background studies will be covered to understand the characteristic of skin lesions. Also, different deep neural network architecture, speed, and size will be covered in the studies to find the best architecture to be used in this project. The specific skin lesions dataset will be discovered in the studies, as well as the methods to train deep neural networks and integrate deep neural network into mobile application. Besides, the application of deep learning in detecting and classifying skin lesions will be covered to explore the existing method of training a deep neural network. The implementation of deep learning in a smartphone will be studied to discover the platform for mobile application development with deep learning integration.

In this project, the number of dataset images of skin lesions is important to produce a good deep learning model. Also, all skin lesions images are required to label manually with bounding box since no existing labelled images are found. Due to this constraint, only limited labelled images can produce. Furthermore, training a deep learning model could be time consuming if computer specifications are not high enough. In this project, a computer with Intel i7 3rd generation CPU, 8gb of RAM was used to train a deep learning model. Therefore, the deep learning model would be trained using CPU only.

CHAPTER 2

LITERATURE REVIEW

2.1 Skin Cancer

Skin cancer is defined as the abnormal growth of skin cells and commonly develops on the skin with or without a long time exposed to the sun. According to The Skin Cancer Foundation (2020), skin cancer is the most common cancer in the United States and worldwide. Among five Americans, one person would develop skin cancer by 70 years old. Besides that, more than two people died because of skin cancer in the United States every hour (The Skin Cancer Foundation, 2020). There are two main types of skin cancer which can be classified as non-melanoma and melanoma. Non-melanoma skin cancer is a type of skin cancer that is not formed by melanoma and the most common types of non-melanoma skin cancer are basal cell carcinoma and squamous cell carcinoma. Basal cell carcinoma begins to develop in the basal cell layer of the skin (Figure 2.1), and Squamous cell carcinoma begins to develop in the squamous layer of the skin. On the other hand, melanoma type of skin cancer is considered to be the most serious among all types of skin cancer (Mayo Clinic.org, 2020). Melanoma started in the melanocytes, which are the cells that make melanin, the pigment that gives skin its colour (Cdc.gov, 2020). These types of skin cancer are commonly caused by malignant skin lesions and it is observable (Rose, 2020). Melanoma possesses a complex cancer staging and important to diagnose because it helps doctor to decide on patients' treatment and predict the chance of recovery (The Skin Cancer Foundation, 2020). The staging of melanoma is summarized in Table 2.1.

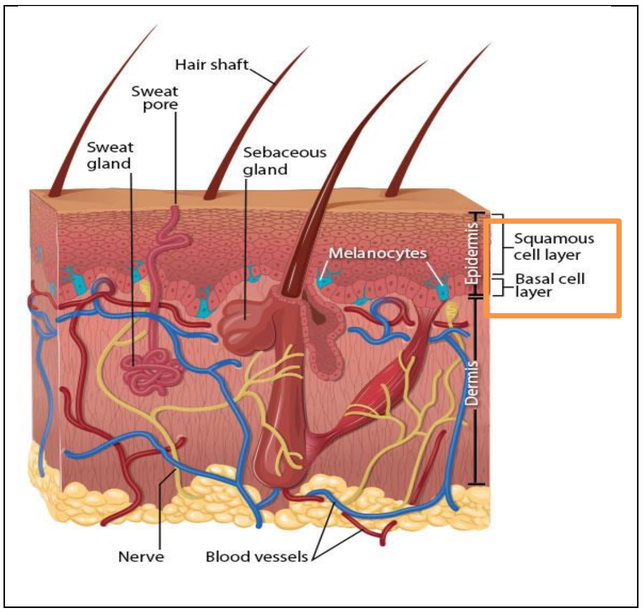


Figure 2.1: Skin Anatomy (CDC, 2020).

Table 2.1: Melanoma Stages (The Skin Cancer Foundation, 2020).

Stages	Description
Stage 0	Melanoma localized at the outermost layer of skin and does not grow deeper into skin.
Stage I	The cancer grown deeper into skin which is smaller than 1mm from the outermost layer of skin. It is localized but invasive.
Stage II	In this stage, the melanoma grown deeper larger than 1mm from the outermost layer of skin and could be grown greater than 4mm. It has a very high risk of spreading the cancer.
Stage III	The cancer has spread to local lymph nodes or lymph vessel. Besides, melanoma that spread to nearby skin or underlying tissue but does not reached lymph nodes are also included in current stage.
Stage IV	The cancer has spread to other body area, including lymph nodes or organs such as lungs, bone, brain, liver.

In our daily life, we can observe some skin lesions called ‘moles’ commonly. Moles are scientifically called ‘nevi’, they are normally benign which is not cancerous and with the appearance of flat shape, brown, dark brown, or even black (AIM at Melanoma Foundation, 2020). Moles are formed due to the accumulation of melanocytes in your skin, although they are harmless, they have the potential to develop into skin cancer (MHealth.org, 2020). According to Lodde et al. research, there are various types of nevi or moles with different characteristics, such as Junctional nevi, Intradermal nevi, Juvenile melanomas, and Blue nevi. Among these types of skin moles, skin lesion with the mixed characteristics of Junctional nevi and Intradermal nevi which called Compound nevi may develop into malignant skin lesions due to junctional component. Besides that, Lodde et al. also mentioned about the giant size of moles may develop into malignant skin lesions according to the respective incidence with 2-13 %. Although moles are seeming to be harmless(benign), there are still some possibilities for it to evolve into malignant skin lesions and result in skin cancer over time (Lodde et al., 2020). Therefore, it is necessary to study the importance of early detection of skin cancer.

2.1.1 Importance of Skin Cancer Early Detection

Early detection of skin cancer plays an important role to increase the survival rate. From Doben and MacGillivray (2009) research, evidence has provided that the five-year survival rate of cancer is highly affected by the time period of cancer diagnosis (Table 2.2). In Table 2.2, high distribution at diagnosis on an early stage of cancer which is ‘localized’ stage, result in a very high 5-year survival rate of 98.7%; whereas at the last stage of cancer, which is distant metastasis, the 5-year survival rate has only 15.5%. The authors also provided that 93% of melanomas cases which were diagnosed early, resulting in a much greater survival rate.

Table 2.2: Dissemination of Disease at 5-Year Survival and Presentation (Doben and MacGillivray, 2009).

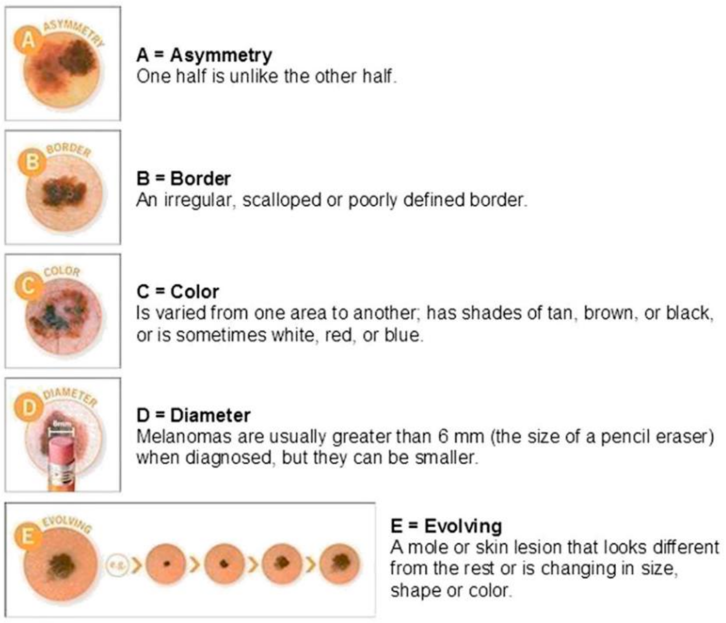
Disease Pattern	Distribution at Diagnosis (%)	5-Year Survival Rate (%)
Localized	81	98.7
Regional spread	12	65.1
Distant metastasis	4	15.5
Unknown stage	4	77.4

Moreover, Glazer et al. (2017) also stated that early detection of skin cancer can avert the morbidity of skin cancer as well as increase the survival rate of the patient. Melanoma would grow horizontally within the first layer of the skin in a very early stage and then started to grow vertically or deeper into the skin over time (Clark Jr. et al., 1989). Due to this fact, early detection of skin cancer able to increase the survival rate is because it results in a higher proportion of thinner depth of skin lesions being removed. Besides that, the prognosis on skin cancer is directly proportional to the vertical depth of the skin lesions and able to further conclude that limit skin cancer burden as well as reduce death could be achieved through early detection of skin cancer.

2.1.2 Self-Skin Examination Method and Efficacy

Despite early detection of skin cancer or melanoma able to increase curability in most cases, differentiation between melanoma and benign skin lesion at the initial stage is a difficult task even for experienced dermatologists (Jerant et al., 2000). A large number of existing researches pointed out a common self-skin examination method for early detect skin cancer, called ‘ABCDE’ criteria. This method was first introduced by Friedman, Rigel and Kopf (1985) with only ‘ABCD’ without ‘E’ criteria, and this method aims to make an early diagnosis of malignant melanoma through observing and differentiating between lesions clinical characteristics, such as asymmetry shape, border irregularity, colour uniformity, and diameter of lesions. Criteria ‘E’, means evolvement of skin lesions in terms of shape, size, and colour is then added on to the ‘ABCD’ criteria in Abbasi et al. (2004) research with shreds of evidence supported. The ‘ABCDE’ criteria for early diagnosis of skin cancer is summarized in Figure 2.2.

With these criteria, if the skin lesions fulfil more of the criteria, the more suspicious for skin cancer (Tsao et al., 2015).



A = Asymmetry
One half is unlike the other half.

B = Border
An irregular, scalloped or poorly defined border.

C = Color
Is varied from one area to another; has shades of tan, brown, or black, or is sometimes white, red, or blue.

D = Diameter
Melanomas are usually greater than 6 mm (the size of a pencil eraser) when diagnosed, but they can be smaller.

E = Evolving
A mole or skin lesion that looks different from the rest or is changing in size, shape or color.

Figure 2.2: ABCDE Criteria for Skin Cancer Diagnosis (Tsao et al., 2015).

Multiple researchers indicated that ‘ABCDE’ criteria could be used by dermatologist or physicians to carry out an early diagnosis of skin cancer for patients, and also it can be educated to laypersons or novice for self-examination on skin cancer (Friedman, Rigel and Kopf, 1985; Tsao et al., 2015; Farberg and Rigel, 2017; Glazer et al., 2017). Besides that, the proposer of this criteria Friedman et al. and also ‘E’ criteria proposer Abbasi et al. both had concluded that this technique does help in early diagnosis of skin cancer and able to enhance layperson’s ability in distinguishing malignant skin lesions. Although this technique seems to be convincing for self-examination on skin cancer, some researchers pointed out its limitations and doubt its efficacy after reviewing the criteria.

For example, Bränström et al. (2002) experimented on whether ‘ABCD’ criteria could help layperson on self-examination of malignant skin lesions. The results of their experiment shown that the criteria did enhance their ability to predict malignant skin lesions, but respondents have difficulty in recognizing benign skin lesions such as nevi or common moles or even overestimated the malignancies of benign skin lesions due to misconceptions about the characteristics of malignant skin lesions and benign skin lesions. Besides that, Tsao et al. (2015) commented about the 'ABCDE' criteria accuracy would affect by the level of experience or subjectivity of physicians and also concluded that no exact clinical trial evidence shown to prove that by using 'ABCDE' criteria can improve public's ability to perform early diagnosis on skin cancer even though the diagnosis accuracy of 'ABCDE' criteria verified in clinical practice. Chamberlain et al. (2003) research results on earlier detection of nodular melanoma have shown that nodular melanoma types of skin lesions sometimes fail to fulfil 'ABCD' criteria due to its shape more to symmetrical, uniform colour, non-pigmented as well as does not evolve in a colour change. Also, Glazer et al. (2017) mentioned that 'D' diameter > 6mm criteria are not very accurate due to the diverse nature of early malignant skin lesions and some malignant skin lesions with a diameter smaller than 6mm have been identified. Therefore due to skin cancer diversity characteristics, it is still a challenge in clinical recognition on malignant skin lesions even for experienced physicians or dermatologists (Glazer et al., 2017).

Besides the flaws of 'ABCDE' criteria stated above, the criteria may still difficult to understand and remember by the public (Tsao et al., 2015). A review on visual images for patient skin self-diagnose indicated that untrained layperson would have problem with the application of 'ABCDE' criteria without appropriate images (McWhirter and Hoffman-Goetz, 2013). Therefore, the development of accurate, sensitive and objective diagnostic tools to aid visual diagnosis is vital to enhance and improve the early recognition outcomes (Farberg and Rigel, 2017).

2.1.3 Summary

In summary, skin cancer is caused by malignant skin lesions such as melanoma and it would become severe or even hard to cure if the patient was later diagnosed. Benign skin lesions could evolve into malignant skin lesions over time. Therefore, it is important to have an early diagnosis or detection of skin lesions to increase the survival rate of a skin cancer patient. Besides, the most common skin self-examination method for early diagnosis of skin cancer called 'ABCDE' criteria can be learned by laypersons to develop the ability to distinguish malignant skin lesions. This method was also adopted by physicians and dermatologists to carry out an early diagnosis for the patient. Due to many reasons, the method possessed some limitations and have been pointed out by various researchers, such as subjectivity of dermatologist or physicians, skin lesions diverse nature characteristic reduce accuracy, and difficult to remember by the public.

However, automated analysis of skin lesions is a trending research topic that intended to develop tools for computer-aided diagnosis of skin cancer (Korotkov and Garcia, 2012). Computerized diagnosis is essential due to the increasing rate of cases, subjectivity of procedure and time (Amelard et al., 2015). Hence, it is encouraged to develop a new skin cancer detection technique to improve early recognition outcomes for the public and dermatologist.

2.2 Object Detection with Deep Learning

Object detection is the process of classifying an object and recognizes its respective location by outputting a bounding box around the object (Pathak, Pandey and Rautaray, 2018). Figure 2.3 shows the general flow of object detection, it clearly shows that detection, localization, and classification are important components in the process. Besides, due to its wide range of applications and technological breakthroughs in transportation, surveillance, life, and medical field recently, object detection has brought some attention (Jiao et al., 2019). Therefore, object detection could be the potential field to develop a new technology for skin lesions detection to improve the traditional early diagnosis method.

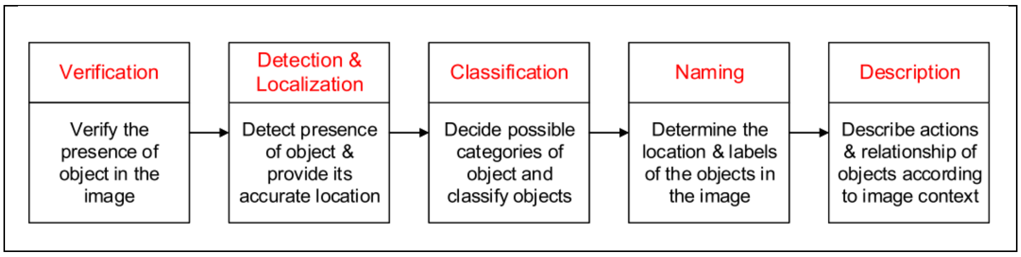

Verification
Verify the presence of object in the image
Detection & Localization
Detect presence of object & provide its accurate location
Classification
Decide possible categories of object and classify objects
Naming
Determine the location & labels of the objects in the image
Description
Describe actions & relationship of objects according to image context

Figure 2.3: General Flow of Object Detection (Pathak, Pandey and Rautaray, 2018).

In traditional object detection, the technique or methods generally based on handcrafted features and simple trainable neural network architecture. Since the development of deep learning, more powerful tools which can learn deeper features and details, are introduced to solve the limitation in traditional object detection (Zhao et al., 2019). With deep learning mechanisms, it can learn high-level features from low-level ones and approach high accuracy for object classification without any extraction of handcrafted features (Nasr-Esfahani et al., 2016). The most representative model of deep learning in image analysis is called Convolutional Neural Network, as known as CNN (Lecun, Bengio and Hinton, 2015).

2.2.1 Overview of CNN Architecture

Convolutional neural network (CNN) is a feedforward network in which input information such as image data flows in one direction to generate some outputs. In general, CNN architecture consists of different types of layers, they are convolutional and pooling layers. Another layer called fully connected layers similar to the artificial neural network is connected after the modules. These modules are stacked together to form a deep model or CNN architecture (Rawat and Wang, 2017). Figure 2.4 illustrated a general CNN architecture with an image classification task. An input image passes into convolutional layers, then outputs of the convolutional layers feed into the fully connected layer to do classification. The best example of CNN model with the same components stated previously is known as AlexNet (Krizhevsky et al., 2012).

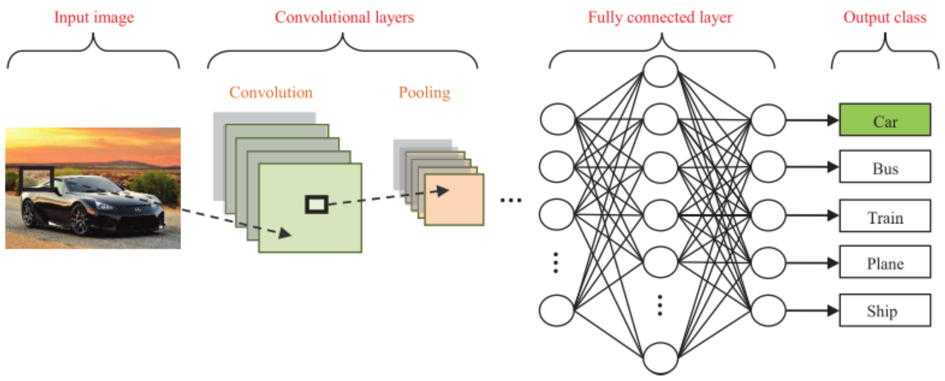


Figure 2.4: CNN General Architecture (Rawat and Wang, 2017).

The convolutional layers aim for features extraction from an image and learn those feature representations from the input image. Feature maps are formed by arranging neurons in the convolutional layers, and each neuron in a feature map connected to previous layers' neurons via a set of trainable weights, or as known as 'filter banks'. Inputs and learned weight convolve together to generate a new feature map, then the feature map would go through a non-linear function such as ReLU (Rectified Linear Unit) or Sigmoid to activate the next neurons. Besides, the pooling layer serves the purpose of reducing the spatial resolution of the feature maps. In general, max-pooling would be used in the architecture, and it selects the largest element value within each receptive field (Lecun, Bengio and Hinton, 2015). Whereas fully connected layers interpret the output features from previous convolutional and pooling layer to perform high-level reasoning. In the problem of classification, it generally uses soft-max operators which choose the class with the highest probability as the final output (Krizhevsky et al., 2012).

After these layers are constructed, the whole model is required to train using labelled dataset to able to classify an object. In normal cases, the dataset would be separated into two groups, which are test set and training set. A training set is the images that the model would be trained on, whereas testing set is the images without feed into training to evaluate the performance of the model after training. The dataset will be normally split into 80% of training set and 20% of testing set. CNN uses learning algorithms to adjust all the parameters which are biases and weights. The most common learning algorithm is called back-propagation; it calculates the loss function to determine how much adjustments to be made on those parameters to approach desire output. A common problem while training a CNN is overfitting, which will affect the model's ability to classify data due to an imbalance of training data. There are multiple ways to overcome overfitting problems such as perform training data augmentation or modify networks with dropout and batch normalization methods (Rawat and Wang, 2017). State-of-the-art results models were normally trained based on a dataset, called ImageNet, which is an image classification competition also as known as ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Image-net.org, 2020).

CNN commonly used for classifying an image. For example, Pai and Giridharan (2019) trained a popular CNN model called VGG16 to classify seven types of skin lesions. Besides the classification purpose of a CNN, CNN also acting as the backbone network to serve as a feature extractor role in object detection (Jiao et al., 2019).

2.2.2 Backbone Network of Object Detection

Object detection needs a good backbone network to perform well. The backbone network serves the purpose of feature extractor for object detection. It takes an input image and output a feature map of the corresponding image. Most of the backbone networks for object detection has taken out the last fully connected layer (Jiao et al., 2019).

Jiao et al. (2019) did a detailed survey on deep learning object detection and mentioned that choosing a CNN backbone for object detection consist of two requirements which are accuracy and efficiency. People can choose existing densely and deeper backbone such as ResNet or lightweight backbone such as MobileNet. Choosing the right backbone for object detection is important depending on the application requirement as it will make a trade-off between speed and accuracy.

Reddy et al. (2018) did a comparison between deep learning models in terms of efficiency and accuracy for user authentication on mobile devices. They aim to find suitable CNN architecture for mobile devices. Therefore, following Reddy et al. works, some of the state-of-the-art model architecture such as ResNet, MobileNet, and VGG, as well as the comparison between each model in terms of efficiency and accuracy will be reviewed in this section.

2.2.2.1 Overview of VGG

The VGG network was introduced by Simonyan and Zisserman in 2015. This network possesses the same configuration shown in Figure 2.4, but VGG consists of more convolution and pooling layers. Besides, according to the authors, VGG uses a small size 3 x 3 convolution filter to replace the large size of 11 x 11 and 5 x 5 convolution filter in AlexNet proposed by Krizhevsky et al. (2012). The architecture of VGG (Figure 2.5) consists of 13 convolution layers, 3 fully connected layers, and 5 max-pooling layers stacked up together, total up with 16 trainable layers. Hence the model is commonly called VGG16. The input size of VGG16 is 224 x 224 fixed dimension of RGB image. All convolution layers are equipped with ReLU (Rectified Linear Unit) (Simonyan and Zisserman, 2015).

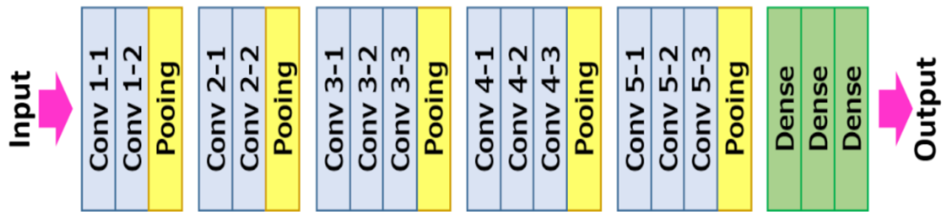


Figure 2.5: VGG16 Layers Definition.

VGG16 has been experimented with by its authors with training on the ImageNet dataset. It achieved a good result with only 7.0% of top-5 test error.

2.2.2.2 Overview of ResNet

ResNet as known as the residual network was developed by He et al. in 2016 to address the solution of degradation in accuracy while training a deep convolutional neural network such as VGG16. He et al.'s literature made complete documentation about ResNet. In an ideal case, a neural network shall get much better in training (low training error) when the neural network has deeper layers. But the authors stated that with the increasing depth of the network, the accuracy of the respective networks to get saturated and degrades in practical cases. Therefore, adding more layers to a traditional deep convolutional network would cause higher training errors (He et al., 2016).

The main idea of ResNet introduces a technique called “Identity shortcut connection” that will skip one or more layers in the convolutional networks as shown in Figure 2.6. The idea behind Figure 2.6 residual block is that instead of going through each stack of layers that directly fit a desired underlying mapping, it can be directly let these layers fit a residual mapping. Identity shortcut connection does not add any extra parameter or computational complexity (He et al., 2016). Besides, these shortcut connections perform identity mapping, and their outputs are added to the output of original stacked layers which propagates the gradient from deep layers to shallow units, hence result in reduces training difficulty. This simple modification of ResNet made the possibility to train a network up to 1000 convolutional layers, and yet still able to produce low training error while increase layers' depth (Wu, Sahoo and Hoi, 2020).

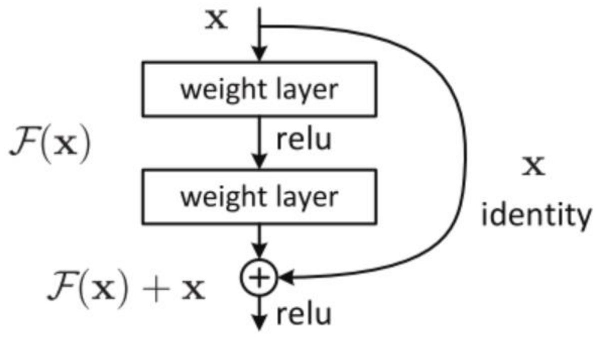

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.6: Residual Learning: Residual Block (He et al., 2016).

After He et al. propose ResNet, the respective model get first place in an ImageNet, the image classification competition in 2015. Also, the authors compare the results of a different number of layers of ResNet in the competition with another model such as VGG16 (Table 2.3). Table 2.3 shows that ResNet with more layers can produce low test error than VGG16.

Table 2.3: Error Rates of Different Model on ImageNet Validation (He et al., 2016).

	Top-1 error (%)	**Top-5 error (%)**
VGG16	28.07	9.33
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	21.43	5.71

2.2.2.3 Overview of MobileNet

MobileNet was developed by Howard et al. in 2017. According to the authors' literature, MobileNet is an efficient model for mobile and embedded vision system. The general trend is to make deeper and more complicated networks for higher accuracy and resulting in these networks do not make itself more efficient in terms of size and speed. Hence, the authors address MobileNet is a network that focuses on optimizing speed and small in size by reducing the number of multiplication and addition operations occurs in the network, as well as reduce overall parameters of the network.

MobileNet architecture mainly built from depthwise separable convolution to reduce computation at the first few layers of the network. Depthwise separable convolution consists of two layers, which are pointwise convolutions layers and depthwise convolutions layers (Howard et al., 2017).

Depthwise convolutions use a single filter to filter each input channel as shown in Figure 2.7, therefore it consists D_k of width and height which normally to be 3 x 3 in value, and a thickness of 1 since it only runs through one channel. But it does not combine all the outputs to create new features because it only filters input channels such as RGB color channels. Therefore, pointwise convolution comes into place to compute the linear combination of depthwise convolution outputs using 1 x 1 convolution filter as shown in Figure 2.8. With these two-component, computational cost (multiplication and addition) is reduced drastically compared to the traditional convolutional networks which filter and combine all inputs to outputs in one step (Howard et al., 2017).

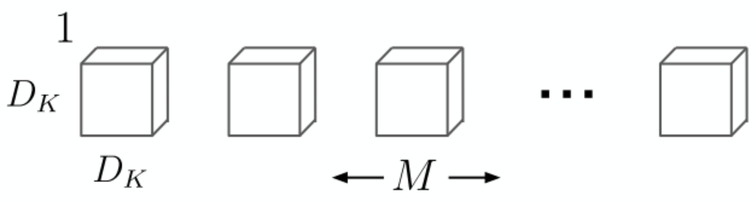


Figure 2.7: Depthwise Convolution Filter (Howard et al., 2017).

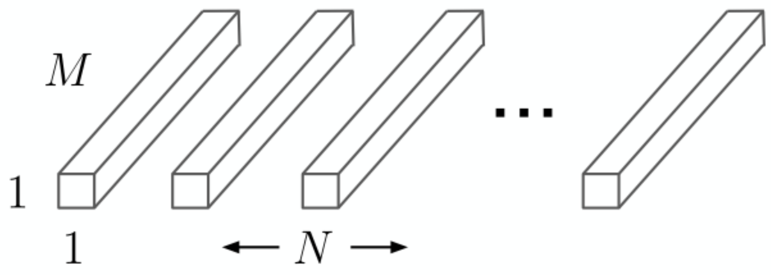


Figure 2.8: 1x1 Pointwise Convolution Filter (Howard et al., 2017).

In this case, MobileNet that uses depthwise separable convolutions able to achieve 8 to 9 times less computational cost than traditional convolutional network but sacrifice some accuracy. A comparison has been made by Howard et al between MobileNet with depthwise separable convolution and MobileNet with full tradition convolution in terms of accuracy, the number of multiplication and addition, and the number of parameters as shown in Table 2.4 below. From Table 2.4, the results show that MobileNet with full convolution contains much more parameters and operation, although it achieved 71.7% of accuracy. On the other hand, MobileNet with depthwise separable convolution although has slightly lower 70.6% accuracy but the number of parameters and operation is much less (Howard et al., 2017).

Table 2.4: Depthwise Separable vs Full Convolution MobileNet (Howard et al., 2017).

Model	**Accuracy**	**No. multiplication and addition (million)**	**Parameters (million)**
MobileNet with full convolution	71.7%	4866	29.3
MobileNet with depthwise separable convolution	70.6%	569	4.2

Later, a new mobile architecture called MobileNetV2 was introduced by Sandler et al. (2018). The architecture of MobileNetV2 is a combination of original MobileNet and inverted ResNet, 3 x 3 convolution layers in ResNet replaced to 3 x 3 depthwise separable convolution. MobileNetV2 architecture contains the initial fully convolution layer with 32 filters followed by 19 residual bottleneck layers which are similar to the residual block mentioned in section 2.2.2.2. Sandler et al. also compared its number of multiplication and addition operations and parameters with the original version of MobileNet. Table 2.5 clearly shows that the operations and parameters that occur in MobileNetV2 are much lower than the original version when tested with the ImageNet dataset.

Table 2.5: MobileNet vs MobileNetV2 (Sandler et al., 2018).

Model	**No. multiplication and addition (million)**	**Parameters (million)**
MobileNet	575	4.2
MobileNetV2	300	3.4

2.2.2.4 Summary

In summary, four CNN models' architecture and idea which are VGG, ResNet, MobileNet, and MobileNetV2 has been reviewed. According to our project's objective, a backbone model with lightweight and efficient characteristics is more favourable. In Reddy et al. (2018) literature, these models have been compared with each other in terms of parameters and number of multiplication and addition operations. The results are summarized in Table 2.6, it shows that MobileNetV2 contains the least parameters and number of operations. Although very deep models such as VGG and ResNet able to achieve high accuracy in the ImageNet dataset, but they require lots of memory or size and the number of multiplication and addition operation, and therefore these models are not suitable for mobile application (Reddy, Rattani and Derakhshani, 2018).

Table 2.6: Comparison of The Number of Parameters and Operation Between CNN Models.

Model	**Parameters**	**No. of Multiplication and Addition operations**
VGG - 19	140 million	19.6 giga
ResNet - 50	23.5 million	4 giga
MobileNet	3.2 million	568 million
MobileNetV2	2.2 million	300 million

Besides, Bianco et al. (2018) did a benchmark analysis on deep neural network architecture and provided that MobileNetV2 is one of the most efficient models with moderate accuracy and lower model complexity (14MB of size) than original MobileNet (17MB of size). Therefore, the MobileNetV2 model has characteristics of both lightweight and efficient to serve as object detection backbone.

2.2.3 Object Detection Framework

Most state-of-the-art object detection utilizes deep learning network especially CNN as their backbone and combines with a detection network. As mentioned in the previous section, CNN deep learning network has the role of feature extractor in object detection. Whereas, detection network is the key idea in object detection because it serves the purpose of performing classification and localization on the object (Jiao et al., 2019).

Object detectors can be classified into two types, two-stage detector, and one stage detector. Both have some difference in architecture, as well as different performance. A two-stage detector tends to have lower detection speed due to its more complex architecture than the one-stage detector. One stage detector has relatively low performance such as classification accuracy compared to a two-stage detector (Wu, Sahoo and Hoi, 2020).

2.2.3.1 Two-stage Detector

Two-stage detector consists of two stages with different tasks performed. The first stage is the proposal generation. During this stage, region proposals that may potentially be the object will be performed on the image. The second stage would have a deep convolutional network to classify all the proposals from the first stage. Figure 2.9 shows the general architecture of a two-stage detector, object detection generally would have a backbone network at the front and follow by object detection network at the end, in this case, the object detection network is a two-stage detector (Wu, Sahoo and Hoi, 2020).

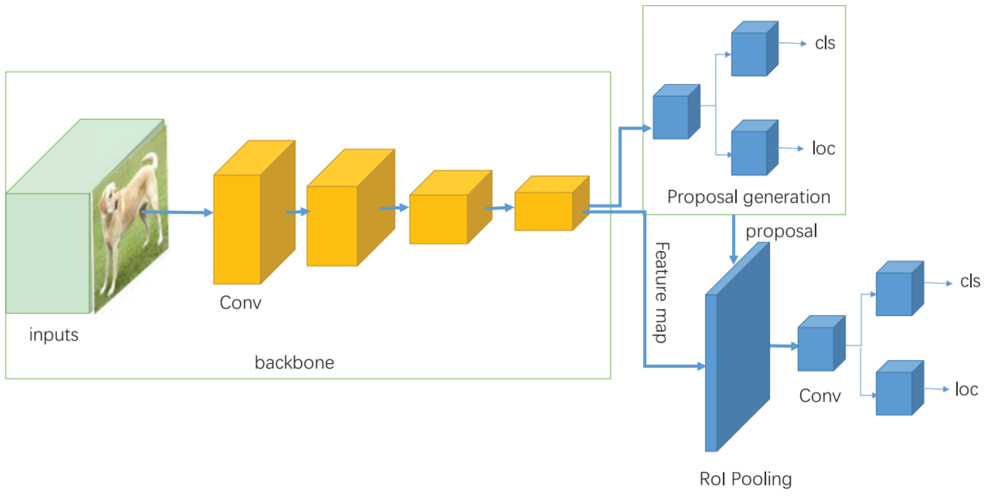


Figure 2.9: General Architecture of Two-Stage Object Detector (Jiao et al., 2019).

Some real examples are using the two-stage detector. The first one is RCNN proposed by Girshick et al. (2014). The architecture of RCNN can be divided into three different parts, they are region proposal generation, deep feature extraction with CNN, classification and localization (Figure 2.10). First stage RCNN performs selective search (Uijlings et al., 2013) and generates roughly 2000 of region proposals within one image, and provides to the second stage. During second stage, the region proposal is then cropped or warped into a fixed dimension and using AlexNet CNN model (Krizhevsky et al., 2012) to extract 4096 features from the proposed region as outputs. Then, different region proposals will be scored on a set of positive and background regions by pre-trained linear Support Vector Machine (SVM). The scored regions are then adjusted with the regression of bounding box and using non-maximum suppression (NMS) to filter out the final object location with a bounding box (Zhao et al., 2019).

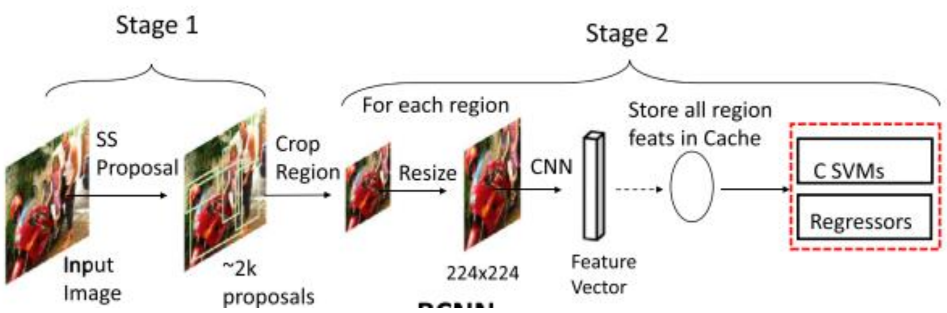


Figure 2.10: Architecture of RCNN (Wu, Sahoo and Hoi, 2020).

A year later, Girshick (2015) proposed a better version of RCNN, called Fast-RCNN. Due to RCNN performs convolution operation on each proposed region without sharing computation, it takes a long time to classify with SVM. Fast-RCNN has been modified to extracts features from the entire input image and passes the RoIPool (region of interest pooling) layer to get fixed dimension features for the classification and bounding box regression. The key-concept of Fast-RCNN is that it extracts features from an input image once, and then pass to CNN for localization and classification task. Compared to RCNN which inputs every single proposed region into CNN, Fast-CNN saves a lot of time and memory to process and store all the features. Also, Fast-RCNN training process is faster than RCNN because Fast-RCNN is a one-stage end-to-end training process whereas RCNN is a multi-stage training process (Girshick, 2015). As shown in Figure 2.11, an input image and various RoI are passes into a fully convolutional network. Then each RoI pooled into a fixed dimension feature map and pass into fully connected layers for predictions.

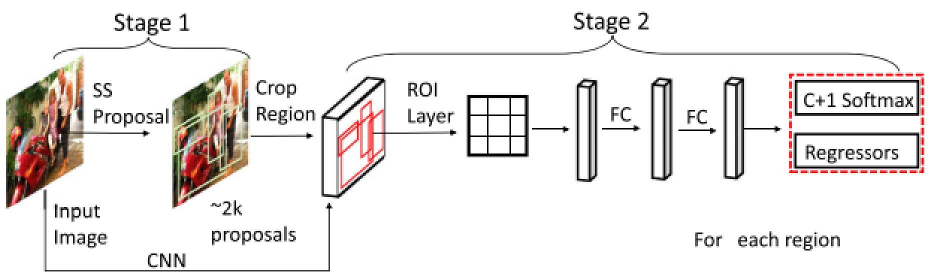


Figure 2.11: Fast-RCNN Architecture (Wu, Sahoo and Hoi, 2020).

Three months later since Fast-RCNN was introduced, a faster version of Fast-RCNN was developed by Ren et al. (2017) and it is called Faster-RCNN.

Due to the utilization of selective search to propose region in Fast-RCNN which is slow, Faster-RCNN replaces it with a Region Proposal Network (RPN) module to increase the speed of generating region proposals because it shares the same feature maps output from the backbone network with the detection network (Ren et al., 2017). In the field of two-stage detector, Faster-RCNN is the most representative detector. The architecture of Faster-RCNN is shown in Figure 2.12.

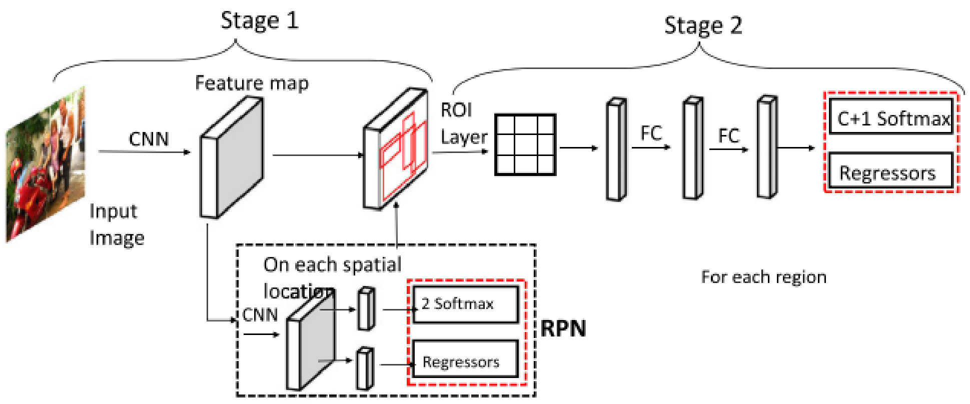


Figure 2.12: Faster-RCNN with RPN Module (Wu, Sahoo and Hoi, 2020).

2.2.3.2 One-stage Detector

One stage detector is different from two-stage detector because it does not require a separate stage for region proposal. The main feature of one stage detector is it consider all regions on the input image as potential objects (Wu, Sahoo and Hoi, 2020). Two representatives of one stage detector are YOLO and SSD.

YOLO (You only look once) was developed by Redmon et al. (2016) for real-time detection implementation. YOLO frame object detection as a single regression problem and with a relatively simple process compared to RCNN, thus make itself extremely fast on detection. YOLO architecture is simple, due to it only predicts less than 100 bounding boxes in one image compared to RCNN which predicts over 2000 proposed region (Jiao et al., 2019). Besides, another reason that makes YOLO so fast is that it combines all the separate component of object detection becomes a single neural network, and predict all bounding boxes across all classes for an input image simultaneously (Redmon et al., 2016).

YOLO first divides the input image into $S \times S$ grid (Figure 2.12) and S is pre-defined if the object centre falls into a grid cell, the particular grid cell is responsible for prediction. Besides, each grid cell would predict some bounding boxes and confidence scores. The predicted confidence scores would reflect the level of confidence of the model thinks the particular box contains an object, as well as how accurate is the predicted box. Meanwhile, besides the bounding boxes, the object class probability in each grid cell is also predicted and can be plotted into a class probability map as shown in Figure 2.13 (Redmon et al., 2016). The whole YOLO object detection architecture is combined with 24 convolution layers and 2 fully connected layers for feature extraction and classification, the YOLO architecture is between the convolution layers and fully connected layer, as shown in Figure 2.14.

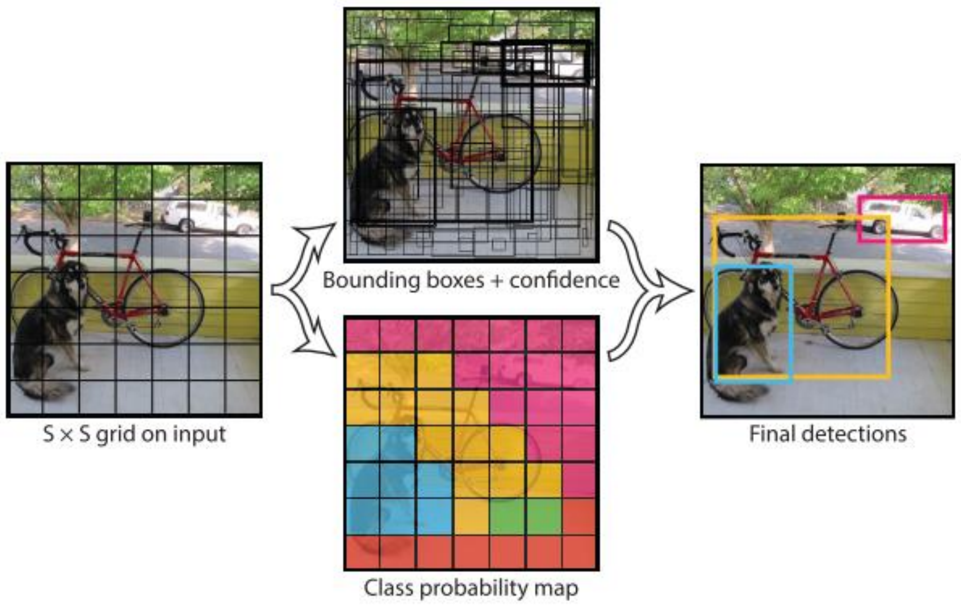


Figure 2.13: Main Idea of YOLO (Redmon et al., 2016).

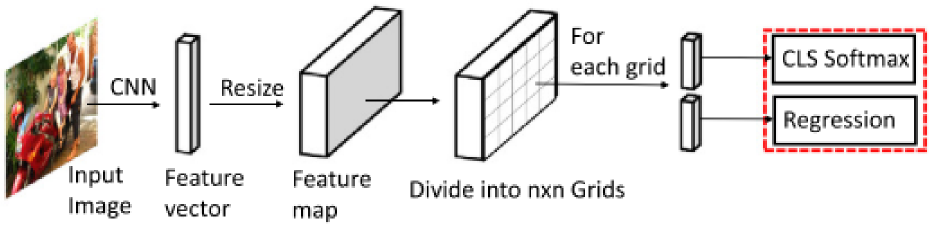


Figure 2.14: Architecture of YOLO Object Detection (Wu et al., 2020).

Although YOLO is fast in detection, it possesses some limitations too. One of the limitations is that YOLO has spatial constraints on predictions of bounding boxes since one grid cell can only predict a limited bounding box. Due to this reason, it limits the prediction of very nearby objects or small objects that group together such as flocks of birds. Also, YOLO has a trade-off in localization accuracy of objects due to it generalize to objects in unusual aspect ratio and generate rough features since it has multiple down sampling operations (Redmon et al., 2016).

Because of these limitations, Liu et al., (2016) proposed a one stage detector called Single Shot Detector (SSD). SSD also divided image feature maps into grid cells but in each cell, multiple scales and sizes of anchor boxes were generated (Figure 2.15). SSD is based on a convolution neural network to produce these anchor boxes and predict the presence of object class instances in the boxes. Then, followed by non-maximum suppression step to generate the final output detections. The SSD convolutional network is normally added as the extra feature layers right after a backbone convolutional network such as VGG16 as shown in Figure 2.16. These extra feature layers predict the offsets to default boxes of different scales and aspect ratios and their corresponding confidences (Liu et al., 2016).

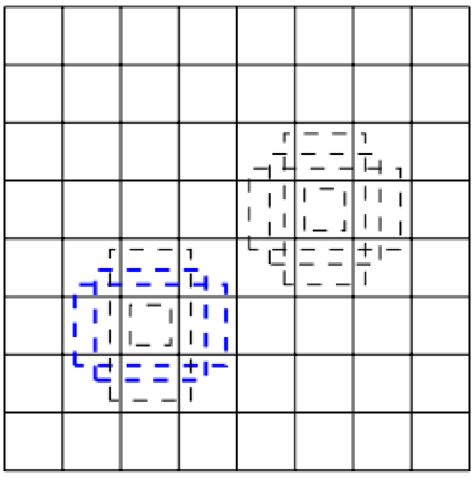


Figure 2.15: Main Idea of SSD Generate Multiple Size Anchor Boxed In Grid Cell (Liu et al., 2016).

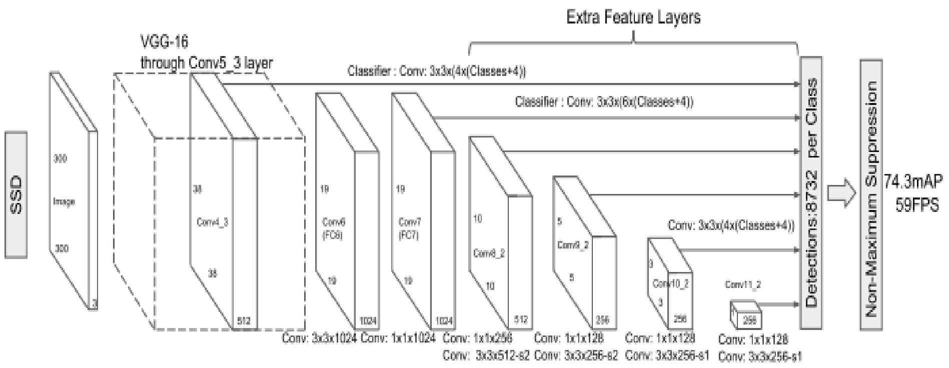


Figure 2.16: Complete Architecture of SSD Detector With VGG16 Backbone (Liu et al., 2016).

2.2.3.3 Comparison

After reviewing both two-stage detector (RCNN, Fast-RCNN, Faster-RCNN) and one stage detector (YOLO, SSD), they have their advantages and limitations. Two-stage detectors able to reach high accuracy but typically slower in detection speed, whereas one stage detectors have much faster detection speed than two-stage detectors but have lower accuracy (Soviany and Ionescu, 2018). A comparison has been made between each other by Wu et al. (2020), all the object detectors were tested with same VGG16 backbone, trained on PASCAL VOC2007, 2012 dataset with 2501 and 5717 images of 20 categories of the object respectively, and evaluated with mean average precision (mAP) which will be covered in the next section. The results are summarized in Table 2.7.

Table 2.7 shows that using the same backbone for feature extraction, every detector performs differently with the same dataset used. SSD performs well with the highest mAP of 79.8% and 78.5% with VOC 2007 and VOC 2012 dataset respectively among the other detectors. Higher mAP results in better model performance.

Table 2.7: Comparison of Detectors Trained on PASCAL VOC Dataset in Terms Of Accuracy (Wu, Sahoo and Hoi, 2020).

Detectors	**Backbone**	**Input Size**	**mAP (%)**	
			VOC 2007	**VOC 2012**
RCNN	VGG16	Arbitrary	66.0	62.4
Fast-RCNN	VGG16	~600 x 1000	70.0	68.4
Faster-RCNN	VGG16	~600 x 1000	73.2	70.4
YOLO	VGG16	448 x 448	66.4	57.9
SSD	VGG16	512 x 512	79.8	78.5

Besides that, Zhao et al. (2019) also made a comparison of testing time on the detectors (Faster-RCNN, YOLO, SSD)which also trained on PASCAL VOC 2007 dataset with a powerful computer. Zhao et al. evaluated them with mAP, testing time (second/image) and also real-time detection frame per second (FPS) but the backbone of some detectors (YOLO, SSD) does not mention very clearly in their literature. The results are summarized in Table 2.8 below, Faster-RCNN with ResNet 101 convolution layers backbone get the best mAP, but the detection time per image was 2.24 seconds on a powerful computer. Whereas SSD trained with 300 x 300 and 512 x 512 input size resulted in 74.3 and 76.8 mAP respectively. The detection time results of both SSD detector are optimistic which further prove that single-stage detector is much faster than two-stage detector, but sacrifice some accuracy.

Table 2.8: Comparison of Testing Consumption on VOC 2007 Dataset (Zhao et al., 2019).

Model	mAP (%)	Test time (sec/img)	Rate (FPS)
Faster RCNN(VGG16)	73.2	0.11	9.1
Faster RCNN (ResNet101)	83.8	2.24	0.4
YOLO	63.4	0.02	45
SSD300	74.3	0.02	46
SSD512	76.8	0.05	19

2.2.3.4 Summary

In summary, two-stage object detector may produce higher accuracy but sacrifice its detection time, whereas one-stage detector produces fast detection but sacrifice some accuracy. Due to smartphone computational power constraint, SSD one stage detector is more suitable in this project due to the reason of it has low detection time meanwhile able to product moderate level of accuracy as shown in the previous studies compared to YOLO.

2.2.4 Transfer Learning and Fine-Tuning

Fine-tuning is defined as the approach that defines the model parameters for the required task from the parameters that pre-trained on other related tasks (Ouyang et al., 2016). Training a deep convolutional from scratch is a difficult task due to a large number of labelled training images are required. Besides training a model from scratch, an alternative is through fine-tuning or transfer learning from a model that has been pre-trained on a very large labelled dataset, and then uses those pre-trained parameters such a way that performs feature extraction on object edges for further training on another specific object detection task (Tajbakhsh et al., 2016). Fine-tuning has been successfully used in several applications (Razavian et al., 2014; Penatti, Nogueira and Santos, 2015; Azizpour et al., 2015).

Tajbakhsh et al. (2016) analysed the comparison of full training or fine-tuning convolutional neural networks for medical images. They have investigated the problem by experimenting using AlexNet to train on the medical image dataset with and without fine-tune. Their results have concluded that using a pre-trained CNN with some fine-tuning on some specific layers in the CNN would outperform or may be performed the same as trained from scratch. Also, other advantages using fine-tuning is that fine-tuned CNN would be more robust to the size of training data than CNN trained from scratch.

Besides that, Shin et al. (2016) also did an experiment on training AlexNet from scratch and with fine-tuning from pre-trained AlexNet with medical images dataset to perform classification. Their result has shown that after using transfer learning, the fine-tuned AlexNet has much lower validation loss and higher validation accuracy compared to AlexNet which trained from scratch. Therefore, they found that using a transfer learning strategy able to produce the best performance results.

Nevertheless, some existing projects made by other researchers use transfer learning or fine-tuning to perform object detection. For example, Goyal et al. (2018) using a transfer learning approach on a pre-trained Faster-RCNN InceptionV2 object detection model to localize foot ulcer. They addressed that due to limited medical imaging datasets, CNN trained from scratch on these datasets does not produce a good result as the main reason to use transfer learning in their project. (Goyal et al., 2018).

In summary, fine-tuning or transfer learning from a pre-trained model is much better compared to train from scratch. Due to limited specific labelled dataset such as medical images, using fine-tuning and transfer learning is a way to solve this problem effectively.

2.2.5 Object Detection Evaluation Metrics

Before diving into evaluation metrics for object detection, it is necessary to introduce some existing object detection challenges, because each challenge using different evaluation metrics to judge on the model performance. Currently, every developed model would use these challenges as a benchmark to test model. PASCAL VOC and MSCOCO is the mainstream benchmark for object detection (Jiao et al., 2019).

The first one is the PASCAL VOC object detection challenge (Everingham et al., 2006) especially the challenges in the years of 2007 and

2012 are widely used. Both challenges provide a mid-scale dataset with 20 categories for object detection, but the number of images in the dataset is different. The second one is MSCOCO object detection challenge (Lin et al., 2014). MSCOCO challenge provides a large-scale dataset with 80 categories. Their number of training images is about 118287.

The first metrics for object detection is Intersection over Union (IoU) also as known as Jaccard Index (DeepAI., 2020). This metric quantified the likeness between the predicted bounding box and ground truth bounding box (labelled image) to measure how good is the predictions (Figure 2.17). The score of IoU ranges from 0 to 1. The higher score of IoU, the more similar of predicted box to the ground-truth box. IoU measures the overlapping area between the predicted box and the ground-truth box over their union (Manal El Aidouni., 2020). The equation of IoU is denoted as below.

$$IoU = \frac{ground\ truth\ \cap predicted}{ground\ truth\ \cup predicted} \tag{2.1}$$

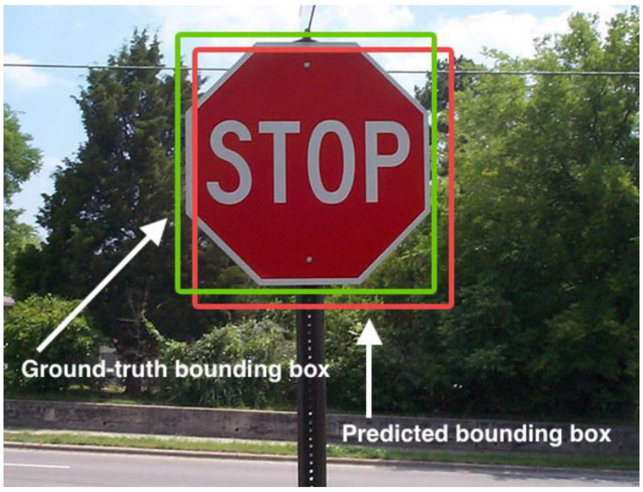


Figure 2.17: Ground Truth and Predicted Bounding Box (DeepAI., 2020).

By registering the IoU score for every detection, a threshold is set to group these scores, where IoU over this threshold are viewed as positive predictions and those below the threshold are viewed as false predictions. All the more accurately, the predictions are grouped into True Positives (TP), False Positives (FP), and False Negatives (FN). The statement above is for the localization problem, but in a classification problem, the IoU threshold is replaced with a classification confidence threshold. The descriptions of TP, FN, and FP are summarized in Table 2.9.

Table 2.9: TP, FP, and FN (Padilla, R., 2020)

True Positive	Correct detection with IoU/confidence larger than the threshold
False Positive	Wrong detection with IoU/confidence smaller than the threshold
False Negative	No prediction occurs in the ground-truth

After determining TP, FP, and FN, some of the basic metrics can be calculated such as Precision and Recall, which are important in evaluating object detection. Precision, also as known as specificity is to measure the probability of the predicted class or bounding boxes matches the actual ground-truth class or boxes. The value of precision ranging from 0 to 1. For example, if the precision score has a value of 0.8, which means that 80% of the time the predictions are correct. The formula of precision is denoted below.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{All\ detections} \tag{2.2}$$

where
TP = number of true positives
FP = number of false positives

Whereas, recall is to measure the probability of ground truth objects are detected correctly. Recall is also known as sensitivity. The recall of an object detector can be calculated using the equation below.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{All\ ground\ truths} \tag{2.3}$$

where
TP = number of true positives

FN = number of false negatives

Therefore, by determining precision and recall it can know that if the object detector has low recall, but high precision means that all the predicted boxes are correct but a lot of unpredicted ground truth objects (high number of false negatives). On the other hand, if the object detector has high recall but low precision means that all ground-truth objects are detected however many of the detections are incorrect. Besides, an object detector will predict bounding boxes, and each bounding box would have an associated confidence score. This confidence score is the probability of the object class shown in the respective bounding box. Therefore, by setting a threshold of confidence score, the detections with a confidence score higher than the threshold are classified as TP, whereas lower than the threshold are classified as FP. Hence, with different confidence thresholds, different precision and recall can be calculated to determine the model's performance and with the aid of the precision-recall curve (PR-curve) as shown in Figure 2.18. In Figure 2.18 each point in the curve represents different precision and recall values with a certain confidence value. Ideally, a model would maintain high precision with recall increases (Manal El Aidouni., 2020).

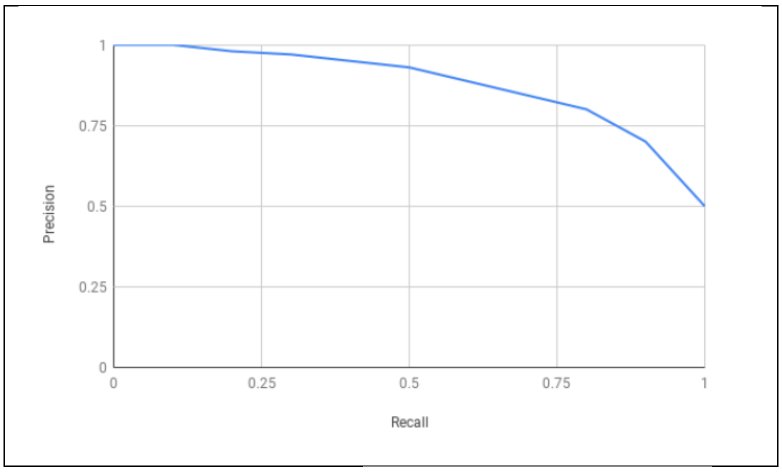


Figure 2.18: PR-curve (Manal El Aidouni., 2020).

Besides, another way to evaluate an object detection model is to calculate the average precision (AP) with the area under the PR-curve (Figure 2.19). AP is the precision averaged across all the recall values and has a range of 0 to 1. After understanding AP, since AP is calculated over 1 class category only, therefore mean average precision (mAP) comes into place if the dataset contains multiple N class categories (Manal El Aidouni., 2020). The mAP averages the sum of AP over several N class, which can be denoted as equation below.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{2.3}$$

where

N = number of class categories

AP = average precision

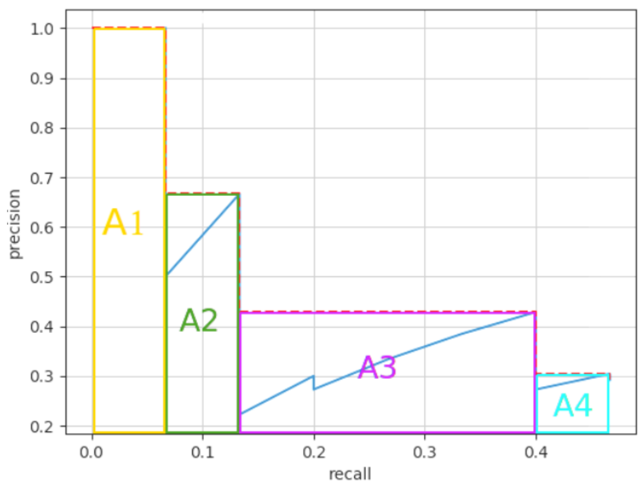


Figure 2.19: Area Under PR-curve (Padilla, R., 2020).

In MSCOCO object detection challenge, mAP metrics are used to evaluate an object detector. Besides, the AP is averaged with 10 different confidence thresholds ranging from 0.5 to 0.95 incrementing with 0.05, thus the higher the AP score indicated that the localization of objects is better. Also, MSCOCO evaluates AP on two different IoU values which are 0.5 and 0.75. Lastly, since MSCOCO contains small objects in their dataset, therefore AP is also evaluated on different sizes of object, such as AP^{small}, AP^{medium}, AP^{large} (Lin et al., 2014).

PASCAL VOC object detection challenge uses PR-curve and AP as model evaluation metrics. The AP is calculated with a 0.5 IoU threshold only (Everingham et al., 2006).

2.2.6 Summary

In summary, object detection is a process of object localization and classification. Due to the rise of deep learning, object detection had improved drastically compared to traditional object detection. Object detection with deep learning does not require any handcraft features but instead using a deep convolutional neural network as a feature extractor. Some of the existing deep CNN such as VGG16 and ResNet, and also lightweight CNN especially for mobile applications such as MobileNet and MobileNetV2 have been widely used by researchers. Besides that, by combining a detection network such as RCNN, YOLO, SSD with CNN, an object detector is made. Unfortunately, these deep neural networks require large datasets to train in order to achieve good performance in the detection task. Due to limited dataset on medical imaging, researchers use existing deep neural network model which pre-trained on large dataset and perform fine-tuning or transfer learning to train the network base on their requirements. Once a model is trained, it is necessary to evaluate the model and take a look at the performance. Some existing object detection benchmarks were introduced because it has been widely used by researchers.

2.3 Application of Deep Learning in Skin Lesions Classification and Detection

Due to the rising trend of deep learning, many researchers take advantage of deep learning to perform skin lesion classification and detection. Especially a lot of researchers contribute to skin lesion classification tasks. They train on different CNN models with different skin lesion dataset. The classification task is found commonly base on classifying between malignant and benign skin lesions. The accuracy metric has been used to evaluate the models, which is defined as the correct predictions over all predictions. From Table 2.10, some researchers (Al-Masni, Kim and Kim, 2020; Hosny, Kassem and Foaud, 2019; Romero-Lopez et al., 2017) use the existing CNN model to fine-tune or transfer learning. However, some others create their own CNN model (Albahar, 2019; Nasr-Esfahani et al., 2016), or customize on existing CNN model meanwhile transfer its parameters to create a new model (Harangi, 2018; Adegun and Viriri, 2020). Among them, Hosny et al. (2019) achieve the highest accuracy with 95.91% by fine-tuning AlexNet.

Table 2.10: Summarized Works for Malignant Vs Benign Skin Lesion Classification.

Authors	**Dataset**	**Model**	**Accuracy**
Al-Masni et al. (2020)	ISIC 2017	InceptionV3	77.04%
		ResNet-50	79.95%
		Inception-ResNetV2	81.79%
		DenseNet-201	81.27%
Nasr-Esfahani et al. (2016)	Med-Node	CNN from scratch	81%
Harangi (2018)	ISIC 2017	Customized CNN	89%
Adegun and Viriri (2020)	ISIC 2017	Customized CNN	95%
	PH2		95%
Albahar (2019)	ISIC 2017	CNN from scratch	94.94%
Hosny et al. (2019)	ISIC 2017	AlexNet	95.91%
Romero-Lopez et al. (2017)	ISIC 2016	VGGNet	81.33%

Since the main focus of this project is about skin lesion localization and classification, only one literature that did a similar project has been found. Taqi et al. (2019) use object detection with deep learning to localize and classify skin lesions. However, the classification task is only performed between skin lesions and background. The authors use ISIC 2018 dataset to train and perform transfer learning on existing pre-trained SSD-MobileNet model which MobileNet as the backbone for feature extraction, SSD as the object detector for bounding box prediction and classification. They presented very convincing results on detecting skin lesions from images with a detection accuracy of 99% in model testing and 96% of mAP during model training. The authors use a platform called TensorFlow Object Detection API to train and evaluate the model. Figure 2.20 shows the general flow of using TensorFlow Object Detection API.

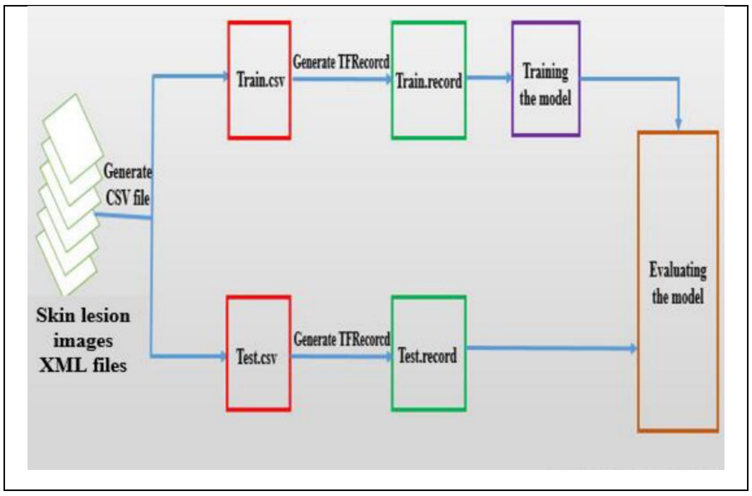


Figure 2.20: General Flow of Using TensorFlow Object Detection API (Taqi et al., 2019).

On the other hand, some similarities have found regarding the skin lesion dataset used among all literature stated above. Most of them using the ISIC dataset, while some using PH2 or Med-Node dataset. ISIC 2017 dataset contains 3 types of skin lesions which are benign skin lesion, melanoma, and seborrheic keratosis (benign), about 2000 images for training, 150 images for validation, and 600 images for testing. Whereas ISIC 2016 only contains 2 types of skin lesions which are benign skin lesion and melanoma, about 900 images for model training, 379 images for model testing (Al-Masni, Kim and Kim, 2020). PH2 dataset contains 3 types of skin lesions which are benign nevi, atypical nevi, and melanoma, about 200 images in the dataset (Mendonca et al., 2013). The Med-Node dataset contains 2 types of skin lesion which are benign nevi and melanoma, about 170 images only in the dataset (Giotis et al., 2015).

In summary, a lot of researchers contribute to skin lesions classification but not detection (localization and classification). However only Taqi et al. (2019) using object detection with deep learning to perform skin lesions detection. Fortunately, they had done complete documentation on the steps of approach. Besides, ISIC skin lesions dataset has been widely used by a lot of researchers since it provided more data images compared to PH2 and Med-Node.

2.4 Deep Learning in Mobile Application

Due to the rise of deep learning technologies nowadays, it enables a lot of mobile applications. There are several advantages of deep learning implemented on mobile devices including low communication bandwidth, quick response time, data privacy, and most importantly is to ease life. To deploy a deep learning model into mobile devices, some existing platforms enable you to train a mobile suitable model such as TensorFlow Lite, Caffe2, CoreML (Deng, 2019).

TensorFlow Lite is a lightweight version of TensorFlow which developed by Google which is an open-source tool that allows anyone to perform model training and deployment on a computer using Python programming language. TensorFlow Lite architecture allows deployment from computers to mobile devices and there is an easy process to bring TensorFlow model to mobile devices just by converting the original TensorFlow model into TensorFlow Lite (TensorFlow, 2020). Caffe2 was developed by Facebook company, it is a lightweight, modular, and also scalable deep learning framework. It provides cross-platform libraries for mobile devices deployment. Besides, Caffe2 models are extremely lightweight which can under 1MB of size. CoreML was developed by Apple company, and it is available on IOS operating system only. It can automatically minimize memory usage and power consumption on iPhone. Besides, models that are built using TensorFlow or Caffe can be converted into CoreML format in just a few lines of code (Deng, 2019). However, according to Ignatov et al. (2019), the easiest way to use deep learning on a mobile phone is TensorFlow Lite because it provides better performance, smaller size, and less requirement. To use TensorFlow Lite model, the trained model is required to convert into '.tflite' format for further implementation. Whereas other platforms such as Caffe2 are much less popular and very few problem descriptions and tutorials (Ignatov et al., 2019).

According to Deng (2019), there are a lot of deep learning models that have made to the public and enable everyone to develop mobile deep learning applications. For example, TensorFlow Model Zoo provides various pre-trained models such as ResNet, MobileNet, VGGNet and more. Besides that, it also provides some of the object detection models such as SSD MobileNet, Faster-RCNN ResNet and others. Furthermore, the author has compared some deep learning models in terms of accuracy, model size, and execution time on iPhone 7 (Table 2.11). The results in Table 2.11 have shown MobileNet has the least model size and execution time however the classification accuracy is below InceptionV3 and ResNet50.

Table 2.11: Benchmark Image Classification Model Performance on iPhone 7 (Deng, 2019).

Model Architecture	**Accuracy**	**Model Size (MB)**	**Execution time (ms)**
VGG16	71	553	208
Inception V3	78	95	90
ResNet50	75	103	64
MobileNet	71	17	32

Due to a large variety of mobile phones nowadays in terms of processing speed and memory, not every deep learning models are suitable to deploy on mobile phones (Ignatov et al., 2019). Therefore, to effectively integrate deep learning with mobile applications, it is necessary to choose a model with low computational cost which in this case MobileNet architecture (Section 2.2.2.3) has the stated potential. Some existing works of literature use MobileNet deep learning model to perform classification and object detection on mobile phones. For example, Taqi et al., (2019) trained SSD-MobileNet with TensorFlow object detection API to perform skin lesion detection on Samsung Galaxy S6 mobile phone.

2.5 Summary

In summary, importance of early detection of skin cancer cannot be underestimated. To replace the conventional method of skin cancer early detection, researchers suggest the development of new technology to support this area. However, due to the rise of deep learning on recent years, the performance of object detection shows great improvement. Train a high accuracy and lightweight architecture object detection deep learning model such as SSD MobileNet V2 then integrated into mobile application could be the possible method.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

The general project workflow is summarized in a flowchart figure below (Figure 3.1). This project uses TensorFlow Object Detection API to train object detection model and Android Studio platform for mobile application development.

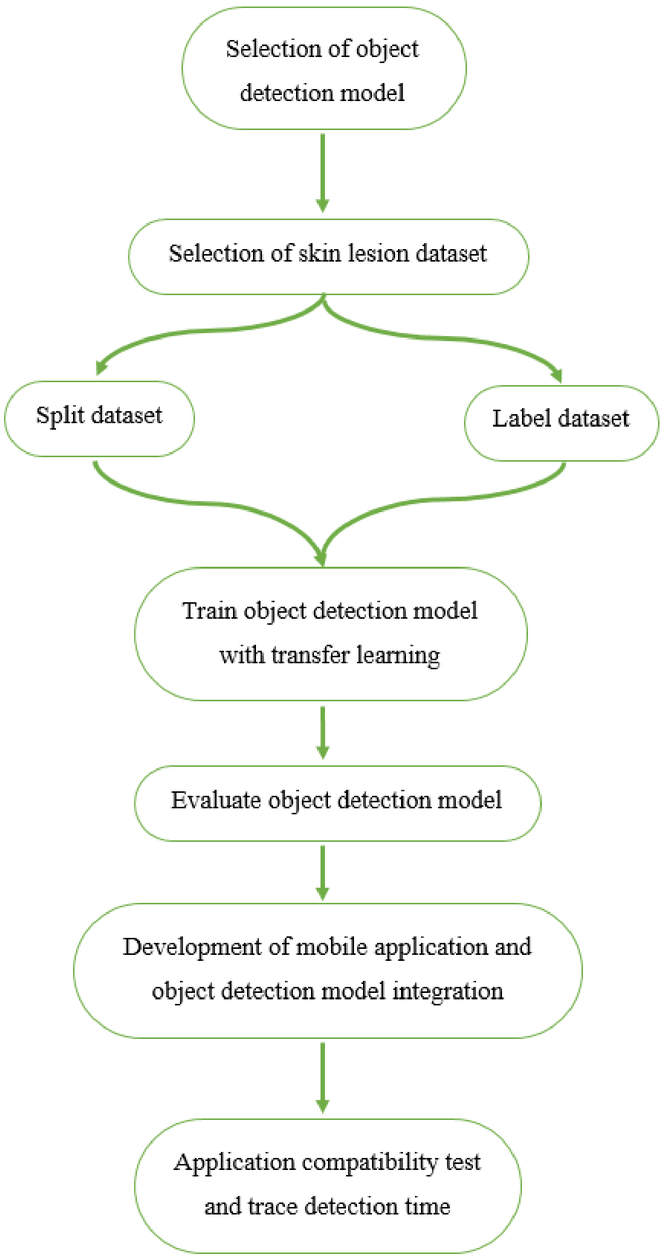

Selection of object detection model
Selection of skin lesion dataset
Split dataset
Label dataset
Train object detection model with transfer learning
Evaluate object detection model
Development of mobile application and object detection model integration
Application compatibility test and trace detection time

Figure 3.1: Project Flowchart.

3.2 Selection of Object Detection Model

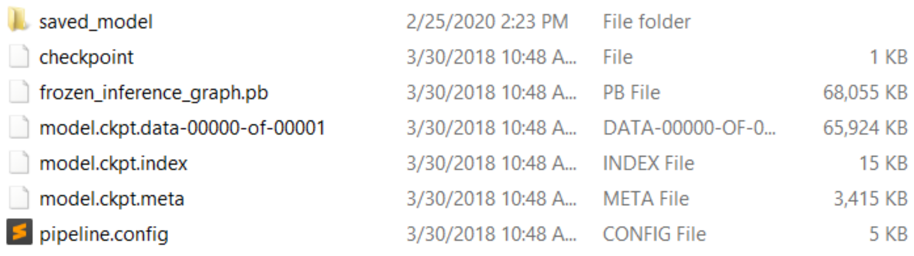
Theoretically, SSD MobileNet model became the first choice for this project after literature studies. However, some comparison will be made across various models to proof the reliability of this choice. For object detection network, the two highest accuracy networks which came across in literature review are chosen, they are SSD and Faster-RCNN. For feature extraction network, a fast detection speed with moderate accuracy network MobileNet (V1 and V2) and a slow detection speed with high accuracy ResNet are chosen. Therefore, comparison will be made between SSD MobileNet V1, SSD Mobilenet V2, and Faster-RCNN ResNet in terms of detection speed, mAP, and model size.

All selected networks will perform transfer learning with their respective pre-trained models on the same dataset. GitHub (2020) website provided these various pre-trained object detection models on the MSCOCO dataset for TensorFlow users. Some object detection models details are provided on the website (Table 3.1).

Table 3.1: Pre-Trained Object Detection Model Online (GitHub., 2020)

Model name	Speed (ms)	COCO mAP	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes

After downloaded and extract the pre-trained model file, inside contains various sub-files (Figure 3.2). ‘model.ckpt’ is the model parameters data files which trained on MSCOCO dataset. The ‘pipeline.config’ file defines the models, which contains all the details about the what model will be trained, what parameters should be used to train the model parameters, what set of metrics will be used to evaluate the model, and also the input dataset path for the model to train on.



saved_model	2/25/2020 2:23 PM	File folder	
checkpoint	3/30/2018 10:48 A...	File	1 KB
frozen_inference_graph.pb	3/30/2018 10:48 A...	PB File	68,055 KB
model.ckpt.data-00000-of-00001	3/30/2018 10:48 A...	DATA-00000-OF-0...	65,924 KB
model.ckpt.index	3/30/2018 10:48 A...	INDEX File	15 KB
model.ckpt.meta	3/30/2018 10:48 A...	META File	3,415 KB
pipeline.config	3/30/2018 10:48 A...	CONFIG File	5 KB

Figure 3.2: Pre-trained Model Files.

3.3 Selection of Skin Lesions Dataset

Various researchers selected ISIC skin lesions dataset to perform classification (Al-Masni, Kim and Kim, 2020; Harangi, 2018; Adegun and Viriri, 2020; Albahar, 2019; Hosny, Kassem and Foaud, 2019; Romero-Lopez et al., 2017) and object detection (Taqi et al., 2019), especially on benign and malignant skin lesions. Fanconi (2020) provides a malignant and benign skin lesion dataset which arranged from ISIC website to address the imbalance number between malignant skin lesions images (2286) and benign skin lesions images (19373) because imbalance class dataset deteriorates the performance of a model (Mazurowski et al., 2008). Therefore, dataset provided by Fanconi (2020) is used in this project, 800 images of skin lesions in the training set, 200 images in the testing set. All images scaled to the same 224 x 224 dimension.

Figure 3.3 and Figure 3.4 show an example of benign and malignant skin lesions image respectively from the dataset.

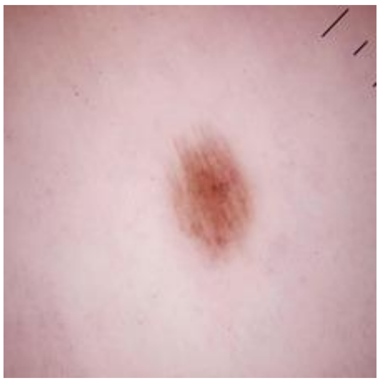


Figure 3.3: Benign Skin Lesion.

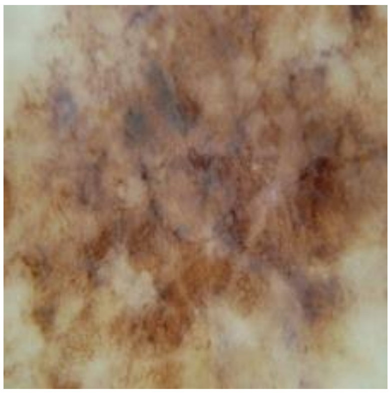


Figure 3.4: Malignant Skin Lesion.

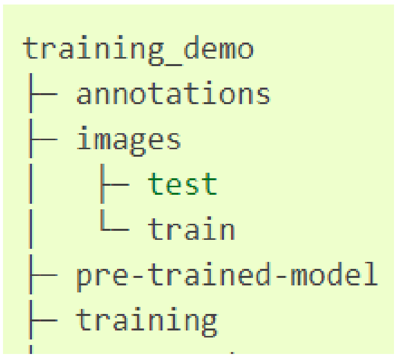
3.4 Selection of Evaluation Metrics

Some evaluation metrics can be selected from an existing object detection challenge such as MSCOCO or PASCAL VOC. Various researchers reviewed these metric and applied them in their research especially MSCOCO metrics (Ren et al., 2017; Pathak, Pandey and Rautaray, 2018; Liu et al., 2016; Goyal et al., 2018; Huang et al., 2017; Wu, Sahoo and Hoi, 2020; Girshick, 2015). Comparing these metrics, MSCOCO challenge uses AP metric that calculates over 10 different IoU thresholds which from 0.5 to 0.95 incrementing with 0.05. Then, calculate mAP with the average of these 10 AP. Additionally, MSCOCO evaluates the object detection model on AP in a distinct object dimension for small, medium, and large. However, PASCAL VOC challenge calculates mAP and AP over 0.5 IoU threshold. Furthermore, the selected dataset does not contain extremely small or large object in the images, hence, some metrics in MSCOCO are redundant and might cause false evaluation on model's performance. Therefore, PASCAL VOC metrics is used to evaluate the model performance in this project.

3.5 TensorFlow Object Detection API

In this project, using Tensorflow Object Detection API is the method to train and evaluate object detection model. To train an object detector more efficiently, it is necessary to prepare an organized workspace where all the required files will be saved into a sub-folder. For example, in Figure 3.5 creates a main folder or workspace called 'training_demo'. Inside the main folder, creates various sub-folders such as 'annotations', 'images', 'pre-trained model', and 'training'.

The annotation folder will store all the dataset label files for example '.csv' and '.record' files. Images folder will store all the dataset images, and the dataset shall split into test set folder and train set folder. Besides that, the pre-trained model folder will store the selected model. Whereas training folder storing '.config' and '.pbtxt' files.



```
training_demo
├─ annotations
├─ images
│   ├─ test
│   └─ train
├─ pre-trained-model
├─ training
```

Figure 3.5: Workspace Example.

3.5.1 Dataset Preparation

To train an object detection model, all the images require its bounding box and class label (Taqi et al., 2019). The bounding box specifies the location of the skin lesions and the class specifies the type of skin lesions, in this case only two classes, benign and malignant are used. Using 'LabelImg' software (tzutalin/labelImg, 2020) able to generate bounding box and class label of an image into object detection label file (Figure 3.6). All the label detail such as bounding box coordinate and image class would be saving with '.xml' format into the images test or train folder (Section 3.4.1) depends on which set of images are labelling (Figure 3.7).

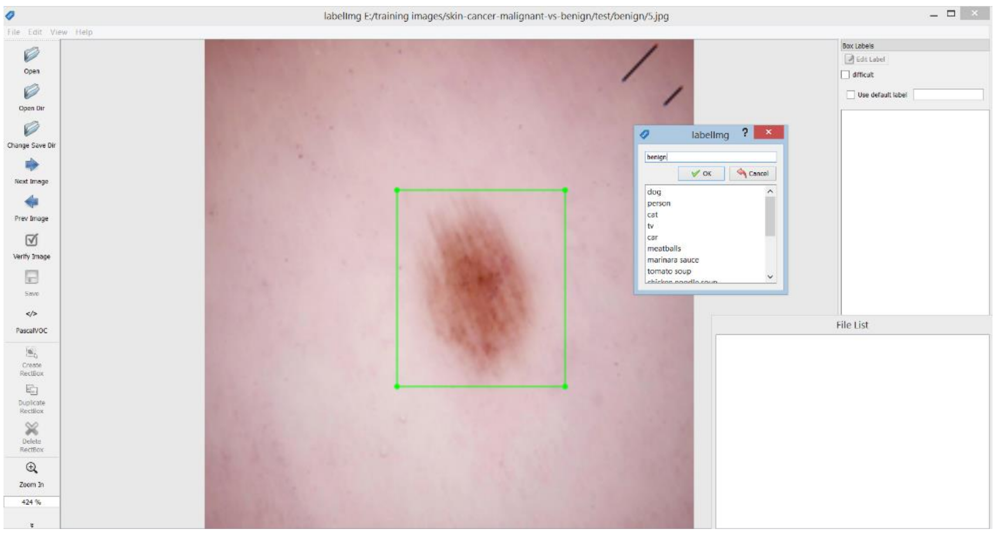
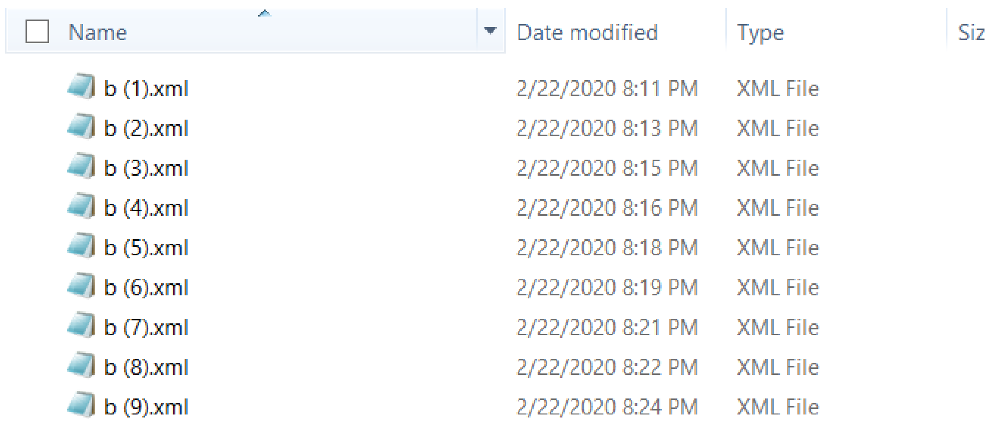


Figure 3.6: Labelimg Software.



Name	Date modified	Type	Siz
b (1).xml	2/22/2020 8:11 PM	XML File	
b (2).xml	2/22/2020 8:13 PM	XML File	
b (3).xml	2/22/2020 8:15 PM	XML File	
b (4).xml	2/22/2020 8:16 PM	XML File	
b (5).xml	2/22/2020 8:18 PM	XML File	
b (6).xml	2/22/2020 8:19 PM	XML File	
b (7).xml	2/22/2020 8:21 PM	XML File	
b (8).xml	2/22/2020 8:22 PM	XML File	
b (9).xml	2/22/2020 8:24 PM	XML File	

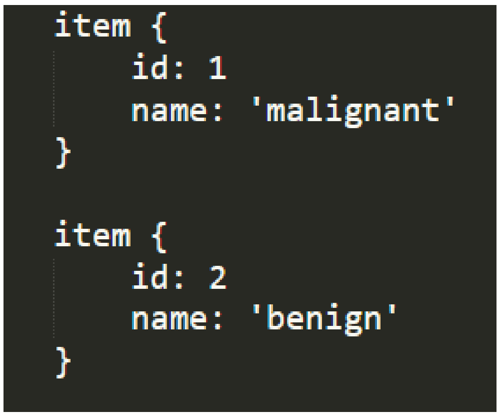
Figure 3.7: Label Detail Saved as '.xml' Format.

Once all data images are labeled and saved into '.xml' format (Section 3.3), all the XML files are required to convert into '.csv' file to combine all XML files. TensorFlow Object Detection API provides a Python script called 'xml to csv.py' to perform the conversion. Then, two CSV files for train set and test set images will be generated at the annotation folder.

Once the CSV files are generated, the next step is to convert the CSV files into TensorFlow application readable file called 'TFrecords'. TensorFlow Object Detection API provides a conversion script written in Python to its user. To convert the CSV files of both train set and test set to TFrecords files, run the script in windows command prompt and further pass in the path of the CSV files and the output path which refers to the annotation folder will do. Therefore, all information about the dataset such as image path, bounding box coordinate, image class are saved in TFrecords (.record) format.

Besides, TensorFlow Object Detection API requires a label map that maps each detection classes into an integer number. This file will be used during the training model process. In this case, the label map files will map two classes which are benign and malignant (Figure 3.8). Then, this file will be saved into the annotation folder with '.pbtxt' format.

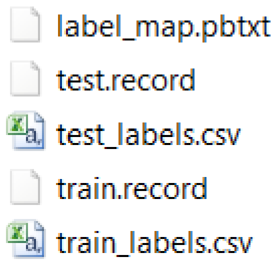
Lastly, the annotation folder should contain these files as shown in Figure 3.9.



```
item {
    id: 1
    name: 'malignant'
}

item {
    id: 2
    name: 'benign'
}
```

Figure 3.8: Label Map.



label_map.pbtxt
test.record
test_labels.csv
train.record
train_labels.csv

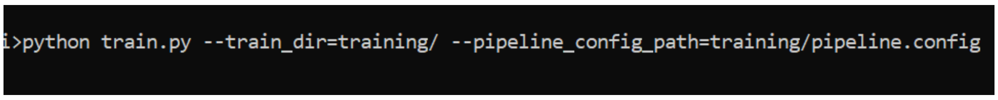
Figure 3.9: Necessary Files in Annotation Folder.

3.5.2 Configure Pipeline and Model Preparation

The next step requires to configure the pipeline of the model. Open the 'pipeline.config' as shown in Figure 3.1, then edit some of the content such as the number of classes, type of feature extractor, fine-tune checkpoint file path, TFrecord file path for train set, label map file path, TFrecord file path for test set, evaluation metrics, and the number of training steps.

3.5.3 Model Training

To train a model, TensorFlow Object Detection API provides a training Python script called ‘train.py’ to allow users to train their model in one command line without writing the script from scratch. Figure 3.10 shows the command line input in windows command prompt to run the Python script. The command line passes in the path of the training folder to save the model into the folder once the training process finished. Also, the model configuration file is required. Once the training started, the command window will show training loss with each training step, the lower the training loss the better the model training performance. The model should be trained until the training loss reaches saturated.

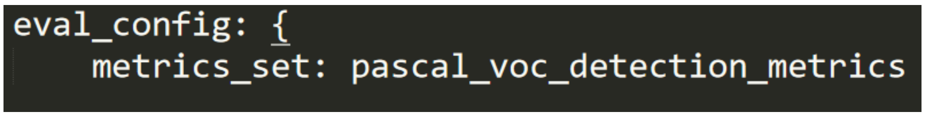


```
i>python train.py --train_dir=training/ --pipeline_config_path=training/pipeline.config
```

Figure 3.10: Model Training Command Line.

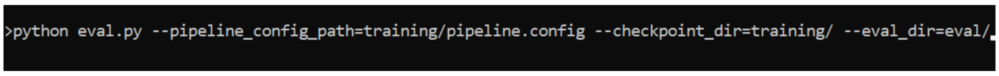
3.5.4 Model Evaluation

Once the model is trained, evaluation will be performed to observe the model’s performance on test set images with trained checkpoint. The evaluation metrics is PASCAL VOC metrics (Figure 3.11) which consist of mAP, and PR-curve (0.5 IoU). TensorFlow Object Detection API also provides a Python script called ‘eval.py’ to evaluate the model. The evaluation process generally similar to training process, run the script in windows command prompt and enter the command line shows in Figure 3.12.



```
eval_config: {
    metrics_set: pascal_voc_detection_metrics
```

Figure 3.11: Configure Evaluation Metrics in The Pipeline Configuration File.

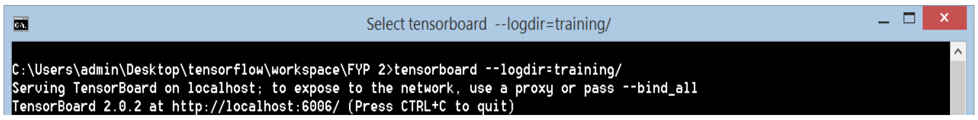


```
>python eval.py --pipeline_config_path=training/pipeline.config --checkpoint_dir=training/ --eval_dir=eval/
```

Figure 3.12: Model Evaluation Command Line.

3.5.5 Tensorboard

Tensorboard allows users to observe training and evaluation info. To monitor the training and evaluation process, run a command line in windows command prompt to activate Tensorboard (Figure 3.13), then an IP address will be output for the user to access via an internet browser. The Tensorboard will then read a log file inside the training folder or evaluation folder and display all the information as shown in Figure 3.14 example. Evaluation metrics mAP and PR-curve obtained from Tensorboard.



```
Select tensorboard  --logdir=training/
C:\Users\admin\Desktop\tensorflow\workspace\FYP 2>tensorboard --logdir=training/
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.0.2 at http://localhost:6006/ (Press CTRL+C to quit)
```

Figure 3.13: Tensorboard Activation Command Line.

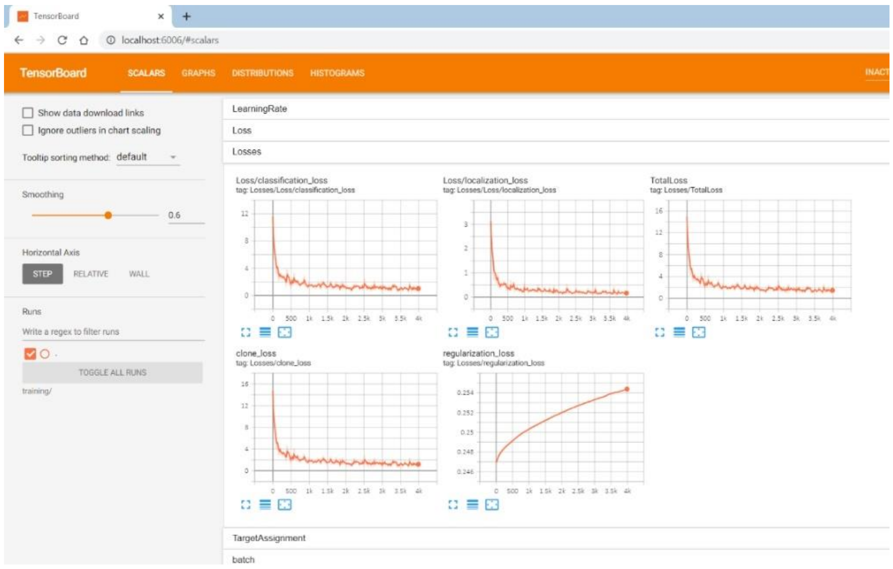
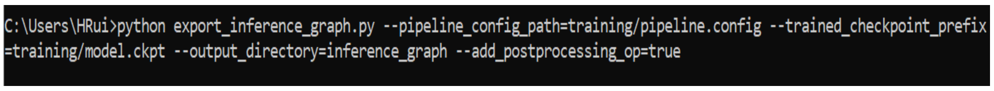


Figure 3.14: Tensorboard Interface.

3.5.6 Convert TensorFlow Lite Model

After the object detection is trained to a satisfactory level, to deploy this model in a mobile application, it requires to convert into TensorFlow Lite model. The trained model requires to export into a frozen inference graph for TensorFlow

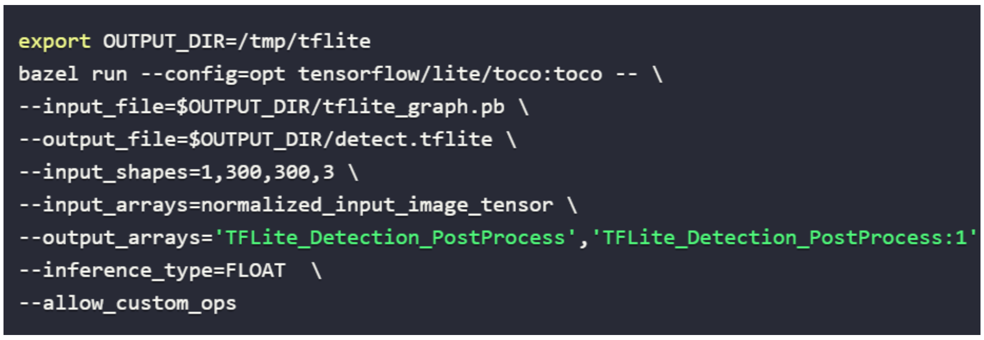
Lite, TensorFlow Object Detection API provides a conversion Python script called 'export_tflit_ssd_graph.py' to its user, this script only supports the conversion with SSD object detector model. Figure 3.15 shows an example of running the script in windows command prompt and some necessary information to pass in. Then, a '.pb' model file will be generated in the specified output directory (Tanner, 2020).



```
C:\Users\HRui>python export_inference_graph.py --pipeline_config_path=training/pipeline.config --trained_checkpoint_prefix
=training/model.ckpt --output_directory=inference_graph --add_postprocessing_op=true
```

Figure 3.15: Export Inference Graph Command Line.

To generate a TensorFlow Lite model, Figure 3.16 shows an example of running a conversion method provided by TensorFlow. The inference graph generated previously is required to pass into the command line. The output file will be saved in '.tflite' format which refers to TensorFlow Lite model (Tanner, 2020).



```
export OUTPUT_DIR=/tmp/tflite
bazel run --config=opt tensorflow/lite/toco:toco -- \
--input_file=$OUTPUT_DIR/tflite_graph.pb \
--output_file=$OUTPUT_DIR/detect.tflite \
--input_shapes=1,300,300,3 \
--input_arrays=normalized_input_image_tensor \
--output_arrays='TFLite_Detection_PostProcess','TFLite_Detection_PostProcess:1'
--inference_type=FLOAT  \
--allow_custom_ops
```

Figure 3.16: Convert Model into Tensorflow Lite Command Line (Tanner, 2020).

3.6 Mobile Application Development

For mobile application development, Android Studio platform with JAVA programming language will be used. Android Studio is limited to Android mobile application development only. Once the development process is complete, the application file can be export into '.apk' file which able to install in every Android smartphone (Verma, Kansal and Malvi, 2018). Figure 3.17 shows the process of android mobile application development.

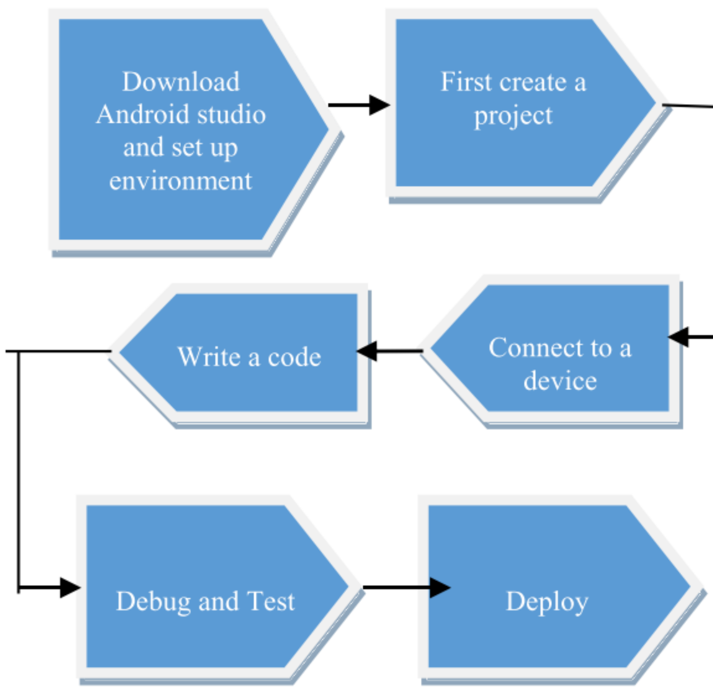
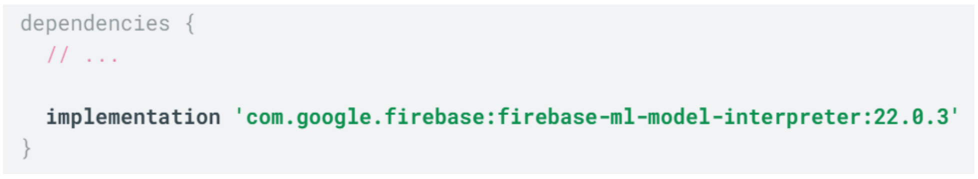


Figure 3.17: General Process to Build an Android Application (Verma, Kansal and Malvi, 2018).

3.6.1 Integration of TensorFlow Lite Model

To integrate the trained model into mobile application, Firebase ML Kit will be use. Firebase ML Kit supports any Tensorflow Lite model using its model interpreter API. To use Firebase ML Kit in Android application, simply adding Firebase ML Kit library into Android application dependencies (Figure 3.18).



```
dependencies {
  // ...

  implementation 'com.google.firebase:firebase-ml-model-interpreter:22.0.3'
}
```

Figure 3.18: Code of Adding Firebase ML Kit Library into Android Dependencies.

Then, create a Tensorflow Lite model interpreter by passing the '.tflite' model file path in the Android assets folder into the API. Next, specify the input dimension and the output dimension of the model using the API. Lastly, run the model using the created interpreter by passing in the input image.

In order to match incoming input image with model input dimension, the image must pre-process with downscaling to 224 x 224 dimension, and normalize input image 8-bit RGB value from range [0,255] into range [-1,1] using the formula (3.1) below which according to the original normalization method from TensorFlow.

$$normalized\ input\ RGB = RGB\ value\ \times \left(\frac{2}{255}\right) - 1 \qquad (3.1)$$

After the image is scaled and normalized, the image will pass into the interpreter for inference. Once the inference is successfully, it is required to process the generated outputs which are bounding box, confidence score, and predicted class. Generally, the bounding box of a prediction will be encoded in a normalized 4 elements array as shown in equation (3.2) below.

$$[\frac{y_{top}}{image\ height}, \frac{x_{left}}{image\ width}, \frac{y_{bottom}}{image\ height}, \frac{x_{right}}{image\ width}] \qquad (3.2)$$

where:
y_{top} = top y-coordinate of bounding box
x_{left} = left x-coordinate of bounding box
y_{bottom} = bottom y-coordinate of bounding box
x_{right} = right x-coordinate of bounding box

Therefore, to obtain the exact bounding box coordinate, every element in the array requires to be extracted and denormalized by multiplying image width and height. Using these exact coordinates will able to draw and display a correct bounding box on the image.

Besides that, for predicted class, the output class value will be encoded into 0 and 1 which are malignant and benign class respectively. By simply creating a reference for the encoded value with the exact class name will obtain the predicted class name. For confidence score of the prediction, it returns a normalized value ranging from 0 to 1. Multiply the value with 100% will get the exact percentage confidence score.

3.6.2 Mobile Application Functionalities

Functionalities of the mobile application will include a main activity which displays image, predicted result and some buttons such as run inference button, setting button, and document button. Also, a camera activity with flash light on to capture skin lesions with constant brightness at the same time automatically crop out region of interest from image. The purpose of this concept is to simulate a normal dermatoscope (cost nearly RM 1200) for example Figure 3.18 below, which commonly used by dermatologist. Moreover, it ensures the image similar to the train image which reduce the background noises. Figure 3.19 below shows that concept of automatic crop feature of the camera activity.

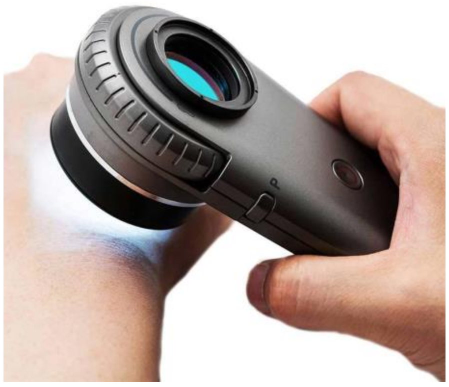


Figure 3.19: Normal Dermatoscope in Market.

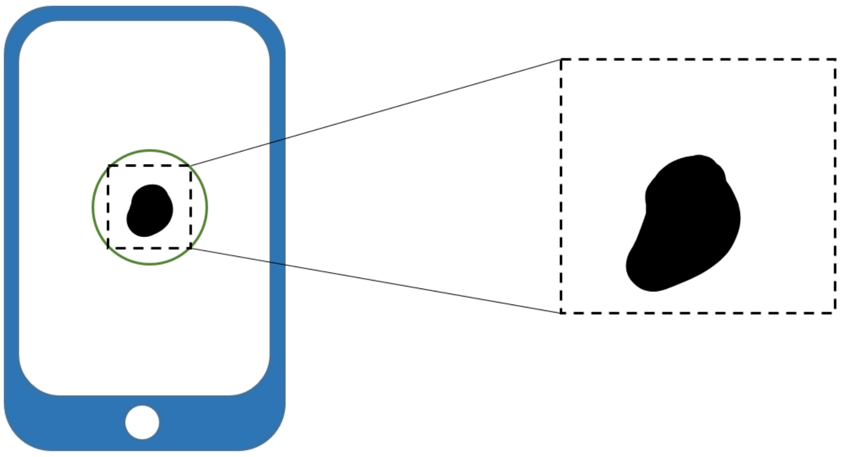


Figure 3.20: Camera Activity Auto Crop Concept to Simulate Dermatoscope.

In camera activity, users require to capture the lesions inside the center circle for auto cropping. Once the image captured, the region of interest will be slice out from the image, anything outside the circle will be discarded. Besides camera activity, add image from phone gallery function is an alternative method to perform inference. If user choose to add image from gallery, additional crop function will prompt user to crop out region of interest from the image. Once image added or captured, a button will need to prompt user to run inference on the image including drawing bounding box around lesions and display confidence, and class value on the image. On the other hand, a setting function enable user to adjust confidence threshold to show predicted result. Lastly, a documentation function will display ABCDE criteria with proper illustration for users to refer if manual detection is preferred and to overcome the public misunderstanding issue of ABCDE criteria mentioned by Tsao et al. (2015). All functionality activities are summarized into:

(i) *Main* Activity (display image and an inference button).
(ii) *Camera* Activity with flash light on and auto crop.
(iii) *Crop Image* Activity (if add image from gallery).
(iv) *Setting* Activity.
(v) *Document* Activity.

Moreover, a save image button will also require ensuring the user able to save the predicted image. Also, an edit button will require if user choose crop image activity to allow user to re-crop the image without re-select image from gallery.

3.6.3 Mobile Application Compatibility Test and Inference Time Tracing

The development of the mobile application mainly focuses on Android version 6.0 onwards. Besides, multiple dimension design layout will be created to tackle different screen sizes of Android smartphones however it should not able to support extremely large or small screen sizes. Also, the overall mobile application size should not exceed 50MB to maintain lightweight.

To test the compatibility of the application, Firebase Test Lab will be used. Firebase Test Lab is a cloud-based platform to test applications running on Android. It enables users to test applications across many types of devices and Android versions. It will automatically run the app, searching for crashes and bugs (Khawas and Shah, 2018). The complete application will be tested with 7 various screen sizes and Android version. If any application crashes occur during the test, the application would not be able to pass the test for that specific device.

Besides that, inference time of object detection model on the mobile application will be traced using Firebase Performance library. Adding Firebase Performance library in the mobile application allows to trace the running time for a part of the code by inserting start and stop trace function provided by the library. Since the model will be optimized to TensorFlow Lite model, the inference time should not exceed 1 seconds or more.

3.7 Summary

In summary, this chapter explains the project workflow thoroughly. The methods of approach are shown in detail. To develop a mobile application to detect skin lesions, the workflow can be summarized into dataset preparation, model training and evaluation, mobile application development, mobile application compatibility testing, and inference time tracing.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the result of object detection models and mobile application will be discussed. In section 4.2 a comparison of various object detection models and validation of the selected model with existing classification model will be discussed. Section 4.3 discusses about the result of mobile application development.

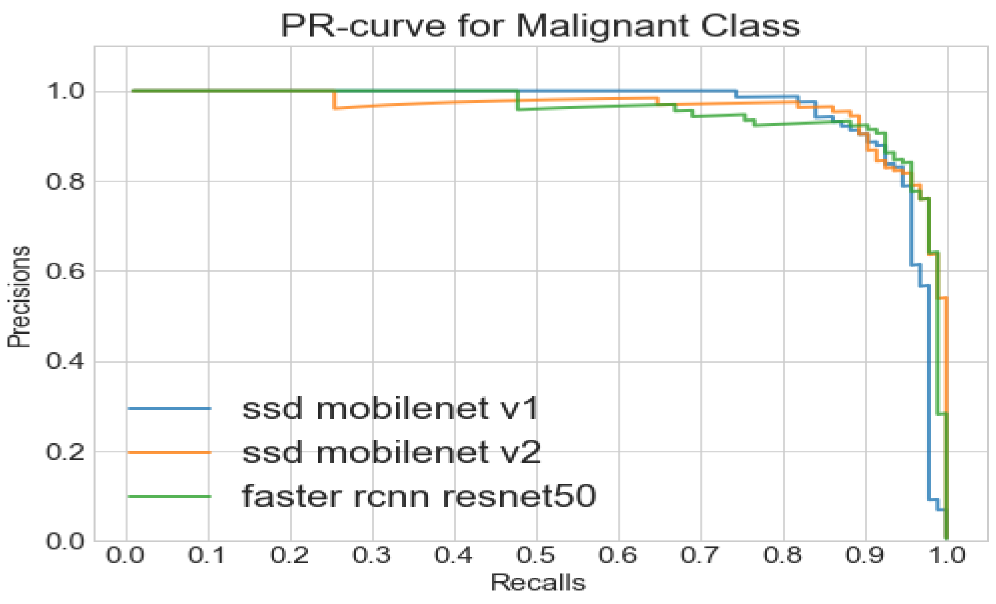
4.2 Object Detection Models Comparison

In this section, three object detection model SSD MobileNet V1, SSD MobileNet V2, and Faster-RCNN ResNet will be compared in terms of their mAP, inference time on a single image, model size, and PR-curve after training. All models are trained using the same dataset and evaluation PASCAL VOC metrics, however different in training step because training process can be terminated once Training Loss does not show any improvement. Data of evaluation mAP and PR-curve are obtained from Tensorboard whereas inference time on single image data obtained from self-written python script. Table 4.1 below shows the respective results from three models. Figure 4.1 shows the PR-curves of three models for benign and malignant skin lesion classes.

Table 4.1: Object Detection Models Comparison in terms of mAP, Inference time, and Model Size.

Model	mAP	Inference time (single image)	Model Size
SSD MobileNet V1	92.89%	1.91s	22 MB
SSD MobileNet V2	93.99%	1.79s	18 MB
Faster-RCNN ResNet	95.29%	14.53s	112 MB

(a)



(b)

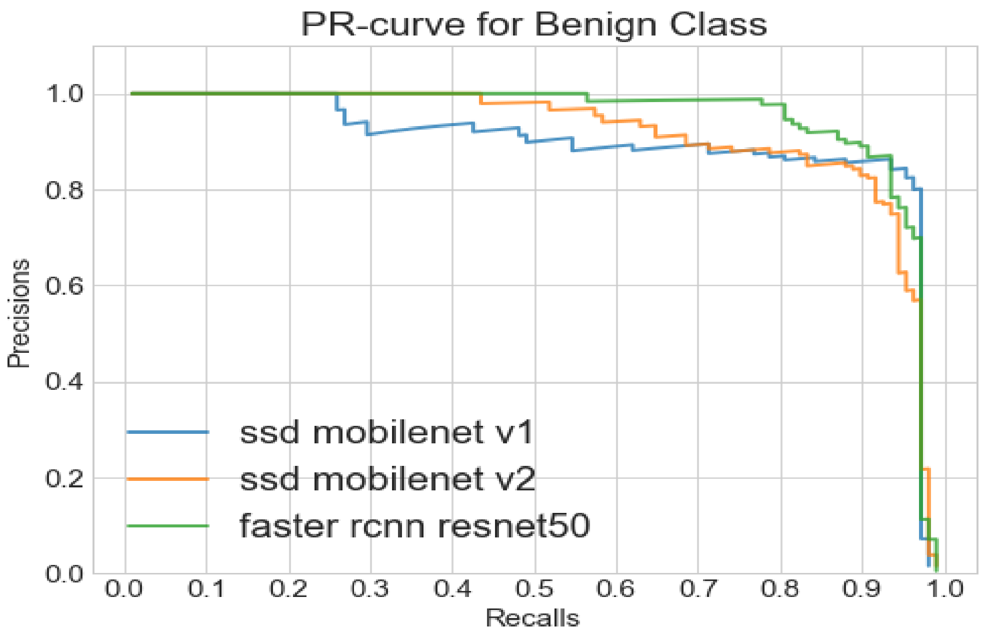


Figure 4.1: (a) PR-curve of three models for malignant skin lesion class. (b) PR-curve of three models for benign skin lesion class.

From the result (Table 4.1), no doubt that Faster-RCNN ResNet obtains the highest score in mAP. According to Bianco et al. (2018) benchmark analysis of feature extraction network and Zhao et al. (2019) reviews of object detection network, Faster-RCNN object detector, and ResNet feature extraction network give higher localization and classification accuracy compared to the others. This high accuracy achievement from Faster-RCNN ResNet has traded off on its inference speed of 14.53 seconds with one single image on a computer. Compare to SSD MobileNet V1 and V2, Faster-RCNN ResNet has a relatively slower inference speed, due to the number of parameters, multiplication and addition operation much higher within ResNet feature extractor (Reddy, Rattani and Derakhshani, 2018). This reason also leads to higher model size of 112 MB for Faster-RCNN ResNet model.

Meanwhile comparing SSD MobileNet V1 and V2 model, SSD MobileNet V2 shows more advantages than SSD MobileNet V1 in terms of all the data due to MobileNet V2 has some improvement from the MobileNet V1 version with network architecture changes. These changes decrease the number of parameters, multiplication, and addition operation in MobileNet V2 at the same time improve its accuracy.

According to Bränström et al. (2002), some degree of overdiagnosis of benign skin lesions is better than any degree of under-diagnosis of malignant skin lesions after they experimented with layperson’s ability to differentiate between these two types of skin lesions. This refers that a benign skin lesion predicted as a malignant class can be acceptable, but not encourage for a malignant skin lesion predicted as a benign class. By following this idea, recalls more important than precisions for malignant class, and precisions more important than recalls for benign class in PR-curve. Figure 4.1 (a) shows that three models have the same recalls of 1 in predicting malignant skin lesions. In this case, precision becomes the priority of performance measuring for malignant class. At recalls close to 1 in Figure 4.1 (a) SSD MobileNet V2 has the highest precision among all the other models, whereas Faster-RCNN ResNet has higher precision than SSD MobileNet V1. From Figure 4.1 (b), Faster-RCNN has the highest precision among other models. Besides that, SSD MobileNet V2 precision is higher than the V1 model. A clearer comparison on the PR-curve can be summarized by taking the area under the PR-curve, also known as Average Precision shown in Table 4.2 below.

Table 4.2: Area Under PR-curve (Average Precision) for Benign and Malignant Class of Three Models.

	Predicting Class	
Models	**Benign Class**	**Malignant Class**
SSD MobileNet V1	90.06%	95.73%
SSD MobileNet V2	91.67%	96.30%
Faster-RCNN ResNet	94.92%	95.66%

From the analysis above, Faster-RCNN has the top accuracy however bad in inference speed and model size. SSD MobileNet V2 rank on the second regarding accuracy but possess highest inference speed and smallest model size. To choose from these two models, since finding a lightweight and fast inference speed model is one of the objectives in this project, this experimental comparison matches the findings in Chapter 2 which proves that the pre-chosen model SSD MobileNet V2 during studies was more suitable for mobile phone implementation.

4.2.1 Model Validation

In this section, trained SSD MobileNet V2 model will be compared with an existing model from a researcher. Since no existing object detection model related to this project, an existing high accuracy ResNet50 classification model is obtained from Fanconi (2020) on Kaggle website. This existing model is also trained with the same dataset however training set contains 2637 images. Due to this reason, only Accuracy metric and Confusion Matrix could use to compare performance between an object detection model and classification model. A Confusion Matrix is a table with True Positive, True Negative, False Positive, False Negative value recorded. The accuracy is as stated in Equation 4.1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{4.1}$$

where:

TP = True Positive

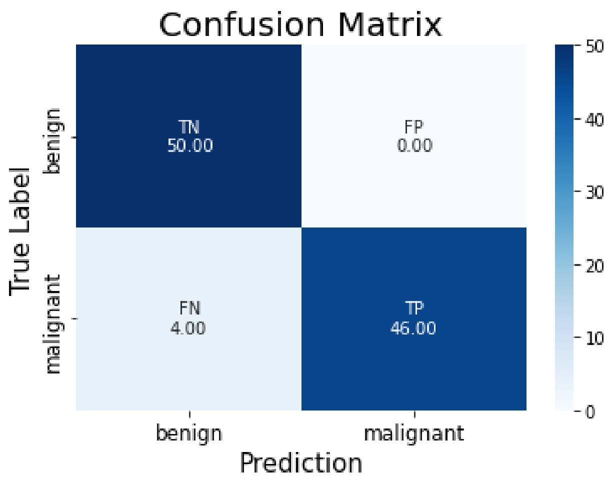
TN = True Negative

FP = False Positive

FN = False Negative

Besides, another dataset which contains dimension resolution 224 x 224 pixels of 50 benign and 50 malignant skin lesions images is selected from ISIC archive website for validation. The validation results are generated using self-written Python script by running inference on the validation dataset with both models. The Confusion Matrix for both models are shown in Figure 4.2.

(a)



(b)

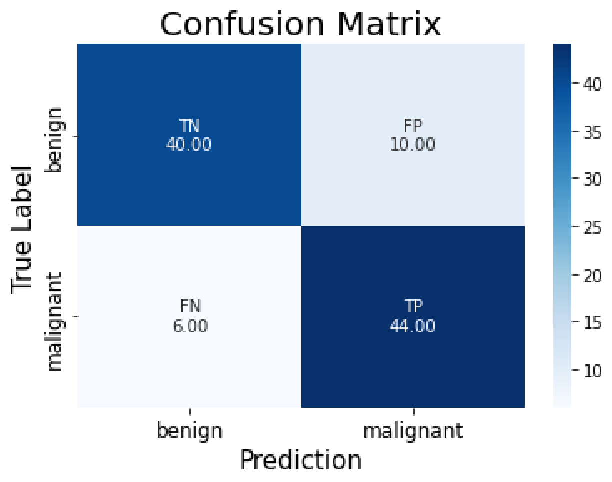


Figure 4.2: Confusion Matrix for (a) SSD MobileNet V2 object detection model, (b) ResNet50 classification model.

Table 4.3: Accuracy of SSD MobileNet V2 object detection model and ResNet50 classification model using a new validation dataset.

Model	Accuracy
SSD MobileNet V2	96%
ResNet50	84%

Figure 4.2 (a) shows that SSD MobileNet V2 has none false predict benign skin lesions as malignant, however has 4 malignant skin lesions predicted as benign. On the other hand, in Figure 4.2 (b) classification ResNet50 model has 10 false predict on benign skin lesions as malignant class, and 6 malignant skin lesions predicted as benign. From these values, Accuracy is calculated as shown in Table 4.3. It shows that SSD MobileNet V2 has higher accuracy than classification ResNet50 model on this validation dataset.

From analysis above, the performance of SSD MobileNet V2 in this project surpasses the existing model provided by the researcher. However, the generalization of the current model on this detection task has not yet been proven since it does not undergo any proper clinical assessment.

4.3 Mobile Application Result

In this section, the screenshots on the graphical user interface of various functions of the mobile application are presented and discussed. The screenshots include all the activities such as:

(i) *Main* Activity (display image and an inference button).

(ii) *Camera* Activity with flashlight on and auto-crop.

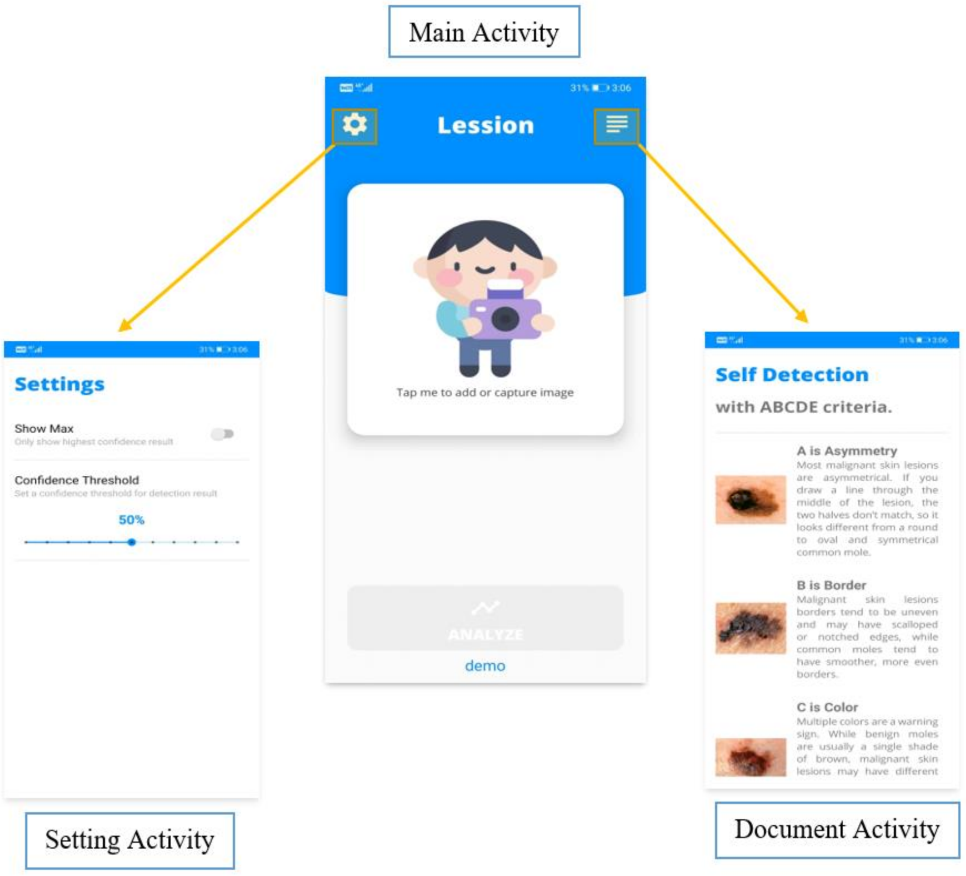
(iii) *Crop Image* Activity (if add image from gallery).

(iv) *Setting* Activity

(v) *Document* Activity

4.3.1 Main, Setting, Document Activities

Figure 4.3 shows the screenshots of *Main*, *Setting*, and *Document* activities and their respective invoke button especially *Setting* and *Document* activity.



Settings

Show Max
Only show highest confidence result

Confidence Threshold
Set a confidence threshold for detection result
50%

Self Detection

with ABCDE criteria.

A is Asymmetry
Most malignant skin lesions are asymmetrical. If you draw a line through the middle of the lesion, the two halves don't match, so it looks different from a round to oval and symmetrical common mole.

B is Border
Malignant skin lesions borders tend to be uneven and may have scalloped or notched edges, while common moles tend to have smoother, more even borders.

C is Color
Multiple colors are a warning sign. While benign moles are usually a single shade of brown, malignant skin lesions may have different

Figure 4.3: *Main*, *Setting*, and *Document* Activities.

Figure 4.3 above shows that *Setting* activity can be invoking from the gear icon at the top left corner of the *Main* activity. In the *Setting* activity, users allow to adjust the confidence threshold of prediction with a seek bar, and able to switch whether to display the prediction which has highest confidence. On the other hand, the *Document* activity can be invoked from the document icon at the top right corner of the *Main* activity. The *Document* activity consists of a scrollable instruction with proper illustration of ABCDE self-detection criteria for the user to refer.

4.3.2 Crop Image and Camera Activities

In this section, interface of *Crop Image* and *Camera* activities is shown and discussed. Besides that, the flow of using these two activities to perform inference on an image is discussed as well.

To add an image for inference, users will be prompted with an intent chooser dialog to select either capture image with camera auto crop or choose an image from gallery to crop region of interest. Figure 4.4 below shows the action flow as stated above.

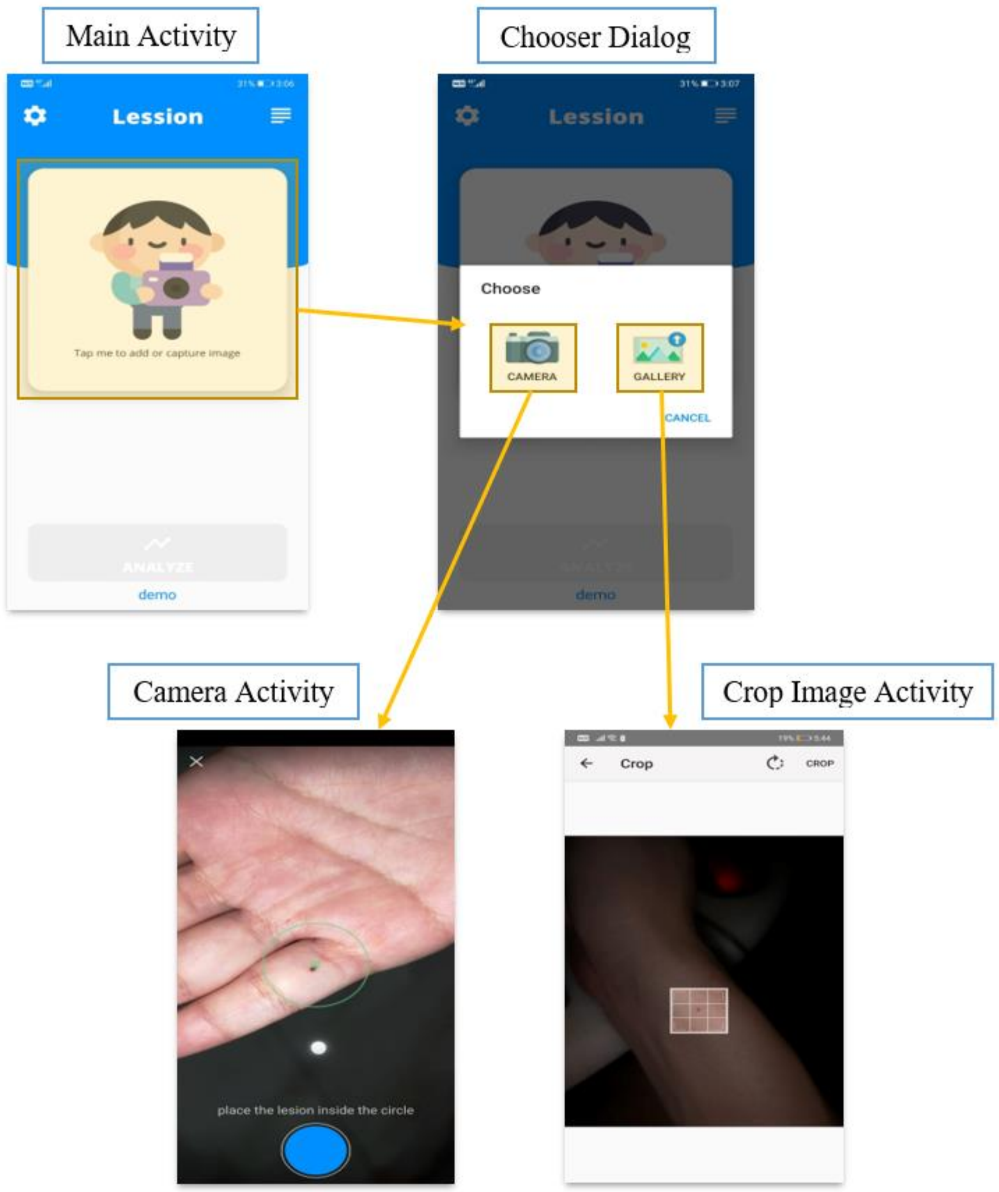


Figure 4.4: Action Flow for *Camera* and *Crop Image* Activity.

As shown in Figure 4.4, the users are required to tap on the center frame located at *Main* activity to invoke the chooser dialog. If the users intend to use the camera to capture a picture of skin lesions, they can tap on the *camera* button on the dialog to invoke the *Camera* activity. In *Camera* activity, to ensure the captured image automatically crop out the region of interest, the users will be required to place the lesion inside the circle. Besides that, the users are able to tap on the screen to ensure the image is well focus and clear.

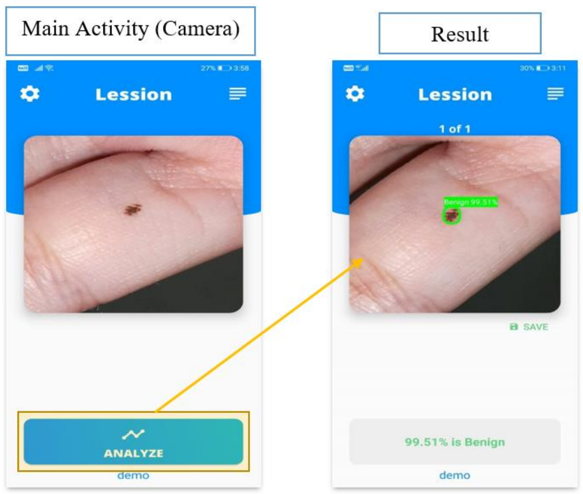
On the other hand, if the users intend to add an image from the gallery, they can tap on the gallery button to invoke an image selection. Once the image is selected, the users will be prompted with a region of interest crop to crop out the lesion from the selected image.

4.3.3 Object Detection Model Integration and Inference

The trained SSD MobileNet V2 object detection model is converted into TensorFlow Lite model and successfully integrated into the mobile application. A TensorFlow Lite interpreter is created using Firebase ML Kit.

Once the image is cropped or captured, the application will exit from the previous activity and return to *Main* activity to perform preprocessing (downscaling and normalization) and inference on the image with the integrated model. The inference process includes passing image into interpreter, run inference, draw bounding box and display info on the image. Figure 4.5 below shows the action flow to perform inference in the mobile application.

(a)



(b)

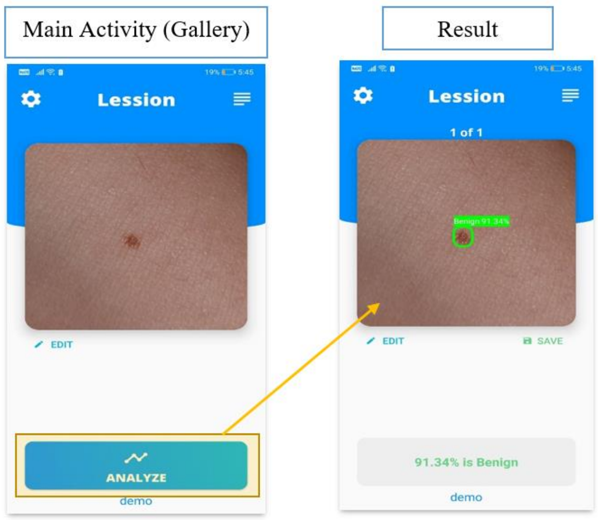


Figure 4.5: Action Flow of Inference (a) if Image Captured from Camera (b) if an Image is being Added from the Gallery.

From Figure 4.5, notice that both (a) and (b) looks similar, however, if the user previously captures an image from camera, the *edit* button at the bottom left corner of the center frame will not show up since the image already automatically cropped. If the user previously crops and add image from gallery, the edit button pops up to allow users able to re-crop the image without reselecting from gallery. After the tap on the *analyze* inference button located at the bottom of *Main* activity, the result will show on the image with bounding box, class, and confidence value. Lastly, users able to save the predicted image using the *save* button at the bottom right corner of the center frame.

4.3.3 Application Compatibility Test

After the development, the application exported into '.apk' file. The '.apk' file is uploaded to Firebase Test Lab for compatibility test. The test runs through all activities and buttons in the application multiple time to ensure every function or action compatible with respective Android version. The application tested with 7 different smartphones with various Android versions ranging from 6.0 onwards and screen resolution. The test result is shown in Table 4.4.

Table 4.4: Firebase Test Lab Results.

Smartphone Model	Screen Resolution	Android Version	Result
Motorola G Play	720 x 1280	6.0	Pass
Huawei Mate 9	1080 x 1920	7.0	Pass
Nexus 6	1440 x 2560	7.1	Pass
Lenovo S5	1080 x 2160	8.0	Pass
Google Pixel 2 XL	1440 x 2880	8.1	Pass
Motorola One	720 x 1520	9.0	Pass
Google Pixel 2	1080 x 1920	10.0	Pass

Table 4.4 above shows that the application passes all the tests without any application crash or bugs occur. Every activity in the mobile application has been tested on each smartphone. Besides, none of the UI such as buttons or shapes in the mobile application overlaps with each other on various screen resolutions.

4.3.4 Inference Time and Application Size

Firebase able traces every inference (include time of drawing bounding box, class, and confidence value on image) on a single image and records its time. This result can obtain from Firebase console. On the last 30 days, from 1st August to 30th August, Firebase already recorded 1300 samples of inference time from 4 different physical smartphones. The inference time distribution graph is shown in Figure 4.6 below.

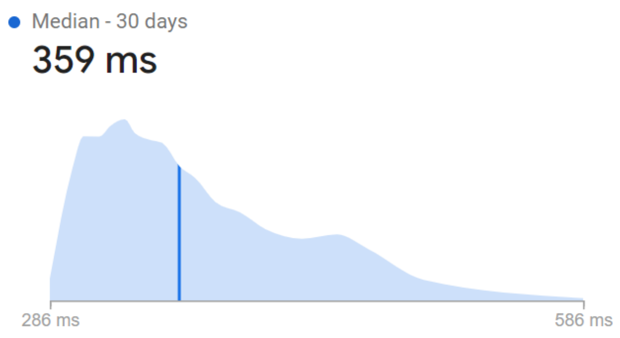


Figure 4.6: Inference Time Distribution of 1300 samples collected for the past 30 days.

Figure 4.6 shows that average inference time falls around 359ms, with maximum value of 586ms and minimum value of 286ms, which is surprisingly fast. The result is excitingly lower than the preset 1 seconds in Chapter 3. Besides that, the inference time showing much lower than the inference time on a computer. One possible reason might be due to the trained model already been converted into a smartphone-optimized model (TensorFlow Lite). Hence, this proves that the integration of TensorFlow Lite object detection model in the mobile application will not cause any computation burden to a smartphone.

For application size, the finalized application size 30.31MB, which does not exceed the preset 50MB in Chapter 3. This proves that a mobile application with a smartphone suitable object detection model will not occupy too much spaces in a smartphone storage.

4.4 Summary

In summary, the model selection result tally with the previous literature findings on object detection model selection, in other words, SSD MobileNet V2 is selected due to its lightweight architecture, meanwhile, achieve 93.9% of evaluation mAP and lowest detection time among others. The validation result also shows that the selected model surpasses other researcher's classification model in terms of accuracy. Besides, the development of mobile application is successful and has achieved the stated objectives in this project.

In short, this mobile application allows users to detect malignant and benign skin lesions using an Android based smartphone which able to replace the conventional ABCDE criteria self-detection method. Complete coding and system including the object detection model of the mobile application in this project have been uploaded to GitHub website (https://github.com/tanhouren/FYP-skin-lesion-detection-mobile-app) to serve as a contribution to skin cancer diagnosis.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In this project, it is found that most existing skin lesions diagnosis with deep learning technology stops at deep learning modelling, and without any further deployment or integration with a readily available device such as smartphone. However, the advantage of integrating object detection deep learning technology and smartphone in the medical field has been discovered throughout the project. This technology is able to provide low-cost diagnosis and without require years of skin lesion diagnosis experience. Moreover, users able to perform diagnosis at home with a smartphone, therefore can provide point-of-care to the users from a remote area.

Although the process of development is challenging due to the immature platform of object detection development (TensorFlow Object Detection API) and require experiences for Android application development, the success of this project has proven that the development of this technology is feasible and should be aware.

In short, the objectives of this project have been achieved. Besides using ABCDE criteria conventional self-detection method, users able to use smartphone to perform self-detection on malignant and benign skin lesions with this mobile application. This mobile application supports a wide range of Android smartphones and does not occupy huge internal storage of smartphones. The integrated object detection model has achieved fast detection time and higher accuracy after validating with the existing classification model.

5.2 Recommendations for future work

The integration of telemedicine technology is highly recommended such that it provides interaction between users and dermatologists, for example giving advice and extra assessment. Also, dermatologists able to receive predicted images from their patients to constantly monitor the condition more effectively.

With telemedicine integrated, the system can achieve as a more complete and professional skin cancer diagnosis tool.

Besides, to obtain more convincing object detection performance, a larger dataset for training is required to improve the generalization of the model on detecting malignant and benign skin lesions. Besides, some image preprocessing methods such as image-denoise or contrast enhancement can be applied to emphasize the features of the skin lesions therefore increase model accuracy. On the other hand, the functionality of the Camera activity in the mobile application should be improved with the aid of real-time detection which able to ensure the stability of every generated detection hence improve user experience. Also, a reminder function can be implemented to remind users to observe or perform detection on suspicious skin lesions periodically since malignant skin lesions will evolve in shape over time.

Lastly, due to the advantages such as processing capability, and high-resolution image capture provided by smartphones nowadays, the integration of object detection deep learning with smartphone application can be applied not only for skin lesions detection. In future, this technology can be applied into other medical field with the same detection method developed in this project such as foot ulcer detection, ear infection detection, and more to provide efficient and low-cost point-of-care diagnosis.

REFERENCES

Abbasi, N.R., Shaw, H.M., Rigel, D.S., Friedman, R.J., McCarthy, W.H., Osman, I., Kopf, A.W. and Polsky, D., 2004. Early Diagnosis of Cutaneous Melanoma. *Jama*, 292(22), p.2771.

Abuzaghleh, O., Barkana, B.D. and Faezipour, M., 2015. Noninvasive Real-Time Automated Skin Lesion Analysis System for Melanoma Early Detection and Prevention. *IEEE Journal of Translational Engineering in Health and Medicine,* [online] 3, pp.1–12. Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/7079463>.

Adegun, A.A. and Viriri, S., 2020. Deep learning-based system for automatic melanoma detection. *IEEE Access*, 8, pp.7160–7172.

Al-Masni, M.A., Kim, D.H. and Kim, T.S., 2020. Multiple skin lesions diagnostics via integrated deep convolutional networks for segmentation and classification. *Comput Methods Programs Biomed*, [online] 190, p.105351. Available at: <https://www.ncbi.nlm.nih.gov/pubmed/32028084>.

Albahar, M.A., 2019. Skin Lesion Classification Using Convolutional Neural Network with Novel Regularizer. *IEEE Access*, 7, pp.38306–38313.

AIM at Melanoma Foundation. 2020. *Moles & Other Lesions - AIM At Melanoma Foundation*. [online] Available at: <https://www.aimatmelanoma.org/about-melanoma/other-lesions/> [Accessed 21 April 2020].

Aidouni, M., 2020. *Evaluating Object Detection Models: Guide To Performance Metrics*. [online] Manal El Aidouni. Available at: <https://manalelaidouni.github.io/manalelaidouni.github.io/Evaluating-Object-Detection-Models-Guide-to-Performance-Metrics.html#precision-x-recall-curve> [Accessed 21 April 2020].

Amelard, R., Glaister, J., Wong, A. and Clausi, D.A., 2015. High-Level Intuitive Features (HLIFs) for intuitive skin lesion description. *IEEE Transactions on Biomedical Engineering*, 62(3), pp.820–831.

Azizpour, H., Razavian, A.S., Sullivan, J., Maki, A. and Carlsson, S., 2015. From generic to specific deep representations for visual recognition. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. pp.36–45.

Bianco, S., Cadene, R., Celona, L. and Napoletano, P., 2018. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6, pp.64270–64277.

Bränström, R., Hedblad, M.A., Krakau, I. and Ullén, H., 2002. Laypersons' perceptual discrimination of pigmented skin lesions. *Journal of the American Academy of Dermatology*, 46(5), pp.667–673.

Cdc.gov., 2020. *What Is Skin Cancer? | CDC*. [online] Available at: <https://www.cdc.gov/cancer/skin/basic_info/what-is-skin-cancer.htm> [Accessed 21 April 2020].

Chamberlain, A.J., Fritschi, L. and Kelly, J.W., 2003. Nodular melanoma: Patients' perceptions of presenting features and implications for earlier detection. *Journal of the American Academy of Dermatology*, [online] 48(5), pp.694–701. Available at: <https://linkinghub.elsevier.com/retrieve/pii/S0190962203000227>.

Clark Jr., W.H., Elder, D.E., Guerry IV, D., Braitman, L.E., Trock, B.J., Schultz, D., Synnestvedt, M. and Halpern, A.C., 1989. Model Predicting Survival in Stage I Melanoma Based on Tumor Progression. *JNCI: Journal of the National Cancer Institute*, [online] 81(24), pp.1893–1904. Available at: <https://doi.org/10.1093/jnci/81.24.1893>.

Deng, Y., 2019. Deep learning on mobile devices: a review. In: S.S. Agaian, S.P. DelMarco and V.K. Asari, eds. *Mobile Multimedia/Image Processing, Security, and Applications 2019*. [online] SPIE.p.11. Available at: <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10993/2518469/Deep-learning-on-mobile-devices-a-review/10.1117/12.2518469.full>.

DeepAI. 2020. *Jaccard Index*. [online] Available at: <https://deepai.org/machine-learning-glossary-and-terms/jaccard-index> [Accessed 21 April 2020].

Doben, A.R. and MacGillivray, D.C., 2009. Current Concepts in Cutaneous Melanoma: Malignant Melanoma. *Surgical Clinics of North America*, [online] 89(3), pp.713–725. Available at: <http://www.sciencedirect.com/science/article/pii/S0039610909000371>.

Everingham, M., Everingham, M., Zisserman, A., Zisserman, A., Williams, C. and Williams, C., 2006. The PASCAL visual object classes challenge 2006 (VOC2006) results. *Workshop in ECCV06, May. Graz, Austria*, [online] 2006(January 2006). Available at: <http://scholar.google.co.uk/scholar?start=10&q=%22bag+of+visual+words%22&hl=en#3>.

Farberg, A.S. and Rigel, D.S., 2017. The Importance of Early Recognition of Skin Cancer. *Dermatologic Clinics*, 35(4), pp.xv–xvi.

Friedman, R.J., Rigel, D.S. and Kopf, A.W., 1985. Early Detection of Malignant Melanoma: The Role of Physician Examination and Self-Examination of the Skin. *CA: A Cancer Journal for Clinicians*, [online] 35(3), pp.130–151. Available at: <https://doi.org/10.3322/canjclin.35.3.130>.

Fanconi, C., 2020. *Skin Cancer: Malignant Vs. Benign*. [online] Kaggle.com. Available at: <https://www.kaggle.com/fanconic/skin-cancer-malignant-vs-benign> [Accessed 17 April 2020].

Giotis, I., Molders, N., Land, S., Biehl, M., Jonkman, M. and Petkov, N., 2015. MED-NODE: A Computer-Assisted Melanoma Diagnosis System using Non-Dermoscopic Images. *Expert Systems with Applications*, 42.

Girshick, R., 2015. Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter, pp.1440–1448.

Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp.580–587.

GitHub. 2020. *Tensorflow/Models*. [online] Available at: <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md> [Accessed 17 April 2020].

GitHub. 2020. *Tzutalin/Labelimg*. [online] Available at: <https://github.com/tzutalin/labelImg> [Accessed 18 April 2020].

Glazer, A.M., Rigel, D.S., Winkelmann, R.R. and Farberg, A.S., 2017. Clinical Diagnosis of Skin Cancer: Enhancing Inspection and Early Recognition. *Dermatologic Clinics*, 35(4), pp.409–416.

Goyal, M., Reeves, N., Rajbhandari, S. and Yap, M.H., 2018. Robust Methods for Real-Time Diabetic Foot Ulcer Detection and Localization on Mobile Devices. *IEEE Journal of Biomedical and Health Informatics*, PP, p.1.

Harangi, B., 2018. Skin lesion classification with ensembles of deep convolutional neural networks. *Journal of Biomedical Informatics*, [online] 86(June), pp.25–32. Available at: <https://doi.org/10.1016/j.jbi.2018.08.006>.

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp.770–778.

Hosny, K.M., Kassem, M.A. and Foaud, M.M., 2019. Classification of skin lesions using transfer learning and augmentation with Alex-net. *PLOS ONE*, [online] 14(5), p.e0217293. Available at: <http://dx.plos.org/10.1371/journal.pone.0217293>.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. [online] Available at: <http://arxiv.org/abs/1704.04861>.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. and Murphy, K., 2017. Speed/accuracy trade-offs for modern convolutional object detectors. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua, pp.3296–3305.

Ignatov, A., Timofte, R., Chou, W., Wang, K., Wu, M., Hartley, T. and Van Gool, L., 2019. AI Benchmark: Running deep neural networks on android smartphones. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11133 LNCS, pp.288–314.

Jerant, A.F., Johnson, J.T., Sheridan, C.D. and Caffrey, T.J., 2000. *Early detection and treatment of skin cancer*. [online] American family physician. Available at: <https://www.aafp.org/afp/2000/0715/p357.html> [Accessed 17 Mar. 2020].

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z. and Qu, R., 2019. A survey of deep learning-based object detection. *IEEE Access*, 7, pp.128837–128868.

Khawas, C. and Shah, P., 2018. Application of Firebase in Android App Development-A Study. *International Journal of Computer Applications*, [online] 179(46), pp.49–53. Available at: <http://www.ijcaonline.org/archives/volume179/number46/khawas-2018-ijca-917200.pdf>.

Korotkov, K. and Garcia, R., 2012. Computerized analysis of pigmented skin lesions: A review. *Artificial Intelligence in Medicine*, [online] 56(2), pp.69–90. Available at: <http://dx.doi.org/10.1016/j.artmed.2012.08.002>.

Krizhevsky, A., Hinton, G.E., Sutskever, I. and Hinton, G.E., 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, 25, pp.1–9.

Lecun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. *Nature*, 521(7553), pp.436–444.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014. *Microsoft COCO: Common objects in context*. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, .

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016. SSD: Single shot multibox detector. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9905 LNCS, pp.21–37.

Lodde, G., Zimmer, L., Livingstone, E., Schadendorf, D. and Ugurel, S., 2020. Malignant melanoma. *Hautarzt*.

Mazurowski, M.A., Habas, P.A., Zurada, J.M., Lo, J.Y., Baker, J.A. and Tourassi, G.D., 2008. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural Networks*, 21(2–3), pp.427–436.

Mayo Clinic. 2020. *Melanoma - Symptoms And Causes*. [online] Available at: <https://www.mayoclinic.org/diseases-conditions/melanoma/symptoms-causes/syc-20374884> [Accessed 21 April 2020].

McWhirter, J.E. and Hoffman-Goetz, L., 2013. Visual images for patient skin self-examination and melanoma detection: A systematic review of published&#xa0;studies. *Journal of the American Academy of Dermatology*, [online] 69(1), pp.47-55.e9. Available at: <https://doi.org/10.1016/j.jaad.2013.01.031>.

Mendonca, T., Ferreira, P.M., Marques, J.S., Marcal, A.R.S. and Rozeira, J., 2013. PH^2 - A dermoscopic image database for research and benchmarking. In: *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. [online] IEEE.pp.5437–5440. Available at: <http://ieeexplore.ieee.org/document/6610779/>.

Mhealth.org. 2020. *Five Things You Should Know About Skin Lesions | Mhealth.Org*. [online] Available at: <https://www.mhealth.org/blog/2018/october-2018/five-things-you-should-know-skin-lesions> [Accessed 21 April 2020].

Moroney, L., 2020. *Using Tensorflow Lite On Android*. [online] Blog.tensorflow.org. Available at: <https://blog.tensorflow.org/2018/03/using-tensorflow-lite-on-android.html> [Accessed 19 April 2020].

Nasr-Esfahani, E., Samavi, S., Karimi, N., Soroushmehr, S.M.R., Jafari, M.H., Ward, K. and Najarian, K., 2016. Melanoma detection by analysis of clinical images using convolutional neural network. In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*.

[online] IEEE.pp.1373–1376. Available at: <http://ieeexplore.ieee.org/document/7590963/>.

Ouyang, W., Wang, X., Zhang, C. and Yang, X., 2016. Factors in finetuning deep model for object detection with long-tail distribution. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp.864–873.

Pai, K. and Giridharan, A., 2019. Convolutional Neural Networks for classifying skin lesions. In: *IEEE Region 10 Annual International Conference, Proceedings/TENCON*. IEEE.pp.1794–1796.

Pathak, A.R., Pandey, M. and Rautaray, S., 2018. Application of Deep Learning for Object Detection. In: *Procedia Computer Science*. [online] Elsevier B.V.pp.1706–1717. Available at: <https://doi.org/10.1016/j.procs.2018.05.144>.

Padilla, R., 2020. *Rafaelpadilla/Object-Detection-Metrics*. [online] GitHub. Available at: <https://github.com/rafaelpadilla/Object-Detection-Metrics> [Accessed 21 April 2020].

Penatti, O.A.B., Nogueira, K. and Santos, J.A. dos, 2015. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In: *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. pp.44–51.

Rawat, W. and Wang, Z., 2017. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, [online] 29(9), pp.2352–2449. Available at: <https://doi.org/10.1162/neco_a_00990>.

Razavian, A.S., Azizpour, H., Sullivan, J. and Carlsson, S., 2014. CNN features off-the-shelf: An astounding baseline for recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp.512–519.

Reddy, N., Rattani, A. and Derakhshani, R., 2018. Comparison of deep learning models for biometric-based mobile user authentication. In: *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems, BTAS 2018*.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem, pp.779–788.

Ren, S., He, K., Girshick, R. and Sun, J., 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on*

Pattern Analysis and Machine Intelligence, 39(6), pp.1137–1149.

Rose, L., 2020. *Recognizing Neoplastic Skin Lesions: A Photo Guide*. [online] Aafp.org. Available at: <https://www.aafp.org/afp/1998/0915/p873.html> [Accessed 21 April 2020].

Romero-Lopez, A., Giro-i-Nieto, X., Burdick, J. and Marques, O., 2017. Skin Lesion Classification from Dermoscopic Images Using Deep Learning Techniques. In: *Biomedical Engineering*. [online] Calgary,AB,Canada: ACTAPRESS.pp.49–54. Available at: <http://www.actapress.com/PaperInfo.aspx?paperId=456417>.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C., 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE.pp.4510–4520.

Shin, H., Roth, H.R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D. and Summers, R.M., 2016. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, 35(5), pp.1285–1298.

Simonyan, K. and Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. pp.1–14.

Soviany, P. and Ionescu, R.T., 2018. Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction. In: *Proceedings - 2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2018*. pp.209–214.

Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B. and Liang, J., 2016. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? *IEEE Transactions on Medical Imaging*, 35(5), pp.1299–1312.

Taqi, A., al azzo, F., Awad, A. and Milanova, M., 2019. Skin Lesion Detection by Android Camera based on SSD-Mo- bilenet and TensorFlow Object Detection API. *International Journal of Advanced Research*, 3(July), pp.5–11.

Tanner, G., 2020. *Convert Your Tensorflow Object Detection Model To Tensorflow Lite.*. [online] Gilberttanner.com. Available at: <https://gilberttanner.com/blog/convert-your-tensorflow-object-detection-model-to-tensorflow-lite> [Accessed 19 April 2020].

TensorFlow. 2020. *Tensorflow*. [online] Available at: <https://www.tensorflow.org/> [Accessed 22 April 2020].

The Skin Cancer Foundation. 2020. *Skin Cancer Facts & Statistics - The Skin Cancer Foundation*. [online] Available at: <https://www.skincancer.org/skin-cancer-information/skin-cancer-facts/> [Accessed 21 April 2020].

The Skin Cancer Foundation. 2020. *Melanoma Stages - The Skin Cancer Foundation*. [online] Available at: <https://www.skincancer.org/skin-cancer-information/melanoma/the-stages-of-melanoma/> [Accessed 21 April 2020].

Tsao, H., Olazagasti, J.M., Cordoro, K.M., Brewer, J.D., Taylor, S.C., Bordeaux, J.S., Chren, M.-M., Sober, A.J., Tegeler, C., Bhushan, R. and Begolka, W.S., 2015. Early detection of melanoma: reviewing the ABCDEs. *Journal of the American Academy of Dermatology*, 72(4), pp.717–723.

Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T. and Smeulders, A.W.M., 2013. Selective Search for Object Recognition. *International Journal of Computer Vision*, [online] 104(2), pp.154–171. Available at: <https://doi.org/10.1007/s11263-013-0620-5>.

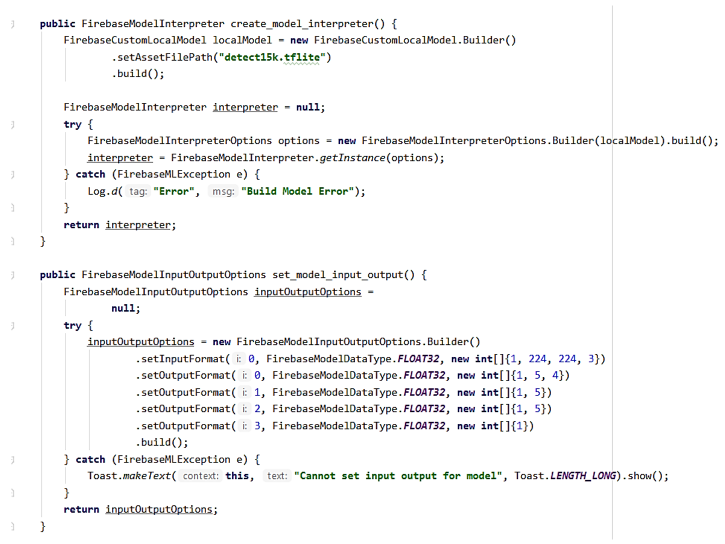
Verma, N., Kansal, S. and Malvi, H., 2018. Development of Native Mobile Application Using Android Studio for Cabs and Some Glimpse of Cross Platform Apps. *International Journal of Applied Engineering Research*, [online] 13(16), pp.12527–12530. Available at: <http://www.ripublication.com>.

Wu, X., Sahoo, D. and Hoi, S.C.H.H., 2020. Recent advances in deep learning for object detection. *Neurocomputing*, [online] (xxxx). Available at: <http://www.sciencedirect.com/science/article/pii/S0925231220301430>.

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*, 30(11), pp.3212–3232.

APPENDICES

APPENDIX A: Coding

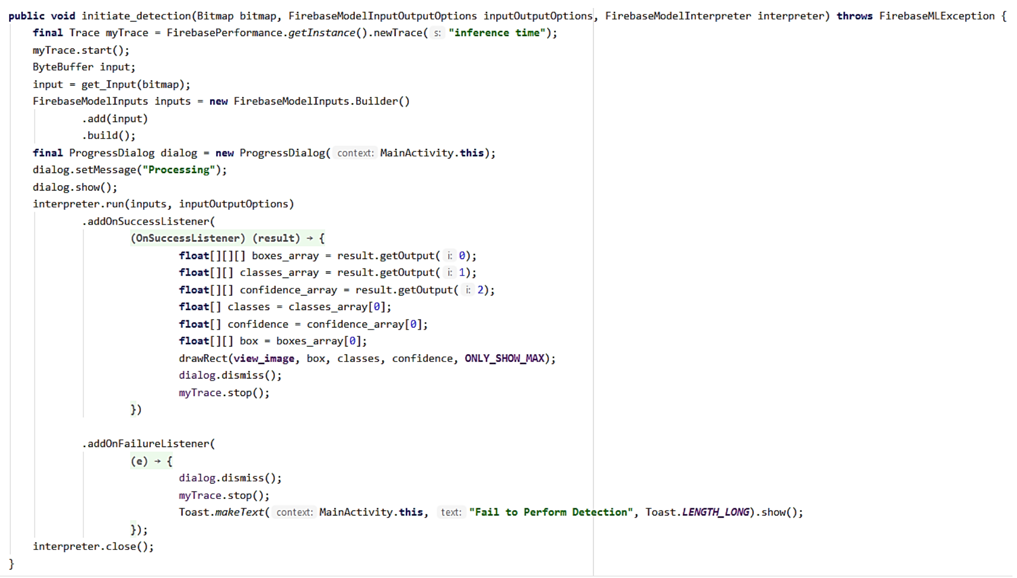


```java
public FirebaseModelInterpreter create_model_interpreter() {
    FirebaseCustomLocalModel localModel = new FirebaseCustomLocalModel.Builder()
            .setAssetFilePath("detect15k.tflite")
            .build();

    FirebaseModelInterpreter interpreter = null;
    try {
        FirebaseModelInterpreterOptions options = new FirebaseModelInterpreterOptions.Builder(localModel).build();
        interpreter = FirebaseModelInterpreter.getInstance(options);
    } catch (FirebaseMLException e) {
        Log.d( tag: "Error",  msg: "Build Model Error");
    }
    return interpreter;
}

public FirebaseModelInputOutputOptions set_model_input_output() {
    FirebaseModelInputOutputOptions inputOutputOptions =
            null;
    try {
        inputOutputOptions = new FirebaseModelInputOutputOptions.Builder()
                .setInputFormat( i: 0, FirebaseModelDataType.FLOAT32, new int[]{1, 224, 224, 3})
                .setOutputFormat( i: 0, FirebaseModelDataType.FLOAT32, new int[]{1, 5, 4})
                .setOutputFormat( i: 1, FirebaseModelDataType.FLOAT32, new int[]{1, 5})
                .setOutputFormat( i: 2, FirebaseModelDataType.FLOAT32, new int[]{1, 5})
                .setOutputFormat( i: 3, FirebaseModelDataType.FLOAT32, new int[]{1})
                .build();
    } catch (FirebaseMLException e) {
        Toast.makeText( context: this,  text: "Cannot set input output for model", Toast.LENGTH_LONG).show();
    }
    return inputOutputOptions;
}
```

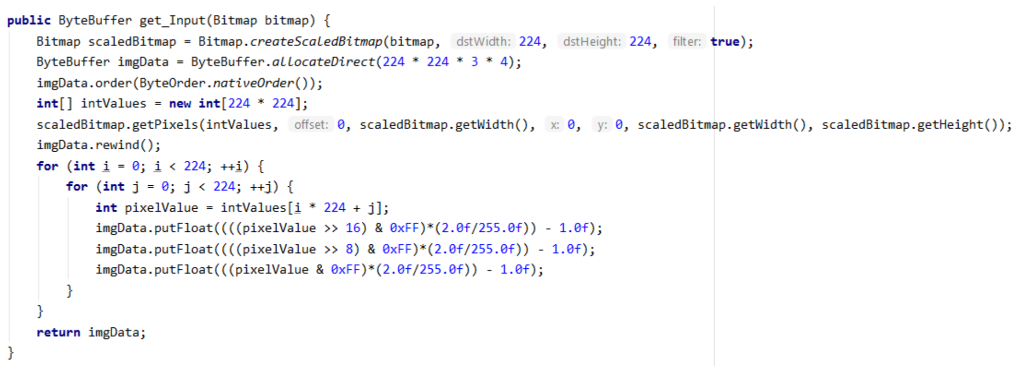
Figure A-1: Android Studio Coding for Creating TensorFlow Lite Interpreter.



```java
public void initiate_detection(Bitmap bitmap, FirebaseModelInputOutputOptions inputOutputOptions, FirebaseModelInterpreter interpreter) throws FirebaseMLException {
    final Trace myTrace = FirebasePerformance.getInstance().newTrace( s: "inference time");
    myTrace.start();
    ByteBuffer input;
    input = get_Input(bitmap);
    FirebaseModelInputs inputs = new FirebaseModelInputs.Builder()
            .add(input)
            .build();
    final ProgressDialog dialog = new ProgressDialog( context: MainActivity.this);
    dialog.setMessage("Processing");
    dialog.show();
    interpreter.run(inputs, inputOutputOptions)
            .addOnSuccessListener(
                    (OnSuccessListener) (result) → {
                            float[][][] boxes_array = result.getOutput( i: 0);
                            float[][] classes_array = result.getOutput( i: 1);
                            float[][] confidence_array = result.getOutput( i: 2);
                            float[] classes = classes_array[0];
                            float[] confidence = confidence_array[0];
                            float[][] box = boxes_array[0];
                            drawRect(view_image, box, classes, confidence, ONLY_SHOW_MAX);
                            dialog.dismiss();
                            myTrace.stop();
                    })

            .addOnFailureListener(
                    (e) → {
                            dialog.dismiss();
                            myTrace.stop();
                            Toast.makeText( context: MainActivity.this,  text: "Fail to Perform Detection", Toast.LENGTH_LONG).show();
                    });
    interpreter.close();
}
```

Figure A-2: Android Studio Code for Inference.



```java
public ByteBuffer get_Input(Bitmap bitmap) {
    Bitmap scaledBitmap = Bitmap.createScaledBitmap(bitmap,  dstWidth: 224,  dstHeight: 224,  filter: true);
    ByteBuffer imgData = ByteBuffer.allocateDirect(224 * 224 * 3 * 4);
    imgData.order(ByteOrder.nativeOrder());
    int[] intValues = new int[224 * 224];
    scaledBitmap.getPixels(intValues,  offset: 0, scaledBitmap.getWidth(),  x: 0,  y: 0, scaledBitmap.getWidth(), scaledBitmap.getHeight());
    imgData.rewind();
    for (int i = 0; i < 224; ++i) {
        for (int j = 0; j < 224; ++j) {
            int pixelValue = intValues[i * 224 + j];
            imgData.putFloat((((pixelValue >> 16) & 0xFF)*(2.0f/255.0f)) - 1.0f);
            imgData.putFloat((((pixelValue >> 8) & 0xFF)*(2.0f/255.0f)) - 1.0f);
            imgData.putFloat(((pixelValue & 0xFF)*(2.0f/255.0f)) - 1.0f);
        }
    }
    return imgData;
}
```

Figure A-2: Android Studio Code for Image Downscaling and Normalization.