AN INVESTIGATION ON SMARTPHONE BASED MACHINE VISION SYSTEM FOR INSPECTION

THEY SHAO PENG

UNIVERSITI TUNKU ABDUL RAHMAN

AN INVESTIGATION ON SMARTPHONE BASED MACHINE VISION SYSTEM FOR INSPECTION

THEY SHAO PENG

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

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

April 2022

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	:	THEY SHAO PENG
ID No.	:	1703177
Date	:	10/3/2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"AN INVESTIGATION ON SMARTPHONE BASED MACHINE VISION SYSTEM FOR INSPECTION"** was prepared by **THEY SHAO PENG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Dr. Chai Tong Yuen
Date	:	12/5/2022

Signature	:	
Co-Supervisor	:	
Date	:	

ACKNOWLEDGEMENTS

First of all, I would like to express my special thanks of gratitude to everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Chai Tong Yuen for his invaluable advice, guidance and his enormous patience throughout the development of the research.

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

ABSTRACT

Machine vision system for inspection is an automated technology that is normally utilized to analyse items on the production line for quality control purposes, it also can be known as automated visual inspection (AVI) system. By applying automated visual inspection, the existence of items, defects, contaminants, flaws and other irregularities in manufactured products can be easily detected in a short time and accurately. However, AVI systems are still inflexible and expensive due to their uniqueness for a specific task and consuming a lot of set-up time and space. With the rapid development of mobile devices, smartphones can be an alternative device for the visual system to solve the existing problems of AVI. Since the smartphone-based AVI system is still at a nascent stage, this led to the motivation to investigate the smartphone-based AVI system. This study is aimed to provide a low-cost AVI system with high efficiency and flexibility. In this project, the object detection models which are You Only Look Once (YOLO) model and Single Shot MultiBox Detector (SSD) model are trained, evaluated and integrated with the smartphone and webcam devices. The performance of the smartphone-based AVI is compared with the webcam-based AVI according to the precision and inference time in this study. Additionally, a mobile application is developed which allows users to implement real-time object detection and object detection from image storage.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AP	average precision
API	application programming interface
AVI	automated visual inspection
CNN	convolutional neural networks
FN	false negatives
FP	false positives
FRCNNF	Faster R-CNN with feature pyramid networks
IoU	intersection area over the union area
mAP	mean average precision
R-CNN	Faster Region-based Convolutional Neural Network
SIN	structure inference net
SINF	SIN based on FRCNNF
SSD	Single Shot MultiBox Detector
TP	true positives
UI	user interface
UX	user experience
YOLO	You Only Look Once

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

The machine vision system is a set of integrated components that can automatically process the image and recognize the image contents (Edwards, 2020). With the development of information technologies, deep learning is promoted as a revolution in machine vision systems by improving the performance of machine vision systems. Deep learning facilitates the machine vision system to analyse and classify objects by using neural networking for data processing (Wilson, 2019). Nowadays, the combination of deep learning and machine vision techniques is widely utilized in various industrial sectors such as manufacturing, electronics, automotive and food industries for inspection, measurement and quality control. Machine vision system for inspection is an automation technology that is used to improve the quality control of manufactured products, it also can be known as automated visual inspection (AVI). By applying AVI, the existence of items, defects, contaminants, flaws and other irregularities in manufactured products can be easily detected in a short time and accurately. Normally, it is operated by acquiring the object images that are being inspected through the camera and then utilizing the appropriate machine vision techniques to search and classify areas of interest in the image (Batchelor and Whelan, 2002).

With the development of the mobile internet, the number of mobile phone users is explosively growing every day. According to Statista Research Department (2021), there are 4.28 billion unique mobile internet users, which indicates that more than 90% of the global internet population is using mobile devices to go online in 2020. Besides, all the smartphones nowadays are equipped with cameras and almost everyone has at least a smartphone. As a result, more and more businesses consider implementing machine vision and deep learning tasks in smartphones. There is a huge interest in detecting medical conditions by using smartphones and wireless technology such as the Qualcomm Tricorder X Prize launched in 2013 to incentivize the development of a wireless device that can detect a range of diseases (Razdan and Bateman, 2015). Nevertheless, the use of smartphones in the manufacturing industry for inspection is still at an emerging stage. Thus, this led to a proposed idea of investigating smartphone-based machine vision system for inspection.

1.2 Importance of the Study

In recent years, smartphone-based machine vision system for industrial inspection is still at a nascent stage. This study may significantly impact on providing an alternative device which is using the smartphone in a machine vision system for industrial inspection. Instead of an industrial camera device, the smartphone provides flexibility, minimizes space usage and reduces the expenses such as maintenance fees in the machine vision system. This research project may shed light on the concepts behind the smartphone-based machine vision system and the concepts behind deep learning in machine vision. The machine vision algorithms proposed may contribute to automated visual inspection where the existence of some items and locations on some manufactured products can be easily detected.

1.3 Problem Statement

To implement an AVI system, some general requirements are required to be examined such as its performance, flexibility, reliability, maintainability and cost-effectiveness (Batchelor and Whelan, 2002). However, AVI systems are still inflexible and expensive due to their uniqueness for a specific task and consuming a lot of set-up time and space. The vision system costs, which included the installation, operating and maintenance fees, are directly proportional to the dimensional accuracy of the system. The higher the cost required, the higher the dimension accuracy is available for the system. Hence, the AVI systems are still unaffordable for most small and medium-sized manufacturing businesses (Razdan and Bateman, 2015).

In most industries, the AVI system is typically conducted through the industrial cameras connected to the computer (Hashim, Abdullah and Prabuwono, 2010). By using the industrial camera, the flexibility of the AVI system is limited due to its various camera specs and types that are chosen for a specific task such as area scan camera, line scan camera, CMOS camera, CCD camera, monochrome camera, colour camera and so on (Fell, 2017). Each automated visual inspection system is unique, and it is difficult to be transferred from one product or production line to another.

Furthermore, the industrial camera-based machine vision system can be considered as a complicated system that mainly consists of the camera system, lighting system and industrial PC. Jia (2009) indicated that the camera lens aperture, camera focus and light intensity of the lighting system in the machine vision system are required to be set and adjusted to a specified value for capturing the sharpest image. Besides, the camera is required to connect with the industrial PC for transmitting captured images and implementing image analysis. Thus, the industrial camera-based machine vision system setup required high technical skills and consumes lots of time for setup.

With the rapid development of mobile devices, smartphones can be an alternative device for the visual system to solve the existing problems of the AVI system mentioned above. Smartphone devices can improve the flexibility of the machine vision system, simplify the system and reduce expenses due to its smaller size, portable feature and advanced camera feature such as autofocus, flash type, multiple cameras and so on. Since smartphone-based machine vision in industries is also still at a nascent stage, this led to the motivation to investigate the smartphone-based machine vision system for inspection.

1.4 Aim and Objectives

The overall aim of this study is to propose a smartphone-based machine vision system for inspection. The system is aimed to provide a low-cost machine vision device with high efficiency and flexibility for small and medium scale enterprises. The specific objectives of this project are shown below:

i. To develop a mobile application for users to access the smartphone camera for inspection.
ii. To devise a learning-based algorithm to automatically detect the existence of items in manufactured products.
iii. To investigate the performance of a smartphone in machine vision inspection can be benchmarked against a webcam used in a production line.

1.5 Scope of the Study

In this project, an Android mobile application that integrates with deep learning-based machine vision is developed. The main functionality of this Android application is accessing the Android smartphone's camera to detect the existence of items in manufactured products. Since screws are the most common items used in manufactured products, the developed Android application focuses on screws detection in manufactured products.

Once the Android application development is finalized, the performance of the smartphone (Huawei Mate 20) is evaluated and compared with the webcam (C920 Pro HD Webcam) in screws detection.

1.6 Contribution of the Study

As mentioned in the problem statement, AVI systems are still inflexible and expensive because of their uniqueness for a specific task and consuming a lot of set-up time and space. Hence, this final year project presents proof of a smartphone device can be an alternative device in machine vision system for industrial inspection and provides a low-cost machine vision device with high efficiency and flexibility. This is because smartphone provides flexibility, minimizes space usage and reduces the expenses such as maintenance fees in the machine vision system, instead of the industrial camera device.

Furthermore, this study illustrates the concepts behind the smartphone-based machine vision system and the concepts behind deep learning in machine vision and the proposed machine vision algorithms contribute to the AVI system where the existence of some items and locations on manufactured products can be easily detected. Additionally, the mobile application developed allows users to implement real-time object detection and implement object detection by selecting images from a photo gallery through the smartphone.

1.7 Outline of the Report

In this report, the contents are breakdown into five main chapters which are introduction, literature review, methodology and workplan, results and discussion, and conclusions and recommendations.

In Chapter 1, the project had introduced which includes general introduction, the importance of the study, problem statement, aim and objectives, the scope of the study, contributions of the study and outline of the report. In Chapter 2, the previous related research and theories that contributed to this project had been reviewed and summarised.

The Chapter 3 of this report illustrated the development process of the investigation on smartphone-based machine vision system for inspection. The results and discussion section had been discussed in Chapter 4 of this report which includes the model evaluation, object detection results in reality and the object detection app. Lastly, Chapter 5 concluded the project and the recommendations for the future work had been discussed.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, a detailed literature review was conducted for an in-depth study of the machine vision system, deep learning in machine vision and mobile apps development to get the fundamental knowledge related to this research topic. Furthermore, related works were studied and analysed in order to gather useful information that can be implemented in the smartphone-based machine vision system for inspection.

2.2 Machine Vision System

Machine vision is the science and technology that utilizes the computer to simulate biological vision. The primary objective of the machine vision system is used to generate or restore the real-world model from the image model and then get to know the real world (Bao et al., 2018). Nowadays, machine vision systems are widely employed in industrial areas, especially the manufacturing system for the inspection, measurement and providing useful visual information and quality during the quality control processes.

Machine vision systems are composed of several components and each component performs its function within the machine vision system. Typically, these components of a machine vision system include lighting, cameras, optics, imaging boards and third-party software that work together to process the image and recognize the image contents automatically (Jia, 2009). In a machine vision system, its processes can be categorized into various stages which are image acquisition, image representation, image processing, image analysis and results. Image capturing and image processing in a machine vision system are normally done by the smart camera whereas an external computer is utilized to implement image analysis and result presentations. In contrast, smartphone device offers an all-in-one solution for all stages of the machine vision process (Razdan and Bateman, 2015).

With the rapid development of advanced smartphones, smartphones can be known as the integration of smart camera and computer that consist of smart camera's features and can perform a lot of functions of a computer. Furthermore, smartphones nowadays are normally equipped with flash LED lighting to illuminate the object for capturing a better image. Therefore, smartphone vision has the potential in revolutionizing the machine vision system.

2.2.1 Smartphone Vs Webcam

In this project, the cameras provided for comparison are (1) smartphone camera from Huawei Mate 20, and (2) smart camera from Logitech C920 HD Pro Webcam. Table 2.1 provides details for this comparison. Logitech C920 HD Pro Webcam is chosen for comparison due to the limited budget in this study and it is compatible with the camera device that is used in industries.

Table 2.1: Comparison Between a Smartphone Camera and a Webcam.

Feature	**Huawei Mate 20 Camera**	**C920 HD Pro Webcam**
Image	Colour	Colour
Numbers of Lenses	Triple (e.g., wide, telephoto and ultrawide)	Single
Max Resolution	1080 p / 60 fps	1080 p / 30 fps
Image Size	12 MP (wide) 8 MP (telephoto) 16 MP (ultrawide)	3 MP
Aperture	f / 1.8 f / 2.4 f / 2.2	Not Available
Privacy Shutter	Yes	No
Focus	Autofocus	Autofocus
Zoom Capability	Up to 2x	Not Available
Lighting	LED (internal)	LED (external)
Auto Light Correction	Yes	Yes
Software Launch	Internal (on-screen)	External (via computer)

Based on the comparison table, it can be seen that the overall features of the Huawei Mate 20 camera are quite better than C920 HD Pro Webcam, especially in image resolution. This is because the Huawei Mate 20 camera is equipped with triple lenses and each lens has its image size value and aperture value that can be captured. In contrast, C920 HD Pro Webcam is only equipped with a single lens, the image size that can be captured is 3 MP and the aperture is not available. Furthermore, Huawei Mate 20 camera is available with 2x optical zoom and is equipped with an internal LED flashlight whereas the webcam is not available in these two features. Additionally, the machine vision software can be launched internally in the smartphone whereas the software must be launched externally with a computer for the webcam. However, Huawei Mate 20 camera and C920 HD Pro Webcam support autofocus and auto light correction.

2.3 Deep Learning in Machine Vision

Deep learning is a method of learning to construct a deep architecture and generate a model by iterating multi-layered functions, which can also be known as a deep neural network. With the development of information technologies, deep learning is promoted as a revolution in machine vision systems by improving the performance of machine vision systems (Zhu et al., 2021). Increasingly, deep learning technology is being implemented in most industries to resolve manufacturing inspections that are too complex, time-consuming and costly to program with traditional machine vision algorithms. In this way, deep learning makes the industrial inspection system more automated, accelerates inspection times and reduces the error rates, rather than the traditional machine vision (Cognex Corporation, 2020).

Basically, deep learning enables multiple-layer computational models to learn and represent data with multiple stages of abstraction, thereby implicitly capturing large-scale complex data structures. This means that its algorithms rely on learning the representation of data and these multiple stages of abstraction can be expressed in various ways such as an intensity value vector for each pixel, a series of edges, areas of a specific shape and so on (Voulodimos et al., 2018). Nowadays, there are several advanced deep learning frameworks such as Recurrent Neural Networks, Convolutional Neural Networks (CNN) and Fully Convolutional Networks that can be applied to image classification, object detection, object tracking, semantic segmentation and instance segmentation.

Since the objective of this study is to devise a learning-based algorithm to automatically detect the existence of items in manufactured products, object detection approaches by using deep learning technology are studied and discussed. Object detection is the process of detecting objects in an image by applying a recognition algorithm on all viewports of the original image. In other words, the task of object detection is to define the targeted objects within images by applying classification and localization (James Le, 2018). CNN with sliding window detector is a historical approach for object detection within an image, where the boxes slide over the entire image in steps and each crop of the image will be classified by CNN. However, this approach is computationally expensive by applying CNN to a wide range of locations and scales. In order to cope with this, several advanced object detection networks are proposed by neural network researchers, such as Faster Region-based Convolutional Neural Network (R-CNN), YOLO, SSD and so on (Alsing, 2018). Thus, these advanced object detection networks that implemented in this research are studied and discussed.

2.3.1 CNN

CNN is the most popular architecture in image classification and it is commonly used as the backbone of object detection networks such as Faster R-CNN, YOLO and SSD to automatically extract various features from the input images (Narvekar, 2020). The contents of CNN can be classified into three main types of neural layers which are convolutional layers, pooling layers and fully connected layers. Each kind of neural layer performs a different role within CNN. Convolutional layer is the primary layer in the convolutional base, and it is usually followed by the pooling layer in order to extract various important features from the input images and perform feature learning. Subsequently, these extracted features from the input images will be flattened and fed into the fully connected layers to implement the classification process (Voulodimos et al., 2018). Figure 2.1 shows the architecture of CNN.

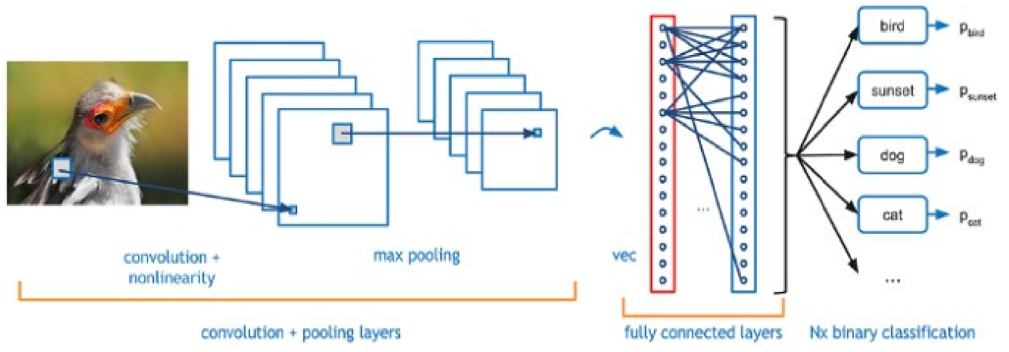


Figure 2.1: Architecture of CNN (James Le, 2018).

The objective of using convolutional layers in CNN is to extract various features such as edges, gradient orientation and colour from the input images. Various kernels are utilized by CNN in the convolutional layers in order to convolve the whole image and the intermediate feature maps and generate various feature maps (Alsing, 2018).

In the pooling layers, the aim in CNN is used to reduce the spatial dimensions of the input volume before feeding them into the subsequent convolutional layer. However, the depth dimension of the input volume is not influenced by the pooling layer (Alsing, 2018). The most commonly used strategies in these pooling layers are average pooling and max pooling. The operating principle of average pooling is to take the average number of each sub-area as the output when the filter convolves around the input volume whereas the maximum number of each sub-area is taken in max-pooling (Voulodimos et al., 2018).

Fully connected layer performs the high-level reasoning in the neural network from the output of convolutional and pooling layers. All the neurons of the fully connected layer are fully connected to all activation of the previous layer and these activations will be computed with the matrix multiplication and the bias offset. Eventually, the 2D feature maps are converted into a 1D feature vector through the fully connected layers and it is utilized to classify images based on the labels (Saha, 2018).

2.3.2 YOLO

YOLO is a regression-based deep learning object detection network, and it is categorized into the one-stage target detection model which directly generates the prediction results without involving the intermediate region detection process. YOLO implements object detection by only requiring a single forward propagation via the neural network and CNN is employed in the YOLO in order to predict different class probabilities and bounding boxes simultaneously (Zhongyao et al., 2020). The architecture of YOLO is shown in Figure 2.2.

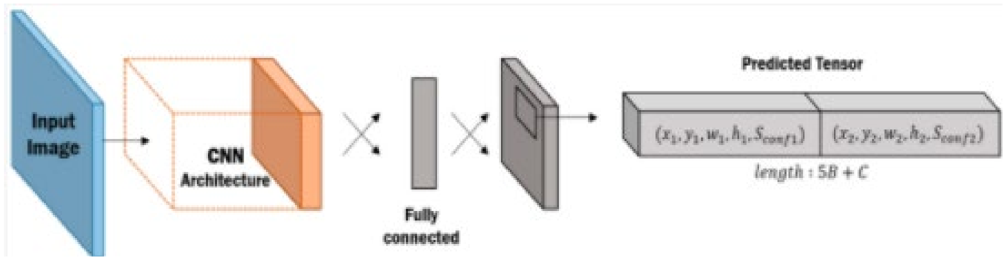


Figure 2.2: Architecture of YOLO (Kim and Cho, 2020).

Initially, the YOLO network divides the input image into a grid with S x S cells based on the resolution and then CNN is utilized to extract the features of the input image. Fully connected layer is inserted after the CNN architecture in order to generate the predicted tensor with the size of (S x S) and the length of (B x 5 + C). S x S represents the number of grid zones, B indicates the number of candidate boundary boxes while C symbolizes the number of objects that can be categorized. For the B bounding boxes, they contain several pieces of information in the form of (x, y, w, h, Sconf) that will be predicted by the grid region. (x, y) represents the bounding box's centre coordinate while (w, h) is the width and height of the bounding box. Sconf is the product of Pr (Object) and Intersection-over-Union (IoU), as shown in Equation 2.1.

$$S_{conf} = \Pr(Object)\ x\ IoU_{pred}^{truth} \tag{2.1}$$

where

Pr(Object) = probability of an object being included in the bounding box

IoU_{pred}^{truth} = area relative to the ground truth and intersection of union

Pr(Object) will be calculated as 1 while the actual coordinates and the centre coordinates of the predicted bounding box are located in the similar grid area, else it will be treated as 0. The formula of IoU is shown in Equation 2.2.

$$IoU_{pred}^{truth} = \frac{area(b.b_{pred} \cap b.b_{truth})}{area(b.b_{pred} \cup b.b_{truth})} \tag{2.2}$$

where

(b.b$_{pred}$) = predicted bounding box

(b.b$_{truth}$) = ground truth

Additionally, Pclass represents the probability that the classified object and the ground truth are matched with each other, which can be calculated as shown in Equation 2.3.

$$P_{class} = \Pr\,(Class|Object) \tag{2.3}$$

Furthermore, the confidence score, CSconf can be obtained after all the grid regions (S x S) are predicted by the tensor of length (B x 5 + C), as shown in Equation 2.4.

$$\begin{aligned} CS &= S_{conf} x\, P_{class} \\ &= \Pr(Object)\, xIOU_{pred}^{truth} xPr(Class|Object) \\ &= \Pr\,(Class) xIOU_{pred}^{truth} \end{aligned} \tag{2.4}$$

Eventually, the highest CS value of the bounding box among the predicted B bounding boxes will be selected as the bounding box of the object. In short, object detection in YOLO is performed as a regression problem and provides the class probabilities of the detected images (Kim and Cho, 2020).

2.3.3 SSD

SSD is categorized as the regression-based deep learning object detection network, and it is also known as a one-stage target detection model which is similar to YOLO. The difference between the SSD and YOLO is that two fully connected layers are utilized in YOLO architecture whereas the varying size of convolutional layers is used in SSD architecture. The combination of anchor boxes and multiscale features maps support rapid detection speed in SSD while still maintaining high detection quality (Alsing, 2018). The architecture of the SSD is illustrated in Figure 2.3.

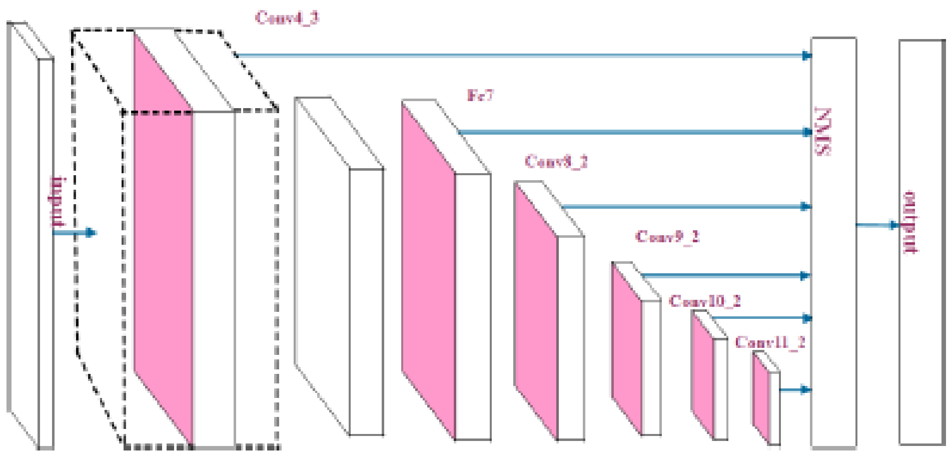


Figure 2.3: Architecture of SSD (Wang et al., 2021).

In SSD architecture, it is comprised of a backbone convolutional base located at the front, followed by a feature pyramid and non-maximum suppression (NMS). Backbone convolutional base normally is a pre-trained image classification such as Mobilenet and Visual Geometry Group (VGG) that is used to extract features. For the feature pyramid, it consists of multiple scales of layers that play a specific role to detect the objects independently while NMS is used to generate the final detection. The detection results of the SSD are directly generated from the various levels of feature maps which do not involve the intermediate region detection process. Hence, SSD can also be known as a region-free method (Wang et al., 2021).

2.3.4 Evaluation Metrics

For object detection networks evaluation like Faster RCNN, YOLO, SSD and so on, the mean average precision (mAP) is commonly used where it is comparing the ground-truth bounding box with the detected box and returning a score (Khandelwal, 2020).

Before discussing mAP, there are several metrics are discussed in this section. IoU is defined as the intersection area over the union area of the detected box and ground-truth bounding box, as shown in Equation 2.2. IoU is normally applied to determine the true positives (TP), false positives (FP) and false negatives (FN). TP is the number of detections that IoU is greater than 0.5, FP is the number of detections that IoU is less than or equal to 0.5 or detected more than once and FN is the number of objects that IoU is less than or equal to 0.5 or not detected. Precision is utilized to determine the accuracy of the predictions where the formula as shown in Equation 2.5. Besides, the formula of recall is shown in Equation 2.6.

$$Precision = \frac{TP}{(TP+FP)} \tag{2.5}$$

$$Recall = \frac{TP}{(TP+FN)} \tag{2.6}$$

Average precision (AP) is the area under the precision-recall curve, whereas mAP is the average of the calculated AP in all the classes of object detection. Thus, these various metrics are normally applied to evaluate the object detection networks (Wadawadagi, 2020).

2.4 Mobile Apps Development

Due to one of the objectives in this study is to develop a mobile application for users to access the smartphone camera for inspection, the mobile apps development processes are studied and discussed. Mobile apps development is the process of setting up a set of coded instructions in the mobile device and it is utilized in order to resolve issues, wireless computing and communication (Patidar, 2021).

In the mobile app development process, it can be divided into several stages which are strategy, analysis and planning, app designing, app development, testing and deployment. At the strategy stage, its steps involve app users identification, identifying the goals and objectives of the mobile application and selection of a mobile platform for the mobile application. In the second stage of the mobile apps development, the detailed functional requirements are required to identify and a product roadmap is also required to prepare (Invonto, 2021).

The subsequent stage of the mobile apps development is designing the mobile application which involves User Interface (UI) and User Experience (UX) design. In the app development stage, the technical architecture should be defined before starting the actual development and programming. Typically, mobile application projects consist of three integral parts which are the frontend framework, backend framework and application programming interface (API). During the mobile app development process, the testing stage is very important to maintain the stability, security and usability of the mobile application. In the final stage of the mobile apps development, it involves app deployment where the apps can be launched on the Apple and Google app stores (Invonto, 2021).

2.5 Related Works

In this chapter section, several research papers that are related to this study are studied and discussed. The related research papers can be categorized into machine vision inspection and smartphone-based machine vision. The aim of studying related research papers is to gather useful information that can be implemented in the smartphone-based machine vision for inspections.

Razdan and Bateman (2015) in their paper have researched smartphone-based machine vision for hole diameter measurements and flaw detection of twist drill bits. In this paper, the HTC One X Smartphone was used as a vision system to measure the hole diameter and detect the drill bits' flaws. Eclipse IDE was used for android app development in Java language and OpenCV operators were used for machine vision algorithms. For machine vision algorithms, traditional methods such as Gaussian blur, Sobel and Canny operators were used. The overall accuracy results in hole diameter measurements and flaw detection of drill bits were not performed well due to lacking advanced mathematical algorithms for image processing. Additionally, this paper also mentioned that smartphone-based machine vision has the potential as a revolution in machine vision inspections due to its lower cost and it is affordable for small scale manufacturers.

There is another research paper studying train component detection by using cascade CNN based on structure knowledge. To detect train components, deep learning-based machine vision was implemented on Pytorch which are YOLOv3 model, RetinaNet model, Faster R-CNN model, Structure Inference Net (SIN), Faster R-CNN with feature pyramid networks (FRCNNF), SIN based on FRCNNF (SINF). In addition, the backbone of all the proposed models was using the Resnet-101 model. Each proposed model was evaluated and compared with each other based on mAP which is commonly used to assess object detection networks. For small object detection, FRCNNF and SINF showed better results with mAP above 70% while all the proposed models performed well in large object detection with mAP above 80% (Zhongyao et al., 2020).

Burresi et al. (2021) proposed missing screws detection approaches and compared different deep learning models for this missing screw detection. The deep learning models that are used in this research paper are YOLOv3, Tiny YOLOv4 and Xception network. Based on the experimental results in this research paper, the accuracy of these deep learning models was exceeding 95%. In this research paper, it mentioned that the Xception network is considered as a binary classification which relies on the position of the panel. In contrast, YOLOv3 is not limited to this scenario. However, evident changes of illumination will affect the performance of YOLOv3. Tiny YOLOv4 is the lighter version of YOLO where its performance was reduced compared with YOLOv3 (Burresi et al., 2021).

The deep learning-based machine vision approaches for defects detection of metal screw surfaces are proposed by Song et al. (2018). To detect micro-defects on screw surfaces, several deep learning models were implemented and compared with each other which are LeNet-5, YOLO, R-CNN, Faster R-CNN, SSD and a proposed DCNN. The proposed DCNN was developed based on LeNet-5. According to the experimental results, the accuracy of SSD and the proposed DCNN were higher compared to others while the prediction time of YOLO and the proposed DCNN were faster (Song et al., 2018).

Although there are several research papers are related to machine vision inspections, most of them do not extend to the mobile apps or others are still using traditional machine vision approaches. Hence, several research papers that related to smartphone-based machine vision were studied to collect more useful information. Parico and Ahamed (2021) proposed pear fruit detection by using YOLOv4 models and the performance of these different types of YOLOv4 models were compared with each other. In addition, the Deep SORT algorithm was utilized for object counting. Another research paper was conducted by Zhang et al. (2020) that studied the icons detection in mobile apps by using different types of deep learning networks which are Fast RCNN, Faster RCNN, SSD300, SSD512, YOLOv3 and IconYOLO.

Jakhete et al. (2019) developed an object recognition app that can be used by visually impaired persons. In this object recognition app, SSD, RCNN and YOLO were implemented as object detection and compared with each other. Eventually, SSD was chosen for this app due to its better performance. A mobile-based grasshopper detection was proposed by Chudzik et al. (2020). For grasshopper detection, RetinaNet with ResNet-50 model and SSD with MobileNetV2 were implemented. Based on some research papers that related to smartphone-based machine vision above, the deep learning models were trained by using training data with TensorFlow and these trained models were added to the mobile application. Hence, the object detection process is directly running on the smartphone application background and generates the prediction results.

To summarize the related research papers, a summary table is constructed which contains some relative information on the machine vision approaches as shown in Table 2.2.

Table 2.2: Summary of Related Research Papers

Title	**Targeted Images**	**Machine Vision Methods**	**Experimental Results**	**Mobile App**

Investigation into the use of Smartphone as a Machine Vision Device for Engineering Metrology and Flaw Detection, with Focus on Drilling	Twist Drill bits and holes	Traditional Methods (e.g., Gaussian Blur, Sobel and Canny operators)	Large Hole: <10% error Small Hole: <30% error Flank Detections: -	Yes
Robust Train Component Detection with Cascade CNN based on Structure Knowledge	Train Component (e.g., screws and nuts)	Deep Learning methods: 1) YOLOv3 2) RetinaNet 3) FRCNN 4) SIN 5) FRCNNF 6) SINF	mAP: 1) 60.58% 2) 82.42% 3) 70.50% 4) 71.27% 5) 86.54% 6) 86.10%	No
Detection Approach Based on an Improved Faster RCNN for Brace Sleeve Screws in High-Speed Railways	Brace Sleeve Screws	Deep Learning methods: 1) Faster RCNN with VGG-16 2) Improved Faster RCNN with VGG-16 3) Faster RCNN with ResNet-101 4) Improved Faster RCNN with ResNet-101	mAP: 1) 85.31% 2) 85.34% 3) 84.80% 4) 85.47%	No

Table 2.2 (Continued)

Title	**Targeted Images**	**Machine Vision Methods**	**Experimental Results**	**Mobile App**

Image-Based Defect Detection in Assembly Line with Machine Learning	Screws	Deep Learning methods: 1) YOLOv3 2) Tiny-YOLOv4 3) Xception	Micro acc. 1) 98.9% 2) 98.1% 3) 98.8% Macro acc.: 1) 97.2% 2) 97.0% 3) 98.6% Precision: 1) 99.3% 2) 97.1% 3) 98.6%	No
Detection of Micro-Defects on Metal Screw Surfaces Based on Deep CNN	Screws	Deep Learning methods: 1) YOLO 2) RCNN 3) Faster RCNN 4) SSD 5) LeNet-5 6) Proposed DCNN	Time: 1) Faster 2) Low 3) Fast 4) Low 5) - 6) Faster Accuracy: 1) Low 2) Low 3) High 4) Higher 5) Higher 6) Higher	No
Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT	Pear Fruits	Deep Learning methods: 1) Tiny-YOLOv4 2) YOLOv4 3) YOLOv4-CSP 4) Deep SORT (Object Counting)	AP: 1) 80.12% 2) 87.98% 3) 86.63%	Yes

Table 2.2 (Continued)

Title	Targeted Images	Machine Vision Methods	Experimental Results	Mobile App

A Benchmark Dataset for Real-Time Detection of Icons in Mobile Apps and a Small-Scale Feature Module	Icons	Deep Learning methods: 1) Fast RCNN 2) Faster RCNN 3) SSD300 4) SSD512 5) YOLOv3 6) IconYOLO	mAP: 1) 70.0% 2) 73.2% 3) 74.3% 4) 76.8% 5) 75.3% 6) 79.1%	Yes
Object Recognition App for Visually Impaired	Domestic Objects	Deep Learning methods: 1) SSD 2) RCNN 3) YOLO	SSD achieved better accuracy and performance compared with others.	Yes
Mobile Real-Time Grasshopper Detection and Data Aggregation Framework	Grasshopper	Deep Learning methods: 1) RetinaNet with ResNet-50 2) SSD with MobileNetV2	mAP: 1) 78.3% 2) 49.4%	Yes

Based on the summary table, YOLO and SSD are the deep learning models that are commonly used in object detection and have good performance in screws detection. Besides, these deep learning models are able to be converted into TensorFlow Lite format and integrated into mobile apps for running object detection on the mobile phone's background. Thus, these deep learning frameworks are chosen and implemented in this project.

2.6 Summary

In summary, the fundamental knowledge that is related to smartphone-based machine vision inspections is studied which can be categorized into several sections such as machine vision system, deep learning in machine vision and mobile apps development.

In the machine vision system section, the general structure and concepts of machine vision systems are studied. The advantages of smartphone-based machine vision are indicated and compared with industrial cameras in this section. Additionally, the specs of the Huawei Mate 20 are compared with C920 Pro HD Webcam which is implemented in this research paper.

Subsequently, deep learning-based machine vision is also studied where the deep learning applications in machine vision can be divided into image classification, object detection, object tracking, semantic segmentation and instance segmentation. The detailed object detection which relates to the objectives of this research paper is discussed. The popular object detection models include YOLO and SSD while CNN is implemented as the backbone of these object detection models. Hence, these deep learning models are discussed in this chapter. Besides, the evaluation metrics for object detection networks are also discussed which are IoU, TP, FP, FN, precision, recall, AP and mAP. Due to the mobile app development required in this study, the basic steps of mobile app development are studied.

Several research papers are also studied that are related to this study to collect more useful information. These research papers are discussed and summarized in Table 2.2. Since smartphone-based machine vision in industrial inspection is still at a nascent stage and it has potential as a revolution in industrial inspection, a smartphone-based machine vision inspection is developed in this paper. Furthermore, SSD and YOLO are considered as popular object detection models and their performances are quite well in several research papers that were studied. Hence, SSD and YOLO are implemented as object detection frameworks in this study. These models are trained by training data through TensorFlow and added to the mobile app. The mobile app is aimed to detect the existence of screws in manufactured products.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In this chapter, the detailed methodology and work plan of this project were discussed. The project scope, project flow and project Gantt Chart were constructed to keep track of the progress of the project and ensure the overall project flow is smoothly conducted.

3.1.1 Project Scope

In this section, the project scope table was constructed which covers the project title, project time frame, project description, project objectives, required equipment and software, as shown in Table 3.1.

Table 3.1: FYP Project Scope.

Title	Investigation on Smartphone-based Machine Vision for Inspection
Date	7 June 2021 to 25 April 2022
Project Description	The mobile application is able to access the Android smartphone's camera for machine vision inspection in screws detection. Investigation on the performance of smartphone-based machine vision inspection can be benchmarked against a webcam used in a production line.
Project Objectives	• Simple and user-friendly mobile application for users to access the Android smartphone's camera for inspection. • A learning-based algorithm that automatically detects the existence of screws in manufactured products. • The performance of smartphones in machine vision inspection is benchmarked against a webcam used in a production line.
Equipment (E) & Software (S)	**E**: LED desk lamp, Tripod Stand, Huawei Mate 20, C920 Pro HD Webcam **S**: Android Studio, Google Colaboratory, Roboflow

3.1.2 Project Flow and Project Planning

The overall project flowchart and project Gantt Chart were constructed and discussed. The overall project was divided into two parts which are FYP1 and FYP2. Each part breaks down into several milestones (M1 to M9). Figure 3.1 illustrates FYP1 project flowchart while Figure 3.2 illustrates FYP2 project flowchart. Furthermore, the detailed Gantt Chart of FYP1 and FYP2 are shown in Figure 3.3 and Figure 3.4.

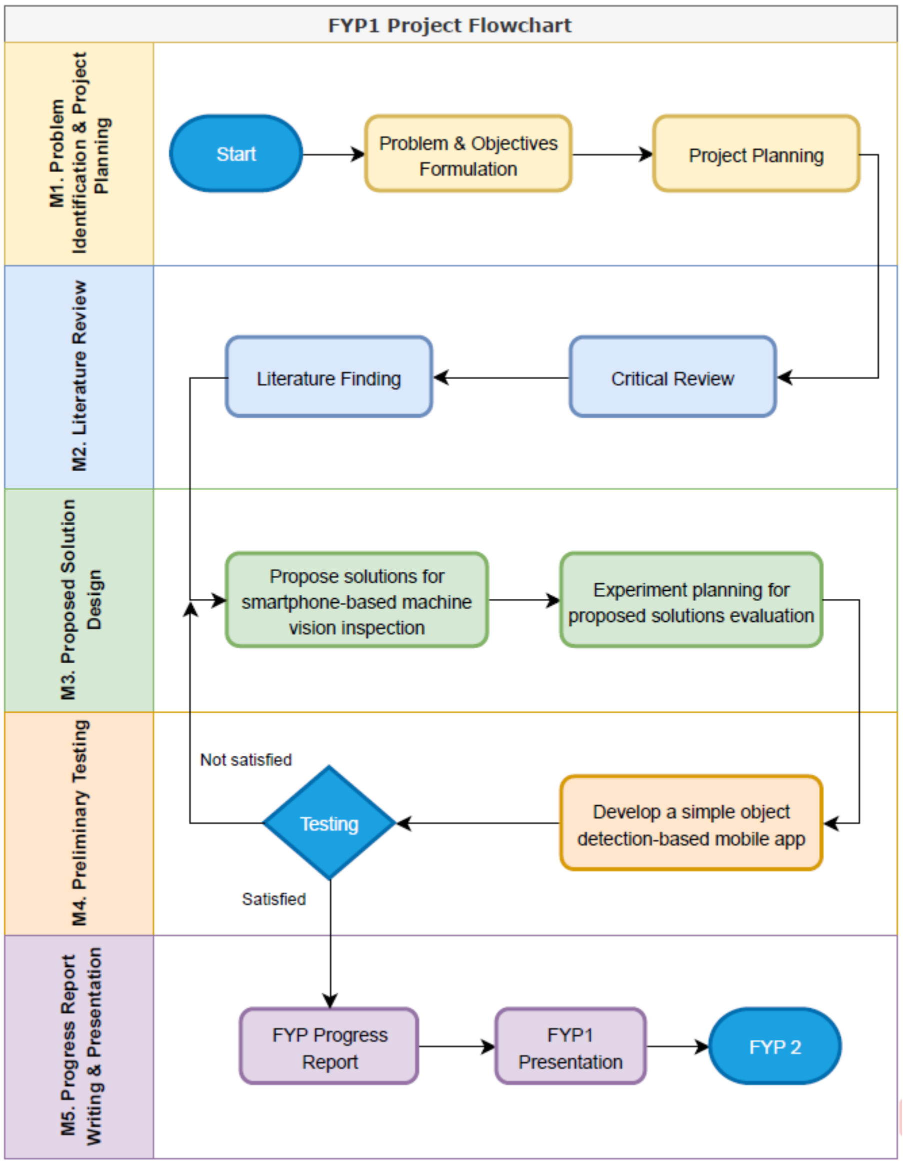

FYP1 Project Flowchart
M1. Problem Identification & Project Planning
Start
Problem & Objectives Formulation
Project Planning
M2. Literature Review
Literature Finding
Critical Review
M3. Proposed Solution Design
Propose solutions for smartphone-based machine vision inspection
Experiment planning for proposed solutions evaluation
M4. Preliminary Testing
Not satisfied
Testing
Develop a simple object detection-based mobile app
Satisfied
M5. Progress Report Writing & Presentation
FYP Progress Report
FYP1 Presentation
FYP 2

Figure 3.1: FYP 1 Project Flowchart.

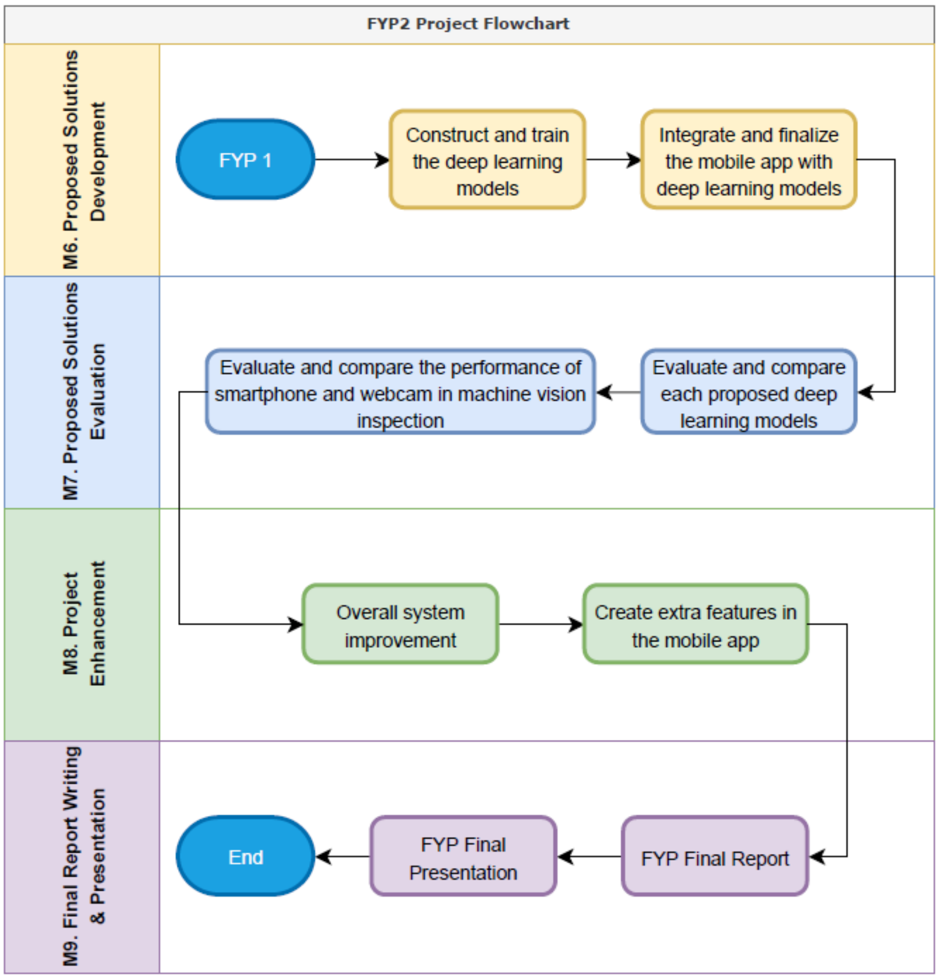
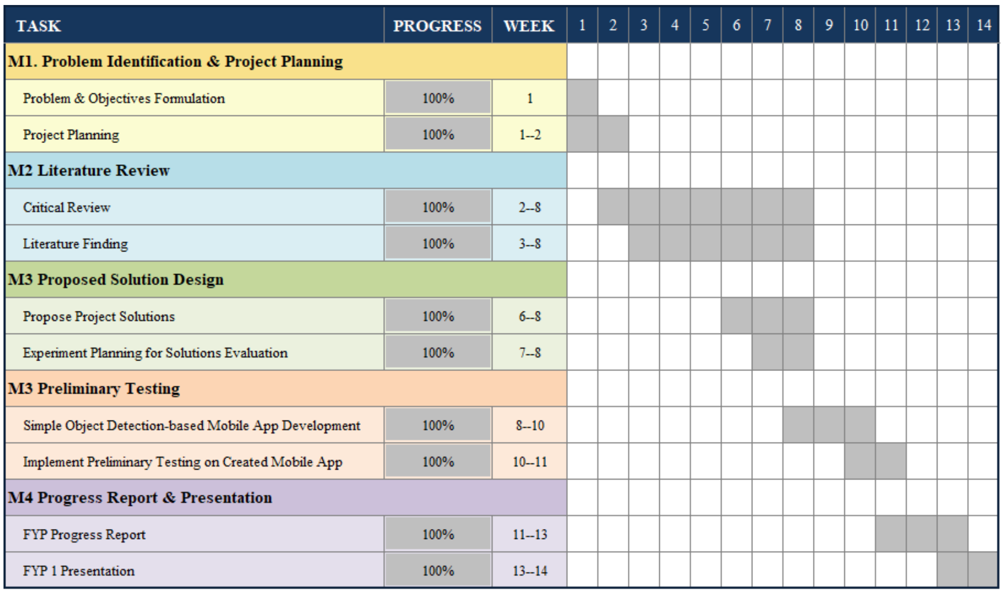
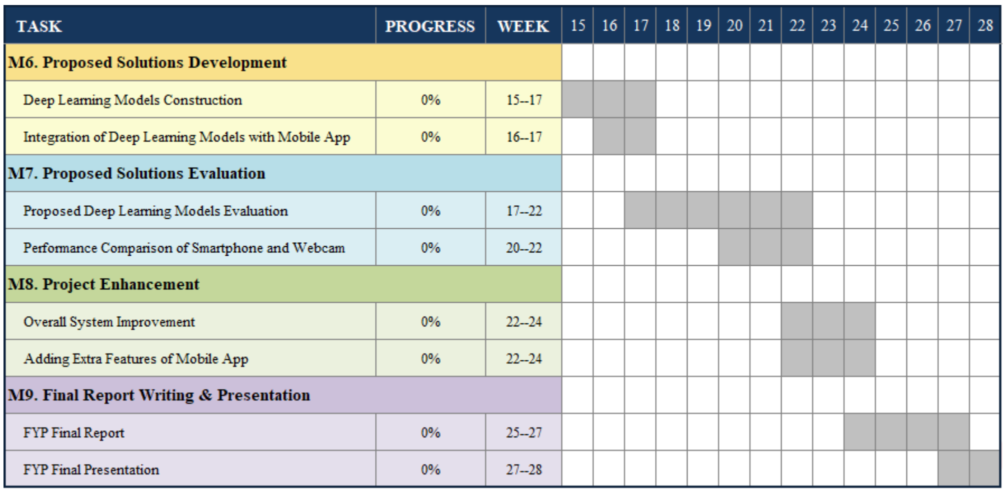

FYP2 Project Flowchart
M6. Proposed Solutions Development
M7. Proposed Solutions Evaluation
M8. Project Enhancement
M9. Final Report Writing & Presentation
FYP 1
Construct and train the deep learning models
Integrate and finalize the mobile app with deep learning models
Evaluate and compare each proposed deep learning models
Evaluate and compare the performance of smartphone and webcam in machine vision inspection
Overall system improvement
Create extra features in the mobile app
FYP Final Report
FYP Final Presentation
End

Figure 3.2: FYP 2 Project Flowchart.



TASK	PROGRESS	WEEK	1	2	3	4	5	6	7	8	9	10	11	12	13	14
M1. Problem Identification & Project Planning																
Problem & Objectives Formulation	100%	1	▨													
Project Planning	100%	1–2	▨	▨												
M2 Literature Review																
Critical Review	100%	2–8		▨	▨	▨	▨	▨	▨	▨						
Literature Finding	100%	3–8			▨	▨	▨	▨	▨	▨						
M3 Proposed Solution Design																
Propose Project Solutions	100%	6–8						▨	▨	▨						
Experiment Planning for Solutions Evaluation	100%	7–8							▨	▨						
M3 Preliminary Testing																
Simple Object Detection-based Mobile App Development	100%	8–10								▨	▨	▨				
Implement Preliminary Testing on Created Mobile App	100%	10–11										▨	▨			
M4 Progress Report & Presentation																
FYP Progress Report	100%	11–13											▨	▨	▨	
FYP 1 Presentation	100%	13–14													▨	▨

Figure 3.3: FYP 1 Project Gantt Chart.



TASK	PROGRESS	WEEK	15	16	17	18	19	20	21	22	23	24	25	26	27	28
M6. Proposed Solutions Development																
Deep Learning Models Construction	0%	15–17	▨	▨	▨											
Integration of Deep Learning Models with Mobile App	0%	16–17		▨	▨											
M7. Proposed Solutions Evaluation																
Proposed Deep Learning Models Evaluation	0%	17–22			▨	▨	▨	▨	▨	▨						
Performance Comparison of Smartphone and Webcam	0%	20–22						▨	▨	▨						
M8. Project Enhancement																
Overall System Improvement	0%	22–24								▨	▨	▨				
Adding Extra Features of Mobile App	0%	22–24								▨	▨	▨				
M9. Final Report Writing & Presentation																
FYP Final Report	0%	25–27										▨	▨	▨	▨	
FYP Final Presentation	0%	27–28													▨	▨

Figure 3.4: FYP 2 Project Gantt Chart.

From the project flowchart and Gantt Chart, there are 9 milestones included in this project and the total project duration is taking about 28 weeks. Hence, it is separated into two parts, FYP 1 and FYP 2 and each of them takes about 14 weeks. First, problem identification and project planning are taken about 2 weeks after project title confirmation. In literature review milestones, critical review and literature findings are conducted within 7 weeks. After gathering useful information from the literature review, the proposed solution design is implemented within 3 weeks, including experiment planning to evaluate the performance of the proposed solutions in this project. Subsequently, preliminary testing is conducted within 4 weeks to ensure that the mobile app for object detection in Android smartphones is functioning properly. From week 11 to 14, the progress report of this project is written and presented.

In FYP part 2, the sixth milestone is proposed solutions development where the time frame is within 3 weeks, including deep learning models construction and integration of the developed deep learning models into the mobile app. The evaluation of proposed deep learning models and the comparison between the performance of smartphone and webcam in machine vision inspection are implemented within 6 weeks. The project enhancement that covers the overall system improvement and extra features of mobile app addition are started from week 22 to 24. The last milestone of the project starts from week 25 that includes writing FYP final report and final presentation. The whole project is ended by the end of week 28.

The detailed project development flowchart which includes proposed solutions development, proposed solution evaluation and project enhancement is constructed to ensure the development flow is smoothly conducted and achieves the desired outcomes, as shown in Figure 3.5.

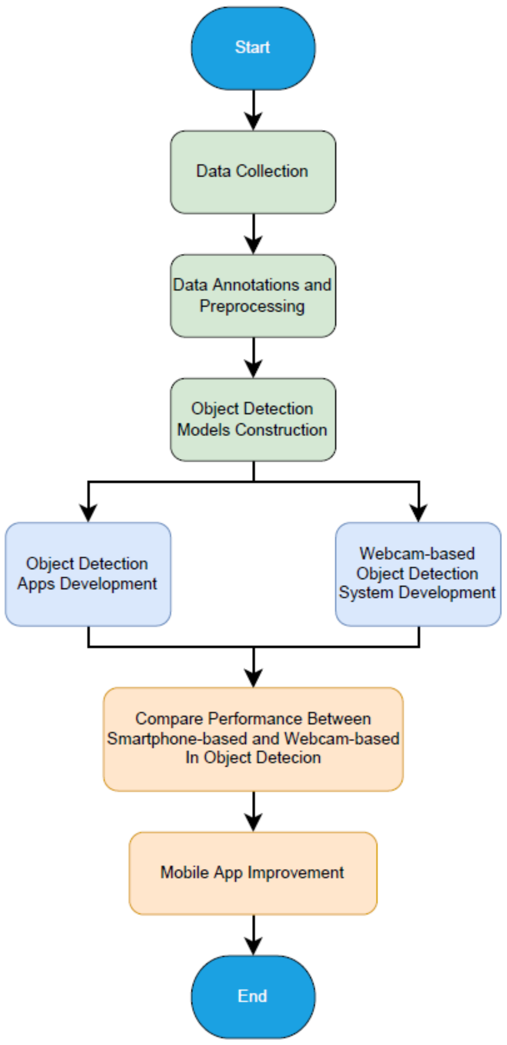

Start
Data Collection
Data Annotations and Preprocessing
Object Detection Models Construction
Object Detection Apps Development
Webcam-based Object Detection System Development
Compare Performance Between Smartphone-based and Webcam-based In Object Detecion
Mobile App Improvement
End

Figure 3.5: Project Development Flowchart.

For the project development, it can be separated into two parts after object detection models construction which is object detection apps development and webcam-based object detection system development. Subsequently, the performance between smartphone-based and webcam-based in object detection is compared. The detailed approaches are further discussed in the following sections.

3.2 Data Collection

Two different datasets were collected for this study which are captured by webcam and smartphone respectively in order to construct object detection models and compare their performances based on smartphone-based and webcam-based in object detection. A total 800 pieces of images that related to screws were captured by each device.

3.3 Data Annotation and Pre-processing

Data annotation is a required step for all AI application constructions by categorizing and labelling the data. In this project, Roboflow software is utilized to implement data annotations by labelling all the screws that found in the dataset, as shown in Figure 3.6.

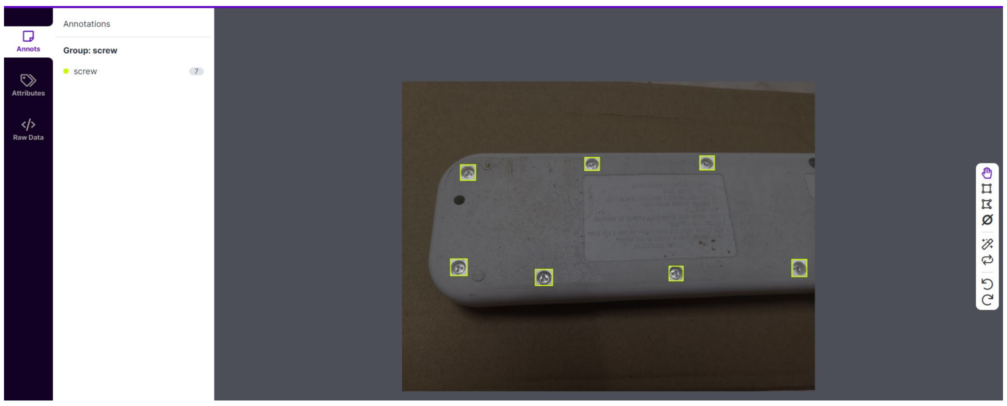


Figure 3.6: Data Annotation in Roboflow Software.

To construct high accuracy deep learning models, data pre-processing is a necessary step where the amount of data is normally directly proportional to the performance of the model. Hence, image augmentation techniques are required to generate massive amounts of data from an existing dataset.

In this project, Roboflow software is also utilized for data pre-processing. After the collected dataset is uploaded to the Roboflow software for labelling, the image augmentation techniques such as image cropping, image rotation, image blurring and image flipping are implemented, as shown in Figure 3.7. The total amounts of the collected dataset were 1920 pieces for each device after data augmentation. These datasets were split into 3 categories which are 88% training set, 8% validation set and 4% testing set. Training data is used to train the deep learning models while validation data and testing data are used to qualify the performance of the models.

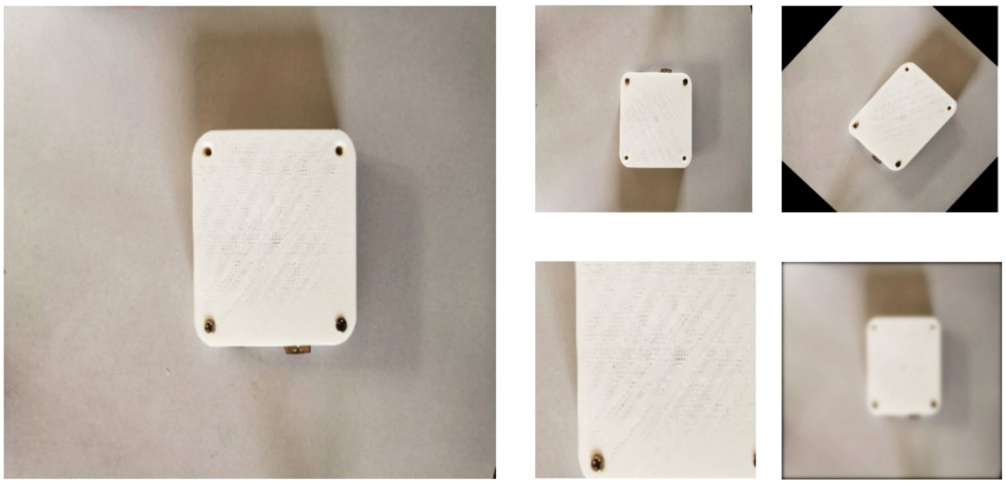


Figure 3.7: Example of Image Augmentation Techniques.

3.4 Object Detection Models Construction

In this project, YOLOv4, YOLOv4-Tiny, SSD MobilenetV2 FPNlite 320x320 and SSD MobilenetV2 FPNlite 640x640 were utilized as object detection models in this project to detect the screws on the manufactured products. Google Colaboratory is utilized in this project to construct the object detection models, including the training and testing process. Google Colaboratory is an environment that is used for deep learning models training on CPUs, GPUs and TPUs.

3.4.1 YOLO Models Construction

To construct the YOLOv4 and YOLOv4-Tiny on Google Colaboratory, some important steps are required to take which are dataset setup, model training with Darknet and TensorFlow Lite's.tflite representation conversion.

Since data annotation and pre-processing were implemented through Roboflow, Roboflow allows converting the dataset to YOLO Darknet format and exporting to the Google Colaboratory. Subsequently, the GPU environment was configured and the Darknet YOLOv4 training environment was installed. Besides, the newly released YOLOv4 and YOLOv4-Tiny training config files were downloaded and configured before training the YOLOv4 and YOLOv4-Tiny models.

There are several parameters for training models that are required to be set which are the number of classes (1), maximum batches (2000) and the number of filters (18). The number of classes is the number of the object class that is required to be detected. The value of maximum batches and number of filters are normally set according to the number of classes, as shown in Equation 3.1 and Equation 3.2.

$$maximum\ batches = number\ of\ classes\ x\ 2000 \qquad (3.1)$$

$$number\ of\ filters = (number\ of\ classes + 5)\ x\ 3 \qquad (3.2)$$

Their detailed training configurations are shown in Appendix A and Appendix B. Once the models were done training, the TensorFlow Lite library was utilized to convert the trained models to TensorFlow Lite format (.tlite) and these models were evaluated based on mAP and the prediction time. Additionally, the architecture of the YOLOv4 and YOLOv4-Tiny models are illustrated in Appendix C and Appendix D where the YOLOv4-Tiny is the lighter version of the YOLOv4 model where its network size is dramatically reduced. However, the input size of the YOLOv4 and YOLOv4-Tiny is 416 x 416.

3.4.2 SSD Models Construction

To construct the SSD MobilenetV2 FPNlite 320x320 and SSD MobilenetV2 FPNlite 640x640, the steps for models training are similar to YOLO models which include dataset setup, model training with TensorFlow-GPU and TensorFlow Lite's.tflite representation conversion.

For dataset setup, the dataset in Roboflow was converted to Pascal VOC format and exported to the Google Colabratory. Subsequently, the TensorFlow models' repository and object detection API were setup. Besides, the newly released SSD MobilenetV2 FPNlite 320x320 and SSD MobilenetV2 FPNlite 640x640 were downloaded and configured before training these models.

There are several parameters for training models that are required to be set which are the number of classes (1), the number of steps, fine-tune checkpoint type (detection) and metrics set (coco detection metrics). Their detailed training configurations are shown in Appendix E and Appendix F. Once the models were done training, the TensorFlow Lite library was utilized to convert the trained models to TensorFlow Lite format (.tlite) and these models were evaluated based on mAP and the prediction time. In addition, the architectures of the SSD MobilenetV2 FPNlite 320x320 and SSD MobilenetV2 FPNlite 640x640 are illustrated in Appendix G and Appendix H where the input size of the SSD MobilenetV2 FPNlite 320x320 is 320 x 320. In contrast, the input size of the SSD MobilenetV2 FPNlite 640x640 is 640 x 640.

3.5 Object Detection Apps Development

Before Android mobile app development, the primary functionalities of the mobile app are listed out, which is allowing users to implement real-time object detection and object detection from smartphone gallery. Subsequently, the UI of the object detection page in the mobile app is designed which includes “Real-time” labelled button and “Storage” labelled button that is used to trigger real-time object detection or upload image from the smartphone gallery for object detection, as shown in Figure 3.8.

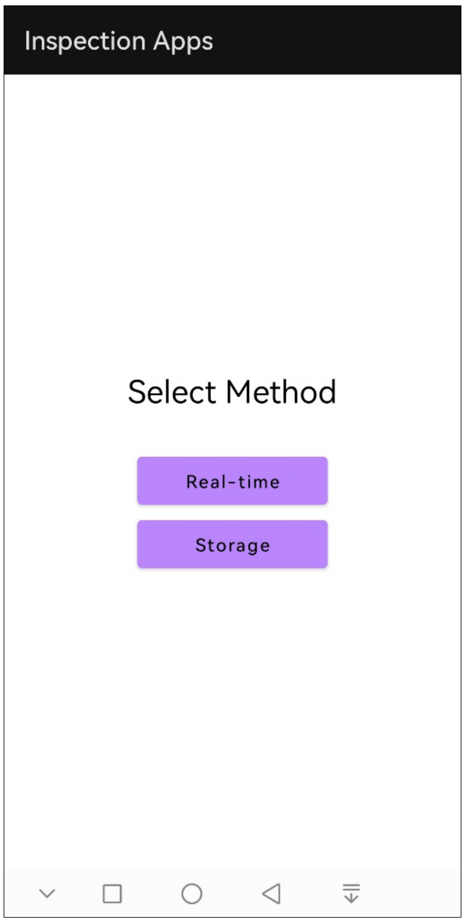


Figure 3.8: UI Design of Object Detection Page.

For the Android mobile app development, Android Studio software is utilized in this project. Since the main functionality of this mobile app is allowing users to real-time object detection or upload an image from the smartphone gallery for object detection, the required dependencies for accessing the smartphone's camera are imported.

To integrate the trained object detection model into the mobile app, the object detection model is required to be converted into the TensorFlow Lite format and integrated into the mobile app. To utilize the trained object detection model in the mobile app, the class method was created to load the TensorFlow lite model and the object class label file. A class method for getting the input image from the camera frame was created and the input image was pre-processed by resizing the image so that it can feed the input size of the TensorFlow Lite model.

The TensorFlow lite interpreter was utilized to interpret the model and implement prediction on the input image based on the model architecture where the output prediction results consist of the detected object class, locations of the detected object in the input image and the scores of the detected object. To display the prediction results, the rectangle boxes were drawn out and its label and scores were labelled out on the input image based on the location of the detected objects. After that, the edited image was replaced with the original image on the camera view. Once the mobile application development was getting done, the mobile application was debugged into the Android smartphone and implementing its functionalities testing. The important coding part with detailed comments was illustrated in Appendix I.

3.6 Webcam-Based Object Detection System Development

For webcam-based object detection, Google Colaboratory was utilized to access the webcam's camera for real-time object detection. The trained object detection models were uploaded to the Google Colaboratory and the video stream was defined by using the OpenCV library.

Subsequently, the frame was read from the webcam and was further processed to fit into the model for detection. The output was displayed and replaced the original input image, as shown in Figure 3.9. The important coding part with detailed comments was illustrated in Appendix J.

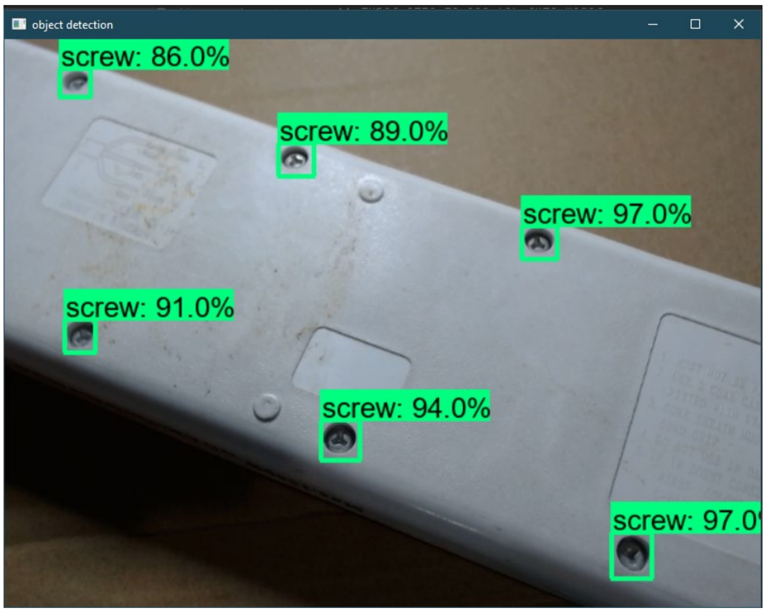


Figure 3.9: Webcam-Based Object Detection.

3.7 Smartphone Vs Webcam in Object Detection

An object detection experiment in reality is conducted to compare the performance of smartphone (Huawei Mate 20) and webcam (C920 Pro HD Webcam) in machine vision inspection.

The basic equipment that is required in this experiment are tripod stand, LED desk lamp and bounding box of 325mm x 225mm x 4mm. The experimental setups of smartphone and webcam for object detection are shown in Figure 3.10 and Figure 3.11. To be fair, the position of tripod stand, LED desk lamp and bounding box are fixed in smartphone and webcam experiments, since the performance of object detection will be affected by the various distance between the camera and targeted objects and different viewpoints. Furthermore, the height and the angle of the tripod stand that holds the smartphone and camera are also fixed and the targeted object is required to be placed within the bounding box.

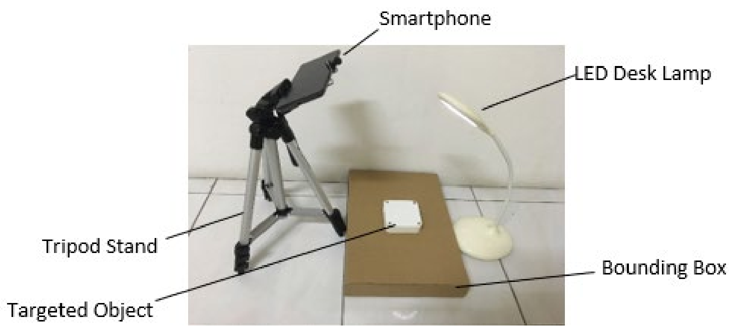


Figure 3.10: Smartphone-based Object Detection Setup.

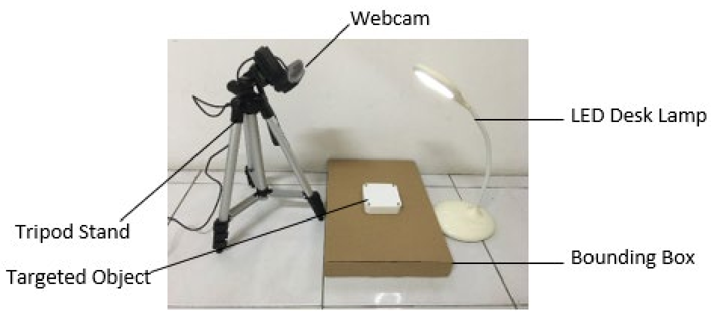


Figure 3.11: Webcam-based Object Detection Setup.

The performances of smartphone and webcam in the machine vision inspection are evaluated based on precision and inference time with actual targeted objects. The results of each part are recorded in Table 3.2 and will be analysed.

Table 3.2: Performance Evaluation Table of Object Detection Systems.

Types of Vision Systems	**Object Detection Models**	**Precision (%)**	**Inference Time (ms)**
Smartphone	**SSD MobilenetV2 FPNlite 320x320**		
	SSD MobilenetV2 FPNlite 640x640		
	YOLO v4-Tiny		
	YOLO v4		
Webcam	**SSD MobilenetV2 FPNlite 320x320**		
	SSD MobilenetV2 FPNlite 640x640		
	YOLO v4-Tiny		
	YOLO v4		

3.8 Summary

In summary, the detailed methodology, work plan and project scope of this project are discussed in this chapter. Furthermore, the detailed project development flow is discussed which includes data collection, data annotation and pre-processing, object detection models construction, object detection apps development, webcam-based object detection system development and experiment for the performance comparison between smartphone-based and webcam-based in object detection.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

In this section, object detection models were evaluated which are YOLOv4, YOLOv4-Tiny, SSD MobilenetV2 FPNlite 320x320 and SSD MobilenetV2 FPNlite 640x640 for smartphone and webcam devices to ensure the object detection models are well trained.

Furthermore, the performances of smartphone and webcam in machine vision inspection were compared and discussed to investigate whether the performance of a smartphone in machine vision inspection can be benchmarked against a webcam used in a production line. Subsequently, the finalised object detection app was also discussed in this section.

4.2 Models Evaluation

Since the training methods of YOLO and SSD are different, their evaluation results are also slightly different. The YOLO models' evaluation results include mAP on validation data and training loss while SSD models' evaluation results include training loss, validation loss, learning rate and mAP on validation data. Their detailed models' evaluation results for each device were discussed in the next subchapters.

4.2.1 Results on Smartphone-Based

In this subchapter, YOLO models and SSD models that using captured image from smartphone as input datasets were evaluated.

To evaluate the YOLOv4 and YOLOv4-Tiny models, the training loss and mAP by using the validation dataset were recorded and illustrated in Figure 4.1 and Figure 4.2. For the mAP evaluations by using the validation dataset, they were started evaluating after 1000 training iterations and evaluating every 100 training iterations.

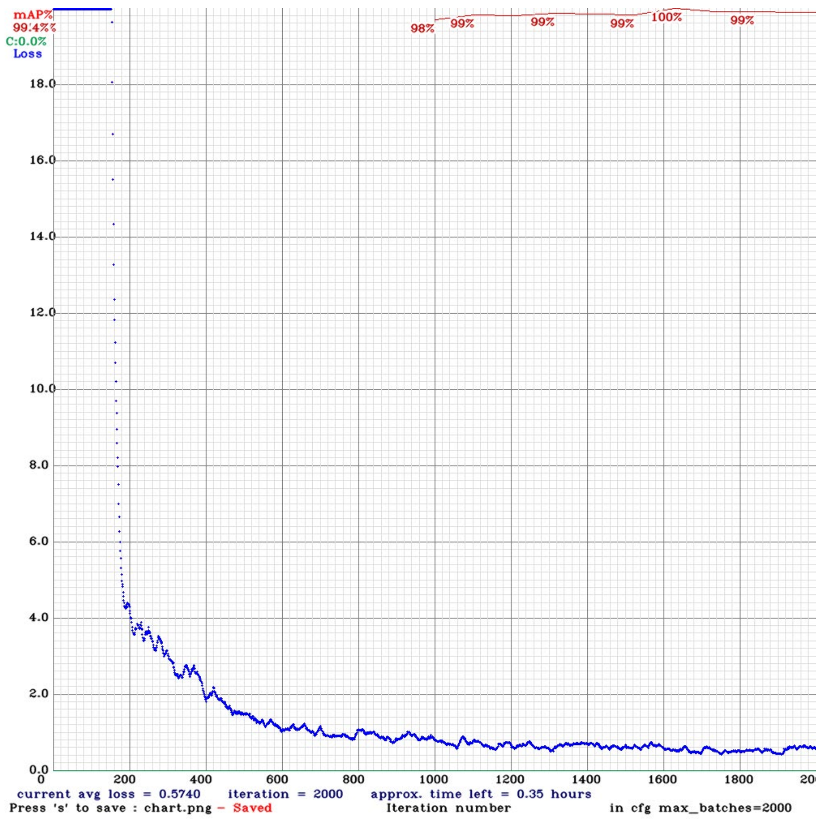


Figure 4.1: Evaluation Result of the Smartphone-Based YOLOv4 Model.

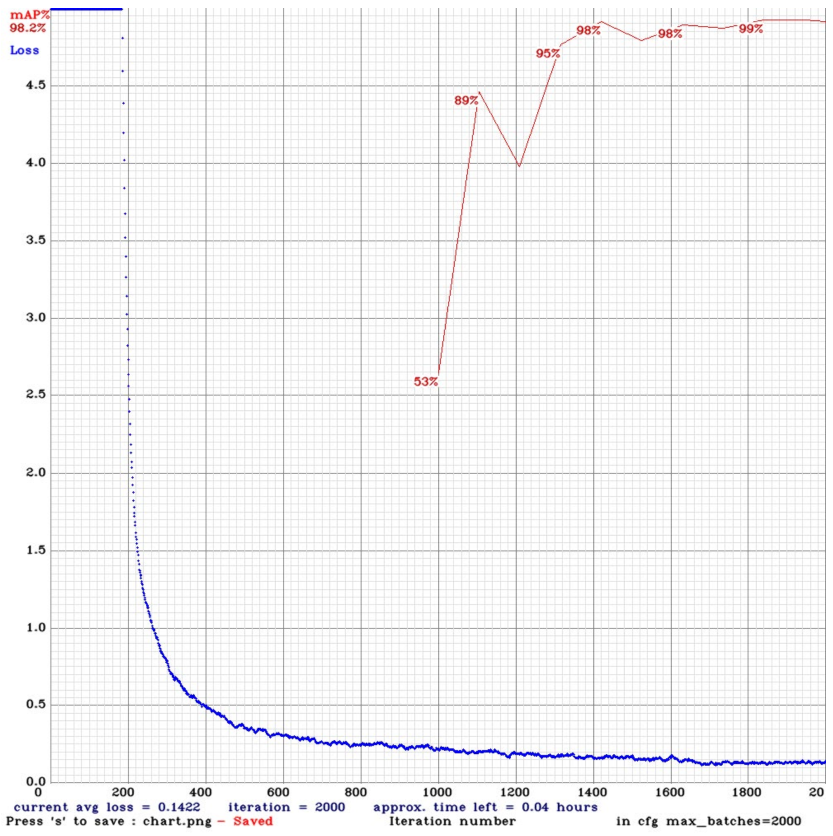


Figure 4.2: Evaluation Result of the Smartphone-Based YOLOv4-Tiny Model.

Based on Figure 4.1 and Figure 4.2, the final mAP of YOLOv4 is 99.4% and its final average training loss is 0.5740. Besides, the final mAP of YOLOv4-Tiny is 98.2% and its final average training loss is 0.1422. Hence, the smartphone-based YOLOv4 and YOLOv4-Tiny models are considered as well trained because the models do not overfit or underfit during training based on the training loss and mAP graph. Since YOLOv4-Tiny is the lighter version of YOLOv4 where its network size is dramatically reduced, its average mAP is slightly lower than the YOLOv4.

To evaluate the SSD MobilenetV2 FPNlite 320x320 and SSD MobilenetV2 FPNlite 640x640 models, the training loss, validation loss, learning rate and mAP on validation data were recorded down and illustrated in Figure 4.3 and Figure 4.4.

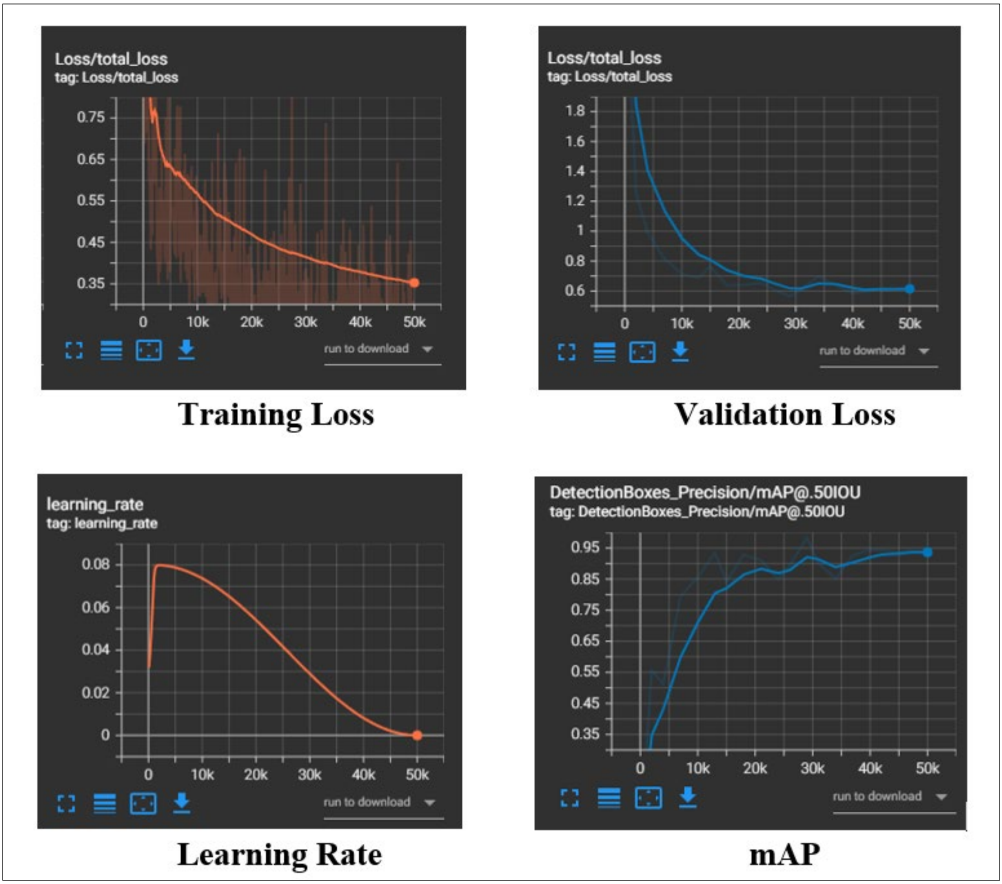


Figure 4.3: Evaluation Results of the Smartphone-Based SSD 320 Model.

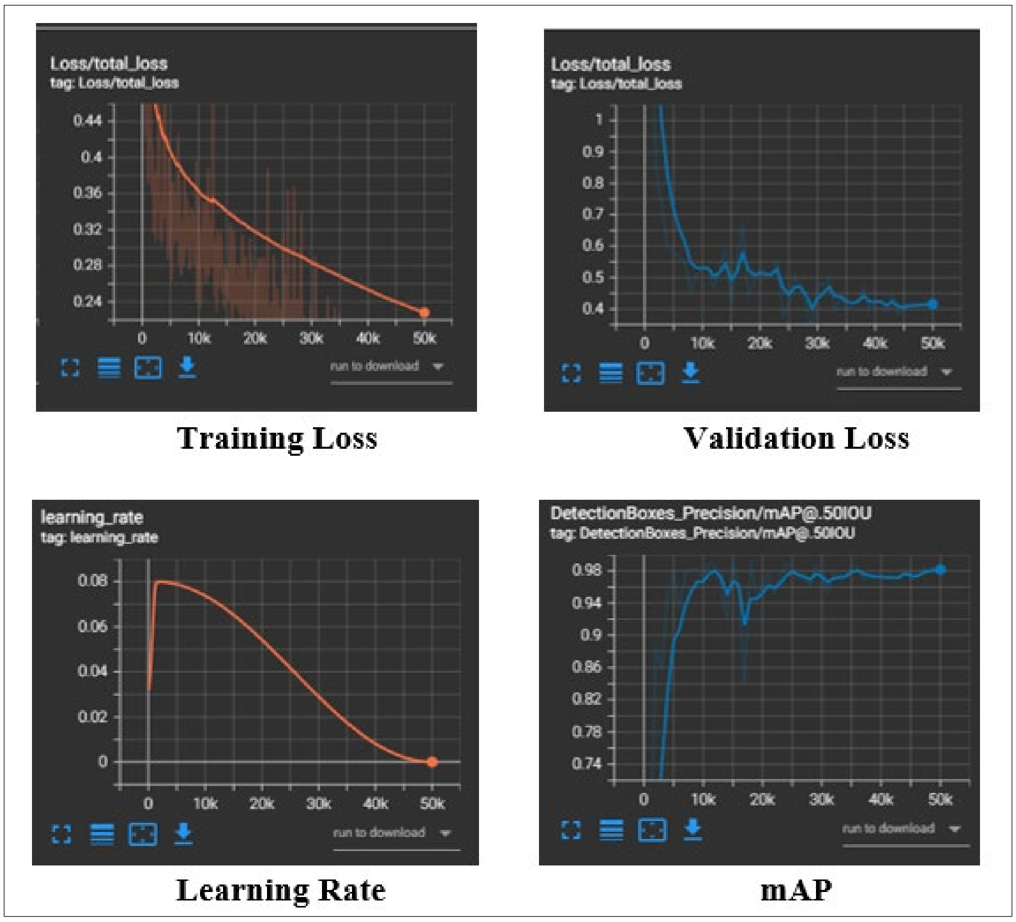


Figure 4.4: Evaluation Results of the Smartphone-Based SSD 640 Model.

Based on Figure 4.3, the final training loss of SSD MobilenetV2 FPNlite 320x320 is 0.35 and its final validation loss is 0.6. The learning rate of this model is also decreased to around 0 and its final mAP is up to around 95%. Based on Figure 4.4, the final training loss of SSD MobilenetV2 FPNlite 640x640 is 0.23 and its final validation loss is 0.4. The learning rate of this model is also reduced to around 0 and its final mAP is up to around 98%.

Therefore, the smartphone-based SSD MobilenetV2 FPNlite 320x320 and SSD MobilenetV2 FPNlite 640x640 models are considered as well trained because the models do not overfit or underfit during training based on the training loss, validation loss, learning rate and mAP graph. Since the input image size to feed the SSD MobilenetV2 FPNlite 320x320 is smaller than SSD MobilenetV2 FPNlite 640x640, its mAP is lower than the SSD MobilenetV2 FPNlite 640x640.

Furthermore, the training time for each model was recorded down and the test dataset was also utilized to evaluate the performance of trained models based on mAP, as shown in Table 4.1.

Table 4.1: Smartphone-Based Object Detection Models Evaluation.

Object Detection Models	**mAP (%)**	**Training Time**
YOLOv4	97.6	5hrs 35mins
YOLOv4-Tiny	96.3	50mins
SSD MobilenetV2 FPNlite 320x320	91.2	1hrs 34mins
SSD MobilenetV2 FPNlite 640x640	95.1	11hrs 18mins

Based on Table 4.1, overall trained object detection models perform well detection on test datasets which are above the 90% mAP. Moreover, the training time of YOLOv4-Tiny is the shortest among these models while SSD MobilenetV2 FPNlite 640x640 took the longest time for training.

4.2.2 Results on Webcam-Based

In this subchapter, YOLO models and SSD models that using captured image from webcam as input datasets were evaluated. The evaluation methods for webcam-based object detection models are similar with smartphone-based object detection models.

For the webcam-based YOLOv4 and YOLOv4-Tiny models, the training loss and mAP by using the validation dataset were recorded and illustrated in Figure 4.5 and Figure 4.6.

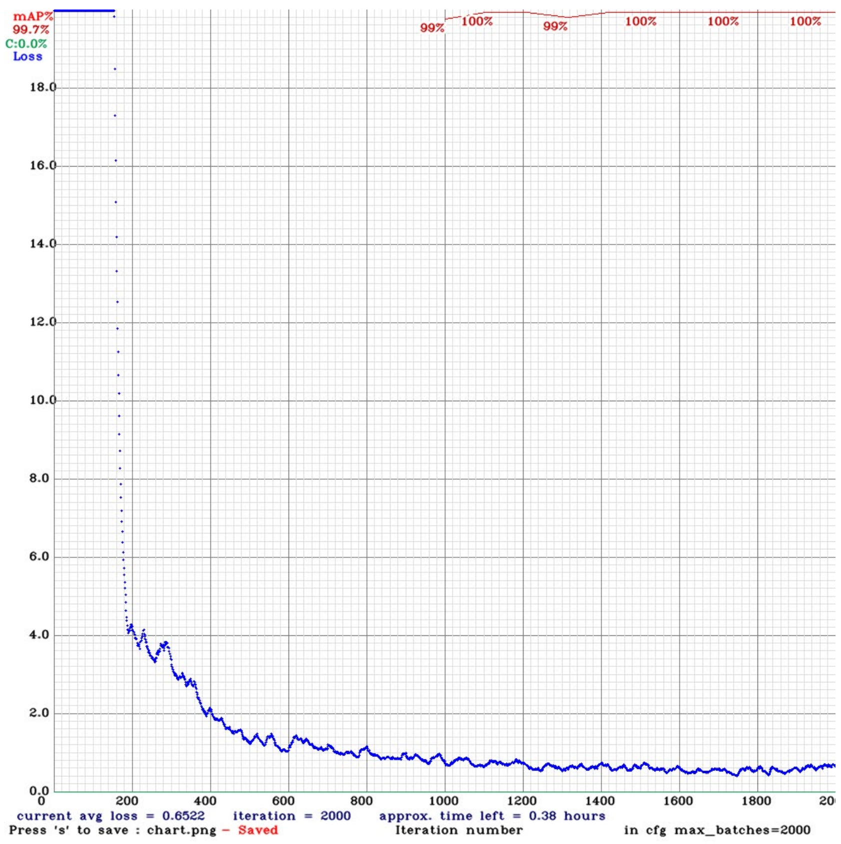


Figure 4.5: Evaluation Result of the Webcam-Based YOLOv4 Model.

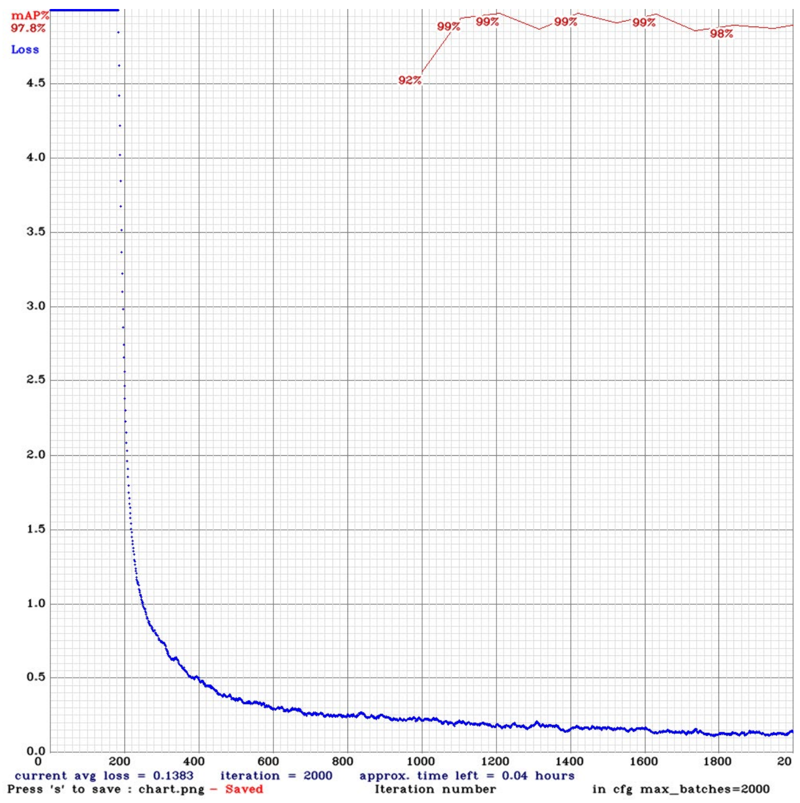


Figure 4.6: Evaluation Result of the Smartphone-Based YOLOv4-Tiny Model.

According to Figure 4.5 and Figure 4.6, the final mAP of YOLOv4 is 99.7% and its final average training loss is 0.6522. Furthermore, the final mAP of YOLOv4-Tiny is 97.8% and its final average training loss is 0.1383. Therefore, the webcam-based YOLOv4 and YOLOv4-Tiny models are considered as well trained because the models do not overfit or underfit during training based on the training loss and mAP graph. Since YOLOv4-Tiny is the lighter version of YOLOv4 where its network size is dramatically reduced, its average mAP is slightly lower than the YOLOv4 which is similar with smartphone-based.

For the webcam-based SSD MobilenetV2 FPNlite 320x320 and SSD MobilenetV2 FPNlite 640x640 models' evaluation, the training loss, validation loss, learning rate and mAP on validation data were recorded and illustrated in Figure 4.7 and Figure 4.8.

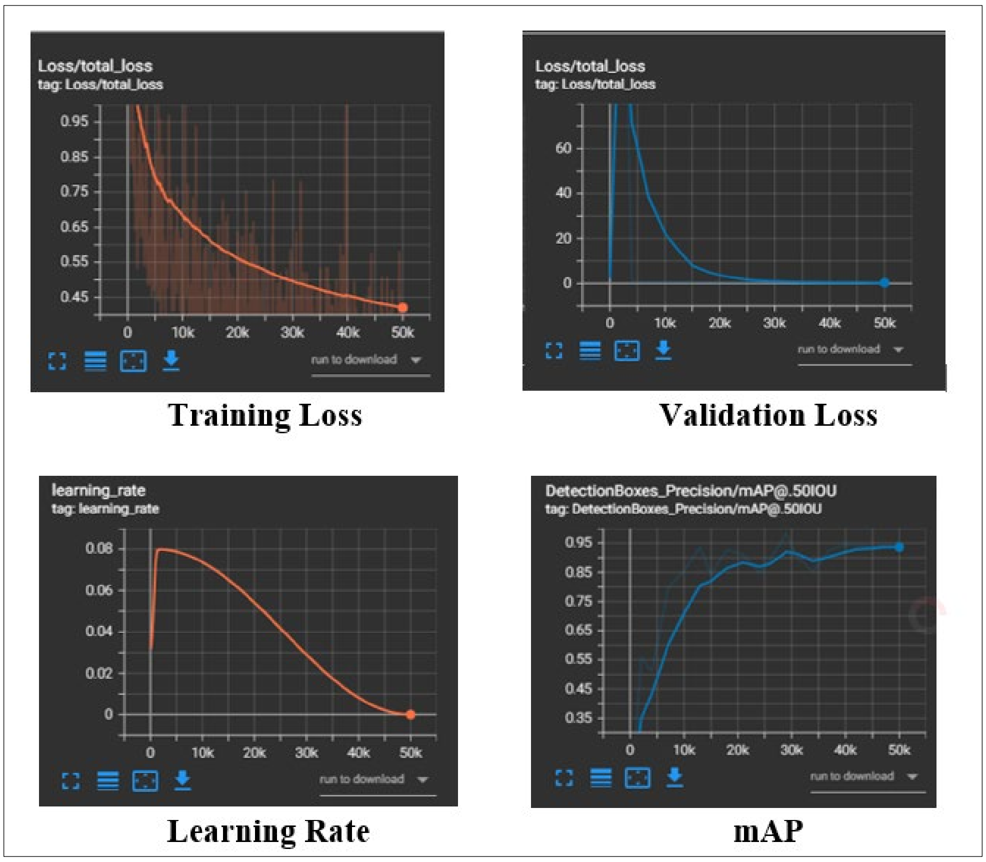


Figure 4.7: Evaluation Results of the Webcam-Based SSD 320 Model.

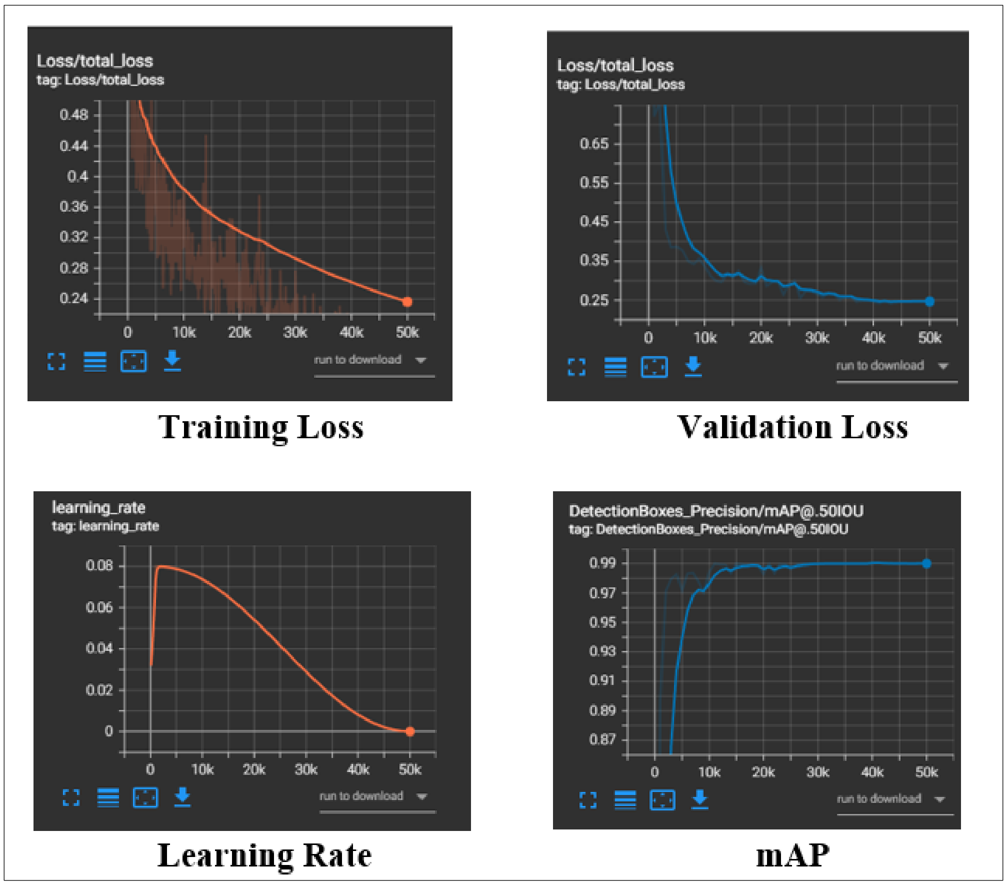


Figure 4.8: Evaluation Results of the Webcam-Based SSD 640 Model.

According to Figure 4.7, the final training loss of webcam-based SSD MobilenetV2 FPNlite 320x320 is around 0.40 and its final validation loss is close to 0. The learning rate of this model is also decreased to around 0 and its final mAP is up to around 95%. According to Figure 4.8, the final training loss of webcam-based SSD MobilenetV2 FPNlite 640x640 is 0.24 and its final validation loss is 0.25. The learning rate of this model is also reduced to around 0 and its final mAP is up to around 99%.

Hence, the webcam-based SSD MobilenetV2 FPNlite 320x320 and SSD MobilenetV2 FPNlite 640x640 models are considered as well trained because the models do not overfit or underfit during training based on the training loss, validation loss, learning rate and mAP graph.

Moreover, the training time for each webcam-based object detection model was noted down and the test dataset was also used to evaluate the performance of trained models based on mAP, as shown in Table 4.2.

Table 4.2: Webcam-Based Object Detection Models Evaluation.

Object Detection Models	**mAP (%)**	**Training Time**
YOLOv4	99.5	5hrs 22mins
YOLOv4-Tiny	97.6	47mins
SSD MobilenetV2 FPNlite 320x320	92.2	1hrs 31mins
SSD MobilenetV2 FPNlite 640x640	98.3	10hrs 11mins

According to Table 4.2, the overall webcam-based trained object detection models perform well detection on test datasets which are above the 90% mAP. Besides, the training time of YOLOv4-Tiny is the shortest among these models while SSD MobilenetV2 FPNlite 640x640 took the longest time for training which is similar with the smartphone-based object detection models.

4.3 Object Detection Results in Reality

To investigate whether the performance of a smartphone in machine vision inspection can be benchmarked against a webcam used in a production line, the object detection experiment in reality is conducted. Therefore, the performances of smartphone and webcam in this object detection experiment were compared and discussed in this subchapter.

For the object detection in reality through the smartphone and webcam, the sample output prediction results are illustrated in Figure 4.9 and Figure 4.10. Furthermore, the table of performance comparison between smartphone-based and webcam-based in real-time screw detection was constructed and illustrated in Table 4.3.

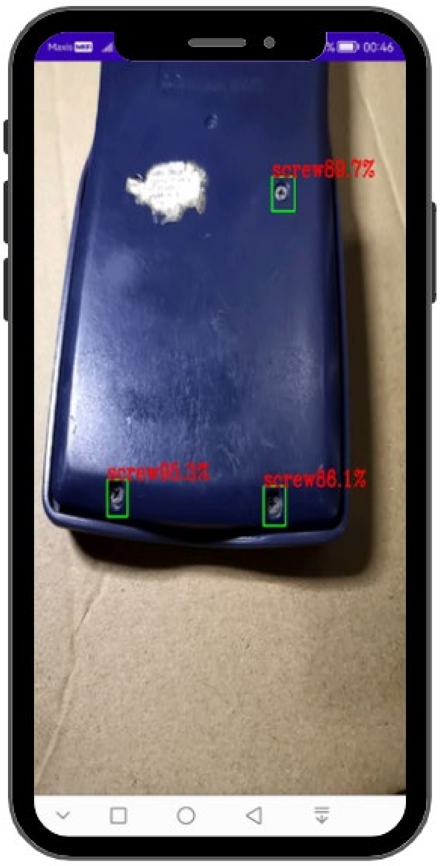


Figure 4.9: Sample Output Prediction Result Through Smartphone.

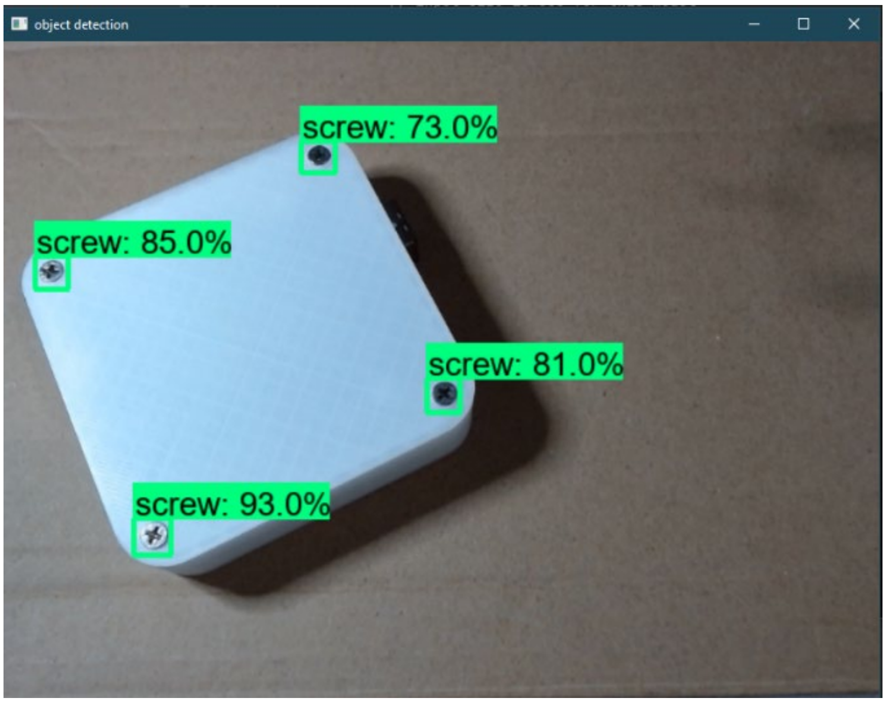


Figure 4.10: Sample Output Prediction Result Through Webcam.

Table 4.3: Performance Comparison Table.

Types of Vision Systems	**Object Detection Models**	**Precision (%)**	**Inference Time (ms)**
Smartphone	**SSD MobilenetV2 FPNlite 320x320**	73.61	83
	SSD MobilenetV2 FPNlite 640x640	92.31	138
	YOLO v4-Tiny	97.43	159
	YOLO v4	97.33	1366
Webcam	**SSD MobilenetV2 FPNlite 320x320**	78.10	69
	SSD MobilenetV2 FPNlite 640x640	94.01	71
	YOLO v4-Tiny	97.80	73
	YOLO v4	97.62	88

Based on Table 4.3, the precision results on smartphone-based are almost similar to webcam-based. The slight differences of the precision between smartphone and webcam are because the object detection models are converted into TensorFlow Lite format before integrating into the mobile app. TensorFlow Lite format is the lightweight version of the TensorFlow model where it is smaller in size, faster and less computationally expensive. Hence, its prediction performance was affected by comparing it with TensorFlow model.

For the inference time, the smartphone-based object detection models took a longer inference time by comparing with webcam-based object detection models. This is because the computing power of the computer that is connected to the webcam is stronger than the smartphone's computing power. However, the inference time of SSD MobilenetV2 FPNlite 320x320, SSD MobilenetV2 FPNlite 640x640 and YOLOv4-Tiny in smartphone-based are considered short and acceptable for the real applications of AVI system.

Other than that, this experiment results illustrated that YOLOv4-Tiny models show the optimal performances in screw detection with both smartphone and webcam devices based on precision and inference time. Based on this performance comparison table, it also proves that the performance of the smartphone in a machine vision inspection system can be benchmarked against the webcam used in the production line, since the precision of each object detection model for both vision devices is close. Therefore, the objectives of this study were achieved, which include devising a learning-based algorithm to automatically detect the existence of items in manufactured products and investigating the performance of a smartphone in a machine vision inspection system that can be benchmarked against a webcam used in the production line.

4.4 Object Detection App Evaluation

In this section, the overall object detection app and its main functionalities are discussed. Since the YOLOv4-Tiny model illustrated the optimal performances in screw detection among other object detection models, it was selected for utilizing and integration into the finalised object detection app.

In this object detection app, its main functionalities are allowing users to implement real-time machine vision inspection on smartphone and implement machine vision inspection by selecting the image from the smartphone gallery.

Once the object detection app is launched, the homepage of the app for the selection of real-time detection and detection from image storage is displayed on the smartphone device's screen, as shown in Figure 4.11. For the homepage of the object detection app, it consists of two buttons which are "Real-time" and "Storage" labelled buttons. The real-time object detection page will be launched once the 'Real-time" labelled button is pressed while the function of selecting the image from the smartphone gallery for object detection will be triggered if the "Storage" labelled button is pressed.

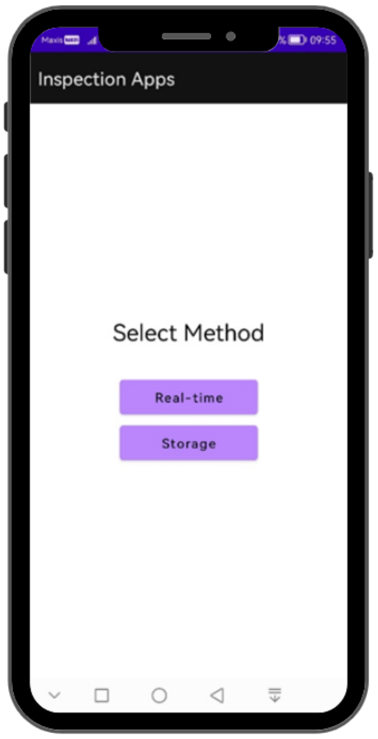


Figure 4.11: Homepage of the Object Detection App.

Figure 4.12 illustrates the real-time object detection page on the smartphone’s screen after the “Real-time” labelled button on the homepage was pressed. At this point, the user is directed to the camera view and is allowed to implement real-time screw detection. In this real-time object detection page, the input image from the camera view is pre-processed before being fed into the object detection model for prediction. For the output prediction, it will return the class labels, locations and score values of the detected objects. Subsequently, these output results will be drawn in the original frame and displayed in the updated frame to the camera view where the prediction result is also illustrated in Figure 4.12.

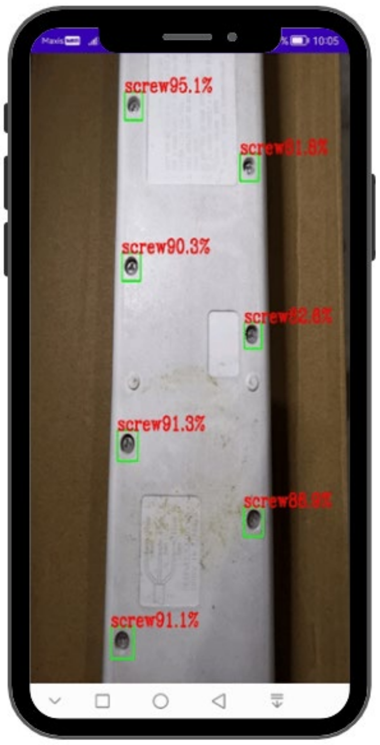


Figure 4.12: Real-time Object Detection Page.

Figure 4.13 shows the section where the user can select the image from the smartphone gallery for detection after the “Storage” labelled button on the homepage was pressed. At this point, the user is directed to the smartphone gallery and is allowed to select the input image from the smartphone gallery. The selected image from the photo gallery is pre-processed before being fed into the object detection model for prediction. For the output prediction, it will return the class labels, locations and score values of the detected objects which is similar to the real-time object detection. Subsequently, these output results will be drawn in the original frame and displayed the updated frame to the camera view where the prediction result is illustrated in Figure 4.14. To reselect the image from the smartphone gallery for detection, the button with “Select Image” label on the top centre of the page can be pressed, as shown in Figure 4.14. Hence, one of the objectives in this study was achieved which is developing a mobile app for users to access the smartphone camera for inspection.

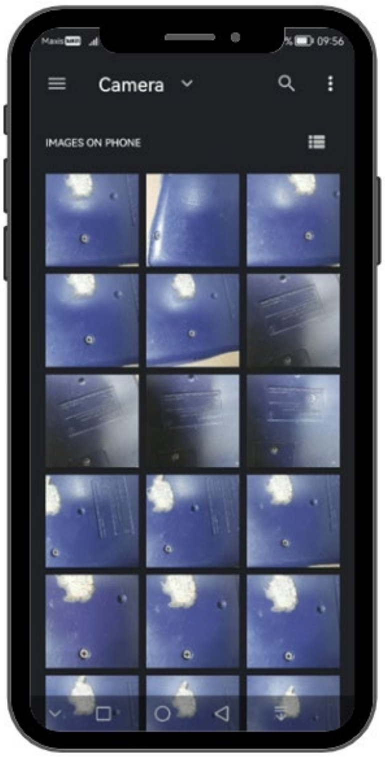


Figure 4.13: Image Selection from Phone Gallery.

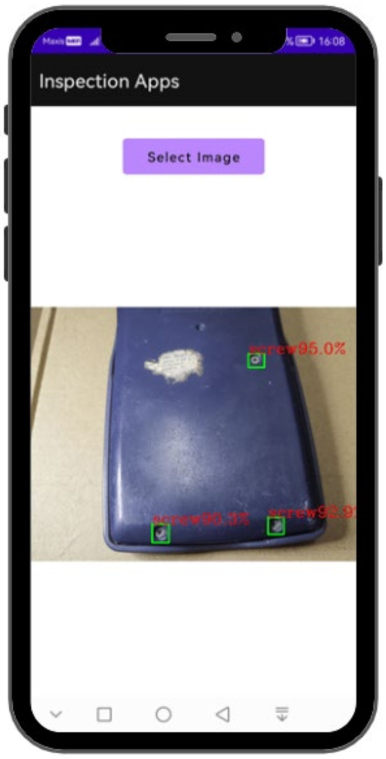


Figure 4.14: Detection Result from Image Storage.

4.5 Summary

In summary, all the objectives of this study were achieved which are developing a mobile app for users to access the smartphone camera for inspection, devising a learning-based algorithm to automatically detect the existence of items in manufactured products and investigating the performance of a smartphone in machine vision inspection system can be benchmarked against a webcam used in the production line.

For the smartphone-based and webcam-based object detection models construction, they were considered as well trained and had good prediction results on the test dataset. Additionally, the precision results of object detection in reality through the smartphone-based are almost similar to the webcam-based and the YOLOv4-Tiny models show the optimal performances in screw detection with both smartphone and webcam devices based on precision and inference time. Hence, the YOLOv4-Tiny model is selected for utilizing and integrating into the finalised object detection app. In this screw detection app, its main functionalities are allowing the user to implement real-time screw inspection on a smartphone and implement screw inspection by selecting the image from the smartphone gallery.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In conclusion, the main objectives of this study had been achieved which include developing a mobile app for users to access the smartphone camera for inspection, devising a learning-based algorithm to automatically detect the existence of items in manufactured products and investigating the performance of a smartphone in machine vision inspection can be benchmarked against a webcam used in the production line.

The proposed object detection models in this study which are YOLOv4, YOLOv4-Tiny, SSD MobilenetV2 FPNlite 320x320 and SSD MobilenetV2 FPNlite 640x640 were considered as well trained and had good prediction results on the test dataset. For the screw detection experiment in reality, the precision results of the smartphone-based are almost similar with the webcam-based. This experiment results obviously proved that smartphone in machine vision inspection can be benchmarked against a webcam used in a production line. Hence, smartphones can be an alternative device for the visual system and provide a low-cost machine vision inspection system with high efficiency and flexibility for small and medium scale enterprises.

The YOLOv4-Tiny models illustrate the optimal performances in screw detection with both smartphone and webcam devices based on precision and prediction time. Therefore, the YOLOv4-Tiny model was chosen for utilizing and integrating into the finalised object detection app. The finalised object detection app was developed to allow users to access the smartphone camera for inspection where the mobile application can automatically detect the existence of screws in manufactured products. The main functionalities of this mobile application are allowing the user to implement real-time screw inspection on a smartphone and implement screw inspection by selecting the image from the smartphone gallery.

5.2 Recommendations for future work

In this study, all the main objectives had been achieved. However, there are some recommendations for the future work of the development of this study. Since the object detection models in this study are only used to detect the screw, the recommendation for future work is to extend the object detection class that can normally be found in manufactured products such as bolts and nuts. Other than object detection, other types of deep learning methods in machine vision inspection can be studied such as image classification, semantic segmentation and instance segmentation in order to detect or classify the defects, contaminants, flaws and other irregularities in manufactured products.

For the object detection app, the UI design can be improved through the popular frontend framework such as Flutter which is a cross-platform mobile application development framework which can be implemented in Android Studio. This is because the Flutter framework supports IOS and Android platforms, rich motion APIs, strong widget support and built-in material design (Borah, 2021). Furthermore, Firebase can be utilized as the backend framework of this mobile application which promotes real-time database, file storage, analytics, authentication, push messaging and configuration features (Demangeon and Janvier, 2019). Hence, the relevant data and prediction results can be stored in the cloud or accessed from the cloud through the Firebase framework. Additionally, there are some mobile application features that can be add-on such as allowing users to trace back the prediction result, history data and data analytics, and customize their own dataset including data annotation on the smartphone, instead of using external platform such as Roboflow.

REFERENCES

Alsing, O., 2018. Mobile Object Detection using TensorFlow Lite and Transfer Learning. [online] Available at: <https://www.diva-portal.org/smash/get/diva2:1242627/FULLTEXT01.pdf>.

Bao, N., Ran, X., Wu, Z., Xue, Y. and Wang, K., 2018. Design of inspection system of glaze defect on the surface of ceramic pot based on machine vision. [online] pp.1486–1492. Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/stamp/stamp.jsp?tp=&arnumber=8285043>.

Batchelor, B.G. and Whelan, P.F., 2002. Intelligent Vision Systems for Industry. *Intelligent Vision Systems for Industry*. [online] Available at: <http://doras.dcu.ie/18215/1/IVSI.pdf>.

Borah, B.R., 2021. *Top Mobile App Development Frameworks in 2021*. Available at: <https://www.clariontech.com/blog/top-mobile-app-development-frameworks-in-2019>.

Burresi, G., Lorusso, M., Graziani, L., Comacchio, A., Trotta, F. and Rizzo, A., 2021. Image-Based Defect Detection in Assembly Line with Machine Learning. *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, [online] pp.1–5. Available at: <https://www.researchgate.net/publication/348389736_Image-Based_Surface_Defect_Detection_Using_Deep_Learning_A_Review>.

Chudzik, P., Mitchell, A., Alkaseem, M., Wu, Y., Fang, S., Hudaib, T., Pearson, S. and Al-Diri, B., 2020. Mobile Real-Time Grasshopper Detection and Data Aggregation Framework. *Scientific Reports*, [online] 10(1), pp.1–11. Available at: <https://www.nature.com/articles/s41598-020-57674-8>.

Cognex Corporation, 2020. DEEP LEARNING FOR FACTORY AUTOMATION Combining artificial intelligence with machine vision. [online] Available at: <https://www.cognex.com/resources/white-papers-articles/deep-learning-for-factory-automation>.

Demangeon, J. and Janvier, A., 2019. *Using Firebase for Backend-as-a-Service_ Pros and Cons*. Available at: <https://marmelab.com/blog/2019/10/23/feedback-on-firebase-in-project-start-up.html>.

Edwards, E., 2020. *An Introduction to Machine Vision and the Machine Vision System*. *Thomas Industry*. Available at: <https://www.thomasnet.com/articles/automation-electronics/machine-vision-systems/>.

Fell, J., 2017. *How to Choose a Machine Vision Camera | 2017-03-01 | Quality Magazine*. *Quality Magazine*. Available at: <https://www.qualitymag.com/articles/93861-how-to-choose-a-machine-vision-camera>.

Hashim, H.S., Abdullah, S.N.H.S. and Prabuwono, A.S., 2010. Automated visual inspection for metal parts based on morphology and fuzzy rules. *ICCAIE 2010 - 2010 International Conference on Computer Applications and Industrial Electronics*, [online] (Iccaie 2010), pp.527–531. Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/5735137?arnumber=5735137>.

Invonto, 2021. *Mobile app development process, Process of Mobile App Development*. Available at: <https://siamcomputing.com/digital-transformation/mobile-app-development-process/>.

Jakhete, S.A., Bagmar, P., Dorle, A., Rajurkar, A. and Pimplikar, P., 2019. Object Recognition App for Visually Impaired. *2019 IEEE Pune Section International Conference, PuneCon 2019*, [online] pp.2019–2022. Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/9105670>.

James Le, 2018. *The 5 Computer Vision Techniques That Will Change How You See The World | by James Le | Heartbeat*. Available at: <https://heartbeat.fritz.ai/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world-1ee19334354b>.

Jia, J., 2009. A machine vision application for industrial assembly inspection. *2009 2nd International Conference on Machine Vision, ICMV 2009*, [online] pp.172–176. Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/5381107>.

Khandelwal, R., 2020. *Evaluating performance of an object detection model | by Renu Khandelwal | Towards Data Science*. Available at: <https://towardsdatascience.com/evaluating-performance-of-an-object-detection-model-137a349c517b>.

Kim, J. and Cho, J., 2020. Exploring a multimodal mixture-Of-YOLOs framework for advanced real-time object detection. *Applied Sciences (Switzerland)*, [online] 10(2). Available at: <https://www.mdpi.com/2076-3417/10/2/612>.

Narvekar, M.C., 2020. Flower classification using CNN and transfer learning in CNN- Agriculture Perspective. [online] pp.660–664. Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/9316030>.

Parico, A.I.B. and Ahamed, T., 2021. Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT. [online] pp.1–32. Available at: <https://www.mdpi.com/1424-8220/21/14/4803>.

Patidar, A., 2021. Towards Analyzing Mobile App Characteristics for Mobile Software Development. [online] pp.786–790. Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/9441097>.

Razdan, V. and Bateman, R., 2015. Investigation into the use of smartphone as a machine vision device for engineering metrology and flaw detection, with focus on drilling. *Automated Visual Inspection and Machine Vision*, [online] 9530(April 2020), p.95300C. Available at: <https://www.researchgate.net/publication/283226812_Investigation_into_the_use_of_smartphone_as_a_machine_vision_device_for_engineering_metrology_and_flaw_detection_with_focus_on_drilling>.

Saha, S., 2018. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way _ by Sumit Saha _ Towards Data Science*. Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.

Song, L., Li, X., Yang, Y., Zhu, X., Guo, Q. and Yang, H., 2018. Detection of micro-defects on metal screw surfaces based on deep convolutional neural networks. *Sensors (Switzerland)*, [online] 18(11). Available at: <https://www.mdpi.com/1424-8220/18/11/3709>.

Statista Research Department, 2021. *Mobile internet usage worldwide - Statistics & Facts | Statista*. Available at: <https://www.statista.com/topics/779/mobile-internet/>.

Voulodimos, A., Doulamis, N., Doulamis, A. and Protopapadakis, E., 2018. Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, [online] 2018. Available at: <https://www.hindawi.com/journals/cin/2018/7068349/>.

Wadawadagi, V., 2020. *Metrics to Use to Evaluate Deep Learning Object Detectors*. Available at: <https://www.kdnuggets.com/2020/08/metrics-evaluate-deep-learning-object-detectors.html>.

Wang, Y., Niu, P., Guo, X., Yang, G. and Chen, J., 2021. Single Shot Multibox Detector with Deconvolutional Region Magnification Procedure. *IEEE Access*, [online] 9, pp.47767–47776. Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/9385156>.

Wilson, A., 2019. *Deep learning brings a new dimension to machine vision | Vision Systems Design*. Available at: <https://www.vision-systems.com/home/article/16736100/deep-learning-brings-a-new-dimension-to-machine-vision>.

Zhang, Q., Pan, X., Liu, F. and Lu, S., 2020. A benchmark dataset for real-time detection of icons in mobile apps and a small-scale feature module. *Pattern Recognition Letters*, [online] 136, pp.87–93. Available at: <https://doi.org/10.1016/j.patrec.2020.04.037>.

Zhongyao, C., Juelin, Z., Cen, C., Xiaoxi, Y., Fan, W., Yue, L. and Zeng, Z., 2020. Robust Train Component Detection with Cascade Convolutional Neural Networks based on Structure Knowledge. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems, ITSC 2020*. [online] Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/9294755>.

Zhu, L., Spachos, P., Pensini, E. and Plataniotis, K.N., 2021. Deep learning and machine vision for food processing: A survey. *Current Research in Food Science*, [online] 4(December 2020), pp.233–249. Available at: <https://doi.org/10.1016/j.crfs.2021.03.009>.

APPENDICES

APPENDIX A: Training Configuration for YOLOv4.

```
[net]
batch=64
subdivisions=24
width=416
height=416
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue = .1

learning_rate=0.001
burn_in=1000
max_batches=2000
policy=steps
steps=1600.0,1800.0
scales=.1,.1

#cutmix=1
mosaic=1

#:104x104 54:52x52 85:26x26 104:13x13 for 416

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=mish

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
```

APPENDIX A: (Continued)

```
pad=1
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-7

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish
```

APPENDIX A: (Continued)

```
# Downsample

[convolutional]
batch_normalize=1
filters=128
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
```

APPENDIX A: (Continued)

```
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-10

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

# Downsample

[convolutional]
batch_normalize=1
filters=256
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
```

APPENDIX A: (Continued)

```
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear
```

APPENDIX A: (Continued)

```
[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
```

APPENDIX A: (Continued)

```
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
```

APPENDIX A: (Continued)

```
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-28

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

# Downsample

[convolutional]
batch_normalize=1
filters=512
size=3
stride=2
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
```

APPENDIX A: (Continued)

```
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
```

APPENDIX A: (Continued)

```
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish
```

APPENDIX A: (Continued)

```
[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
```

```
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=mish

[route]
layers = -1,-28

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

# Downsample

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=2
pad=1
activation=mish
```

APPENDIX A: (Continued)

```
[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[route]
layers = -2

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
```

```
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=mish

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=mish

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
```

APPENDIX A: (Continued)

```
pad=1
activation=mish

[route]
layers = -1,-16

[convolutional]
batch_normalize=1
filters=1024
size=1
stride=1
pad=1
activation=mish

##########################

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

### SPP ###
[maxpool]
stride=1
size=5

[route]
layers=-2

[maxpool]
stride=1
size=9
```

APPENDIX A: (Continued)

```
[route]
layers=-4

[maxpool]
stride=1
size=13

[route]
layers=-1,-3,-5,-6
### End SPP ###

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = 85

[convolutional]
batch_normalize=1
filters=256
```

APPENDIX A: (Continued)

```
size=1
stride=1
pad=1
activation=leaky

[route]
layers = -1, -3

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
```

APPENDIX A: (Continued)

```
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = 54

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[route]
layers = -1, -3

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky
```

```
[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

##########################

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=18
activation=linear

[yolo]
mask = 0,1,2
anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459, 401
classes=1
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
scale_x_y = 1.2
iou_thresh=0.213
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
nms_kind=greedynms
beta_nms=0.6
max_delta=5


[route]
layers = -4

[convolutional]
batch_normalize=1
size=3
stride=2
pad=1
```

```
filters=256
activation=leaky

[route]
layers = -1, -16

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=512
```

```
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=18
activation=linear

[yolo]
mask = 3,4,5
anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459, 401
classes=1
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
scale_x_y = 1.1
iou_thresh=0.213
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
nms_kind=greedynms
beta_nms=0.6
max_delta=5


[route]
layers = -4

[convolutional]
batch_normalize=1
size=3
stride=2
pad=1
filters=512
activation=leaky

[route]
layers = -1, -37

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
```

```
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=18
activation=linear

[yolo]
mask = 6,7,8
anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459, 401
classes=1
num=9
jitter=.3
```

```
ignore_thresh = .7
```

```
truth_thresh = 1
random=1
scale_x_y = 1.05
iou_thresh=0.213
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
nms_kind=greedynms
beta_nms=0.6
max_delta=5
```

APPENDIX B: Training Configuration for YOLOv4-Tiny.

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=16
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.00261
burn_in=1000
max_batches = 2000
policy=steps
steps=1600.0,1800.0
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[route]
layers=-1
groups=2
```

APPENDIX B: (Continued)

```
group_id=1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

[route]
layers = -1,-2

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[route]
layers = -6,-1

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[route]
layers=-1
groups=2
group_id=1

[convolutional]
batch_normalize=1
filters=64
```

```
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[route]
layers = -1,-2

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[route]
layers = -6,-1

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[route]
layers=-1
groups=2
group_id=1

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky
```

APPENDIX B: (Continued)

```
[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[route]
layers = -1,-2

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[route]
layers = -6,-1

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

##################################

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky
```

APPENDIX B: (Continued)

```
[convolutional]
size=1
stride=1
pad=1
filters=18
activation=linear

[yolo]
mask = 3,4,5
anchors = 10,14,  23,27,  37,58,  81,82,  135,169,  344,319
classes=1
num=6
jitter=.3
scale_x_y = 1.05
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
ignore_thresh = .7
truth_thresh = 1
random=0
nms_kind=greedynms
beta_nms=0.6

[route]
layers = -4

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = -1, 23

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[convolutional]
size=1
stride=1
pad=1
```

APPENDIX B: (Continued)

```
filters=18
activation=linear

[yolo]
mask = 1,2,3
anchors = 10,14,  23,27,  37,58,  81,82,  135,169,  344,319
classes=1
num=6
jitter=.3
scale_x_y = 1.05
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
ignore_thresh = .7
truth_thresh = 1
random=0
nms_kind=greedynms
beta_nms=0.6
```

APPENDIX C: Architecture of YOLOv4.

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to                     
==================================================================================================
input_1 (InputLayer)            [(None, 416, 416, 3) 0                                            
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 416, 416, 32) 864         input_1[0][0]                    
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 416, 416, 32) 128         conv2d[0][0]                     
__________________________________________________________________________________________________
tf_op_layer_Softplus (TensorFlo [(None, 416, 416, 32 0           batch_normalization[0][0]        
__________________________________________________________________________________________________
tf_op_layer_Tanh (TensorFlowOpL [(None, 416, 416, 32 0           tf_op_layer_Softplus[0][0]       
__________________________________________________________________________________________________
tf_op_layer_Mul (TensorFlowOpLa [(None, 416, 416, 32 0           batch_normalization[0][0]        
                                                                 tf_op_layer_Tanh[0][0]           
__________________________________________________________________________________________________
zero_padding2d (ZeroPadding2D)  (None, 417, 417, 32) 0           tf_op_layer_Mul[0][0]            
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 208, 208, 64) 18432       zero_padding2d[0][0]             
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 208, 208, 64) 256         conv2d_1[0][0]                   
__________________________________________________________________________________________________
tf_op_layer_Softplus_1 (TensorF [(None, 208, 208, 64 0           batch_normalization_1[0][0]      
__________________________________________________________________________________________________
tf_op_layer_Tanh_1 (TensorFlowO [(None, 208, 208, 64 0           tf_op_layer_Softplus_1[0][0]     
__________________________________________________________________________________________________
tf_op_layer_Mul_1 (TensorFlowOp [(None, 208, 208, 64 0           batch_normalization_1[0][0]      
                                                                 tf_op_layer_Tanh_1[0][0]         
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 208, 208, 64) 4096        tf_op_layer_Mul_1[0][0]          
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 208, 208, 64) 256         conv2d_3[0][0]                   
__________________________________________________________________________________________________
tf_op_layer_Softplus_3 (TensorF [(None, 208, 208, 64 0           batch_normalization_3[0][0]      
__________________________________________________________________________________________________
tf_op_layer_Tanh_3 (TensorFlowO [(None, 208, 208, 64 0           tf_op_layer_Softplus_3[0][0]     
__________________________________________________________________________________________________
tf_op_layer_Mul_3 (TensorFlowOp [(None, 208, 208, 64 0           batch_normalization_3[0][0]      
                                                                 tf_op_layer_Tanh_3[0][0]         
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 208, 208, 32) 2048        tf_op_layer_Mul_3[0][0]          
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 208, 208, 32) 128         conv2d_4[0][0]                   
__________________________________________________________________________________________________
tf_op_layer_Softplus_4 (TensorF [(None, 208, 208, 32 0           batch_normalization_4[0][0]      
__________________________________________________________________________________________________
tf_op_layer_Tanh_4 (TensorFlowO [(None, 208, 208, 32 0           tf_op_layer_Softplus_4[0][0]     
__________________________________________________________________________________________________
tf_op_layer_Mul_4 (TensorFlowOp [(None, 208, 208, 32 0           batch_normalization_4[0][0]      
                                                                 tf_op_layer_Tanh_4[0][0]         
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 208, 208, 64) 18432       tf_op_layer_Mul_4[0][0]          
__________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 208, 208, 64) 256         conv2d_5[0][0]                   
__________________________________________________________________________________________________
tf_op_layer_Softplus_5 (TensorF [(None, 208, 208, 64 0           batch_normalization_5[0][0]      
__________________________________________________________________________________________________
tf_op_layer_Tanh_5 (TensorFlowO [(None, 208, 208, 64 0           tf_op_layer_Softplus_5[0][0]     
__________________________________________________________________________________________________
tf_op_layer_Mul_5 (TensorFlowOp [(None, 208, 208, 64 0           batch_normalization_5[0][0]      
                                                                 tf_op_layer_Tanh_5[0][0]         
__________________________________________________________________________________________________
tf_op_layer_AddV2 (TensorFlowOp [(None, 208, 208, 64 0           tf_op_layer_Mul_3[0][0]          
                                                                 tf_op_layer_Mul_5[0][0]          
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 208, 208, 64) 4096        tf_op_layer_AddV2[0][0]          
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 208, 208, 64) 4096        tf_op_layer_Mul_1[0][0]          
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 208, 208, 64) 256         conv2d_6[0][0]                   
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 208, 208, 64) 256         conv2d_2[0][0]                   
__________________________________________________________________________________________________
tf_op_layer_Softplus_6 (TensorF [(None, 208, 208, 64 0           batch_normalization_6[0][0]      
__________________________________________________________________________________________________
tf_op_layer_Softplus_2 (TensorF [(None, 208, 208, 64 0           batch_normalization_2[0][0]      
__________________________________________________________________________________________________
tf_op_layer_Tanh_6 (TensorFlowO [(None, 208, 208, 64 0           tf_op_layer_Softplus_6[0][0]     
__________________________________________________________________________________________________
tf_op_layer_Tanh_2 (TensorFlowO [(None, 208, 208, 64 0           tf_op_layer_Softplus_2[0][0]     
__________________________________________________________________________________________________
tf_op_layer_Mul_6 (TensorFlowOp [(None, 208, 208, 64 0           batch_normalization_6[0][0]      
                                                                 tf_op_layer_Tanh_6[0][0]         
__________________________________________________________________________________________________
tf_op_layer_Mul_2 (TensorFlowOp [(None, 208, 208, 64 0           batch_normalization_2[0][0]      
                                                                 tf_op_layer_Tanh_2[0][0]         
__________________________________________________________________________________________________
tf_op_layer_concat (TensorFlowO [(None, 208, 208, 12 0           tf_op_layer_Mul_6[0][0]          
                                                                 tf_op_layer_Mul_2[0][0]          
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 208, 208, 64) 8192        tf_op_layer_concat[0][0]         
__________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 208, 208, 64) 256         conv2d_7[0][0]                   
__________________________________________________________________________________________________
tf_op_layer_Softplus_7 (TensorF [(None, 208, 208, 64 0           batch_normalization_7[0][0]      
```

APPENDIX C: (Continued)

```
__________________________________________________________________________________________________
tf_op_layer_Tanh_7 (TensorFlowO [(None, 208, 208, 64 0           tf_op_layer_Softplus_7[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_7 (TensorFlowOp [(None, 208, 208, 64 0           batch_normalization_7[0][0]
                                                                 tf_op_layer_Tanh_7[0][0]
__________________________________________________________________________________________________
zero_padding2d_1 (ZeroPadding2D (None, 209, 209, 64) 0           tf_op_layer_Mul_7[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 104, 104, 128 73728       zero_padding2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, 104, 104, 128 512         conv2d_8[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_8 (TensorF [(None, 104, 104, 12 0           batch_normalization_8[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_8 (TensorFlowO [(None, 104, 104, 12 0           tf_op_layer_Softplus_8[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_8 (TensorFlowOp [(None, 104, 104, 12 0           batch_normalization_8[0][0]
                                                                 tf_op_layer_Tanh_8[0][0]
__________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 104, 104, 64) 8192        tf_op_layer_Mul_8[0][0]
__________________________________________________________________________________________________
batch_normalization_10 (BatchNo (None, 104, 104, 64) 256         conv2d_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_10 (Tensor [(None, 104, 104, 64 0           batch_normalization_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_10 (TensorFlow [(None, 104, 104, 64 0           tf_op_layer_Softplus_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_10 (TensorFlowO [(None, 104, 104, 64 0           batch_normalization_10[0][0]
                                                                 tf_op_layer_Tanh_10[0][0]
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 104, 104, 64) 4096        tf_op_layer_Mul_10[0][0]
__________________________________________________________________________________________________
batch_normalization_11 (BatchNo (None, 104, 104, 64) 256         conv2d_11[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_11 (Tensor [(None, 104, 104, 64 0           batch_normalization_11[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_11 (TensorFlow [(None, 104, 104, 64 0           tf_op_layer_Softplus_11[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_11 (TensorFlowO [(None, 104, 104, 64 0           batch_normalization_11[0][0]
                                                                 tf_op_layer_Tanh_11[0][0]
__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 104, 104, 64) 36864       tf_op_layer_Mul_11[0][0]
__________________________________________________________________________________________________
batch_normalization_12 (BatchNo (None, 104, 104, 64) 256         conv2d_12[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_12 (Tensor [(None, 104, 104, 64 0           batch_normalization_12[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_12 (TensorFlow [(None, 104, 104, 64 0           tf_op_layer_Softplus_12[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_12 (TensorFlowO [(None, 104, 104, 64 0           batch_normalization_12[0][0]
                                                                 tf_op_layer_Tanh_12[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_1 (TensorFlow [(None, 104, 104, 64 0           tf_op_layer_Mul_10[0][0]
                                                                 tf_op_layer_Mul_12[0][0]
__________________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 104, 104, 64) 4096        tf_op_layer_AddV2_1[0][0]
__________________________________________________________________________________________________
batch_normalization_13 (BatchNo (None, 104, 104, 64) 256         conv2d_13[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_13 (Tensor [(None, 104, 104, 64 0           batch_normalization_13[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_13 (TensorFlow [(None, 104, 104, 64 0           tf_op_layer_Softplus_13[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_13 (TensorFlowO [(None, 104, 104, 64 0           batch_normalization_13[0][0]
                                                                 tf_op_layer_Tanh_13[0][0]
__________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 104, 104, 64) 36864       tf_op_layer_Mul_13[0][0]
__________________________________________________________________________________________________
batch_normalization_14 (BatchNo (None, 104, 104, 64) 256         conv2d_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_14 (Tensor [(None, 104, 104, 64 0           batch_normalization_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_14 (TensorFlow [(None, 104, 104, 64 0           tf_op_layer_Softplus_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_14 (TensorFlowO [(None, 104, 104, 64 0           batch_normalization_14[0][0]
                                                                 tf_op_layer_Tanh_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_2 (TensorFlow [(None, 104, 104, 64 0           tf_op_layer_AddV2_1[0][0]
                                                                 tf_op_layer_Mul_14[0][0]
__________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 104, 104, 64) 4096        tf_op_layer_AddV2_2[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 104, 104, 64) 8192        tf_op_layer_Mul_8[0][0]
__________________________________________________________________________________________________
batch_normalization_15 (BatchNo (None, 104, 104, 64) 256         conv2d_15[0][0]
__________________________________________________________________________________________________
batch_normalization_9 (BatchNor (None, 104, 104, 64) 256         conv2d_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_15 (Tensor [(None, 104, 104, 64 0           batch_normalization_15[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_9 (TensorF [(None, 104, 104, 64 0           batch_normalization_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_15 (TensorFlow [(None, 104, 104, 64 0           tf_op_layer_Softplus_15[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_9 (TensorFlowO [(None, 104, 104, 64 0           tf_op_layer_Softplus_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_15 (TensorFlowO [(None, 104, 104, 64 0           batch_normalization_15[0][0]
                                                                 tf_op_layer_Tanh_15[0][0]
__________________________________________________________________________________________________
```

APPENDIX C: (Continued)

```
tf_op_layer_Mul_9 (TensorFlowOp [(None, 104, 104, 64 0           batch_normalization_9[0][0]
                                                                 tf_op_layer_Tanh_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_1 (TensorFlo [(None, 104, 104, 12 0           tf_op_layer_Mul_15[0][0]
                                                                 tf_op_layer_Mul_9[0][0]
__________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 104, 104, 128 16384       tf_op_layer_concat_1[0][0]
__________________________________________________________________________________________________
batch_normalization_16 (BatchNo (None, 104, 104, 128 512         conv2d_16[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_16 (Tensor [(None, 104, 104, 12 0           batch_normalization_16[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_16 (TensorFlow [(None, 104, 104, 12 0           tf_op_layer_Softplus_16[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_16 (TensorFlowO [(None, 104, 104, 12 0           batch_normalization_16[0][0]
                                                                 tf_op_layer_Tanh_16[0][0]
__________________________________________________________________________________________________
zero_padding2d_2 (ZeroPadding2D (None, 105, 105, 128 0           tf_op_layer_Mul_16[0][0]
__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 52, 52, 256)  294912      zero_padding2d_2[0][0]
__________________________________________________________________________________________________
batch_normalization_17 (BatchNo (None, 52, 52, 256)  1024        conv2d_17[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_17 (Tensor [(None, 52, 52, 256) 0           batch_normalization_17[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_17 (TensorFlow [(None, 52, 52, 256) 0           tf_op_layer_Softplus_17[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_17 (TensorFlowO [(None, 52, 52, 256) 0           batch_normalization_17[0][0]
                                                                 tf_op_layer_Tanh_17[0][0]
__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 52, 52, 128)  32768       tf_op_layer_Mul_17[0][0]
__________________________________________________________________________________________________
batch_normalization_19 (BatchNo (None, 52, 52, 128)  512         conv2d_19[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_19 (Tensor [(None, 52, 52, 128) 0           batch_normalization_19[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_19 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_19[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_19 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_19[0][0]
                                                                 tf_op_layer_Tanh_19[0][0]
__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 52, 52, 128)  16384       tf_op_layer_Mul_19[0][0]
__________________________________________________________________________________________________
batch_normalization_20 (BatchNo (None, 52, 52, 128)  512         conv2d_20[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_20 (Tensor [(None, 52, 52, 128) 0           batch_normalization_20[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_20 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_20[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_20 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_20[0][0]
                                                                 tf_op_layer_Tanh_20[0][0]
__________________________________________________________________________________________________
conv2d_21 (Conv2D)              (None, 52, 52, 128)  147456      tf_op_layer_Mul_20[0][0]
__________________________________________________________________________________________________
batch_normalization_21 (BatchNo (None, 52, 52, 128)  512         conv2d_21[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_21 (Tensor [(None, 52, 52, 128) 0           batch_normalization_21[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_21 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_21[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_21 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_21[0][0]
                                                                 tf_op_layer_Tanh_21[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_3 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Mul_19[0][0]
                                                                 tf_op_layer_Mul_21[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 52, 52, 128)  16384       tf_op_layer_AddV2_3[0][0]
__________________________________________________________________________________________________
batch_normalization_22 (BatchNo (None, 52, 52, 128)  512         conv2d_22[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_22 (Tensor [(None, 52, 52, 128) 0           batch_normalization_22[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_22 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_22[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_22 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_22[0][0]
                                                                 tf_op_layer_Tanh_22[0][0]
__________________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, 52, 52, 128)  147456      tf_op_layer_Mul_22[0][0]
__________________________________________________________________________________________________
batch_normalization_23 (BatchNo (None, 52, 52, 128)  512         conv2d_23[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_23 (Tensor [(None, 52, 52, 128) 0           batch_normalization_23[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_23 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_23[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_23 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_23[0][0]
                                                                 tf_op_layer_Tanh_23[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_4 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_AddV2_3[0][0]
                                                                 tf_op_layer_Mul_23[0][0]
__________________________________________________________________________________________________
conv2d_24 (Conv2D)              (None, 52, 52, 128)  16384       tf_op_layer_AddV2_4[0][0]
__________________________________________________________________________________________________
batch_normalization_24 (BatchNo (None, 52, 52, 128)  512         conv2d_24[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_24 (Tensor [(None, 52, 52, 128) 0           batch_normalization_24[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_24 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_24[0][0]
__________________________________________________________________________________________________
```

APPENDIX C: (Continued)

```
tf_op_layer_Mul_24 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_24[0][0]
                                                                 tf_op_layer_Tanh_24[0][0]
__________________________________________________________________________________________________
conv2d_25 (Conv2D)              (None, 52, 52, 128)  147456      tf_op_layer_Mul_24[0][0]
__________________________________________________________________________________________________
batch_normalization_25 (BatchNo (None, 52, 52, 128)  512         conv2d_25[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_25 (Tensor [(None, 52, 52, 128) 0           batch_normalization_25[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_25 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_25[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_25 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_25[0][0]
                                                                 tf_op_layer_Tanh_25[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_5 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_AddV2_4[0][0]
                                                                 tf_op_layer_Mul_25[0][0]
__________________________________________________________________________________________________
conv2d_26 (Conv2D)              (None, 52, 52, 128)  16384       tf_op_layer_AddV2_5[0][0]
__________________________________________________________________________________________________
batch_normalization_26 (BatchNo (None, 52, 52, 128)  512         conv2d_26[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_26 (Tensor [(None, 52, 52, 128) 0           batch_normalization_26[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_26 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_26[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_26 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_26[0][0]
                                                                 tf_op_layer_Tanh_26[0][0]
__________________________________________________________________________________________________
conv2d_27 (Conv2D)              (None, 52, 52, 128)  147456      tf_op_layer_Mul_26[0][0]
__________________________________________________________________________________________________
batch_normalization_27 (BatchNo (None, 52, 52, 128)  512         conv2d_27[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_27 (Tensor [(None, 52, 52, 128) 0           batch_normalization_27[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_27 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_27[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_27 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_27[0][0]
                                                                 tf_op_layer_Tanh_27[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_6 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_AddV2_5[0][0]
                                                                 tf_op_layer_Mul_27[0][0]
__________________________________________________________________________________________________
conv2d_28 (Conv2D)              (None, 52, 52, 128)  16384       tf_op_layer_AddV2_6[0][0]
__________________________________________________________________________________________________
batch_normalization_28 (BatchNo (None, 52, 52, 128)  512         conv2d_28[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_28 (Tensor [(None, 52, 52, 128) 0           batch_normalization_28[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_28 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_28[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_28 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_28[0][0]
                                                                 tf_op_layer_Tanh_28[0][0]
__________________________________________________________________________________________________
conv2d_29 (Conv2D)              (None, 52, 52, 128)  147456      tf_op_layer_Mul_28[0][0]
__________________________________________________________________________________________________
batch_normalization_29 (BatchNo (None, 52, 52, 128)  512         conv2d_29[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_29 (Tensor [(None, 52, 52, 128) 0           batch_normalization_29[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_29 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_29[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_29 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_29[0][0]
                                                                 tf_op_layer_Tanh_29[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_7 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_AddV2_6[0][0]
                                                                 tf_op_layer_Mul_29[0][0]
__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, 52, 52, 128)  16384       tf_op_layer_AddV2_7[0][0]
__________________________________________________________________________________________________
batch_normalization_30 (BatchNo (None, 52, 52, 128)  512         conv2d_30[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_30 (Tensor [(None, 52, 52, 128) 0           batch_normalization_30[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_30 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_30[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_30 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_30[0][0]
                                                                 tf_op_layer_Tanh_30[0][0]
__________________________________________________________________________________________________
conv2d_31 (Conv2D)              (None, 52, 52, 128)  147456      tf_op_layer_Mul_30[0][0]
__________________________________________________________________________________________________
batch_normalization_31 (BatchNo (None, 52, 52, 128)  512         conv2d_31[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_31 (Tensor [(None, 52, 52, 128) 0           batch_normalization_31[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_31 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_31[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_31 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_31[0][0]
                                                                 tf_op_layer_Tanh_31[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_8 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_AddV2_7[0][0]
                                                                 tf_op_layer_Mul_31[0][0]
__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 52, 52, 128)  16384       tf_op_layer_AddV2_8[0][0]
__________________________________________________________________________________________________
batch_normalization_32 (BatchNo (None, 52, 52, 128)  512         conv2d_32[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_32 (Tensor [(None, 52, 52, 128) 0           batch_normalization_32[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_32 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_32[0][0]
```

APPENDIX C: (Continued)

```
__________________________________________________________________________________________________
tf_op_layer_Mul_32 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_32[0][0]
                                                                 tf_op_layer_Tanh_32[0][0]
__________________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, 52, 52, 128)  147456      tf_op_layer_Mul_32[0][0]
__________________________________________________________________________________________________
batch_normalization_33 (BatchNo (None, 52, 52, 128)  512         conv2d_33[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_33 (Tensor [(None, 52, 52, 128) 0           batch_normalization_33[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_33 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_33[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_33 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_33[0][0]
                                                                 tf_op_layer_Tanh_33[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_9 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_AddV2_8[0][0]
                                                                 tf_op_layer_Mul_33[0][0]
__________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 52, 52, 128)  16384       tf_op_layer_AddV2_9[0][0]
__________________________________________________________________________________________________
batch_normalization_34 (BatchNo (None, 52, 52, 128)  512         conv2d_34[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_34 (Tensor [(None, 52, 52, 128) 0           batch_normalization_34[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_34 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_34[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_34 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_34[0][0]
                                                                 tf_op_layer_Tanh_34[0][0]
__________________________________________________________________________________________________
conv2d_35 (Conv2D)              (None, 52, 52, 128)  147456      tf_op_layer_Mul_34[0][0]
__________________________________________________________________________________________________
batch_normalization_35 (BatchNo (None, 52, 52, 128)  512         conv2d_35[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_35 (Tensor [(None, 52, 52, 128) 0           batch_normalization_35[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_35 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_35[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_35 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_35[0][0]
                                                                 tf_op_layer_Tanh_35[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_10 (TensorFlo [(None, 52, 52, 128) 0           tf_op_layer_AddV2_9[0][0]
                                                                 tf_op_layer_Mul_35[0][0]
__________________________________________________________________________________________________
conv2d_36 (Conv2D)              (None, 52, 52, 128)  16384       tf_op_layer_AddV2_10[0][0]
__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 52, 52, 128)  32768       tf_op_layer_Mul_17[0][0]
__________________________________________________________________________________________________
batch_normalization_36 (BatchNo (None, 52, 52, 128)  512         conv2d_36[0][0]
__________________________________________________________________________________________________
batch_normalization_18 (BatchNo (None, 52, 52, 128)  512         conv2d_18[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_36 (Tensor [(None, 52, 52, 128) 0           batch_normalization_36[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_18 (Tensor [(None, 52, 52, 128) 0           batch_normalization_18[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_36 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_36[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_18 (TensorFlow [(None, 52, 52, 128) 0           tf_op_layer_Softplus_18[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_36 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_36[0][0]
                                                                 tf_op_layer_Tanh_36[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_18 (TensorFlowO [(None, 52, 52, 128) 0           batch_normalization_18[0][0]
                                                                 tf_op_layer_Tanh_18[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_2 (TensorFlo [(None, 52, 52, 256) 0           tf_op_layer_Mul_36[0][0]
                                                                 tf_op_layer_Mul_18[0][0]
__________________________________________________________________________________________________
conv2d_37 (Conv2D)              (None, 52, 52, 256)  65536       tf_op_layer_concat_2[0][0]
__________________________________________________________________________________________________
batch_normalization_37 (BatchNo (None, 52, 52, 256)  1024        conv2d_37[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_37 (Tensor [(None, 52, 52, 256) 0           batch_normalization_37[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_37 (TensorFlow [(None, 52, 52, 256) 0           tf_op_layer_Softplus_37[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_37 (TensorFlowO [(None, 52, 52, 256) 0           batch_normalization_37[0][0]
                                                                 tf_op_layer_Tanh_37[0][0]
__________________________________________________________________________________________________
zero_padding2d_3 (ZeroPadding2D (None, 53, 53, 256)  0           tf_op_layer_Mul_37[0][0]
__________________________________________________________________________________________________
conv2d_38 (Conv2D)              (None, 26, 26, 512)  1179648     zero_padding2d_3[0][0]
__________________________________________________________________________________________________
batch_normalization_38 (BatchNo (None, 26, 26, 512)  2048        conv2d_38[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_38 (Tensor [(None, 26, 26, 512) 0           batch_normalization_38[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_38 (TensorFlow [(None, 26, 26, 512) 0           tf_op_layer_Softplus_38[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_38 (TensorFlowO [(None, 26, 26, 512) 0           batch_normalization_38[0][0]
                                                                 tf_op_layer_Tanh_38[0][0]
__________________________________________________________________________________________________
conv2d_40 (Conv2D)              (None, 26, 26, 256)  131072      tf_op_layer_Mul_38[0][0]
__________________________________________________________________________________________________
batch_normalization_40 (BatchNo (None, 26, 26, 256)  1024        conv2d_40[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_40 (Tensor [(None, 26, 26, 256) 0           batch_normalization_40[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_40 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_40[0][0]
```

APPENDIX C: (Continued)

```
__________________________________________________________________________________________________
tf_op_layer_Mul_40 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_40[0][0]
                                                                 tf_op_layer_Tanh_40[0][0]
__________________________________________________________________________________________________
conv2d_41 (Conv2D)              (None, 26, 26, 256)  65536       tf_op_layer_Mul_40[0][0]
__________________________________________________________________________________________________
batch_normalization_41 (BatchNo (None, 26, 26, 256)  1024        conv2d_41[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_41 (Tensor [(None, 26, 26, 256) 0           batch_normalization_41[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_41 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_41[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_41 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_41[0][0]
                                                                 tf_op_layer_Tanh_41[0][0]
__________________________________________________________________________________________________
conv2d_42 (Conv2D)              (None, 26, 26, 256)  589824      tf_op_layer_Mul_41[0][0]
__________________________________________________________________________________________________
batch_normalization_42 (BatchNo (None, 26, 26, 256)  1024        conv2d_42[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_42 (Tensor [(None, 26, 26, 256) 0           batch_normalization_42[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_42 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_42[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_42 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_42[0][0]
                                                                 tf_op_layer_Tanh_42[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_11 (TensorFlo [(None, 26, 26, 256) 0           tf_op_layer_Mul_40[0][0]
                                                                 tf_op_layer_Mul_42[0][0]
__________________________________________________________________________________________________
conv2d_43 (Conv2D)              (None, 26, 26, 256)  65536       tf_op_layer_AddV2_11[0][0]
__________________________________________________________________________________________________
batch_normalization_43 (BatchNo (None, 26, 26, 256)  1024        conv2d_43[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_43 (Tensor [(None, 26, 26, 256) 0           batch_normalization_43[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_43 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_43[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_43 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_43[0][0]
                                                                 tf_op_layer_Tanh_43[0][0]
__________________________________________________________________________________________________
conv2d_44 (Conv2D)              (None, 26, 26, 256)  589824      tf_op_layer_Mul_43[0][0]
__________________________________________________________________________________________________
batch_normalization_44 (BatchNo (None, 26, 26, 256)  1024        conv2d_44[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_44 (Tensor [(None, 26, 26, 256) 0           batch_normalization_44[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_44 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_44[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_44 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_44[0][0]
                                                                 tf_op_layer_Tanh_44[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_12 (TensorFlo [(None, 26, 26, 256) 0           tf_op_layer_AddV2_11[0][0]
                                                                 tf_op_layer_Mul_44[0][0]
__________________________________________________________________________________________________
conv2d_45 (Conv2D)              (None, 26, 26, 256)  65536       tf_op_layer_AddV2_12[0][0]
__________________________________________________________________________________________________
batch_normalization_45 (BatchNo (None, 26, 26, 256)  1024        conv2d_45[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_45 (Tensor [(None, 26, 26, 256) 0           batch_normalization_45[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_45 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_45[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_45 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_45[0][0]
                                                                 tf_op_layer_Tanh_45[0][0]
__________________________________________________________________________________________________
conv2d_46 (Conv2D)              (None, 26, 26, 256)  589824      tf_op_layer_Mul_45[0][0]
__________________________________________________________________________________________________
batch_normalization_46 (BatchNo (None, 26, 26, 256)  1024        conv2d_46[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_46 (Tensor [(None, 26, 26, 256) 0           batch_normalization_46[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_46 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_46[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_46 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_46[0][0]
                                                                 tf_op_layer_Tanh_46[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_13 (TensorFlo [(None, 26, 26, 256) 0           tf_op_layer_AddV2_12[0][0]
                                                                 tf_op_layer_Mul_46[0][0]
__________________________________________________________________________________________________
conv2d_47 (Conv2D)              (None, 26, 26, 256)  65536       tf_op_layer_AddV2_13[0][0]
__________________________________________________________________________________________________
batch_normalization_47 (BatchNo (None, 26, 26, 256)  1024        conv2d_47[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_47 (Tensor [(None, 26, 26, 256) 0           batch_normalization_47[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_47 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_47[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_47 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_47[0][0]
                                                                 tf_op_layer_Tanh_47[0][0]
__________________________________________________________________________________________________
conv2d_48 (Conv2D)              (None, 26, 26, 256)  589824      tf_op_layer_Mul_47[0][0]
__________________________________________________________________________________________________
batch_normalization_48 (BatchNo (None, 26, 26, 256)  1024        conv2d_48[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_48 (Tensor [(None, 26, 26, 256) 0           batch_normalization_48[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_48 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_48[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_48 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_48[0][0]
```

APPENDIX C: (Continued)

```
                                                                  tf_op_layer_Tanh_48[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_14 (TensorFlo [(None, 26, 26, 256) 0           tf_op_layer_AddV2_13[0][0]
                                                                  tf_op_layer_Mul_48[0][0]
__________________________________________________________________________________________________
conv2d_49 (Conv2D)              (None, 26, 26, 256)  65536       tf_op_layer_AddV2_14[0][0]
__________________________________________________________________________________________________
batch_normalization_49 (BatchNo (None, 26, 26, 256)  1024        conv2d_49[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_49 (Tensor [(None, 26, 26, 256) 0           batch_normalization_49[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_49 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_49[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_49 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_49[0][0]
                                                                  tf_op_layer_Tanh_49[0][0]
__________________________________________________________________________________________________
conv2d_50 (Conv2D)              (None, 26, 26, 256)  589824      tf_op_layer_Mul_49[0][0]
__________________________________________________________________________________________________
batch_normalization_50 (BatchNo (None, 26, 26, 256)  1024        conv2d_50[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_50 (Tensor [(None, 26, 26, 256) 0           batch_normalization_50[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_50 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_50[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_50 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_50[0][0]
                                                                  tf_op_layer_Tanh_50[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_15 (TensorFlo [(None, 26, 26, 256) 0           tf_op_layer_AddV2_14[0][0]
                                                                  tf_op_layer_Mul_50[0][0]
__________________________________________________________________________________________________
conv2d_51 (Conv2D)              (None, 26, 26, 256)  65536       tf_op_layer_AddV2_15[0][0]
__________________________________________________________________________________________________
batch_normalization_51 (BatchNo (None, 26, 26, 256)  1024        conv2d_51[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_51 (Tensor [(None, 26, 26, 256) 0           batch_normalization_51[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_51 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_51[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_51 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_51[0][0]
                                                                  tf_op_layer_Tanh_51[0][0]
__________________________________________________________________________________________________
conv2d_52 (Conv2D)              (None, 26, 26, 256)  589824      tf_op_layer_Mul_51[0][0]
__________________________________________________________________________________________________
batch_normalization_52 (BatchNo (None, 26, 26, 256)  1024        conv2d_52[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_52 (Tensor [(None, 26, 26, 256) 0           batch_normalization_52[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_52 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_52[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_52 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_52[0][0]
                                                                  tf_op_layer_Tanh_52[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_16 (TensorFlo [(None, 26, 26, 256) 0           tf_op_layer_AddV2_15[0][0]
                                                                  tf_op_layer_Mul_52[0][0]
__________________________________________________________________________________________________
conv2d_53 (Conv2D)              (None, 26, 26, 256)  65536       tf_op_layer_AddV2_16[0][0]
__________________________________________________________________________________________________
batch_normalization_53 (BatchNo (None, 26, 26, 256)  1024        conv2d_53[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_53 (Tensor [(None, 26, 26, 256) 0           batch_normalization_53[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_53 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_53[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_53 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_53[0][0]
                                                                  tf_op_layer_Tanh_53[0][0]
__________________________________________________________________________________________________
conv2d_54 (Conv2D)              (None, 26, 26, 256)  589824      tf_op_layer_Mul_53[0][0]
__________________________________________________________________________________________________
batch_normalization_54 (BatchNo (None, 26, 26, 256)  1024        conv2d_54[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_54 (Tensor [(None, 26, 26, 256) 0           batch_normalization_54[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_54 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_54[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_54 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_54[0][0]
                                                                  tf_op_layer_Tanh_54[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_17 (TensorFlo [(None, 26, 26, 256) 0           tf_op_layer_AddV2_16[0][0]
                                                                  tf_op_layer_Mul_54[0][0]
__________________________________________________________________________________________________
conv2d_55 (Conv2D)              (None, 26, 26, 256)  65536       tf_op_layer_AddV2_17[0][0]
__________________________________________________________________________________________________
batch_normalization_55 (BatchNo (None, 26, 26, 256)  1024        conv2d_55[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_55 (Tensor [(None, 26, 26, 256) 0           batch_normalization_55[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_55 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_55[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_55 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_55[0][0]
                                                                  tf_op_layer_Tanh_55[0][0]
__________________________________________________________________________________________________
conv2d_56 (Conv2D)              (None, 26, 26, 256)  589824      tf_op_layer_Mul_55[0][0]
__________________________________________________________________________________________________
batch_normalization_56 (BatchNo (None, 26, 26, 256)  1024        conv2d_56[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_56 (Tensor [(None, 26, 26, 256) 0           batch_normalization_56[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_56 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_56[0][0]
```

APPENDIX C: (Continued)

```
__________________________________________________________________________________________________
tf_op_layer_Mul_56 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_56[0][0]
                                                                 tf_op_layer_Tanh_56[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_18 (TensorFlo [(None, 26, 26, 256) 0           tf_op_layer_AddV2_17[0][0]
                                                                 tf_op_layer_Mul_56[0][0]
__________________________________________________________________________________________________
conv2d_57 (Conv2D)              (None, 26, 26, 256)  65536       tf_op_layer_AddV2_18[0][0]
__________________________________________________________________________________________________
conv2d_39 (Conv2D)              (None, 26, 26, 256)  131072      tf_op_layer_Mul_38[0][0]
__________________________________________________________________________________________________
batch_normalization_57 (BatchNo (None, 26, 26, 256)  1024        conv2d_57[0][0]
__________________________________________________________________________________________________
batch_normalization_39 (BatchNo (None, 26, 26, 256)  1024        conv2d_39[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_57 (Tensor [(None, 26, 26, 256) 0           batch_normalization_57[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_39 (Tensor [(None, 26, 26, 256) 0           batch_normalization_39[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_57 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_57[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_39 (TensorFlow [(None, 26, 26, 256) 0           tf_op_layer_Softplus_39[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_57 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_57[0][0]
                                                                 tf_op_layer_Tanh_57[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_39 (TensorFlowO [(None, 26, 26, 256) 0           batch_normalization_39[0][0]
                                                                 tf_op_layer_Tanh_39[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_3 (TensorFlo [(None, 26, 26, 512) 0           tf_op_layer_Mul_57[0][0]
                                                                 tf_op_layer_Mul_39[0][0]
__________________________________________________________________________________________________
conv2d_58 (Conv2D)              (None, 26, 26, 512)  262144      tf_op_layer_concat_3[0][0]
__________________________________________________________________________________________________
batch_normalization_58 (BatchNo (None, 26, 26, 512)  2048        conv2d_58[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_58 (Tensor [(None, 26, 26, 512) 0           batch_normalization_58[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_58 (TensorFlow [(None, 26, 26, 512) 0           tf_op_layer_Softplus_58[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_58 (TensorFlowO [(None, 26, 26, 512) 0           batch_normalization_58[0][0]
                                                                 tf_op_layer_Tanh_58[0][0]
__________________________________________________________________________________________________
zero_padding2d_4 (ZeroPadding2D (None, 27, 27, 512)  0           tf_op_layer_Mul_58[0][0]
__________________________________________________________________________________________________
conv2d_59 (Conv2D)              (None, 13, 13, 1024) 4718592     zero_padding2d_4[0][0]
__________________________________________________________________________________________________
batch_normalization_59 (BatchNo (None, 13, 13, 1024) 4096        conv2d_59[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_59 (Tensor [(None, 13, 13, 1024 0           batch_normalization_59[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_59 (TensorFlow [(None, 13, 13, 1024 0           tf_op_layer_Softplus_59[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_59 (TensorFlowO [(None, 13, 13, 1024 0           batch_normalization_59[0][0]
                                                                 tf_op_layer_Tanh_59[0][0]
__________________________________________________________________________________________________
conv2d_61 (Conv2D)              (None, 13, 13, 512)  524288      tf_op_layer_Mul_59[0][0]
__________________________________________________________________________________________________
batch_normalization_61 (BatchNo (None, 13, 13, 512)  2048        conv2d_61[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_61 (Tensor [(None, 13, 13, 512) 0           batch_normalization_61[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_61 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_61[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_61 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_61[0][0]
                                                                 tf_op_layer_Tanh_61[0][0]
__________________________________________________________________________________________________
conv2d_62 (Conv2D)              (None, 13, 13, 512)  262144      tf_op_layer_Mul_61[0][0]
__________________________________________________________________________________________________
batch_normalization_62 (BatchNo (None, 13, 13, 512)  2048        conv2d_62[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_62 (Tensor [(None, 13, 13, 512) 0           batch_normalization_62[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_62 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_62[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_62 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_62[0][0]
                                                                 tf_op_layer_Tanh_62[0][0]
__________________________________________________________________________________________________
conv2d_63 (Conv2D)              (None, 13, 13, 512)  2359296     tf_op_layer_Mul_62[0][0]
__________________________________________________________________________________________________
batch_normalization_63 (BatchNo (None, 13, 13, 512)  2048        conv2d_63[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_63 (Tensor [(None, 13, 13, 512) 0           batch_normalization_63[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_63 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_63[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_63 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_63[0][0]
                                                                 tf_op_layer_Tanh_63[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_19 (TensorFlo [(None, 13, 13, 512) 0           tf_op_layer_Mul_61[0][0]
                                                                 tf_op_layer_Mul_63[0][0]
__________________________________________________________________________________________________
conv2d_64 (Conv2D)              (None, 13, 13, 512)  262144      tf_op_layer_AddV2_19[0][0]
__________________________________________________________________________________________________
batch_normalization_64 (BatchNo (None, 13, 13, 512)  2048        conv2d_64[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_64 (Tensor [(None, 13, 13, 512) 0           batch_normalization_64[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_64 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_64[0][0]
```

APPENDIX C: (Continued)

```
__________________________________________________________________________________________________
tf_op_layer_Mul_64 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_64[0][0]
                                                                 tf_op_layer_Tanh_64[0][0]
__________________________________________________________________________________________________
conv2d_65 (Conv2D)              (None, 13, 13, 512)  2359296     tf_op_layer_Mul_64[0][0]
__________________________________________________________________________________________________
batch_normalization_65 (BatchNo (None, 13, 13, 512)  2048        conv2d_65[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_65 (Tensor [(None, 13, 13, 512) 0           batch_normalization_65[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_65 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_65[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_65 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_65[0][0]
                                                                 tf_op_layer_Tanh_65[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_20 (TensorFlo [(None, 13, 13, 512) 0           tf_op_layer_AddV2_19[0][0]
                                                                 tf_op_layer_Mul_65[0][0]
__________________________________________________________________________________________________
conv2d_66 (Conv2D)              (None, 13, 13, 512)  262144      tf_op_layer_AddV2_20[0][0]
__________________________________________________________________________________________________
batch_normalization_66 (BatchNo (None, 13, 13, 512)  2048        conv2d_66[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_66 (Tensor [(None, 13, 13, 512) 0           batch_normalization_66[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_66 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_66[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_66 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_66[0][0]
                                                                 tf_op_layer_Tanh_66[0][0]
__________________________________________________________________________________________________
conv2d_67 (Conv2D)              (None, 13, 13, 512)  2359296     tf_op_layer_Mul_66[0][0]
__________________________________________________________________________________________________
batch_normalization_67 (BatchNo (None, 13, 13, 512)  2048        conv2d_67[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_67 (Tensor [(None, 13, 13, 512) 0           batch_normalization_67[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_67 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_67[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_67 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_67[0][0]
                                                                 tf_op_layer_Tanh_67[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_21 (TensorFlo [(None, 13, 13, 512) 0           tf_op_layer_AddV2_20[0][0]
                                                                 tf_op_layer_Mul_67[0][0]
__________________________________________________________________________________________________
conv2d_68 (Conv2D)              (None, 13, 13, 512)  262144      tf_op_layer_AddV2_21[0][0]
__________________________________________________________________________________________________
batch_normalization_68 (BatchNo (None, 13, 13, 512)  2048        conv2d_68[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_68 (Tensor [(None, 13, 13, 512) 0           batch_normalization_68[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_68 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_68[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_68 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_68[0][0]
                                                                 tf_op_layer_Tanh_68[0][0]
__________________________________________________________________________________________________
conv2d_69 (Conv2D)              (None, 13, 13, 512)  2359296     tf_op_layer_Mul_68[0][0]
__________________________________________________________________________________________________
batch_normalization_69 (BatchNo (None, 13, 13, 512)  2048        conv2d_69[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_69 (Tensor [(None, 13, 13, 512) 0           batch_normalization_69[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_69 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_69[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_69 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_69[0][0]
                                                                 tf_op_layer_Tanh_69[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_22 (TensorFlo [(None, 13, 13, 512) 0           tf_op_layer_AddV2_21[0][0]
                                                                 tf_op_layer_Mul_69[0][0]
__________________________________________________________________________________________________
conv2d_70 (Conv2D)              (None, 13, 13, 512)  262144      tf_op_layer_AddV2_22[0][0]
__________________________________________________________________________________________________
conv2d_60 (Conv2D)              (None, 13, 13, 512)  524288      tf_op_layer_Mul_59[0][0]
__________________________________________________________________________________________________
batch_normalization_70 (BatchNo (None, 13, 13, 512)  2048        conv2d_70[0][0]
__________________________________________________________________________________________________
batch_normalization_60 (BatchNo (None, 13, 13, 512)  2048        conv2d_60[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_70 (Tensor [(None, 13, 13, 512) 0           batch_normalization_70[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_60 (Tensor [(None, 13, 13, 512) 0           batch_normalization_60[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_70 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_70[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tanh_60 (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_Softplus_60[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_70 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_70[0][0]
                                                                 tf_op_layer_Tanh_70[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_60 (TensorFlowO [(None, 13, 13, 512) 0           batch_normalization_60[0][0]
                                                                 tf_op_layer_Tanh_60[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_4 (TensorFlo [(None, 13, 13, 1024 0           tf_op_layer_Mul_70[0][0]
                                                                 tf_op_layer_Mul_60[0][0]
__________________________________________________________________________________________________
conv2d_71 (Conv2D)              (None, 13, 13, 1024) 1048576     tf_op_layer_concat_4[0][0]
__________________________________________________________________________________________________
batch_normalization_71 (BatchNo (None, 13, 13, 1024) 4096        conv2d_71[0][0]
__________________________________________________________________________________________________
tf_op_layer_Softplus_71 (Tensor [(None, 13, 13, 1024 0           batch_normalization_71[0][0]
__________________________________________________________________________________________________
```

APPENDIX C: (Continued)

```
tf_op_layer_Tanh_71 (TensorFlow [(None, 13, 13, 1024 0           tf_op_layer_Softplus_71[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_71 (TensorFlowO [(None, 13, 13, 1024 0           batch_normalization_71[0][0]
                                                                 tf_op_layer_Tanh_71[0][0]
__________________________________________________________________________________________________
conv2d_72 (Conv2D)              (None, 13, 13, 512)  524288      tf_op_layer_Mul_71[0][0]
__________________________________________________________________________________________________
batch_normalization_72 (BatchNo (None, 13, 13, 512)  2048        conv2d_72[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu (TensorFl [(None, 13, 13, 512) 0           batch_normalization_72[0][0]
__________________________________________________________________________________________________
conv2d_73 (Conv2D)              (None, 13, 13, 1024) 4718592     tf_op_layer_LeakyRelu[0][0]
__________________________________________________________________________________________________
batch_normalization_73 (BatchNo (None, 13, 13, 1024) 4096        conv2d_73[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_1 (Tensor [(None, 13, 13, 1024 0           batch_normalization_73[0][0]
__________________________________________________________________________________________________
conv2d_74 (Conv2D)              (None, 13, 13, 512)  524288      tf_op_layer_LeakyRelu_1[0][0]
__________________________________________________________________________________________________
batch_normalization_74 (BatchNo (None, 13, 13, 512)  2048        conv2d_74[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_2 (Tensor [(None, 13, 13, 512) 0           batch_normalization_74[0][0]
__________________________________________________________________________________________________
tf_op_layer_MaxPool (TensorFlow [(None, 13, 13, 512) 0           tf_op_layer_LeakyRelu_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_MaxPool_1 (TensorFl [(None, 13, 13, 512) 0           tf_op_layer_LeakyRelu_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_MaxPool_2 (TensorFl [(None, 13, 13, 512) 0           tf_op_layer_LeakyRelu_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_5 (TensorFlo [(None, 13, 13, 2048 0           tf_op_layer_MaxPool[0][0]
                                                                 tf_op_layer_MaxPool_1[0][0]
                                                                 tf_op_layer_MaxPool_2[0][0]
                                                                 tf_op_layer_LeakyRelu_2[0][0]
__________________________________________________________________________________________________
conv2d_75 (Conv2D)              (None, 13, 13, 512)  1048576     tf_op_layer_concat_5[0][0]
__________________________________________________________________________________________________
batch_normalization_75 (BatchNo (None, 13, 13, 512)  2048        conv2d_75[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_3 (Tensor [(None, 13, 13, 512) 0           batch_normalization_75[0][0]
__________________________________________________________________________________________________
conv2d_76 (Conv2D)              (None, 13, 13, 1024) 4718592     tf_op_layer_LeakyRelu_3[0][0]
__________________________________________________________________________________________________
batch_normalization_76 (BatchNo (None, 13, 13, 1024) 4096        conv2d_76[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_4 (Tensor [(None, 13, 13, 1024 0           batch_normalization_76[0][0]
__________________________________________________________________________________________________
conv2d_77 (Conv2D)              (None, 13, 13, 512)  524288      tf_op_layer_LeakyRelu_4[0][0]
__________________________________________________________________________________________________
batch_normalization_77 (BatchNo (None, 13, 13, 512)  2048        conv2d_77[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_5 (Tensor [(None, 13, 13, 512) 0           batch_normalization_77[0][0]
__________________________________________________________________________________________________
conv2d_78 (Conv2D)              (None, 13, 13, 256)  131072      tf_op_layer_LeakyRelu_5[0][0]
__________________________________________________________________________________________________
conv2d_79 (Conv2D)              (None, 26, 26, 256)  131072      tf_op_layer_Mul_58[0][0]
__________________________________________________________________________________________________
batch_normalization_78 (BatchNo (None, 13, 13, 256)  1024        conv2d_78[0][0]
__________________________________________________________________________________________________
batch_normalization_79 (BatchNo (None, 26, 26, 256)  1024        conv2d_79[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_6 (Tensor [(None, 13, 13, 256) 0           batch_normalization_78[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_7 (Tensor [(None, 26, 26, 256) 0           batch_normalization_79[0][0]
__________________________________________________________________________________________________
tf_op_layer_ResizeBilinear (Ten [(None, 26, 26, 256) 0           tf_op_layer_LeakyRelu_6[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_6 (TensorFlo [(None, 26, 26, 512) 0           tf_op_layer_LeakyRelu_7[0][0]
                                                                 tf_op_layer_ResizeBilinear[0][0]
__________________________________________________________________________________________________
conv2d_80 (Conv2D)              (None, 26, 26, 256)  131072      tf_op_layer_concat_6[0][0]
__________________________________________________________________________________________________
batch_normalization_80 (BatchNo (None, 26, 26, 256)  1024        conv2d_80[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_8 (Tensor [(None, 26, 26, 256) 0           batch_normalization_80[0][0]
__________________________________________________________________________________________________
conv2d_81 (Conv2D)              (None, 26, 26, 512)  1179648     tf_op_layer_LeakyRelu_8[0][0]
__________________________________________________________________________________________________
batch_normalization_81 (BatchNo (None, 26, 26, 512)  2048        conv2d_81[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_9 (Tensor [(None, 26, 26, 512) 0           batch_normalization_81[0][0]
__________________________________________________________________________________________________
conv2d_82 (Conv2D)              (None, 26, 26, 256)  131072      tf_op_layer_LeakyRelu_9[0][0]
__________________________________________________________________________________________________
batch_normalization_82 (BatchNo (None, 26, 26, 256)  1024        conv2d_82[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_10 (Tenso [(None, 26, 26, 256) 0           batch_normalization_82[0][0]
__________________________________________________________________________________________________
conv2d_83 (Conv2D)              (None, 26, 26, 512)  1179648     tf_op_layer_LeakyRelu_10[0][0]
__________________________________________________________________________________________________
batch_normalization_83 (BatchNo (None, 26, 26, 512)  2048        conv2d_83[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_11 (Tenso [(None, 26, 26, 512) 0           batch_normalization_83[0][0]
__________________________________________________________________________________________________
conv2d_84 (Conv2D)              (None, 26, 26, 256)  131072      tf_op_layer_LeakyRelu_11[0][0]
__________________________________________________________________________________________________
batch_normalization_84 (BatchNo (None, 26, 26, 256)  1024        conv2d_84[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_12 (Tenso [(None, 26, 26, 256) 0           batch_normalization_84[0][0]
__________________________________________________________________________________________________
```

APPENDIX C: (Continued)

```
conv2d_85 (Conv2D)              (None, 26, 26, 128)  32768       tf_op_layer_LeakyRelu_12[0][0]
__________________________________________________________________________________________________
conv2d_86 (Conv2D)              (None, 52, 52, 128)  32768       tf_op_layer_Mul_37[0][0]
__________________________________________________________________________________________________
batch_normalization_85 (BatchNo (None, 26, 26, 128)  512         conv2d_85[0][0]
__________________________________________________________________________________________________
batch_normalization_86 (BatchNo (None, 52, 52, 128)  512         conv2d_86[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_13 (Tenso [(None, 26, 26, 128) 0           batch_normalization_85[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_14 (Tenso [(None, 52, 52, 128) 0           batch_normalization_86[0][0]
__________________________________________________________________________________________________
tf_op_layer_ResizeBilinear_1 (T [(None, 52, 52, 128) 0           tf_op_layer_LeakyRelu_13[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_7 (TensorFlo [(None, 52, 52, 256) 0           tf_op_layer_LeakyRelu_14[0][0]
                                                                 tf_op_layer_ResizeBilinear_1[0][0
__________________________________________________________________________________________________
conv2d_87 (Conv2D)              (None, 52, 52, 128)  32768       tf_op_layer_concat_7[0][0]
__________________________________________________________________________________________________
batch_normalization_87 (BatchNo (None, 52, 52, 128)  512         conv2d_87[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_15 (Tenso [(None, 52, 52, 128) 0           batch_normalization_87[0][0]
__________________________________________________________________________________________________
conv2d_88 (Conv2D)              (None, 52, 52, 256)  294912      tf_op_layer_LeakyRelu_15[0][0]
__________________________________________________________________________________________________
batch_normalization_88 (BatchNo (None, 52, 52, 256)  1024        conv2d_88[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_16 (Tenso [(None, 52, 52, 256) 0           batch_normalization_88[0][0]
__________________________________________________________________________________________________
conv2d_89 (Conv2D)              (None, 52, 52, 128)  32768       tf_op_layer_LeakyRelu_16[0][0]
__________________________________________________________________________________________________
batch_normalization_89 (BatchNo (None, 52, 52, 128)  512         conv2d_89[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_17 (Tenso [(None, 52, 52, 128) 0           batch_normalization_89[0][0]
__________________________________________________________________________________________________
conv2d_90 (Conv2D)              (None, 52, 52, 256)  294912      tf_op_layer_LeakyRelu_17[0][0]
__________________________________________________________________________________________________
batch_normalization_90 (BatchNo (None, 52, 52, 256)  1024        conv2d_90[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_18 (Tenso [(None, 52, 52, 256) 0           batch_normalization_90[0][0]
__________________________________________________________________________________________________
conv2d_91 (Conv2D)              (None, 52, 52, 128)  32768       tf_op_layer_LeakyRelu_18[0][0]
__________________________________________________________________________________________________
batch_normalization_91 (BatchNo (None, 52, 52, 128)  512         conv2d_91[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_19 (Tenso [(None, 52, 52, 128) 0           batch_normalization_91[0][0]
__________________________________________________________________________________________________
zero_padding2d_5 (ZeroPadding2D (None, 53, 53, 128)  0           tf_op_layer_LeakyRelu_19[0][0]
__________________________________________________________________________________________________
conv2d_94 (Conv2D)              (None, 26, 26, 256)  294912      zero_padding2d_5[0][0]
__________________________________________________________________________________________________
batch_normalization_93 (BatchNo (None, 26, 26, 256)  1024        conv2d_94[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_21 (Tenso [(None, 26, 26, 256) 0           batch_normalization_93[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_8 (TensorFlo [(None, 26, 26, 512) 0           tf_op_layer_LeakyRelu_21[0][0]
                                                                 tf_op_layer_LeakyRelu_12[0][0]
__________________________________________________________________________________________________
conv2d_95 (Conv2D)              (None, 26, 26, 256)  131072      tf_op_layer_concat_8[0][0]
__________________________________________________________________________________________________
batch_normalization_94 (BatchNo (None, 26, 26, 256)  1024        conv2d_95[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_22 (Tenso [(None, 26, 26, 256) 0           batch_normalization_94[0][0]
__________________________________________________________________________________________________
conv2d_96 (Conv2D)              (None, 26, 26, 512)  1179648     tf_op_layer_LeakyRelu_22[0][0]
__________________________________________________________________________________________________
batch_normalization_95 (BatchNo (None, 26, 26, 512)  2048        conv2d_96[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_23 (Tenso [(None, 26, 26, 512) 0           batch_normalization_95[0][0]
__________________________________________________________________________________________________
conv2d_97 (Conv2D)              (None, 26, 26, 256)  131072      tf_op_layer_LeakyRelu_23[0][0]
__________________________________________________________________________________________________
batch_normalization_96 (BatchNo (None, 26, 26, 256)  1024        conv2d_97[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_24 (Tenso [(None, 26, 26, 256) 0           batch_normalization_96[0][0]
__________________________________________________________________________________________________
conv2d_98 (Conv2D)              (None, 26, 26, 512)  1179648     tf_op_layer_LeakyRelu_24[0][0]
__________________________________________________________________________________________________
batch_normalization_97 (BatchNo (None, 26, 26, 512)  2048        conv2d_98[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_25 (Tenso [(None, 26, 26, 512) 0           batch_normalization_97[0][0]
__________________________________________________________________________________________________
conv2d_99 (Conv2D)              (None, 26, 26, 256)  131072      tf_op_layer_LeakyRelu_25[0][0]
__________________________________________________________________________________________________
batch_normalization_98 (BatchNo (None, 26, 26, 256)  1024        conv2d_99[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_26 (Tenso [(None, 26, 26, 256) 0           batch_normalization_98[0][0]
__________________________________________________________________________________________________
zero_padding2d_6 (ZeroPadding2D (None, 27, 27, 256)  0           tf_op_layer_LeakyRelu_26[0][0]
__________________________________________________________________________________________________
conv2d_102 (Conv2D)             (None, 13, 13, 512)  1179648     zero_padding2d_6[0][0]
__________________________________________________________________________________________________
batch_normalization_100 (BatchN (None, 13, 13, 512)  2048        conv2d_102[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_28 (Tenso [(None, 13, 13, 512) 0           batch_normalization_100[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_9 (TensorFlo [(None, 13, 13, 1024 0           tf_op_layer_LeakyRelu_28[0][0]
                                                                 tf_op_layer_LeakyRelu_5[0][0]
__________________________________________________________________________________________________
```

APPENDIX C: (Continued)

```
conv2d_103 (Conv2D)             (None, 13, 13, 512)  524288      tf_op_layer_concat_9[0][0]
__________________________________________________________________________________________________
batch_normalization_101 (BatchN (None, 13, 13, 512)  2048        conv2d_103[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_29 (Tenso [(None, 13, 13, 512) 0           batch_normalization_101[0][0]
__________________________________________________________________________________________________
conv2d_104 (Conv2D)             (None, 13, 13, 1024) 4718592     tf_op_layer_LeakyRelu_29[0][0]
__________________________________________________________________________________________________
batch_normalization_102 (BatchN (None, 13, 13, 1024) 4096        conv2d_104[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_30 (Tenso [(None, 13, 13, 1024 0           batch_normalization_102[0][0]
__________________________________________________________________________________________________
conv2d_105 (Conv2D)             (None, 13, 13, 512)  524288      tf_op_layer_LeakyRelu_30[0][0]
__________________________________________________________________________________________________
batch_normalization_103 (BatchN (None, 13, 13, 512)  2048        conv2d_105[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_31 (Tenso [(None, 13, 13, 512) 0           batch_normalization_103[0][0]
__________________________________________________________________________________________________
conv2d_106 (Conv2D)             (None, 13, 13, 1024) 4718592     tf_op_layer_LeakyRelu_31[0][0]
__________________________________________________________________________________________________
batch_normalization_104 (BatchN (None, 13, 13, 1024) 4096        conv2d_106[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_32 (Tenso [(None, 13, 13, 1024 0           batch_normalization_104[0][0]
__________________________________________________________________________________________________
conv2d_107 (Conv2D)             (None, 13, 13, 512)  524288      tf_op_layer_LeakyRelu_32[0][0]
__________________________________________________________________________________________________
batch_normalization_105 (BatchN (None, 13, 13, 512)  2048        conv2d_107[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_33 (Tenso [(None, 13, 13, 512) 0           batch_normalization_105[0][0]
__________________________________________________________________________________________________
conv2d_92 (Conv2D)              (None, 52, 52, 256)  294912      tf_op_layer_LeakyRelu_19[0][0]
__________________________________________________________________________________________________
conv2d_100 (Conv2D)             (None, 26, 26, 512)  1179648     tf_op_layer_LeakyRelu_26[0][0]
__________________________________________________________________________________________________
conv2d_108 (Conv2D)             (None, 13, 13, 1024) 4718592     tf_op_layer_LeakyRelu_33[0][0]
__________________________________________________________________________________________________
batch_normalization_92 (BatchNo (None, 52, 52, 256)  1024        conv2d_92[0][0]
__________________________________________________________________________________________________
batch_normalization_99 (BatchNo (None, 26, 26, 512)  2048        conv2d_100[0][0]
__________________________________________________________________________________________________
batch_normalization_106 (BatchN (None, 13, 13, 1024) 4096        conv2d_108[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_20 (Tenso [(None, 52, 52, 256) 0           batch_normalization_92[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_27 (Tenso [(None, 26, 26, 512) 0           batch_normalization_99[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_34 (Tenso [(None, 13, 13, 1024 0           batch_normalization_106[0][0]
__________________________________________________________________________________________________
conv2d_93 (Conv2D)              (None, 52, 52, 18)   4626        tf_op_layer_LeakyRelu_20[0][0]
__________________________________________________________________________________________________
conv2d_101 (Conv2D)             (None, 26, 26, 18)   9234        tf_op_layer_LeakyRelu_27[0][0]
__________________________________________________________________________________________________
conv2d_109 (Conv2D)             (None, 13, 13, 18)   18450       tf_op_layer_LeakyRelu_34[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape (TensorFlowOp [(4,)]               0           conv2d_93[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_1 (TensorFlow [(4,)]               0           conv2d_101[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_2 (TensorFlow [(4,)]               0           conv2d_109[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice (Tens [()]                 0           tf_op_layer_Shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_1 (Te [()]                 0           tf_op_layer_Shape_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_2 (Te [()]                 0           tf_op_layer_Shape_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape/shape (Tens [(5,)]               0           tf_op_layer_strided_slice[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_3/shape (Te [(5,)]               0           tf_op_layer_strided_slice_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_6/shape (Te [(5,)]               0           tf_op_layer_strided_slice_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape (TensorFlow [(None, 52, 52, 3, 6 0           conv2d_93[0][0]
                                                                 tf_op_layer_Reshape/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_3 (TensorFl [(None, 26, 26, 3, 6 0           conv2d_101[0][0]
                                                                 tf_op_layer_Reshape_3/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_6 (TensorFl [(None, 13, 13, 3, 6 0           conv2d_109[0][0]
                                                                 tf_op_layer_Reshape_6/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_split (TensorFlowOp [(None, 52, 52, 3, 2 0           tf_op_layer_Reshape[0][0]
__________________________________________________________________________________________________
tf_op_layer_split_1 (TensorFlow [(None, 26, 26, 3, 2 0           tf_op_layer_Reshape_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_split_2 (TensorFlow [(None, 13, 13, 3, 2 0           tf_op_layer_Reshape_6[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid (TensorFlow [(None, 52, 52, 3, 2 0           tf_op_layer_split[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tile/multiples (Ten [(5,)]               0           tf_op_layer_strided_slice[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_3 (TensorFl [(None, 26, 26, 3, 2 0           tf_op_layer_split_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tile_1/multiples (T [(5,)]               0           tf_op_layer_strided_slice_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_6 (TensorFl [(None, 13, 13, 3, 2 0           tf_op_layer_split_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tile_2/multiples (T [(5,)]               0           tf_op_layer_strided_slice_2[0][0]
__________________________________________________________________________________________________
```

APPENDIX C: (Continued)

```
tf_op_layer_Mul_72 (TensorFlowO [(None, 52, 52, 3, 2 0           tf_op_layer_Sigmoid[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tile (TensorFlowOpL [(None, 52, 52, 3, 2 0           tf_op_layer_Tile/multiples[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_76 (TensorFlowO [(None, 26, 26, 3, 2 0           tf_op_layer_Sigmoid_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tile_1 (TensorFlowO [(None, 26, 26, 3, 2 0           tf_op_layer_Tile_1/multiples[0][0
__________________________________________________________________________________________________
tf_op_layer_Mul_80 (TensorFlowO [(None, 13, 13, 3, 2 0           tf_op_layer_Sigmoid_6[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tile_2 (TensorFlowO [(None, 13, 13, 3, 2 0           tf_op_layer_Tile_2/multiples[0][0
__________________________________________________________________________________________________
tf_op_layer_Sub (TensorFlowOpLa [(None, 52, 52, 3, 2 0           tf_op_layer_Mul_72[0][0]
__________________________________________________________________________________________________
tf_op_layer_Cast (TensorFlowOpL [(None, 52, 52, 3, 2 0           tf_op_layer_Tile[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sub_1 (TensorFlowOp [(None, 26, 26, 3, 2 0           tf_op_layer_Mul_76[0][0]
__________________________________________________________________________________________________
tf_op_layer_Cast_1 (TensorFlowO [(None, 26, 26, 3, 2 0           tf_op_layer_Tile_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sub_2 (TensorFlowOp [(None, 13, 13, 3, 2 0           tf_op_layer_Mul_80[0][0]
__________________________________________________________________________________________________
tf_op_layer_Cast_2 (TensorFlowO [(None, 13, 13, 3, 2 0           tf_op_layer_Tile_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_23 (TensorFlo [(None, 52, 52, 3, 2 0           tf_op_layer_Sub[0][0]
                                                                 tf_op_layer_Cast[0][0]
__________________________________________________________________________________________________
tf_op_layer_Exp (TensorFlowOpLa [(None, 52, 52, 3, 2 0           tf_op_layer_split[0][1]
__________________________________________________________________________________________________
tf_op_layer_AddV2_24 (TensorFlo [(None, 26, 26, 3, 2 0           tf_op_layer_Sub_1[0][0]
                                                                 tf_op_layer_Cast_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Exp_1 (TensorFlowOp [(None, 26, 26, 3, 2 0           tf_op_layer_split_1[0][1]
__________________________________________________________________________________________________
tf_op_layer_AddV2_25 (TensorFlo [(None, 13, 13, 3, 2 0           tf_op_layer_Sub_2[0][0]
                                                                 tf_op_layer_Cast_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Exp_2 (TensorFlowOp [(None, 13, 13, 3, 2 0           tf_op_layer_split_2[0][1]
__________________________________________________________________________________________________
tf_op_layer_Mul_73 (TensorFlowO [(None, 52, 52, 3, 2 0           tf_op_layer_AddV2_23[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_74 (TensorFlowO [(None, 52, 52, 3, 2 0           tf_op_layer_Exp[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_77 (TensorFlowO [(None, 26, 26, 3, 2 0           tf_op_layer_AddV2_24[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_78 (TensorFlowO [(None, 26, 26, 3, 2 0           tf_op_layer_Exp_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_81 (TensorFlowO [(None, 13, 13, 3, 2 0           tf_op_layer_AddV2_25[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_82 (TensorFlowO [(None, 13, 13, 3, 2 0           tf_op_layer_Exp_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_10 (TensorFl [(None, 52, 52, 3, 4 0           tf_op_layer_Mul_73[0][0]
                                                                 tf_op_layer_Mul_74[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_2/shape (Te [(3,)]               0           tf_op_layer_strided_slice[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_11 (TensorFl [(None, 26, 26, 3, 4 0           tf_op_layer_Mul_77[0][0]
                                                                 tf_op_layer_Mul_78[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_5/shape (Te [(3,)]               0           tf_op_layer_strided_slice_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_12 (TensorFl [(None, 13, 13, 3, 4 0           tf_op_layer_Mul_81[0][0]
                                                                 tf_op_layer_Mul_82[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_8/shape (Te [(3,)]               0           tf_op_layer_strided_slice_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_1 (TensorFl [(None, 52, 52, 3, 1 0           tf_op_layer_split[0][2]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_2 (TensorFl [(None, 52, 52, 3, 1 0           tf_op_layer_split[0][3]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_4 (TensorFl [(None, 26, 26, 3, 1 0           tf_op_layer_split_1[0][2]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_5 (TensorFl [(None, 26, 26, 3, 1 0           tf_op_layer_split_1[0][3]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_7 (TensorFl [(None, 13, 13, 3, 1 0           tf_op_layer_split_2[0][2]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_8 (TensorFl [(None, 13, 13, 3, 1 0           tf_op_layer_split_2[0][3]
__________________________________________________________________________________________________
tf_op_layer_Reshape_2 (TensorFl [(None, None, 4)]    0           tf_op_layer_concat_10[0][0]
                                                                 tf_op_layer_Reshape_2/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_5 (TensorFl [(None, None, 4)]    0           tf_op_layer_concat_11[0][0]
                                                                 tf_op_layer_Reshape_5/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_8 (TensorFl [(None, None, 4)]    0           tf_op_layer_concat_12[0][0]
                                                                 tf_op_layer_Reshape_8/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_75 (TensorFlowO [(None, 52, 52, 3, 1 0           tf_op_layer_Sigmoid_1[0][0]
                                                                 tf_op_layer_Sigmoid_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_1/shape (Te [(3,)]               0           tf_op_layer_strided_slice[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_79 (TensorFlowO [(None, 26, 26, 3, 1 0           tf_op_layer_Sigmoid_4[0][0]
                                                                 tf_op_layer_Sigmoid_5[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_4/shape (Te [(3,)]               0           tf_op_layer_strided_slice_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_83 (TensorFlowO [(None, 13, 13, 3, 1 0           tf_op_layer_Sigmoid_7[0][0]
                                                                 tf_op_layer_Sigmoid_8[0][0]
```

APPENDIX C: (Continued)

```
__________________________________________________________________________________________________
tf_op_layer_Reshape_7/shape (Te [(3,)]               0           tf_op_layer_strided_slice_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_13 (TensorFl [(None, None, 4)]    0           tf_op_layer_Reshape_2[0][0]
                                                                 tf_op_layer_Reshape_5[0][0]
                                                                 tf_op_layer_Reshape_8[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_1 (TensorFl [(None, None, 1)]    0           tf_op_layer_Mul_75[0][0]
                                                                 tf_op_layer_Reshape_1/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_4 (TensorFl [(None, None, 1)]    0           tf_op_layer_Mul_79[0][0]
                                                                 tf_op_layer_Reshape_4/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_7 (TensorFl [(None, None, 1)]    0           tf_op_layer_Mul_83[0][0]
                                                                 tf_op_layer_Reshape_7/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_3 (TensorFlow [(3,)]               0           tf_op_layer_concat_13[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_14 (TensorFl [(None, None, 1)]    0           tf_op_layer_Reshape_1[0][0]
                                                                 tf_op_layer_Reshape_4[0][0]
                                                                 tf_op_layer_Reshape_7[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_3 (Te [(2,)]               0           tf_op_layer_Shape_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_Max (TensorFlowOpLa [(None, None)]       0           tf_op_layer_concat_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_4 (TensorFlow [(3,)]               0           tf_op_layer_concat_13[0][0]
__________________________________________________________________________________________________
tf_op_layer_Prod (TensorFlowOpL [()]                 0           tf_op_layer_strided_slice_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_5 (TensorFlow [(3,)]               0           tf_op_layer_concat_13[0][0]
__________________________________________________________________________________________________
tf_op_layer_GreaterEqual (Tenso [(None, None)]       0           tf_op_layer_Max[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_4 (Te [(0,)]               0           tf_op_layer_Shape_4[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_15/values_1  [(1,)]               0           tf_op_layer_Prod[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_5 (Te [(1,)]               0           tf_op_layer_Shape_5[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_10 (TensorF [(None,)]            0           tf_op_layer_GreaterEqual[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_15 (TensorFl [(2,)]               0           tf_op_layer_strided_slice_4[0][0]
                                                                 tf_op_layer_concat_15/values_1[0]
                                                                 tf_op_layer_strided_slice_5[0][0]
__________________________________________________________________________________________________
tf_op_layer_Where (TensorFlowOp [(None, 1)]          0           tf_op_layer_Reshape_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_9 (TensorFl [(None, 4)]          0           tf_op_layer_concat_13[0][0]
                                                                 tf_op_layer_concat_15[0][0]
__________________________________________________________________________________________________
tf_op_layer_Squeeze (TensorFlow [(None,)]            0           tf_op_layer_Where[0][0]
__________________________________________________________________________________________________
tf_op_layer_GatherV2 (TensorFlo [(None, 4)]          0           tf_op_layer_Reshape_9[0][0]
                                                                 tf_op_layer_Squeeze[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_9 (TensorFlow [(3,)]               0           tf_op_layer_concat_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_10 (TensorFlo [(2,)]               0           tf_op_layer_GatherV2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_6 (TensorFlow [(3,)]               0           tf_op_layer_concat_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_9 (Te [()]                 0           tf_op_layer_Shape_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_10 (T [()]                 0           tf_op_layer_Shape_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_6 (Te [(2,)]               0           tf_op_layer_Shape_6[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_13/shape (T [(3,)]               0           tf_op_layer_strided_slice_9[0][0]
                                                                 tf_op_layer_strided_slice_10[0][0
__________________________________________________________________________________________________
tf_op_layer_Shape_7 (TensorFlow [(3,)]               0           tf_op_layer_concat_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_Prod_1 (TensorFlowO [()]                 0           tf_op_layer_strided_slice_6[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_8 (TensorFlow [(3,)]               0           tf_op_layer_concat_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_13 (TensorF [(None, None, None)] 0           tf_op_layer_GatherV2[0][0]
                                                                 tf_op_layer_Reshape_13/shape[0][0
__________________________________________________________________________________________________
tf_op_layer_strided_slice_7 (Te [(0,)]               0           tf_op_layer_Shape_7[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_16/values_1  [(1,)]               0           tf_op_layer_Prod_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_8 (Te [(1,)]               0           tf_op_layer_Shape_8[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_12 (TensorF [(None,)]            0           tf_op_layer_GreaterEqual[0][0]
__________________________________________________________________________________________________
tf_op_layer_split_3 (TensorFlow [(None, None, 2), (N 0           tf_op_layer_Reshape_13[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_16 (TensorFl [(2,)]               0           tf_op_layer_strided_slice_7[0][0]
                                                                 tf_op_layer_concat_16/values_1[0]
                                                                 tf_op_layer_strided_slice_8[0][0]
__________________________________________________________________________________________________
tf_op_layer_Where_1 (TensorFlow [(None, 1)]          0           tf_op_layer_Reshape_12[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_14 (T [(None, None, 2)]    0           tf_op_layer_split_3[0][1]
__________________________________________________________________________________________________
tf_op_layer_Reshape_11 (TensorF [(None, 1)]          0           tf_op_layer_concat_14[0][0]
```

APPENDIX C: (Continued)

```
                                                                  tf_op_layer_concat_16[0][0]
__________________________________________________________________________________________________
tf_op_layer_Squeeze_1 (TensorFl [(None,)]            0           tf_op_layer_Where_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_13 (T [(None, None, 2)]    0           tf_op_layer_split_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_RealDiv (TensorFlow [(None, None, 2)]    0           tf_op_layer_strided_slice_14[0][0
__________________________________________________________________________________________________
tf_op_layer_RealDiv_2 (TensorFl [(None, None, 2)]    0           tf_op_layer_strided_slice_14[0][0
__________________________________________________________________________________________________
tf_op_layer_GatherV2_1 (TensorF [(None, 1)]          0           tf_op_layer_Reshape_11[0][0]
                                                                  tf_op_layer_Squeeze_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sub_3 (TensorFlowOp [(None, None, 2)]    0           tf_op_layer_strided_slice_13[0][0
                                                                  tf_op_layer_RealDiv[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_26 (TensorFlo [(None, None, 2)]    0           tf_op_layer_strided_slice_13[0][0
                                                                  tf_op_layer_RealDiv_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_11 (TensorFlo [(3,)]               0           tf_op_layer_concat_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_12 (TensorFlo [(2,)]               0           tf_op_layer_GatherV2_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_RealDiv_1 (TensorFl [(None, None, 2)]    0           tf_op_layer_Sub_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_RealDiv_3 (TensorFl [(None, None, 2)]    0           tf_op_layer_AddV2_26[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_11 (T [()]                 0           tf_op_layer_Shape_11[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_12 (T [()]                 0           tf_op_layer_Shape_12[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_15 (T [(None, None, 1)]    0           tf_op_layer_RealDiv_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_16 (T [(None, None, 1)]    0           tf_op_layer_RealDiv_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_17 (T [(None, None, 1)]    0           tf_op_layer_RealDiv_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_18 (T [(None, None, 1)]    0           tf_op_layer_RealDiv_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_14/shape (T [(3,)]               0           tf_op_layer_strided_slice_11[0][0
                                                                  tf_op_layer_strided_slice_12[0][0
__________________________________________________________________________________________________
tf_op_layer_concat_17 (TensorFl [(None, None, 4)]    0           tf_op_layer_strided_slice_15[0][0
                                                                  tf_op_layer_strided_slice_16[0][0
                                                                  tf_op_layer_strided_slice_17[0][0
                                                                  tf_op_layer_strided_slice_18[0][0
__________________________________________________________________________________________________
tf_op_layer_Reshape_14 (TensorF [(None, None, None)] 0           tf_op_layer_GatherV2_1[0][0]
                                                                  tf_op_layer_Reshape_14/shape[0][0
__________________________________________________________________________________________________
tf_op_layer_concat_18 (TensorFl [(None, None, None)] 0           tf_op_layer_concat_17[0][0]
                                                                  tf_op_layer_Reshape_14[0][0]
==================================================================================================
Total params: 64,003,990
Trainable params: 63,937,686
Non-trainable params: 66,304
```

APPENDIX D: Architecture of YOLOv4-Tiny.

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 416, 416, 3) 0
__________________________________________________________________________________________________
zero_padding2d (ZeroPadding2D)  (None, 417, 417, 3)  0           input_1[0][0]
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 208, 208, 32) 864         zero_padding2d[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 208, 208, 32) 128         conv2d[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu (TensorFl [(None, 208, 208, 32 0           batch_normalization[0][0]
__________________________________________________________________________________________________
zero_padding2d_1 (ZeroPadding2D (None, 209, 209, 32) 0           tf_op_layer_LeakyRelu[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 104, 104, 64) 18432       zero_padding2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 104, 104, 64) 256         conv2d_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_1 (Tensor [(None, 104, 104, 64 0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 104, 104, 64) 36864       tf_op_layer_LeakyRelu_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 104, 104, 64) 256         conv2d_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_2 (Tensor [(None, 104, 104, 64 0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_split (TensorFlowOp [(None, 104, 104, 32 0           tf_op_layer_LeakyRelu_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 104, 104, 32) 9216        tf_op_layer_split[0][1]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 104, 104, 32) 128         conv2d_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_3 (Tensor [(None, 104, 104, 32 0           batch_normalization_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 104, 104, 32) 9216        tf_op_layer_LeakyRelu_3[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 104, 104, 32) 128         conv2d_4[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_4 (Tensor [(None, 104, 104, 32 0           batch_normalization_4[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat (TensorFlowO [(None, 104, 104, 64 0           tf_op_layer_LeakyRelu_4[0][0]
                                                                 tf_op_layer_LeakyRelu_3[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 104, 104, 64) 4096        tf_op_layer_concat[0][0]
__________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 104, 104, 64) 256         conv2d_5[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_5 (Tensor [(None, 104, 104, 64 0           batch_normalization_5[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_1 (TensorFlo [(None, 104, 104, 12 0           tf_op_layer_LeakyRelu_2[0][0]
                                                                 tf_op_layer_LeakyRelu_5[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 52, 52, 128)  0           tf_op_layer_concat_1[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 52, 52, 128)  147456      max_pooling2d[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 52, 52, 128)  512         conv2d_6[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_6 (Tensor [(None, 52, 52, 128) 0           batch_normalization_6[0][0]
__________________________________________________________________________________________________
tf_op_layer_split_1 (TensorFlow [(None, 52, 52, 64), 0           tf_op_layer_LeakyRelu_6[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 52, 52, 64)   36864       tf_op_layer_split_1[0][1]
__________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 52, 52, 64)   256         conv2d_7[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_7 (Tensor [(None, 52, 52, 64)] 0           batch_normalization_7[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 52, 52, 64)   36864       tf_op_layer_LeakyRelu_7[0][0]
__________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, 52, 52, 64)   256         conv2d_8[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_8 (Tensor [(None, 52, 52, 64)] 0           batch_normalization_8[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_2 (TensorFlo [(None, 52, 52, 128) 0           tf_op_layer_LeakyRelu_8[0][0]
                                                                 tf_op_layer_LeakyRelu_7[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 52, 52, 128)  16384       tf_op_layer_concat_2[0][0]
__________________________________________________________________________________________________
batch_normalization_9 (BatchNor (None, 52, 52, 128)  512         conv2d_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_9 (Tensor [(None, 52, 52, 128) 0           batch_normalization_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_3 (TensorFlo [(None, 52, 52, 256) 0           tf_op_layer_LeakyRelu_6[0][0]
                                                                 tf_op_layer_LeakyRelu_9[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 26, 26, 256)  0           tf_op_layer_concat_3[0][0]
__________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 26, 26, 256)  589824      max_pooling2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_10 (BatchNo (None, 26, 26, 256)  1024        conv2d_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_10 (Tenso [(None, 26, 26, 256) 0           batch_normalization_10[0][0]
__________________________________________________________________________________________________
```

APPENDIX D: (Continued)

```
tf_op_layer_split_2 (TensorFlow [(None, 26, 26, 128) 0           tf_op_layer_LeakyRelu_10[0][0]
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 26, 26, 128)  147456      tf_op_layer_split_2[0][1]
__________________________________________________________________________________________________
batch_normalization_11 (BatchNo (None, 26, 26, 128)  512         conv2d_11[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_11 (Tenso [(None, 26, 26, 128) 0           batch_normalization_11[0][0]
__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 26, 26, 128)  147456      tf_op_layer_LeakyRelu_11[0][0]
__________________________________________________________________________________________________
batch_normalization_12 (BatchNo (None, 26, 26, 128)  512         conv2d_12[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_12 (Tenso [(None, 26, 26, 128) 0           batch_normalization_12[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_4 (TensorFlo [(None, 26, 26, 256) 0           tf_op_layer_LeakyRelu_12[0][0]
                                                                 tf_op_layer_LeakyRelu_11[0][0]
__________________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 26, 26, 256)  65536       tf_op_layer_concat_4[0][0]
__________________________________________________________________________________________________
batch_normalization_13 (BatchNo (None, 26, 26, 256)  1024        conv2d_13[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_13 (Tenso [(None, 26, 26, 256) 0           batch_normalization_13[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_5 (TensorFlo [(None, 26, 26, 512) 0           tf_op_layer_LeakyRelu_10[0][0]
                                                                 tf_op_layer_LeakyRelu_13[0][0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 13, 13, 512)  0           tf_op_layer_concat_5[0][0]
__________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 13, 13, 512)  2359296     max_pooling2d_2[0][0]
__________________________________________________________________________________________________
batch_normalization_14 (BatchNo (None, 13, 13, 512)  2048        conv2d_14[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_14 (Tenso [(None, 13, 13, 512) 0           batch_normalization_14[0][0]
__________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 13, 13, 256)  131072      tf_op_layer_LeakyRelu_14[0][0]
__________________________________________________________________________________________________
batch_normalization_15 (BatchNo (None, 13, 13, 256)  1024        conv2d_15[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_15 (Tenso [(None, 13, 13, 256) 0           batch_normalization_15[0][0]
__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 13, 13, 128)  32768       tf_op_layer_LeakyRelu_15[0][0]
__________________________________________________________________________________________________
batch_normalization_17 (BatchNo (None, 13, 13, 128)  512         conv2d_18[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_17 (Tenso [(None, 13, 13, 128) 0           batch_normalization_17[0][0]
__________________________________________________________________________________________________
tf_op_layer_ResizeBilinear (Ten [(None, 26, 26, 128) 0           tf_op_layer_LeakyRelu_17[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_6 (TensorFlo [(None, 26, 26, 384) 0           tf_op_layer_ResizeBilinear[0][0]
                                                                 tf_op_layer_LeakyRelu_13[0][0]
__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 26, 26, 256)  884736      tf_op_layer_concat_6[0][0]
__________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 13, 13, 512)  1179648     tf_op_layer_LeakyRelu_15[0][0]
__________________________________________________________________________________________________
batch_normalization_18 (BatchNo (None, 26, 26, 256)  1024        conv2d_19[0][0]
__________________________________________________________________________________________________
batch_normalization_16 (BatchNo (None, 13, 13, 512)  2048        conv2d_16[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_18 (Tenso [(None, 26, 26, 256) 0           batch_normalization_18[0][0]
__________________________________________________________________________________________________
tf_op_layer_LeakyRelu_16 (Tenso [(None, 13, 13, 512) 0           batch_normalization_16[0][0]
__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 26, 26, 18)   4626        tf_op_layer_LeakyRelu_18[0][0]
__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 13, 13, 18)   9234        tf_op_layer_LeakyRelu_16[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape (TensorFlowOp [(4,)]               0           conv2d_20[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_1 (TensorFlow [(4,)]               0           conv2d_17[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice (Tens [()]                 0           tf_op_layer_Shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_1 (Te [()]                 0           tf_op_layer_Shape_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape/shape (Tens [(5,)]               0           tf_op_layer_strided_slice[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_3/shape (Te [(5,)]               0           tf_op_layer_strided_slice_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape (TensorFlow [(None, 26, 26, 3, 6 0           conv2d_20[0][0]
                                                                 tf_op_layer_Reshape/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_3 (TensorFl [(None, 13, 13, 3, 6 0           conv2d_17[0][0]
                                                                 tf_op_layer_Reshape_3/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_split_3 (TensorFlow [(None, 26, 26, 3, 2 0           tf_op_layer_Reshape[0][0]
__________________________________________________________________________________________________
tf_op_layer_split_4 (TensorFlow [(None, 13, 13, 3, 2 0           tf_op_layer_Reshape_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid (TensorFlow [(None, 26, 26, 3, 2 0           tf_op_layer_split_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tile/multiples (Ten [(5,)]               0           tf_op_layer_strided_slice[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_3 (TensorFl [(None, 13, 13, 3, 2 0           tf_op_layer_split_4[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tile_1/multiples (T [(5,)]               0           tf_op_layer_strided_slice_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul (TensorFlowOpLa [(None, 26, 26, 3, 2 0           tf_op_layer_Sigmoid[0][0]
__________________________________________________________________________________________________
```

APPENDIX D: (Continued)

```
tf_op_layer_Tile (TensorFlowOpL [(None, 26, 26, 3, 2 0           tf_op_layer_Tile/multiples[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_4 (TensorFlowOp [(None, 13, 13, 3, 2 0           tf_op_layer_Sigmoid_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_Tile_1 (TensorFlowO [(None, 13, 13, 3, 2 0           tf_op_layer_Tile_1/multiples[0][0
__________________________________________________________________________________________________
tf_op_layer_Sub (TensorFlowOpLa [(None, 26, 26, 3, 2 0           tf_op_layer_Mul[0][0]
__________________________________________________________________________________________________
tf_op_layer_Cast (TensorFlowOpL [(None, 26, 26, 3, 2 0           tf_op_layer_Tile[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sub_1 (TensorFlowOp [(None, 13, 13, 3, 2 0           tf_op_layer_Mul_4[0][0]
__________________________________________________________________________________________________
tf_op_layer_Cast_1 (TensorFlowO [(None, 13, 13, 3, 2 0           tf_op_layer_Tile_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2 (TensorFlowOp [(None, 26, 26, 3, 2 0           tf_op_layer_Sub[0][0]
                                                                 tf_op_layer_Cast[0][0]
__________________________________________________________________________________________________
tf_op_layer_Exp (TensorFlowOpLa [(None, 26, 26, 3, 2 0           tf_op_layer_split_3[0][1]
__________________________________________________________________________________________________
tf_op_layer_AddV2_1 (TensorFlow [(None, 13, 13, 3, 2 0           tf_op_layer_Sub_1[0][0]
                                                                 tf_op_layer_Cast_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Exp_1 (TensorFlowOp [(None, 13, 13, 3, 2 0           tf_op_layer_split_4[0][1]
__________________________________________________________________________________________________
tf_op_layer_Mul_1 (TensorFlowOp [(None, 26, 26, 3, 2 0           tf_op_layer_AddV2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_2 (TensorFlowOp [(None, 26, 26, 3, 2 0           tf_op_layer_Exp[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_5 (TensorFlowOp [(None, 13, 13, 3, 2 0           tf_op_layer_AddV2_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_6 (TensorFlowOp [(None, 13, 13, 3, 2 0           tf_op_layer_Exp_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_7 (TensorFlo [(None, 26, 26, 3, 4 0           tf_op_layer_Mul_1[0][0]
                                                                 tf_op_layer_Mul_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_2/shape (Te [(3,)]               0           tf_op_layer_strided_slice[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_8 (TensorFlo [(None, 13, 13, 3, 4 0           tf_op_layer_Mul_5[0][0]
                                                                 tf_op_layer_Mul_6[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_5/shape (Te [(3,)]               0           tf_op_layer_strided_slice_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_1 (TensorFl [(None, 26, 26, 3, 1 0           tf_op_layer_split_3[0][2]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_2 (TensorFl [(None, 26, 26, 3, 1 0           tf_op_layer_split_3[0][3]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_4 (TensorFl [(None, 13, 13, 3, 1 0           tf_op_layer_split_4[0][2]
__________________________________________________________________________________________________
tf_op_layer_Sigmoid_5 (TensorFl [(None, 13, 13, 3, 1 0           tf_op_layer_split_4[0][3]
__________________________________________________________________________________________________
tf_op_layer_Reshape_2 (TensorFl [(None, None, 4)]    0           tf_op_layer_concat_7[0][0]
                                                                 tf_op_layer_Reshape_2/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_5 (TensorFl [(None, None, 4)]    0           tf_op_layer_concat_8[0][0]
                                                                 tf_op_layer_Reshape_5/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_3 (TensorFlowOp [(None, 26, 26, 3, 1 0           tf_op_layer_Sigmoid_1[0][0]
                                                                 tf_op_layer_Sigmoid_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_1/shape (Te [(3,)]               0           tf_op_layer_strided_slice[0][0]
__________________________________________________________________________________________________
tf_op_layer_Mul_7 (TensorFlowOp [(None, 13, 13, 3, 1 0           tf_op_layer_Sigmoid_4[0][0]
                                                                 tf_op_layer_Sigmoid_5[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_4/shape (Te [(3,)]               0           tf_op_layer_strided_slice_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_9 (TensorFlo [(None, None, 4)]    0           tf_op_layer_Reshape_2[0][0]
                                                                 tf_op_layer_Reshape_5[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_1 (TensorFl [(None, None, 1)]    0           tf_op_layer_Mul_3[0][0]
                                                                 tf_op_layer_Reshape_1/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_4 (TensorFl [(None, None, 1)]    0           tf_op_layer_Mul_7[0][0]
                                                                 tf_op_layer_Reshape_4/shape[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_2 (TensorFlow [(3,)]               0           tf_op_layer_concat_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_10 (TensorFl [(None, None, 1)]    0           tf_op_layer_Reshape_1[0][0]
                                                                 tf_op_layer_Reshape_4[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_2 (Te [(2,)]               0           tf_op_layer_Shape_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Max (TensorFlowOpLa [(None, None)]       0           tf_op_layer_concat_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_3 (TensorFlow [(3,)]               0           tf_op_layer_concat_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_Prod (TensorFlowOpL [()]                 0           tf_op_layer_strided_slice_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_4 (TensorFlow [(3,)]               0           tf_op_layer_concat_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_GreaterEqual (Tenso [(None, None)]       0           tf_op_layer_Max[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_3 (Te [(0,)]               0           tf_op_layer_Shape_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_11/values_1  [(1,)]               0           tf_op_layer_Prod[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_4 (Te [(1,)]               0           tf_op_layer_Shape_4[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_7 (TensorFl [(None,)]            0           tf_op_layer_GreaterEqual[0][0]
```

APPENDIX D: (Continued)

```
__________________________________________________________________________________________________
tf_op_layer_concat_11 (TensorFl [(2,)]               0           tf_op_layer_strided_slice_3[0][0]
                                                                 tf_op_layer_concat_11/values_1[0]
                                                                 tf_op_layer_strided_slice_4[0][0]
__________________________________________________________________________________________________
tf_op_layer_Where (TensorFlowOp [(None, 1)]          0           tf_op_layer_Reshape_7[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_6 (TensorFl [(None, 4)]          0           tf_op_layer_concat_9[0][0]
                                                                 tf_op_layer_concat_11[0][0]
__________________________________________________________________________________________________
tf_op_layer_Squeeze (TensorFlow [(None,)]            0           tf_op_layer_Where[0][0]
__________________________________________________________________________________________________
tf_op_layer_GatherV2 (TensorFlo [(None, 4)]          0           tf_op_layer_Reshape_6[0][0]
                                                                 tf_op_layer_Squeeze[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_8 (TensorFlow [(3,)]               0           tf_op_layer_concat_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_9 (TensorFlow [(2,)]               0           tf_op_layer_GatherV2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_5 (TensorFlow [(3,)]               0           tf_op_layer_concat_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_8 (Te [()]                 0           tf_op_layer_Shape_8[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_9 (Te [()]                 0           tf_op_layer_Shape_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_5 (Te [(2,)]               0           tf_op_layer_Shape_5[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_10/shape (T [(3,)]               0           tf_op_layer_strided_slice_8[0][0]
                                                                 tf_op_layer_strided_slice_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_6 (TensorFlow [(3,)]               0           tf_op_layer_concat_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_Prod_1 (TensorFlowO [()]                 0           tf_op_layer_strided_slice_5[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_7 (TensorFlow [(3,)]               0           tf_op_layer_concat_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_10 (TensorF [(None, None, None)] 0           tf_op_layer_GatherV2[0][0]
                                                                 tf_op_layer_Reshape_10/shape[0][0
__________________________________________________________________________________________________
tf_op_layer_strided_slice_6 (Te [(0,)]               0           tf_op_layer_Shape_6[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_12/values_1  [(1,)]               0           tf_op_layer_Prod_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_7 (Te [(1,)]               0           tf_op_layer_Shape_7[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_9 (TensorFl [(None,)]            0           tf_op_layer_GreaterEqual[0][0]
__________________________________________________________________________________________________
tf_op_layer_split_5 (TensorFlow [(None, None, 2), (N 0           tf_op_layer_Reshape_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_12 (TensorFl [(2,)]               0           tf_op_layer_strided_slice_6[0][0]
                                                                 tf_op_layer_concat_12/values_1[0]
                                                                 tf_op_layer_strided_slice_7[0][0]
__________________________________________________________________________________________________
tf_op_layer_Where_1 (TensorFlow [(None, 1)]          0           tf_op_layer_Reshape_9[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_13 (T [(None, None, 2)]    0           tf_op_layer_split_5[0][1]
__________________________________________________________________________________________________
tf_op_layer_Reshape_8 (TensorFl [(None, 1)]          0           tf_op_layer_concat_10[0][0]
                                                                 tf_op_layer_concat_12[0][0]
__________________________________________________________________________________________________
tf_op_layer_Squeeze_1 (TensorFl [(None,)]            0           tf_op_layer_Where_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_12 (T [(None, None, 2)]    0           tf_op_layer_split_5[0][0]
__________________________________________________________________________________________________
tf_op_layer_RealDiv (TensorFlow [(None, None, 2)]    0           tf_op_layer_strided_slice_13[0][0
__________________________________________________________________________________________________
tf_op_layer_RealDiv_2 (TensorFl [(None, None, 2)]    0           tf_op_layer_strided_slice_13[0][0
__________________________________________________________________________________________________
tf_op_layer_GatherV2_1 (TensorF [(None, 1)]          0           tf_op_layer_Reshape_8[0][0]
                                                                 tf_op_layer_Squeeze_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Sub_2 (TensorFlowOp [(None, None, 2)]    0           tf_op_layer_strided_slice_12[0][0
                                                                 tf_op_layer_RealDiv[0][0]
__________________________________________________________________________________________________
tf_op_layer_AddV2_2 (TensorFlow [(None, None, 2)]    0           tf_op_layer_strided_slice_12[0][0
                                                                 tf_op_layer_RealDiv_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_10 (TensorFlo [(3,)]               0           tf_op_layer_concat_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_Shape_11 (TensorFlo [(2,)]               0           tf_op_layer_GatherV2_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_RealDiv_1 (TensorFl [(None, None, 2)]    0           tf_op_layer_Sub_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_RealDiv_3 (TensorFl [(None, None, 2)]    0           tf_op_layer_AddV2_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_10 (T [()]                 0           tf_op_layer_Shape_10[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_11 (T [()]                 0           tf_op_layer_Shape_11[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_14 (T [(None, None, 1)]    0           tf_op_layer_RealDiv_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_15 (T [(None, None, 1)]    0           tf_op_layer_RealDiv_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_16 (T [(None, None, 1)]    0           tf_op_layer_RealDiv_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_17 (T [(None, None, 1)]    0           tf_op_layer_RealDiv_3[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_11/shape (T [(3,)]               0           tf_op_layer_strided_slice_10[0][0
                                                                 tf_op_layer_strided_slice_11[0][0
```

APPENDIX D: (Continued)

```
_________________________________________________________________________________________________
tf_op_layer_concat_13 (TensorFl [(None, None, 4)]    0           tf_op_layer_strided_slice_14[0][0
                                                                 tf_op_layer_strided_slice_15[0][0
                                                                 tf_op_layer_strided_slice_16[0][0
                                                                 tf_op_layer_strided_slice_17[0][0
_________________________________________________________________________________________________
tf_op_layer_Reshape_11 (TensorF [(None, None, None)] 0           tf_op_layer_GatherV2_1[0][0]
                                                                 tf_op_layer_Reshape_11/shape[0][0
_________________________________________________________________________________________________
tf_op_layer_concat_14 (TensorFl [(None, None, None)] 0           tf_op_layer_concat_13[0][0]
                                                                 tf_op_layer_Reshape_11[0][0]
=================================================================================================
Total params: 5,880,324
Trainable params: 5,874,116
Non-trainable params: 6,208
```

APPENDIX E: Training Configuration for SSD MobilenetV2 320.

```
model {
  ssd {
    num_classes: 1
    image_resizer {
      fixed_shape_resizer {
        height: 320
        width: 320
      }
    }
    feature_extractor {
      type: "ssd_mobilenet_v2_fpn_keras"
      depth_multiplier: 1.0
      min_depth: 16
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 3.9999998989515007e-05
          }
        }
        initializer {
          random_normal_initializer {
            mean: 0.0
            stddev: 0.009999999776482582
          }
        }
        activation: RELU_6
        batch_norm {
          decay: 0.996999979019165
          scale: true
          epsilon: 0.0010000000474974513
        }
      }
      use_depthwise: true
      override_base_feature_extractor_hyperparams: true
      fpn {
        min_level: 3
        max_level: 7
        additional_layer_depth: 128
      }
    }
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
```

APPENDIX E: (Continued)

```
    unmatched_threshold: 0.5
      ignore_thresholds: false
      negatives_lower_than_unmatched: true
      force_match_for_each_row: true
      use_matmul_gather: true
    }
  }
  similarity_calculator {
    iou_similarity {
    }
  }
  box_predictor {
    weight_shared_convolutional_box_predictor {
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 3.9999998989515007e-05
          }
        }
        initializer {
          random_normal_initializer {
            mean: 0.0
            stddev: 0.009999999776482582
          }
        }
        activation: RELU_6
        batch_norm {
          decay: 0.996999979019165
          scale: true
          epsilon: 0.0010000000474974513
        }
      }
      depth: 128
      num_layers_before_predictor: 4
      kernel_size: 3
      class_prediction_bias_init: -4.599999904632568
      share_prediction_tower: true
      use_depthwise: true
    }
  }
  anchor_generator {
    multiscale_anchor_generator {
      min_level: 3
      max_level: 7
      anchor_scale: 4.0
      aspect_ratios: 1.0
      aspect_ratios: 2.0
      aspect_ratios: 0.5
      scales_per_octave: 2
    }
  }
  post_processing {
```

APPENDIX E: (Continued)

```
    batch_non_max_suppression {
      score_threshold: 9.99999993922529e-09
      iou_threshold: 0.6000000238418579
      max_detections_per_class: 100
      max_total_detections: 100
      use_static_shapes: false
     }
     score_converter: SIGMOID
    }
    normalize_loss_by_num_matches: true
    loss {
     localization_loss {
      weighted_smooth_l1 {
      }
     }
     classification_loss {
      weighted_sigmoid_focal {
       gamma: 2.0
       alpha: 0.25
      }
     }
     classification_weight: 1.0
     localization_weight: 1.0
    }
    encode_background_as_zeros: true
    normalize_loc_loss_by_codesize: true
    inplace_batchnorm_update: true
    freeze_batchnorm: false
  }
}
train_config {
  batch_size: 2
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_crop_image {
      min_object_covered: 0.0
      min_aspect_ratio: 0.75
      max_aspect_ratio: 3.0
      min_area: 0.75
      max_area: 1.0
      overlap_thresh: 0.0
    }
  }
  sync_replicas: true
  optimizer {
    momentum_optimizer {
      learning_rate {
        cosine_decay_learning_rate {
          learning_rate_base: 0.07999999821186066
```

APPENDIX E: (Continued)

```
        total_steps: 50000
         warmup_learning_rate: 0.026666000485420227
         warmup_steps: 1000
       }
     }
     momentum_optimizer_value: 0.8999999761581421
   }
   use_moving_average: false
  }
  fine_tune_checkpoint:                                        "pre-trained-
models/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0"
  num_steps: 50000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "detection"
  fine_tune_checkpoint_version: V2
  use_bfloat16: false
}
train_input_reader {
  label_map_path: "annotations/label_map.pbtxt"
  tf_record_input_reader {
    input_path: "annotations/version2/mobile_screw_dataset/train.record"
  }
}
eval_config {
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path: "annotations/label_map.pbtxt"
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path: "annotations/version2/mobile_screw_dataset/test.record"
  }
}
```

APPENDIX F: Training Configuration for SSD MobilenetV2 640.

```
model {
 ssd {
  num_classes: 1
  image_resizer {
   fixed_shape_resizer {
    height: 640
    width: 640
   }
  }
  feature_extractor {
   type: "ssd_mobilenet_v2_fpn_keras"
   depth_multiplier: 1.0
   min_depth: 16
   conv_hyperparams {
    regularizer {
     l2_regularizer {
      weight: 3.9999998989515007e-05
     }
    }
    initializer {
     random_normal_initializer {
      mean: 0.0
      stddev: 0.009999999776482582
     }
    }
    activation: RELU_6
    batch_norm {
     decay: 0.996999979019165
     scale: true
     epsilon: 0.0010000000474974513
    }
   }
   use_depthwise: true
   override_base_feature_extractor_hyperparams: true
   fpn {
    min_level: 3
    max_level: 7
    additional_layer_depth: 128
   }
  }
  box_coder {
   faster_rcnn_box_coder {
    y_scale: 10.0
    x_scale: 10.0
    height_scale: 5.0
    width_scale: 5.0
   }
  }
  matcher {
   argmax_matcher {
    matched_threshold: 0.5
```

APPENDIX F: (Continued)

```
   unmatched_threshold: 0.5
    ignore_thresholds: false
    negatives_lower_than_unmatched: true
    force_match_for_each_row: true
    use_matmul_gather: true
   }
  }
  similarity_calculator {
   iou_similarity {
   }
  }
  box_predictor {
   weight_shared_convolutional_box_predictor {
    conv_hyperparams {
     regularizer {
      l2_regularizer {
       weight: 3.9999998989515007e-05
      }
     }
     initializer {
      random_normal_initializer {
       mean: 0.0
       stddev: 0.009999999776482582
      }
     }
     activation: RELU_6
     batch_norm {
      decay: 0.996999979019165
      scale: true
      epsilon: 0.0010000000474974513
     }
    }
    depth: 128
    num_layers_before_predictor: 4
    kernel_size: 3
    class_prediction_bias_init: -4.599999904632568
    share_prediction_tower: true
    use_depthwise: true
   }
  }
  anchor_generator {
   multiscale_anchor_generator {
    min_level: 3
    max_level: 7
    anchor_scale: 4.0
    aspect_ratios: 1.0
    aspect_ratios: 2.0
    aspect_ratios: 0.5
    scales_per_octave: 2
   }
  }
  post_processing {
```

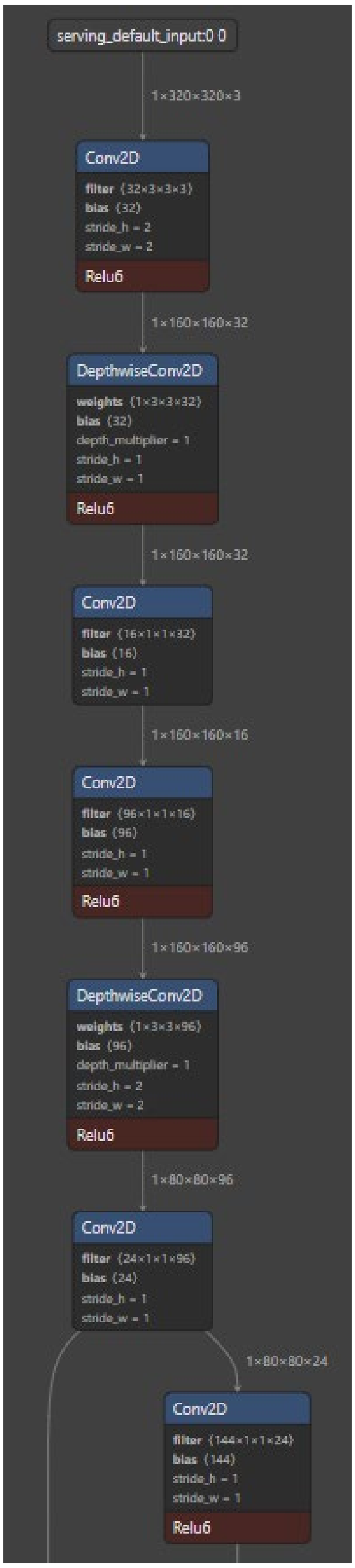
APPENDIX F: (Continued)

```
   batch_non_max_suppression {
        score_threshold: 9.99999993922529e-09
        iou_threshold: 0.6000000238418579
        max_detections_per_class: 100
        max_total_detections: 100
        use_static_shapes: false
      }
      score_converter: SIGMOID
    }
    normalize_loss_by_num_matches: true
    loss {
      localization_loss {
        weighted_smooth_l1 {
        }
      }
      classification_loss {
        weighted_sigmoid_focal {
          gamma: 2.0
          alpha: 0.25
        }
      }
      classification_weight: 1.0
      localization_weight: 1.0
    }
    encode_background_as_zeros: true
    normalize_loc_loss_by_codesize: true
    inplace_batchnorm_update: true
    freeze_batchnorm: false
  }
}
train_config {
  batch_size: 4
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_crop_image {
      min_object_covered: 0.0
      min_aspect_ratio: 0.75
      max_aspect_ratio: 3.0
      min_area: 0.75
      max_area: 1.0
      overlap_thresh: 0.0
    }
  }
  sync_replicas: true
  optimizer {
    momentum_optimizer {
      learning_rate {
        cosine_decay_learning_rate {
          learning_rate_base: 0.07999999821186066
```

APPENDIX F: (Continued)

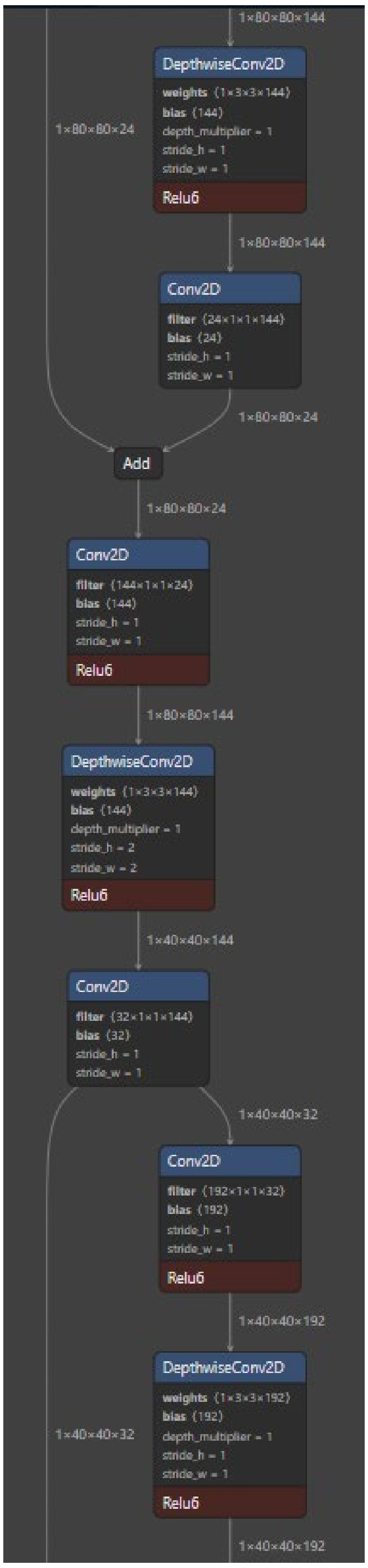
```
        total_steps: 50000
         warmup_learning_rate: 0.026666000485420227
         warmup_steps: 1000
        }
      }
      momentum_optimizer_value: 0.8999999761581421
    }
    use_moving_average: false
  }
  fine_tune_checkpoint:                                          "pre-trained-
models/ssd_mobilenet_v2_fpnlite_640x640_coco17_tpu-8/checkpoint/ckpt-0"
  num_steps: 50000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "detection"
  fine_tune_checkpoint_version: V2
}
train_input_reader {
  label_map_path: "annotations/label_map.pbtxt"
  tf_record_input_reader {
    input_path: "annotations/version2/mobile_screw_dataset/train.record"
  }
}
eval_config {
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path: "annotations/label_map.pbtxt"
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path: "annotations/version2/mobile_screw_dataset/test.record"
  }
}
```

APPENDIX G: Architecture of SSD MobilenetV2 320.

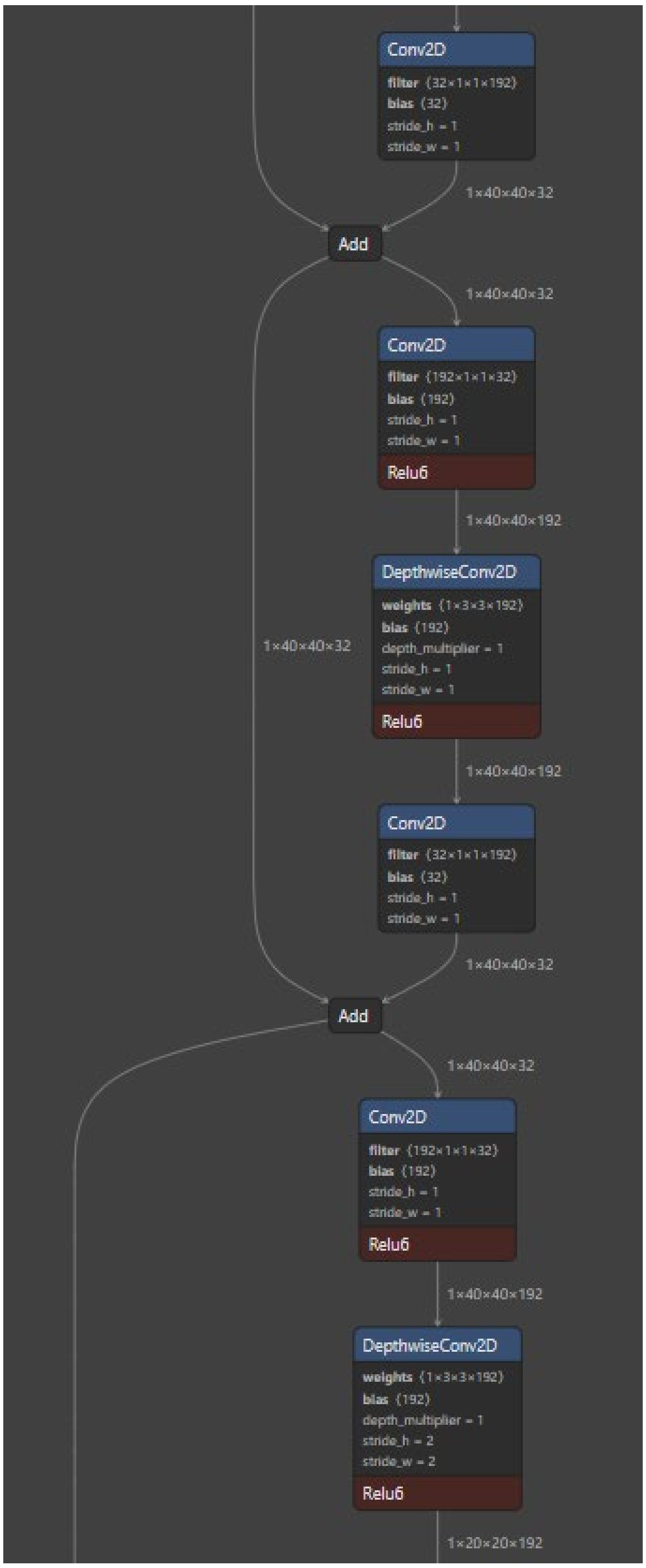


]

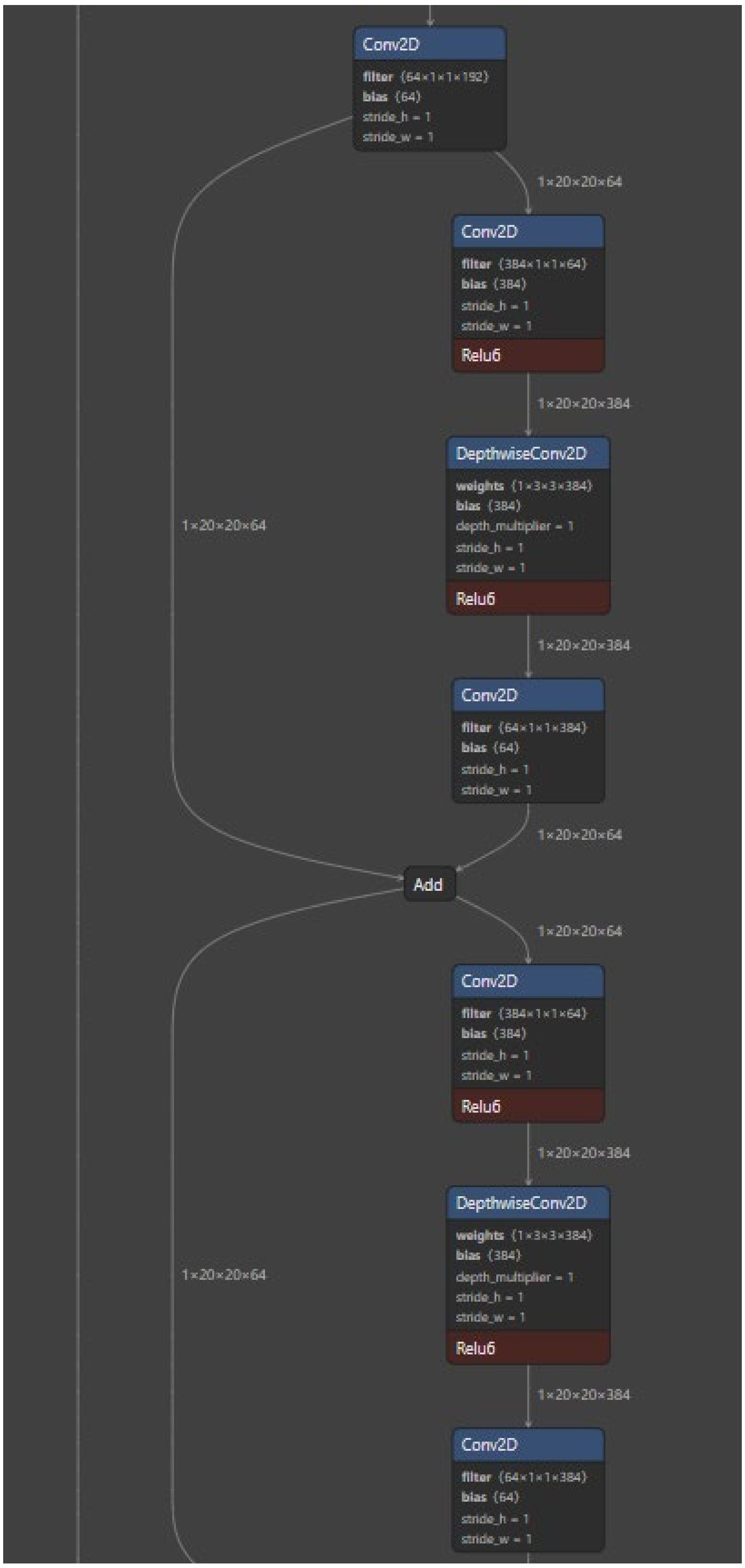
APPENDIX G: (Continued)



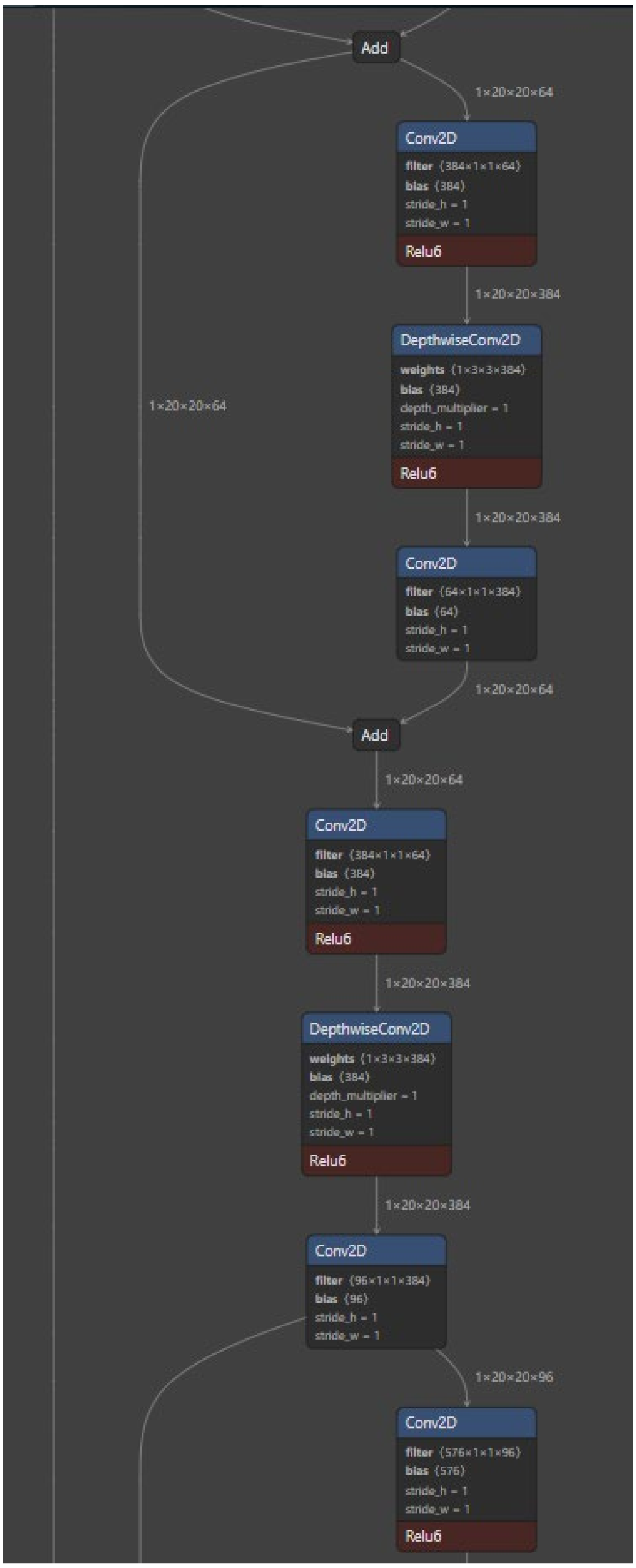
APPENDIX G: (Continued)



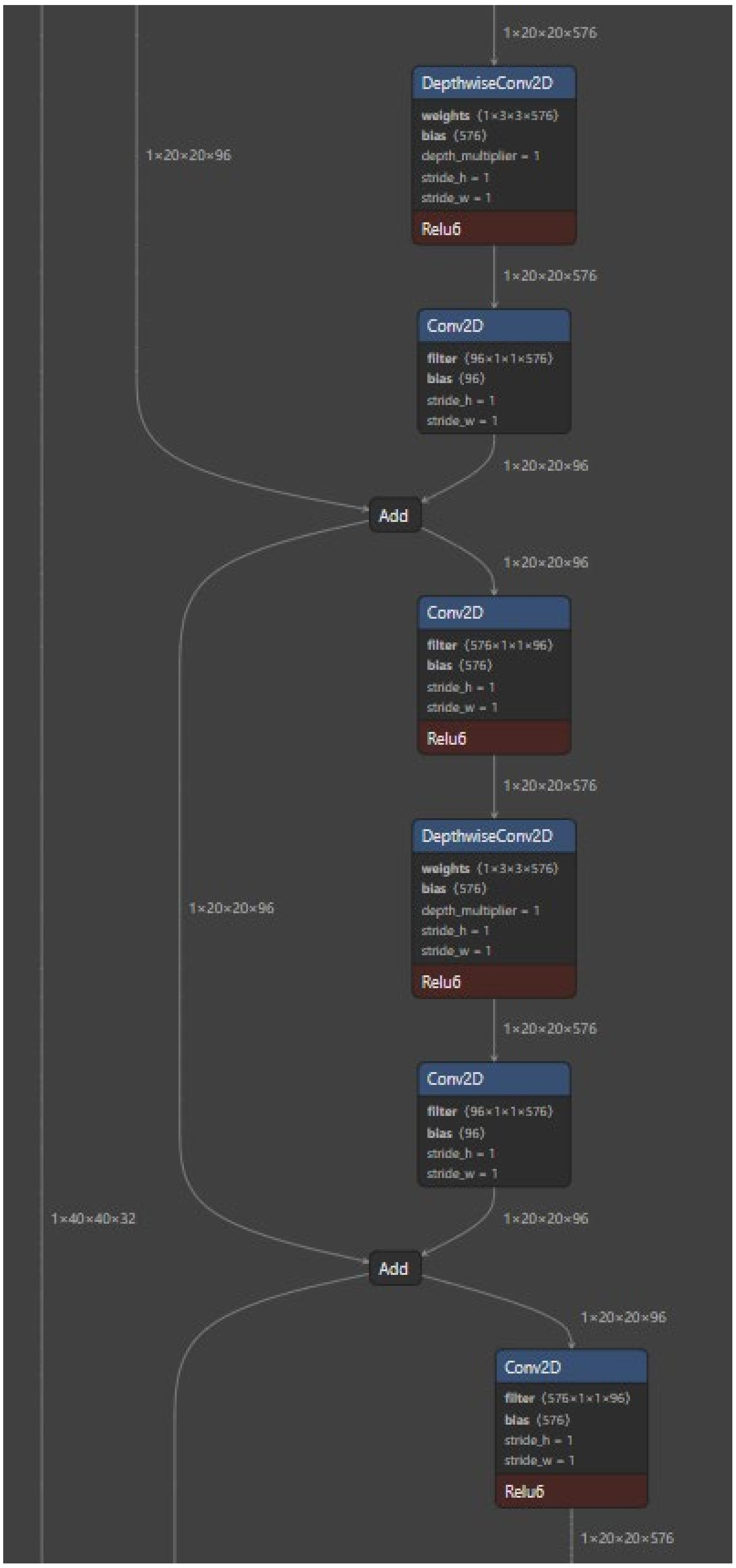
APPENDIX G: (Continued)



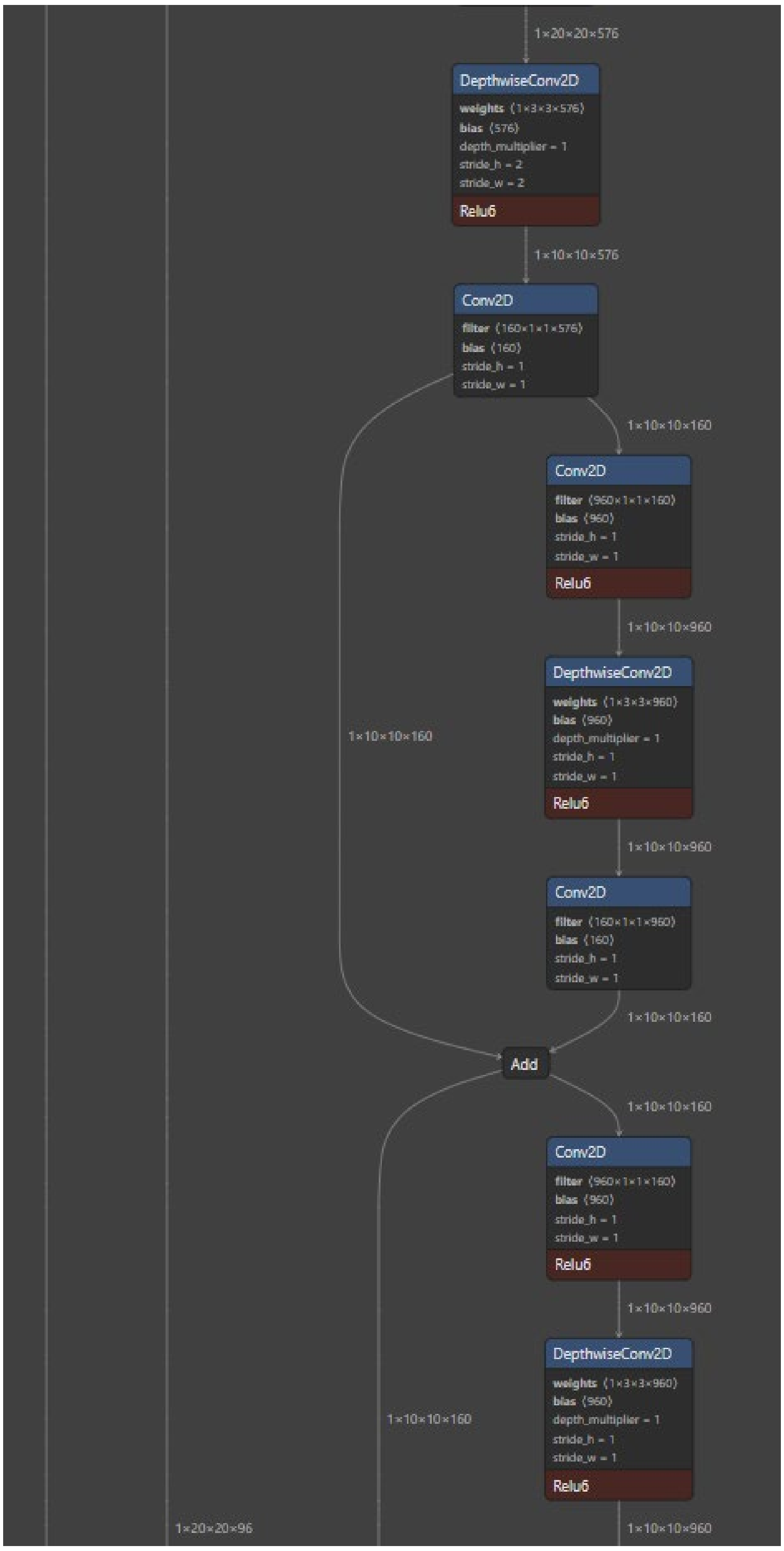
APPENDIX G: (Continued)



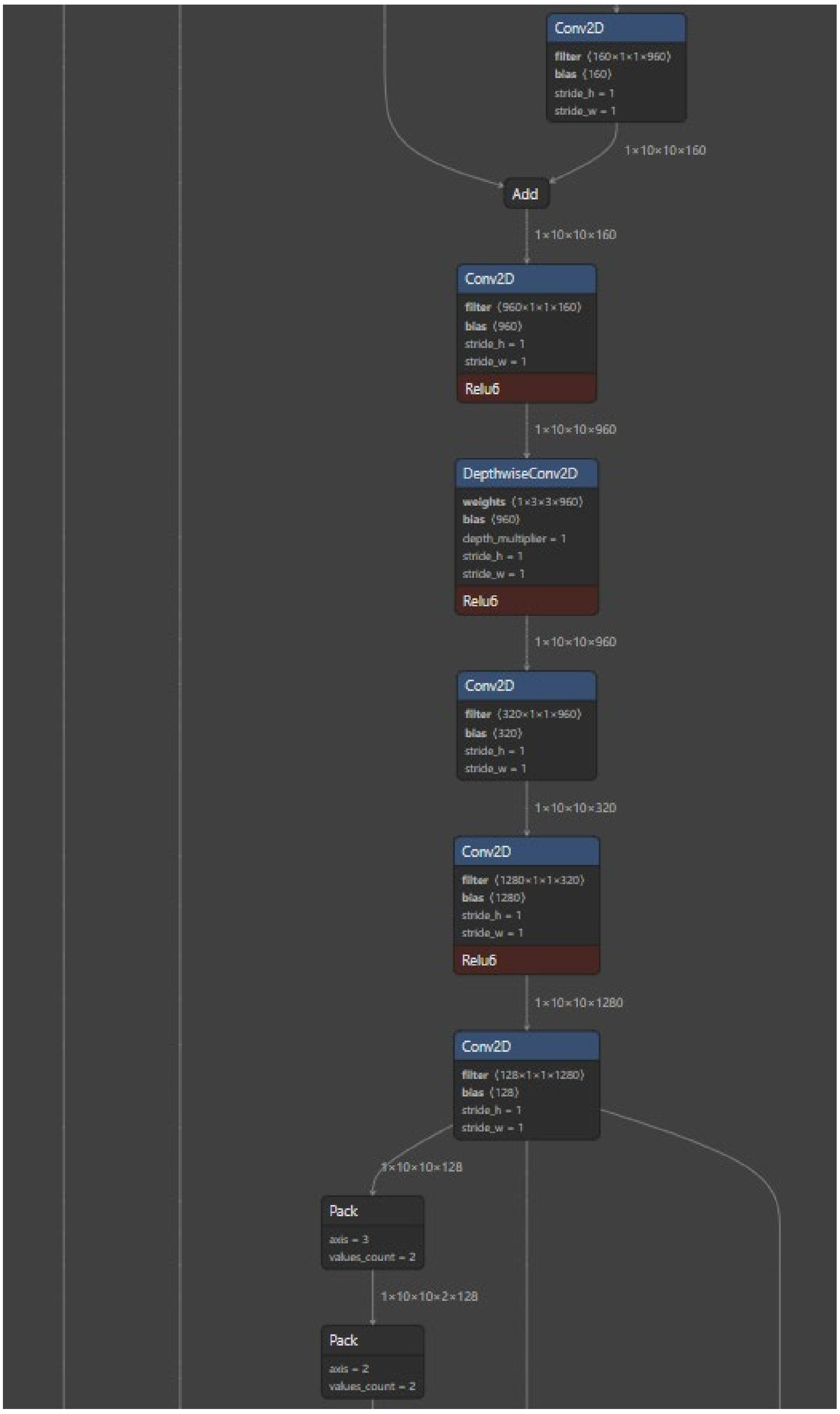
APPENDIX G: (Continued)



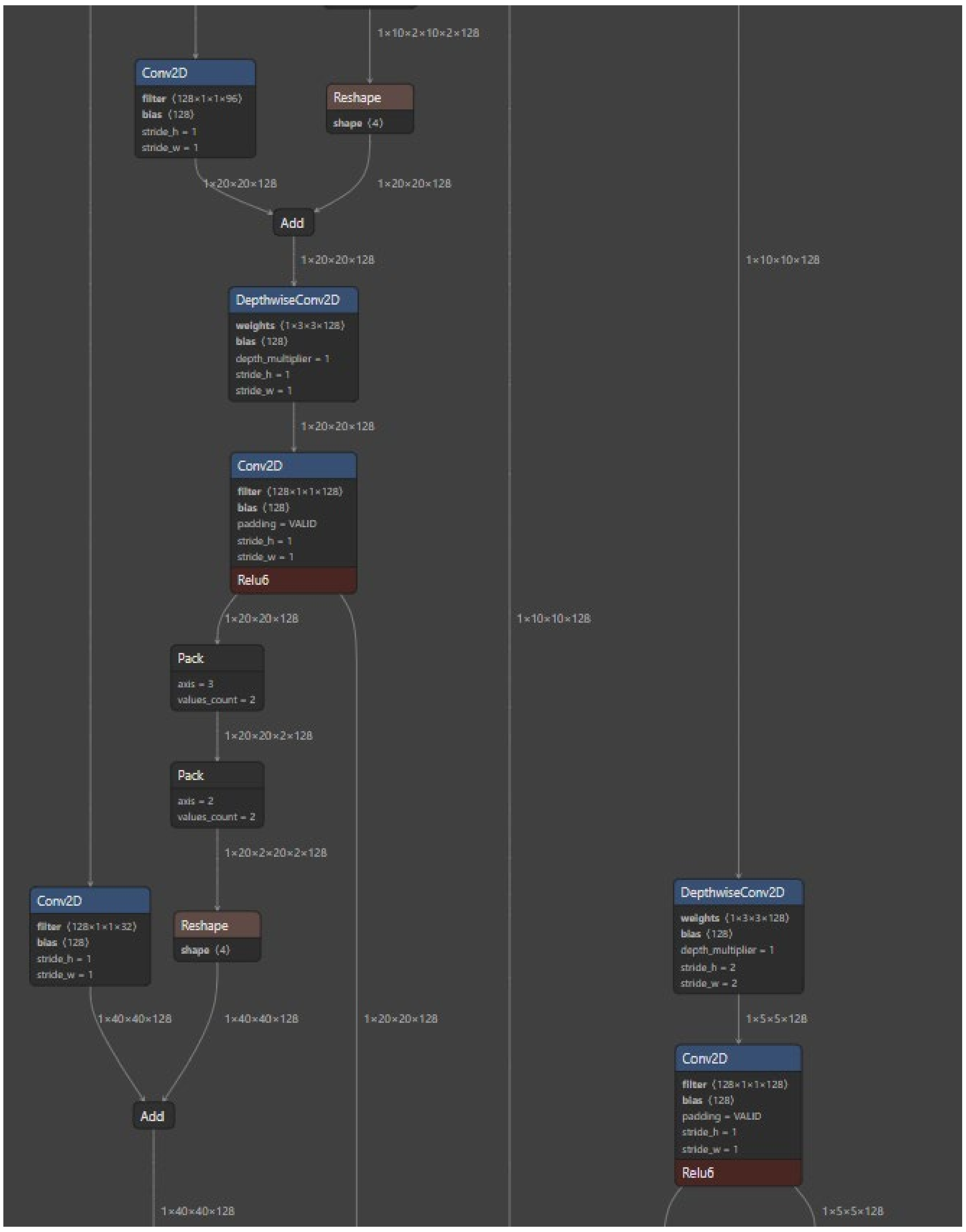
APPENDIX G: (Continued)



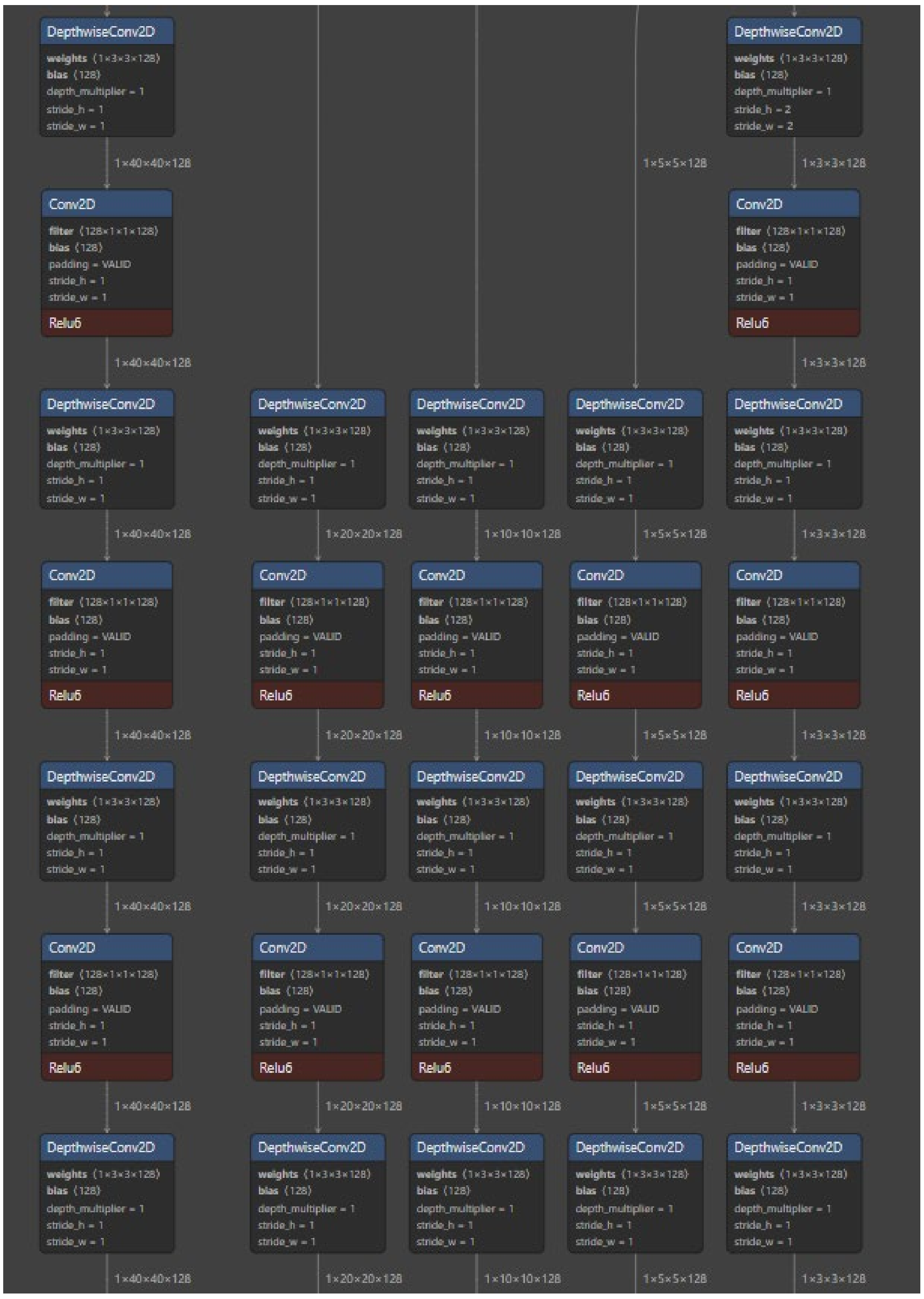
APPENDIX G: (Continued)



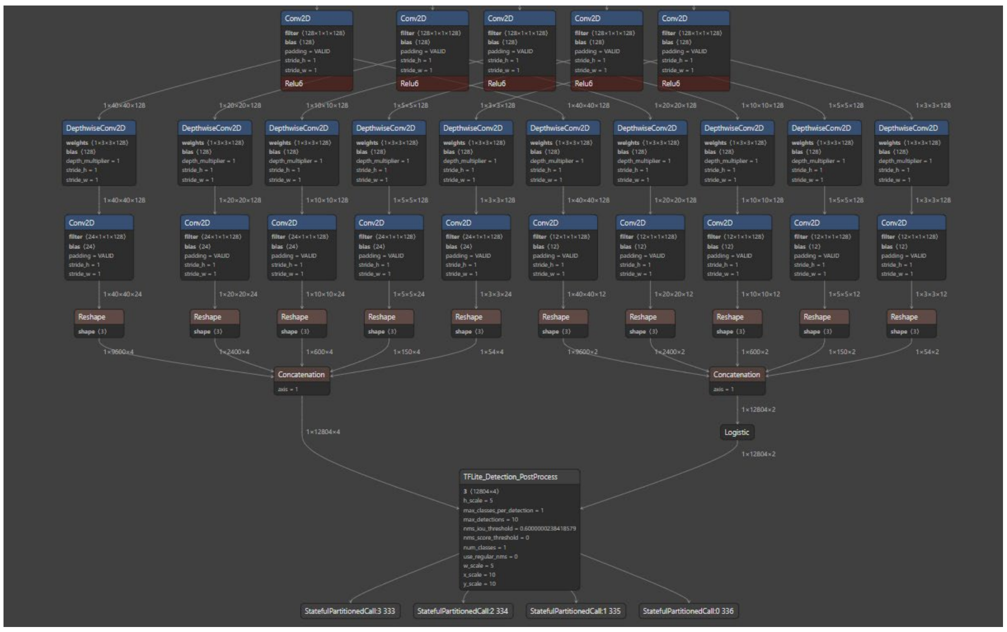
APPENDIX G: (Continued)



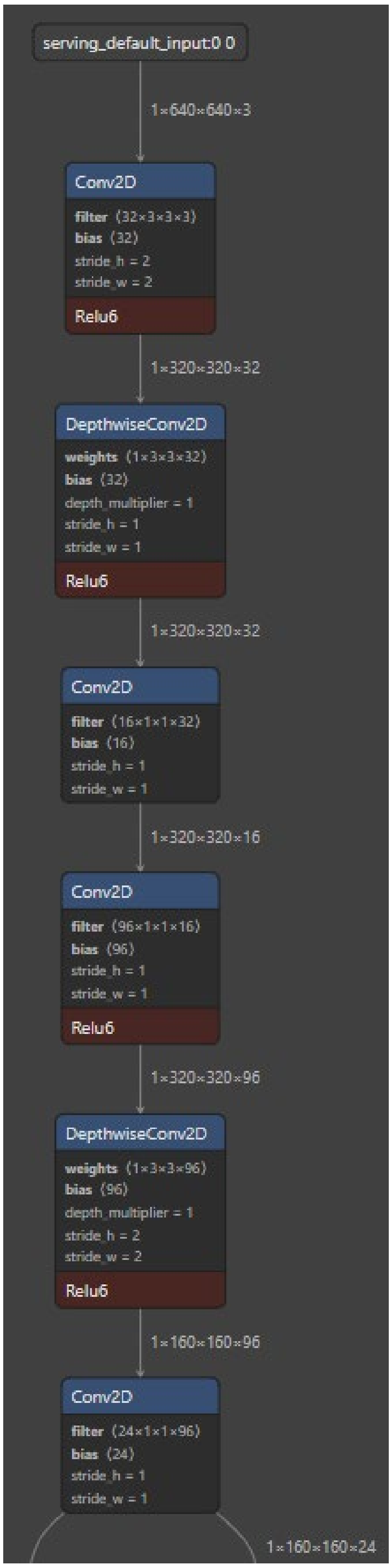
APPENDIX G: (Continued)



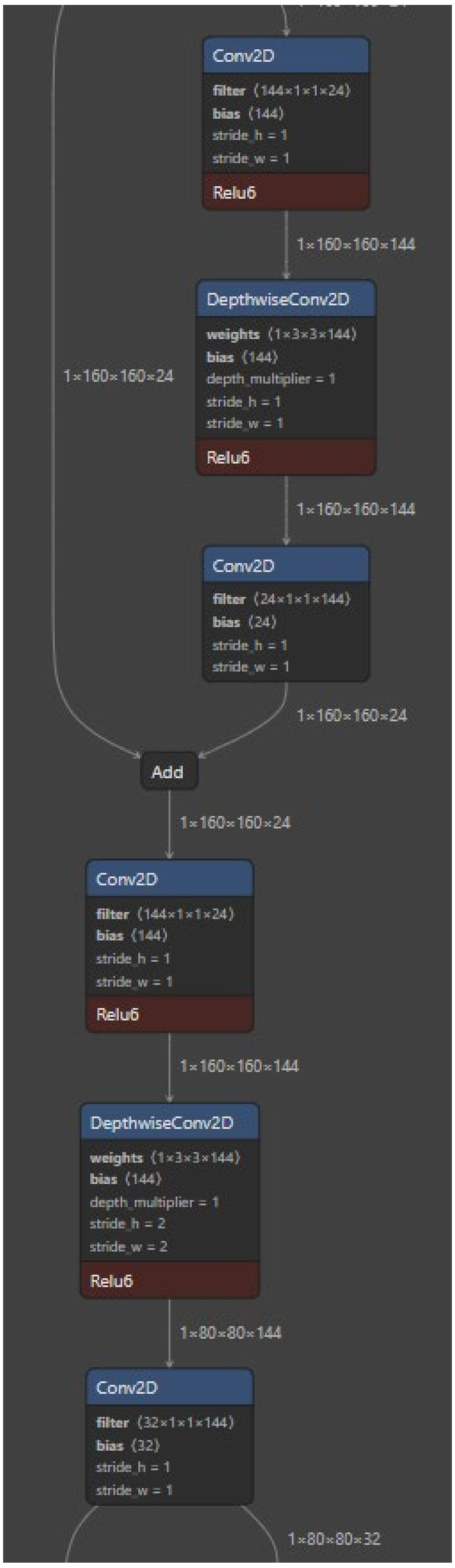
APPENDIX G: (Continued)



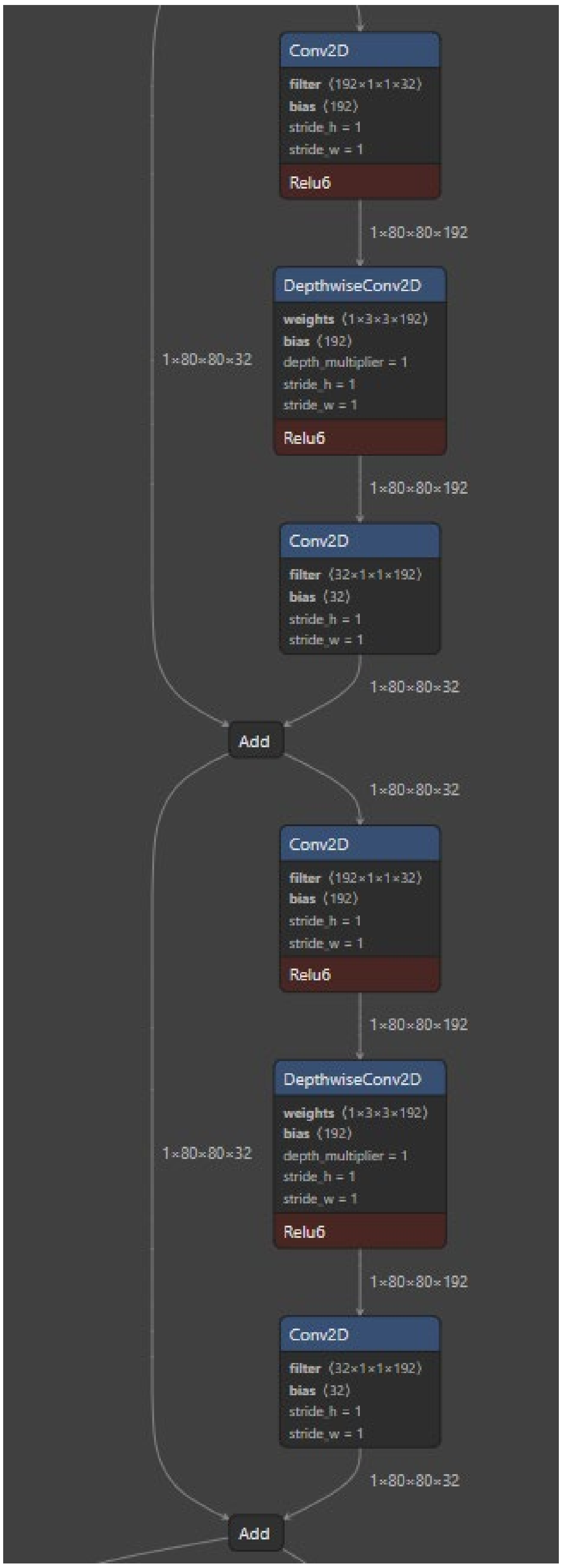
APPENDIX H: Architecture of SSD MobilenetV2 FPNlite 640.



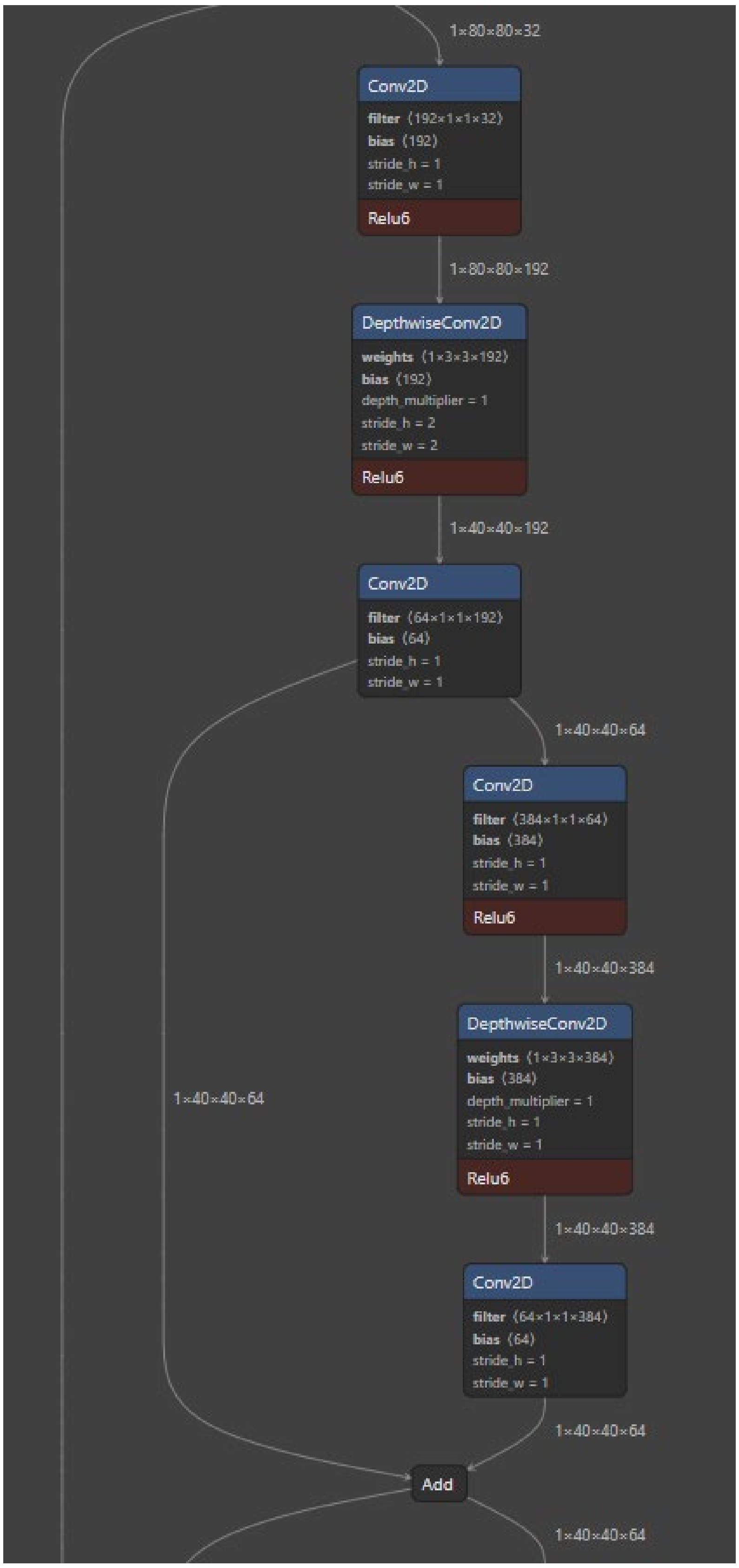
APPENDIX H: (Continued)



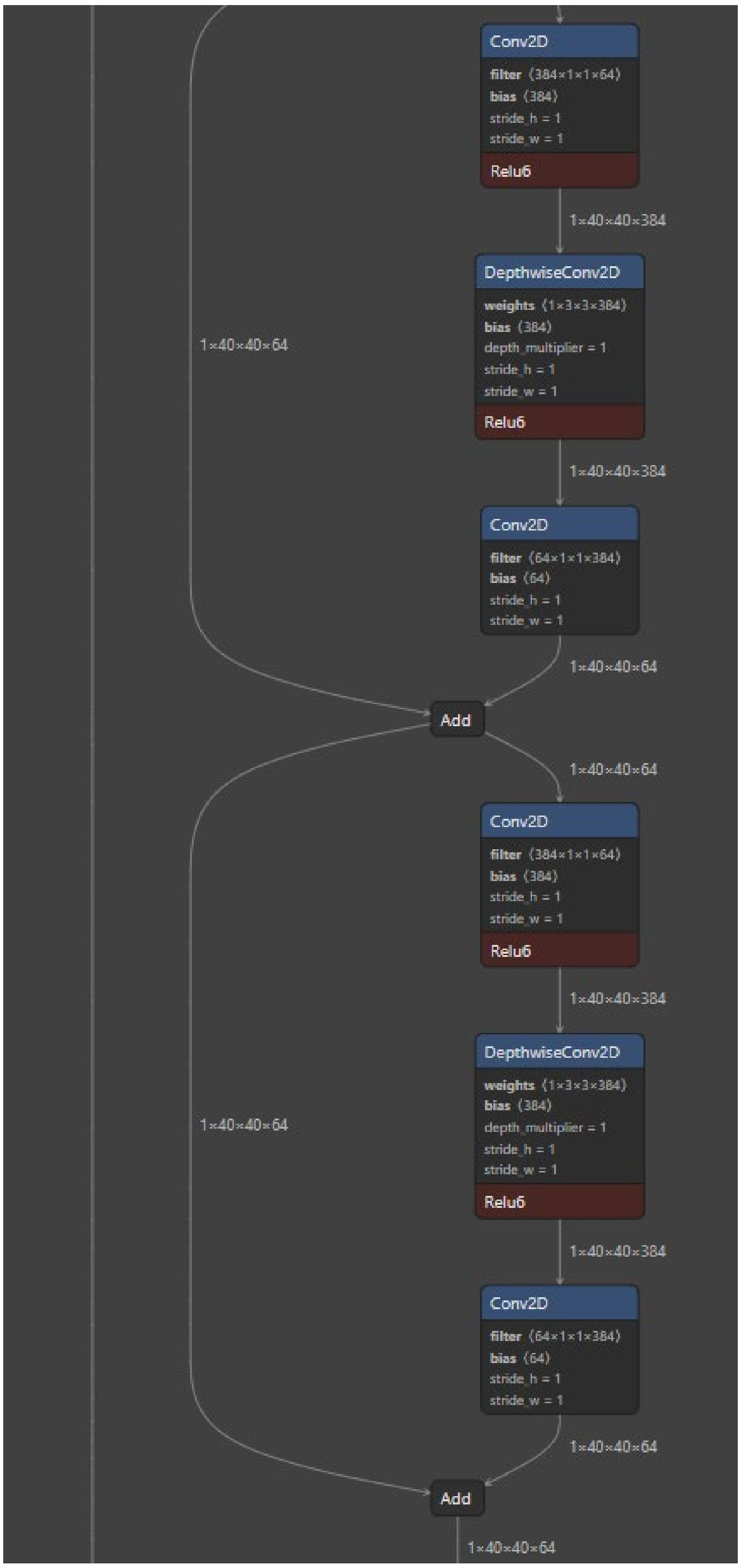
APPENDIX H: (Continued)



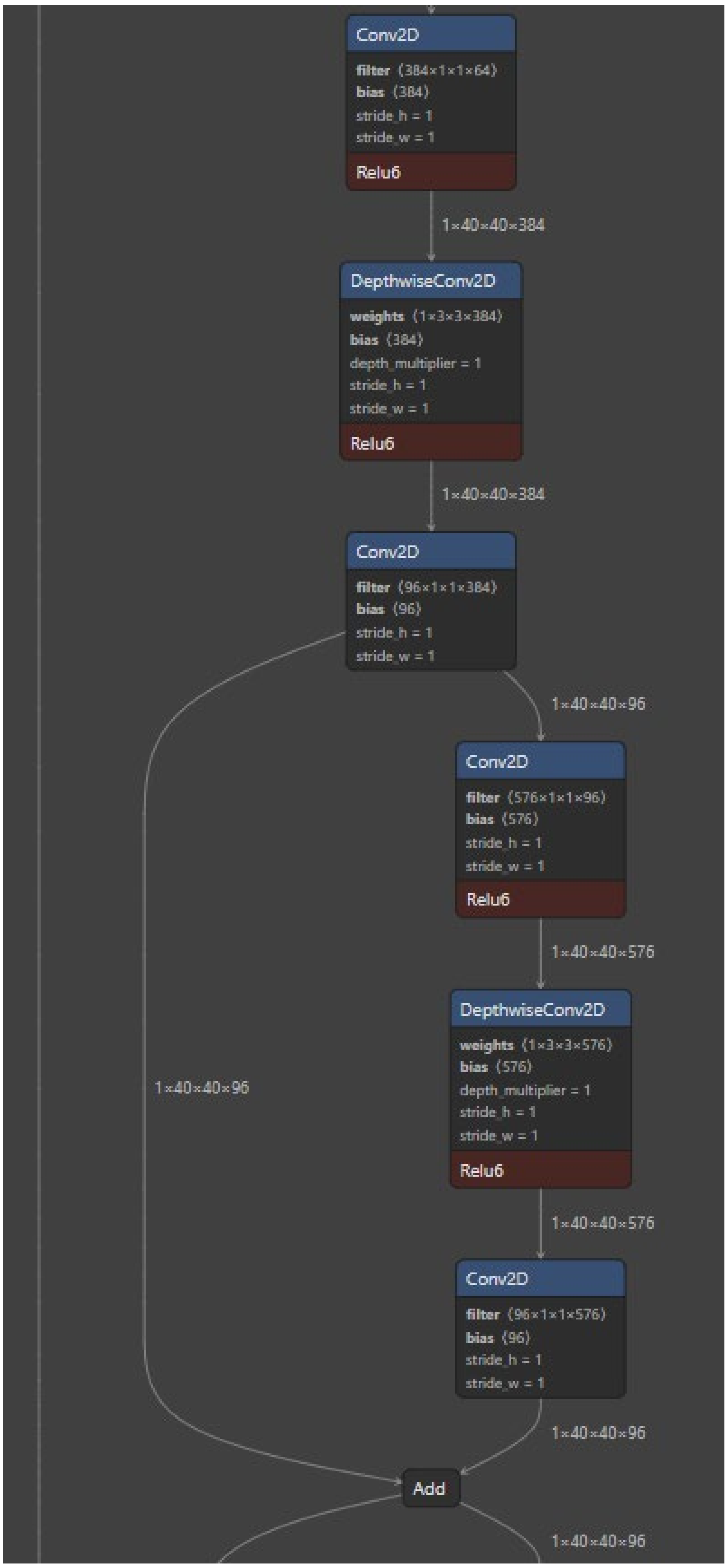
APPENDIX H: (Continued)



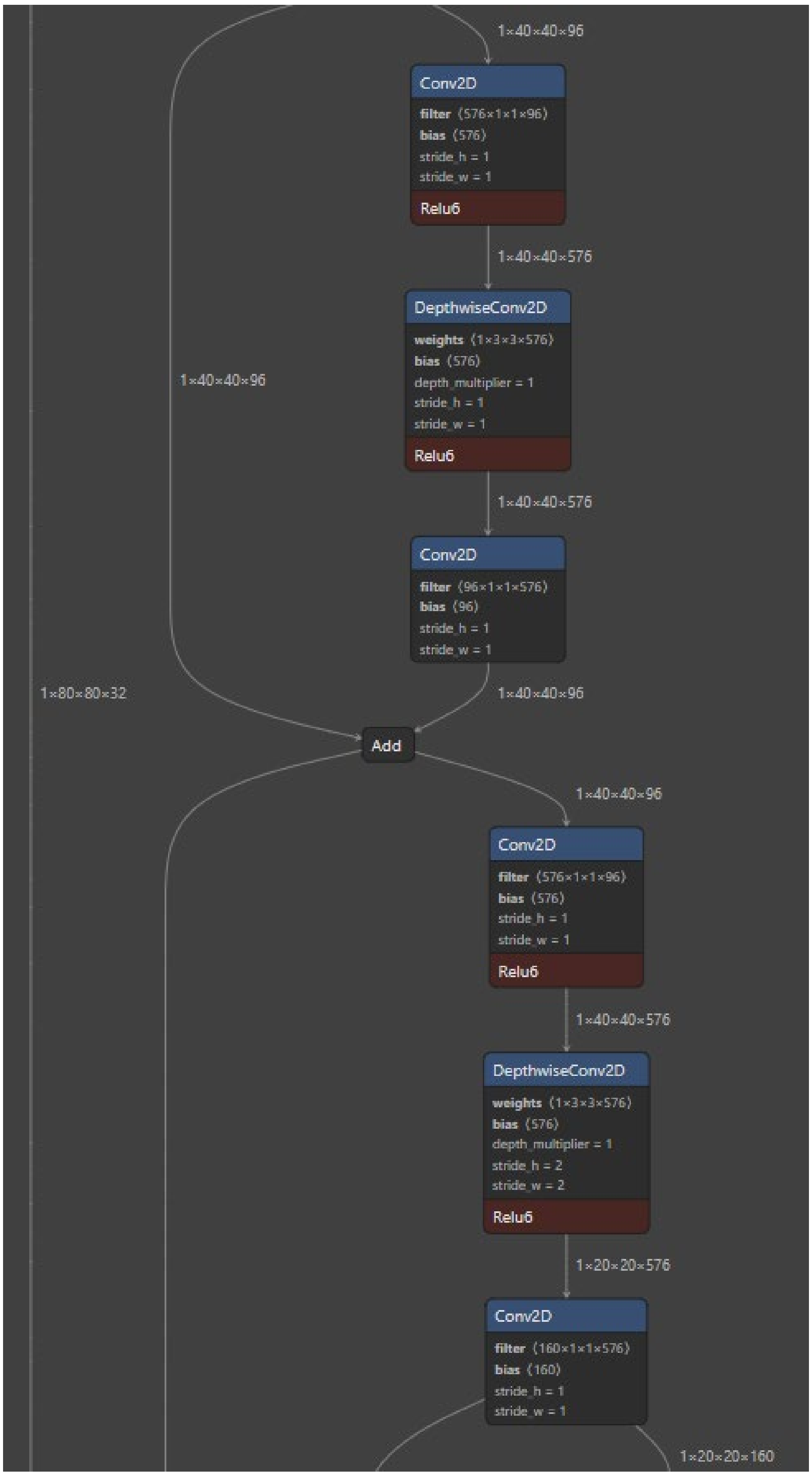
APPENDIX H: (Continued)



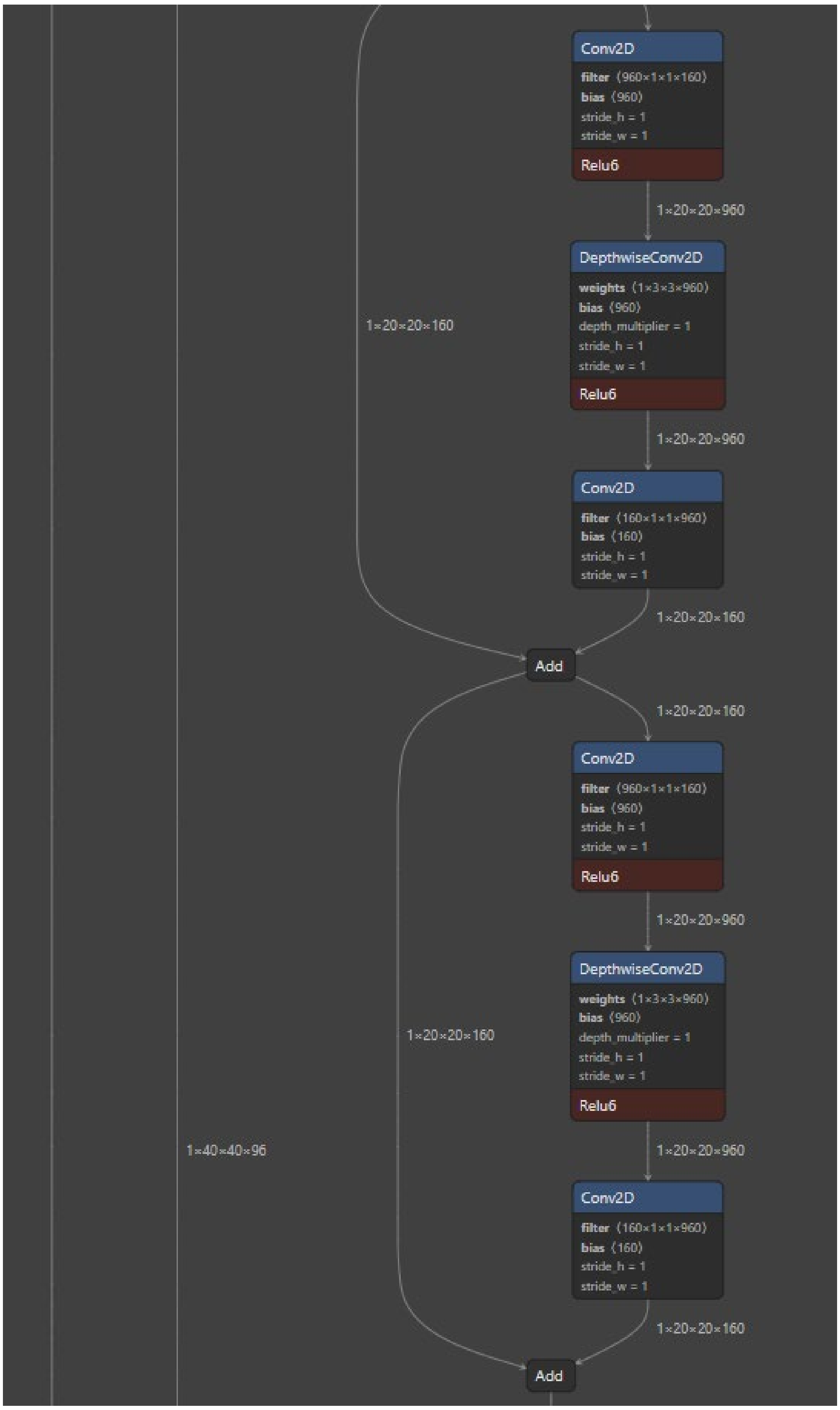
APPENDIX H: (Continued)



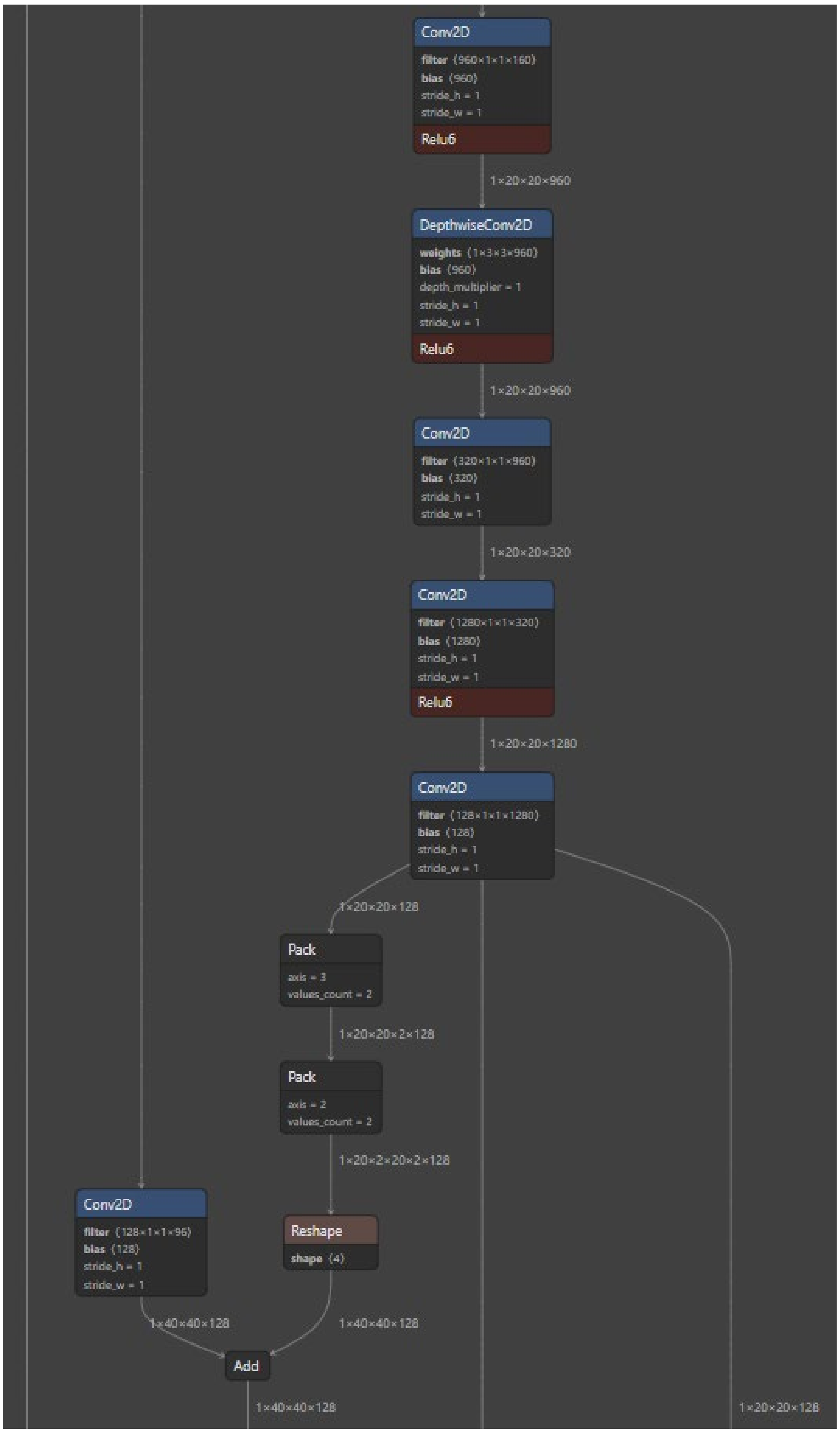
APPENDIX H: (Continued)



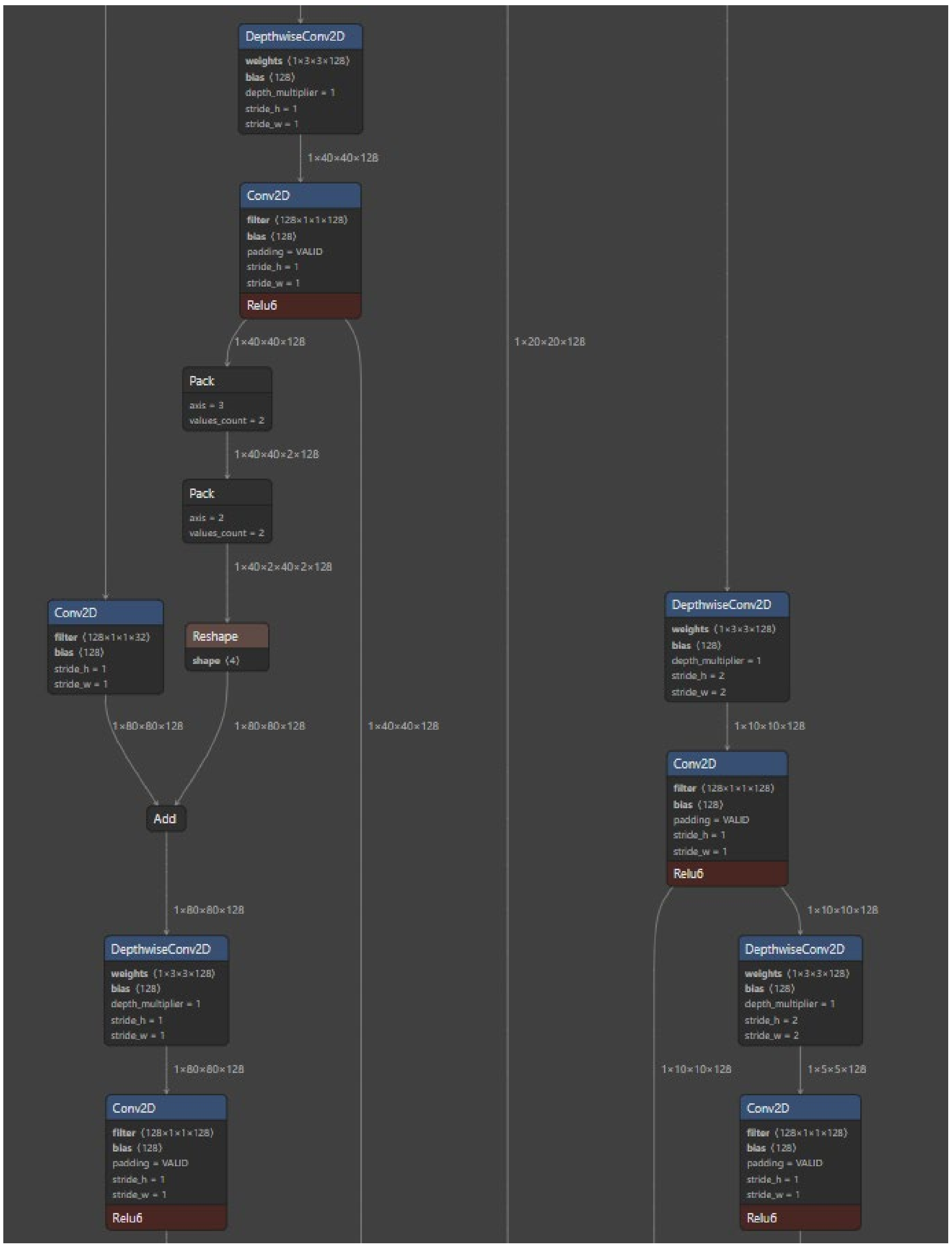
APPENDIX H: (Continued)



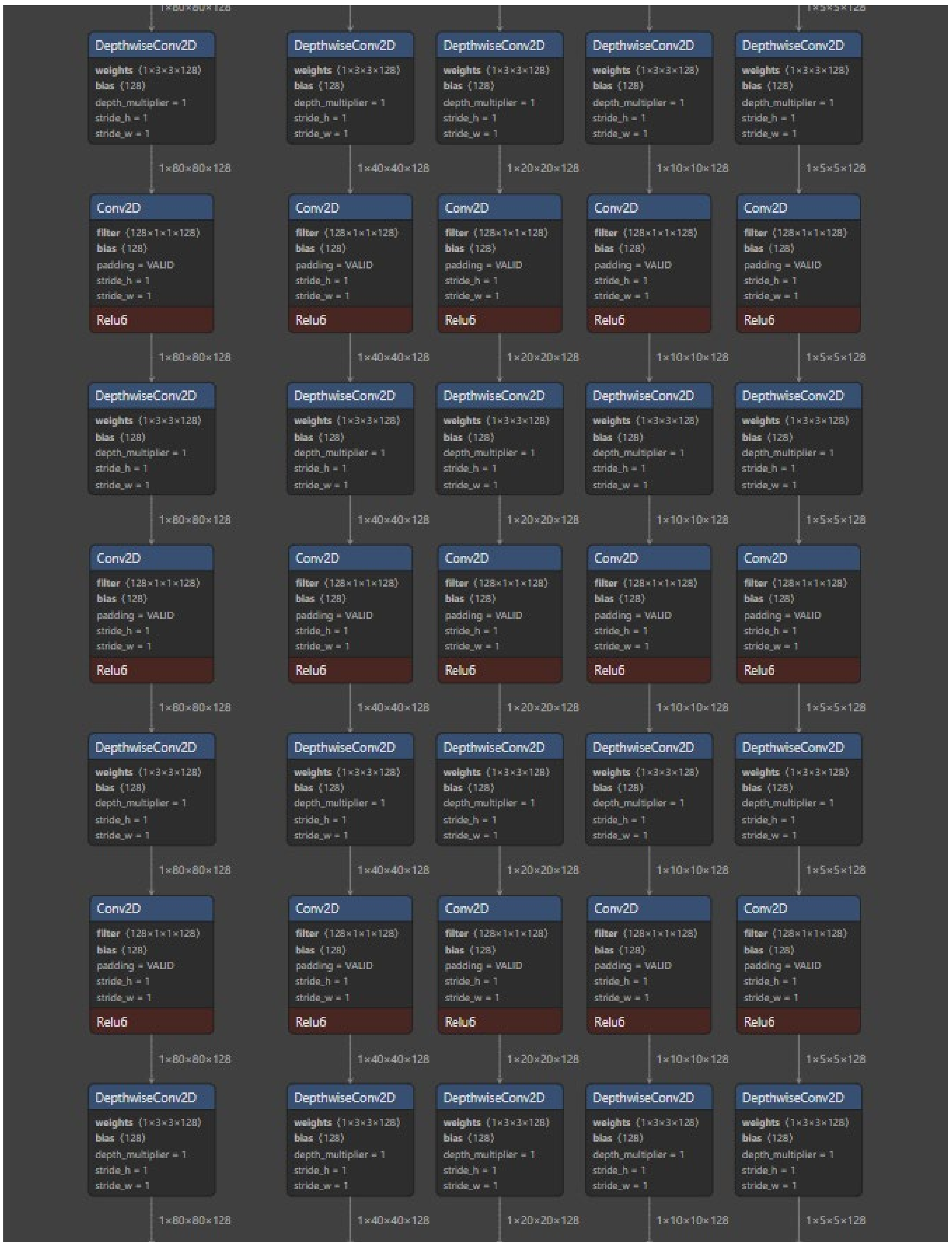
APPENDIX H: (Continued)



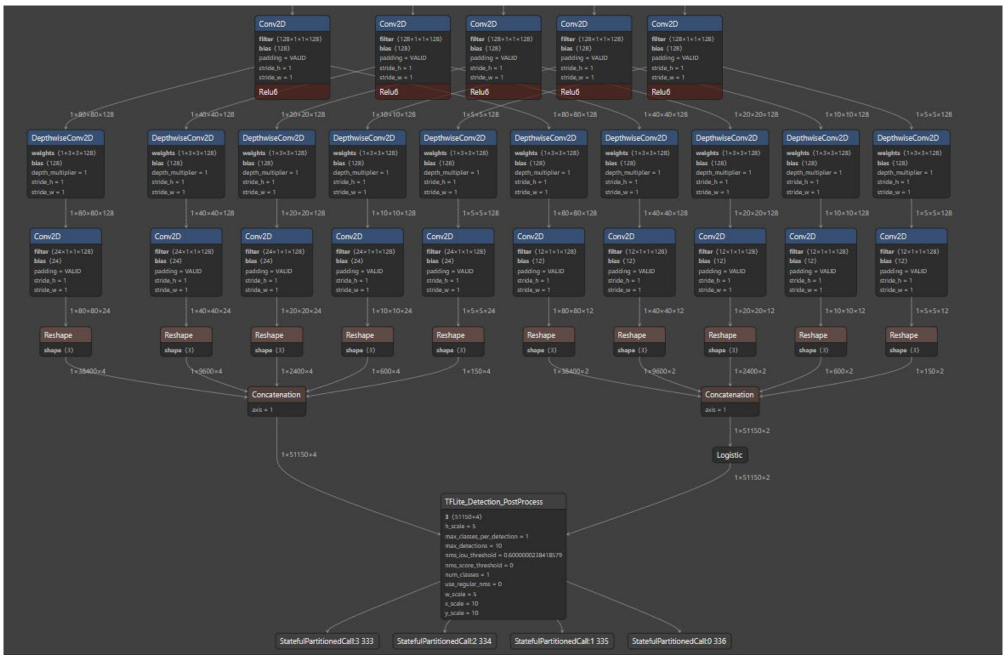
APPENDIX H: (Continued)



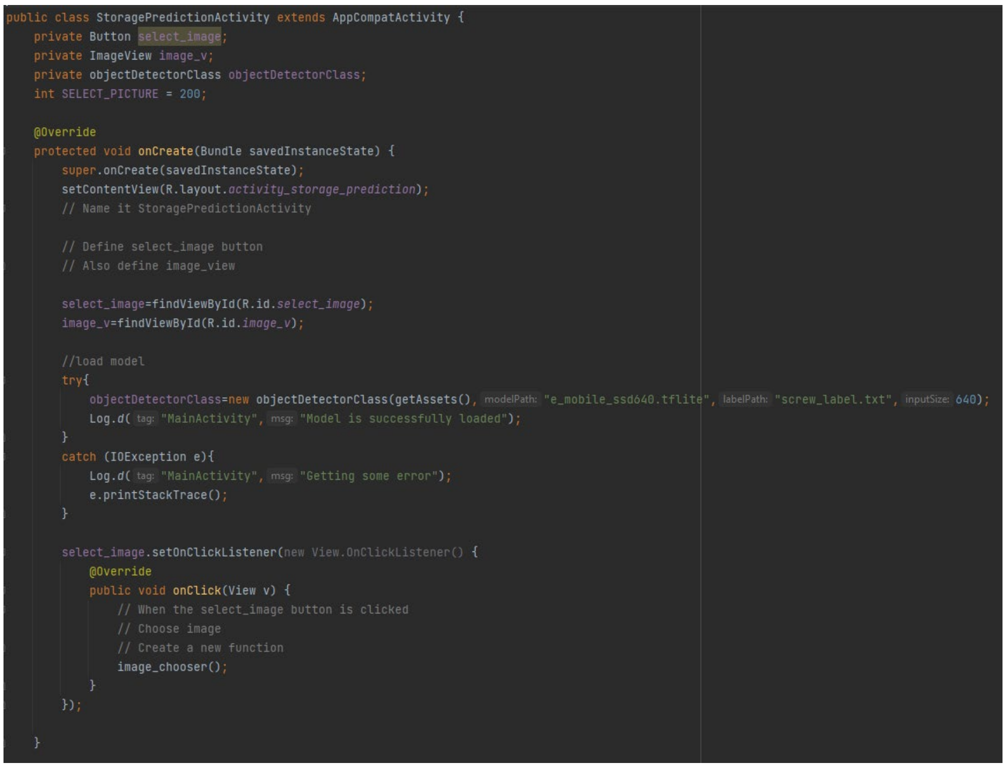
APPENDIX H: (Continued)



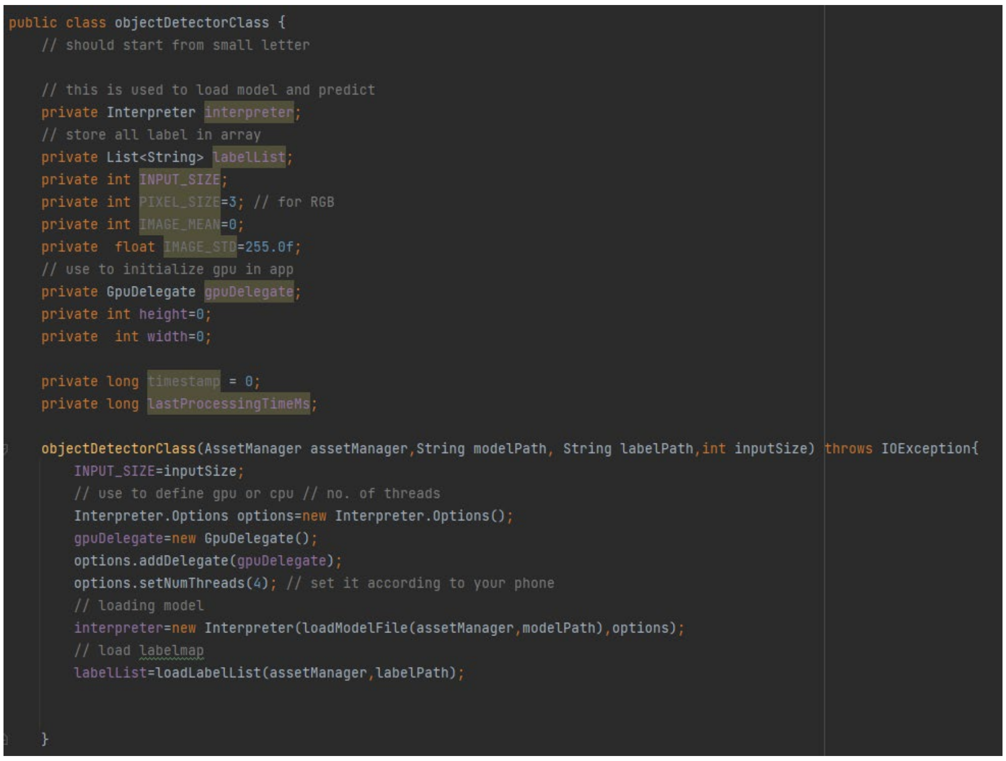
APPENDIX H: (Continued)



APPENDIX I: Object Detection Methods on Mobile App.

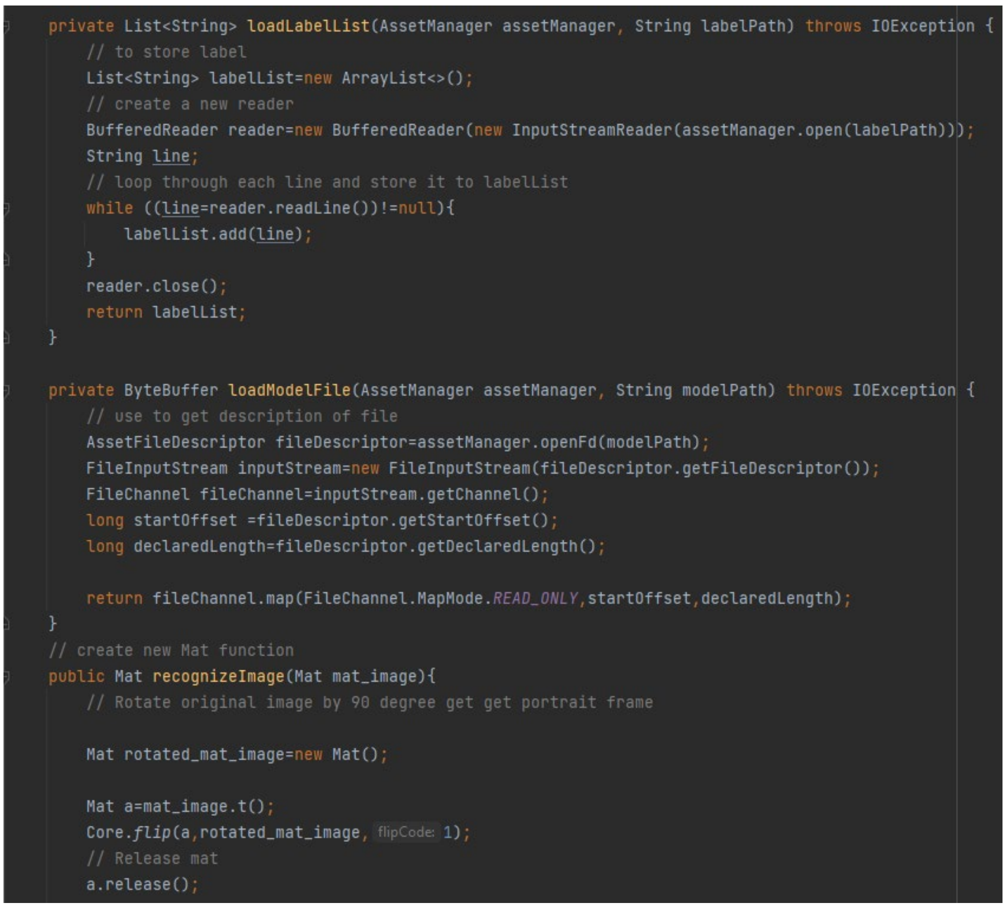


```java
public class StoragePredictionActivity extends AppCompatActivity {
    private Button select_image;
    private ImageView image_v;
    private objectDetectorClass objectDetectorClass;
    int SELECT_PICTURE = 200;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_storage_prediction);
        // Name it StoragePredictionActivity

        // Define select_image button
        // Also define image_view

        select_image=findViewById(R.id.select_image);
        image_v=findViewById(R.id.image_v);

        //load model
        try{
            objectDetectorClass=new objectDetectorClass(getAssets(), modelPath: "e_mobile_ssd640.tflite", labelPath: "screw_label.txt", inputSize: 640);
            Log.d( tag: "MainActivity", msg: "Model is successfully loaded");
        }
        catch (IOException e){
            Log.d( tag: "MainActivity", msg: "Getting some error");
            e.printStackTrace();
        }

        select_image.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // When the select_image button is clicked
                // Choose image
                // Create a new function
                image_chooser();
            }
        });

    }
```

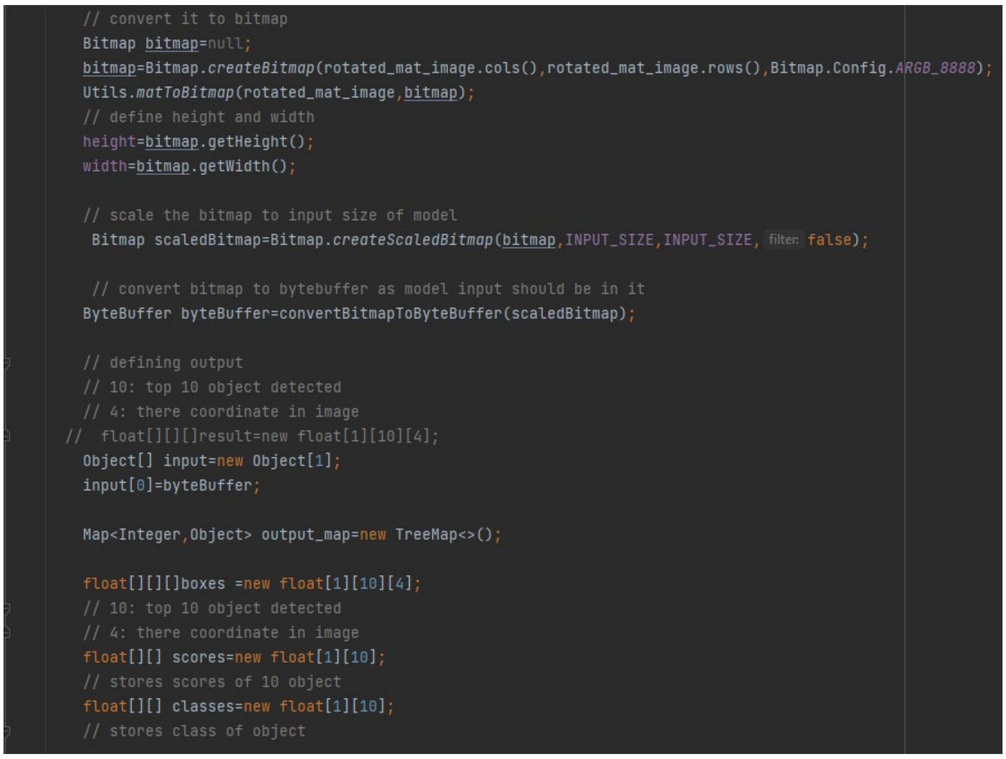


```java
public class objectDetectorClass {
    // should start from small letter

    // this is used to load model and predict
    private Interpreter interpreter;
    // store all label in array
    private List<String> labelList;
    private int INPUT_SIZE;
    private int PIXEL_SIZE=3; // for RGB
    private int IMAGE_MEAN=0;
    private  float IMAGE_STD=255.0f;
    // use to initialize gpu in app
    private GpuDelegate gpuDelegate;
    private int height=0;
    private  int width=0;

    private long timestamp = 0;
    private long lastProcessingTimeMs;

    objectDetectorClass(AssetManager assetManager,String modelPath, String labelPath,int inputSize) throws IOException{
        INPUT_SIZE=inputSize;
        // use to define gpu or cpu // no. of threads
        Interpreter.Options options=new Interpreter.Options();
        gpuDelegate=new GpuDelegate();
        options.addDelegate(gpuDelegate);
        options.setNumThreads(4); // set it according to your phone
        // loading model
        interpreter=new Interpreter(loadModelFile(assetManager,modelPath),options);
        // load labelmap
        labelList=loadLabelList(assetManager,labelPath);


    }
```

APPENDIX I: (Continued)

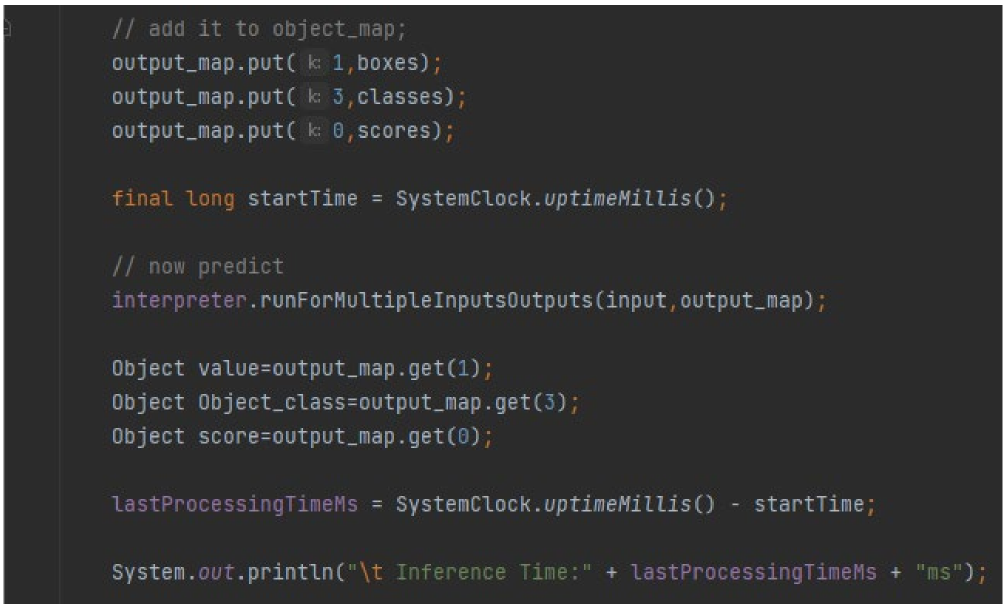


```java
private List<String> loadLabelList(AssetManager assetManager, String labelPath) throws IOException {
    // to store label
    List<String> labelList=new ArrayList<>();
    // create a new reader
    BufferedReader reader=new BufferedReader(new InputStreamReader(assetManager.open(labelPath)));
    String line;
    // loop through each line and store it to labelList
    while ((line=reader.readLine())!=null){
        labelList.add(line);
    }
    reader.close();
    return labelList;
}

private ByteBuffer loadModelFile(AssetManager assetManager, String modelPath) throws IOException {
    // use to get description of file
    AssetFileDescriptor fileDescriptor=assetManager.openFd(modelPath);
    FileInputStream inputStream=new FileInputStream(fileDescriptor.getFileDescriptor());
    FileChannel fileChannel=inputStream.getChannel();
    long startOffset =fileDescriptor.getStartOffset();
    long declaredLength=fileDescriptor.getDeclaredLength();

    return fileChannel.map(FileChannel.MapMode.READ_ONLY,startOffset,declaredLength);
}
// create new Mat function
public Mat recognizeImage(Mat mat_image){
    // Rotate original image by 90 degree get get portrait frame

    Mat rotated_mat_image=new Mat();

    Mat a=mat_image.t();
    Core.flip(a,rotated_mat_image, flipCode: 1);
    // Release mat
    a.release();
```

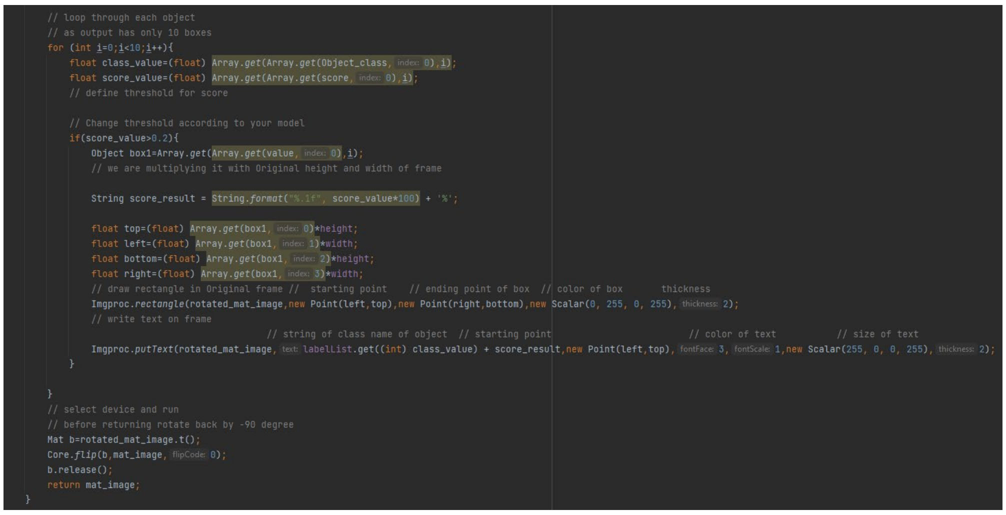


```java
    // convert it to bitmap
    Bitmap bitmap=null;
    bitmap=Bitmap.createBitmap(rotated_mat_image.cols(),rotated_mat_image.rows(),Bitmap.Config.ARGB_8888);
    Utils.matToBitmap(rotated_mat_image,bitmap);
    // define height and width
    height=bitmap.getHeight();
    width=bitmap.getWidth();

    // scale the bitmap to input size of model
     Bitmap scaledBitmap=Bitmap.createScaledBitmap(bitmap,INPUT_SIZE,INPUT_SIZE, filter: false);

     // convert bitmap to bytebuffer as model input should be in it
    ByteBuffer byteBuffer=convertBitmapToByteBuffer(scaledBitmap);

    // defining output
    // 10: top 10 object detected
    // 4: there coordinate in image
  //  float[][][]result=new float[1][10][4];
    Object[] input=new Object[1];
    input[0]=byteBuffer;

    Map<Integer,Object> output_map=new TreeMap<>();

    float[][][]boxes =new float[1][10][4];
    // 10: top 10 object detected
    // 4: there coordinate in image
    float[][] scores=new float[1][10];
    // stores scores of 10 object
    float[][] classes=new float[1][10];
    // stores class of object
```

APPENDIX I: (Continued)



```java
        // add it to object_map;
        output_map.put( k: 1,boxes);
        output_map.put( k: 3,classes);
        output_map.put( k: 0,scores);

        final long startTime = SystemClock.uptimeMillis();

        // now predict
        interpreter.runForMultipleInputsOutputs(input,output_map);

        Object value=output_map.get(1);
        Object Object_class=output_map.get(3);
        Object score=output_map.get(0);

        lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;

        System.out.println("\t Inference Time:" + lastProcessingTimeMs + "ms");
```



```java
        // loop through each object
        // as output has only 10 boxes
        for (int i=0;i<10;i++){
            float class_value=(float) Array.get(Array.get(Object_class, index: 0),i);
            float score_value=(float) Array.get(Array.get(score, index: 0),i);
            // define threshold for score

            // Change threshold according to your model
            if(score_value>0.2){
                Object box1=Array.get(Array.get(value, index: 0),i);
                // we are multiplying it with Original height and width of frame

                String score_result = String.format("%.1f", score_value*100) + '%';

                float top=(float) Array.get(box1, index: 0)*height;
                float left=(float) Array.get(box1, index: 1)*width;
                float bottom=(float) Array.get(box1, index: 2)*height;
                float right=(float) Array.get(box1, index: 3)*width;
                // draw rectangle in Original frame //  starting point    // ending point of box  // color of box       thickness
                Imgproc.rectangle(rotated_mat_image,new Point(left,top),new Point(right,bottom),new Scalar(0, 255, 0, 255), thickness: 2);
                // write text on frame
                                        // string of class name of object  // starting point                         // color of text           // size of text
                Imgproc.putText(rotated_mat_image, text: labelList.get((int) class_value) + score_result,new Point(left,top), fontFace: 3, fontScale: 1,new Scalar(255, 0, 0, 255), thickness: 2);
            }

        }
        // select device and run
        // before returning rotate back by -90 degree
        Mat b=rotated_mat_image.t();
        Core.flip(b,mat_image, flipCode: 0);
        b.release();
        return mat_image;
    }
```

APPENDIX J: Object Detection Methods on Webcam.

```
#!/usr/bin/env python
# coding: utf-8
"""
Detect Objects Using Webcam
===============================
"""

import os
import time

DATA_DIR = os.path.join(os.getcwd(), 'data')
MODELS_DIR = os.path.join(DATA_DIR, 'models')
for dir in [DATA_DIR, MODELS_DIR]:
    if not os.path.exists(dir):
        os.mkdir(dir)

MODEL_NAME = 'v2_webcam_ssd640'
PATH_TO_CKPT = os.path.join(MODELS_DIR, os.path.join(MODEL_NAME,
'checkpoint/'))
PATH_TO_CFG = os.path.join(MODELS_DIR, os.path.join(MODEL_NAME,
'pipeline.config'))
LABEL_FILENAME = 'mscoco_label_map.pbtxt'
PATH_TO_LABELS = os.path.join(MODELS_DIR, os.path.join(MODEL_NAME,
LABEL_FILENAME))

# %%
# Load the model
# ~~~~~~~~~~~~~~
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow logging
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import config_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder

tf.get_logger().setLevel('ERROR')           # Suppress TensorFlow logging (2)

# Enable GPU dynamic memory allocation
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(PATH_TO_CFG)
model_config = configs['model']
detection_model = model_builder.build(model_config=model_config,
is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(PATH_TO_CKPT, 'ckpt-0')).expect_partial()
```

APPENDIX J: (Continued)

```
@tf.function
def detect_fn(image):
   """Detect objects in image."""

   image, shapes = detection_model.preprocess(image)
   prediction_dict = detection_model.predict(image, shapes)
   detections = detection_model.postprocess(prediction_dict, shapes)

   return detections, prediction_dict, tf.reshape(shapes, [-1])


# %%
# Load label map data (for plotting)
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
category_index                                                      =
label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS,
                                                    use_display_name=True)

# %%
# Define the video stream
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
import cv2

cap = cv2.VideoCapture(0)

import numpy as np

while True:
   # Read frame from camera
   ret, image_np = cap.read()

   # Expand dimensions since the model expects images to have shape: [1, None, None,
3]
   image_np_expanded = np.expand_dims(image_np, axis=0)

   input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)

   start = time.time()

   detections, predictions_dict, shapes = detect_fn(input_tensor)

   label_id_offset = 1
   image_np_with_detections = image_np.copy()

   end = time.time()
   print("Time: ",end - start)

   viz_utils.visualize_boxes_and_labels_on_image_array(
        image_np_with_detections,
        detections['detection_boxes'][0].numpy(),
        (detections['detection_classes'][0].numpy() + label_id_offset).astype(int),
```

APPENDIX J: (Continued)

```
            detections['detection_scores'][0].numpy(),
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=200,
            min_score_thresh=.20,
            agnostic_mode=False)



    # Display output
    cv2.imshow('object detection', cv2.resize(image_np_with_detections, (800, 600)))

    if cv2.waitKey(25) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```