

**ACTION DETECTION SYSTEM FOR ALERTING DRIVER
USING COMPUTER VISION**

BY

KHOO CHIA HONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Kampar Campus)

JANUARY 2020

REPORT STATUS DECLARATION FORM

Title: Action Detection System For Alerting Driver Using Computer Vision

Academic Session: JAN 2020

I KHOO CHIA HONG
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

Khoo Chia Hong

(Author's signature)

JML

(Supervisor's signature)

Address:

2A-3-7 E-Park,

Jalan Batu Uban,

11700 Gelugor, Pulau Pinang.

Lau Phooi Yee, PhD

Supervisor's name

Date: 20 April 2020

Date: 20 April 2020

**ACTION DETECTION SYSTEM FOR ALERTING DRIVER
USING COMPUTER VISION**

BY

KHOO CHIA HONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Kampar Campus)

JANUARY 2020

DECLARATION OF ORIGINALITY

I declare that this report entitled “**ACTION DETECTION SYSTEM FOR ALERTING DRIVER USING COMPUTER VISION**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : *Khoo Chia Hong*

Name : Khoo Chia Hong

Date : 20 April 2020

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere thanks and appreciation to my supervisor, Dr. Lau Phooi Yee who had given me this opportunity to engage in this project related to ADAS (Advanced Driver Assistance System). It is my first step to establish a computer vision and real-time development field. A million thanks to you.

Besides that, I would like to thank all my friends for support and giving me idea me during the project development. Finally, I would also like to thank my family for their love, support and continuous encouragement throughout the entire course especially my father who keep helped me in both financial and academic problem throughout the entire course.

ABSTRACT

Nowadays, the increasing number of careless drivers on the road had resulted in more accident cases. Driver's decisions and behaviors are the keys to maintain road safety. However, many drivers tend to do secondary task like playing with their phone, adjusting radio player, eating or drinking, answering phone calls, and worse case is reading phone text.

In previous efforts, many kinds of approaches had been introduced to try to solve the task to recognize and capture potential problem related to careless driving inside the car. In this project, the work will mainly focus on the driver secondary tasks recognition using the action detection method. A camera will be set up inside the car for the real-time extract of driver's action. The video will undergo a process to extract out the human pose frames without background called human pose estimator framework. Inside this framework, raw image will be input into a CNN network that compute human key points activation maps. After that key points coordinate will be computed using the output activation maps and drawn on a new blank frame. Then the frames will be input into Pose-based Convolutional Neural Network for the action classification. If an action performed by driver is considered a dangerous secondary task, alert will be given.

The proposed framework was able to achieve a higher speed compare to others people framework if it is being run on Raspberry Pi CPU. It is able to detect 10 different driver actions where only talking to passengers and normal driving will not trigger the buzzer to give alert to driver.

TABLE OF CONTENTS

TITLE PAGE	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	xi
LIST OF TABLES	xiii
LIST OF ABBREVIATIONS	xiv
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement	1
1.2 Background and Motivation	2
1.3 Objectives	4
1.4 Report Organization	5
CHAPTER 2 LITERATURE REVIEW	6
2.1 Driver Action Recognition using Convolutional Neural Network	6
2.2 Skin-Like Regions	7
2.3 Merge Multiple Layer Output in CNN	8
2.4 Six-Axis Motion Processor	9

2.5	Action Detection Using Pose Estimation	11
2.6	Optical Flow and SVM Classifier	12
2.7	SURF Key Points	13
2.8	Skeleton Motion History	14
2.9	Human Pose Estimation	16
2.10	Human Action Detection Using Raspberry Pi	19
CHAPTER 3 SYSTEM DESIGN		21
3.1	Proposed System Framework	21
3.1.1	Step 1: System Boot	23
3.1.2	Step 2: Image Acquisition	24
3.1.3	Step 3: Pre-processing	24
3.1.4	Step 4: Human Pose Estimation	25
3.1.5	Step 5: Driver Action Detection	30
3.1.6	Step 6: Buzzer Action	31
3.2	System Requirement	32
3.2.1	System Hardware Requirement	32
3.2.2	System Software Requirement	35
3.3	System Setup	36
CHAPTER 4 SYSTEM ANALYSIS		39
4.1	Analysis Overview	39

4.2	Camera Selection Analysis	39
4.2.1	Pi Camera VS USB Webcam	39
4.2.2	Pi Camera VS Smartphone Camera	40
4.3	Camera Location Setup and Analysis	41
4.3.1	Camera Setup Location 1	41
4.3.2	Camera Setup Location 2	42
4.3.3	Camera Setup Location 3	43
4.4	Raspberry Pi Selection Analysis	45
4.5	Human Pose Estimation Analysis	46
4.6	Classification Models Analysis	49
4.6.1	Model 1	49
4.6.2	Model 2	52
4.6.3	Model 3	56
CHAPTER 5 IMPLEMENTATION		59
5.1	Implementation Overview	59
5.2	Image Acquisition	59
5.3	Image Pre-processing	61
5.4	Human Pose Estimation	61
5.5	Driver Action Detection	64
5.6	Buzzer Alert Notification	64
CHAPTER 6 SYSTEM TESTING		65

6.1	Verification Plan	65
6.2	Short Clip Testing Overview	66
6.2.1	Experiment 1 – Short Sleeve	66
6.2.2	Experiment 2 – Long Sleeve	68
6.3	Live Based Testing	70
CHAPTER 7 CONCLUSION		72
REFERENCE		74

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	Architecture of Proposed CNN Network Contains Three Stages	6
Figure 2.2	Implementation of Gaussian Mixture Model for Skin Modeling	7
Figure 2.3	Framework of the proposed DedistractedNet	8
Figure 2.4	Structure of MV-CNN	10
Figure 2.5	Flow diagram of pose-based 3D CNN model	11
Figure 2.6	Block Diagram of Human Action Recognition using Optical Flow and SVM	12
Figure 2.7	Flow of Driver Action Classification using SURF Key Points	13
Figure 2.8a	Flow of Create Skl MHI	14
Figure 2.8b	Model Architecture of 2D-DCNN	15
Figure 2.9a	OpenPose Model Architecture	16
Figure 2.9b	Line Integral Algorithm	17
Figure 2.9c	Assignment Algorithm	17
Figure 2.9d	OpenPose Pipeline	18
Figure 2.10a	System Architecture for Activity Recognition System	19
Figure 2.10b	Raspberry Pi Based Framework for Activity Recognition	20
Figure 3.1	Flow Chart of Proposed Framework	21
Figure 3.1.1	System Boot	23
Figure 3.1.3	Image Pre-processing	24
Figure 3.1.4a	Heatmap Generation Model	25
Figure 3.1.4b	Overview of Heatmap Generation Model	25
Figure 3.1.4c	Architecture of Pre Stage	26
Figure 3.1.4d	Architecture of Initial Stage	26
Figure 3.1.4e	Architecture of Refinement Stage	27

Figure 3.1.4f	Generate Key Points Indexes	28
Figure 3.1.5	Model Architecture of Action Classification	31
Figure 3.1.6	Buzzer Action	31
Figure 3.2.1a	Raspberry Pi 4 Model B Module	32
Figure 3.2.1b	Raspberry Pi Camera	32
Figure 3.2.1c	Speaker Buzzer	32
Figure 3.2.1d	GPS Golder Mount	32
Figure 3.2.1e	USB Car Charger	32
Figure 3.2.1f	2m Type-C Cable	32
Figure 3.2.1g	Raspberry Pi 40-pins GPIO Layout	33
Figure 3.2.1h	Buzzer Location	34
Figure 3.2.2a	Python IDLE	35
Figure 3.2.2b	OpenCV Library	35
Figure 3.2.2c	PyTorch Library	35
Figure 3.2.2d	Numpy Library	35
Figure 3.2.2e	Keras Library	35
Figure 3.2.2f	Tensorflow Library	35
Figure 3.3a	Speaker Buzzer Connect to Raspberry Pi	36
Figure 3.3b	Camera module connect to Raspberry Pi	36
Figure 3.3c	Mount Raspberry Pi on GPS Holder	37
Figure 3.3d	Provide Power Supply to Raspberry Pi	37
Figure 3.3e	Setting Raspberry Pi inside Vehicle	38
Figure 4.2.1a	Pi Camera	39
Figure 4.2.1b	USB Webcam	39
Figure 4.2.2a	Pi Camera	40
Figure 4.2.2b	Phone Camera	40
Figure 4.3.1a	Camera Setup Location 1	41
Figure 4.3.1b	Frame Capture from Location 1	42
Figure 4.3.1c	Pose Estimate Output	42
Figure 4.3.2a	Camera Setup Location 2	42
Figure 4.3.2b	Frame Capture from Location 2	43

Figure 4.3.3a	Camera Setup at Location 3	43
Figure 4.3.3b	Frame Capture from Location 3	44
Figure 4.6.1a	Outcome of Evaluation of Model 1	50
Figure 4.6.1b	Architecture of Model 1	50
Figure 4.6.1c	Details of Model 1	51
Figure 4.6.2a	Outcome of Evaluation of Model 2	53
Figure 4.6.2b	Architecture of Model 2	53
Figure 4.6.2c	Details of Model 2	54
Figure 4.6.3a	Outcome of Evaluation of Model 3	56
Figure 4.6.3b	Architecture of Model 3	56
Figure 4.6.3c	Details of Model 3	57
Figure 5.2	Frame Acquisition	59
Figure 5.2b	Mounting Camera View 1	60
Figure 5.2c	Mounting Camera View 2	60
Figure 5.3	Frame Captured by Camera	61
Figure 5.4a	Left Elbow Heatmap	61
Figure 5.4b	Left Shoulder Heatmap	61
Figure 5.4c	Right Shoulder Heatmap	62
Figure 5.4d	Right Elbow Heatmap	62
Figure 5.4e	Right Wrist Heatmap	62
Figure 5.4f	Head Heatmap	62
Figure 5.4g	Neck Heatmap	62
Figure 5.4h	Right Wrist Heatmap	62
Figure 5.4i	Body Part Location	63
Figure 5.4j	Generate Human Pose Frame	64
Figure 5.5	Predicted Output	64
Figure 6.3a	Frames Obtained from Raspberry Pi	70

LIST OF TABLES

Table Number	Title	Page
Table 2.8	Parameter and Output of Hidden Layers	15
Table 4.3	Comparison at Different Location	44
Table 4.4	Comparison of Different Raspberry Pi Model	45
Table 4.5a	Comparison of Human Pose Estimation Framework	46
Table 4.5b	Output from Different Conditions	47
Table 3.1.5	Result from each Models	58
Table 6.1	Verification Plan	65
Table 6.2.1	Detection Results 1	66
Table 6.2.2	Detection Results 2	68
Table 6.3	Confusion Matrix on Classification Results	71

LIST OF ABBREVIATIONS

<i>CCTV</i>	Close-Circuit Television
<i>RGB</i>	Red, Green, Blue
<i>Wi-Fi</i>	Wireless Fidelity
<i>CNN</i>	Convolutional Neural Network
<i>GMM</i>	Gaussian Mixture Model
<i>R*CNN</i>	Region-Convolutional Neural Network
<i>MV-CNN</i>	Multi-View Convolutional Neural Network
<i>JDA</i>	Joint Data Augmentation
<i>GPU</i>	Graphic Processing Units
<i>ADAS</i>	Advanced Driver Assistance System
<i>SVM</i>	Support Vector Machine
<i>SURF</i>	Speeded Up Robust Features
<i>KNN</i>	K Nearest Neighbor
<i>HOG</i>	Histogram of Oriented Gradients
<i>USB</i>	Universal Serial Bus
<i>CPU</i>	Central Processing Unit
<i>GPIO</i>	General-Purpose Input / Output
<i>PAF</i>	Part Affinity Fields
<i>NMS</i>	Non-Maximum Suppression

CHAPTER 1: INTRODUCTION

1.1 Problem Statement

Nowadays, most of the traffic accidents were reported to be caused by secondary tasks performed by the driver while driving at the roadside. Cdc.gov (2019) report that in 2012, 421,000 people face traffic accidents due to distracted drivers. Drivers usually tend to play their mobile phone while driving at the roadside such as texting to a smartphone, answering phone calls, surfing internet, and using social application such as Facebook, Instagram, Twitter, and etc. It is because most of the driver especially officers need to keep contact with their customer even though they are not in their working hour. Moreover, adjusting car radio also consider a dangerous act which will also catch attention of driver away from primary driving task (Lee et al., 2018). Furthermore, a driver might consume food and take a drink while driving which is also one of the causes of a traffic accident. The accident risk had increased when eating during driving after simulation with some participants. Nowadays, most of the fast-food restaurants featuring the welcoming drive-thru windows all around the part of the world. Most of the driver choose not to waste their time looking for eateries hence drive-thru becomes their major choice, and this led to many drivers choose to consume their food on the way to go, especially highway drivers which are more dangerous compared to normal drivers as the driving speed can reach 110KM/H. Rolison et al. (2018) report that young female driver had 71.57% involve in traffic accident due to distraction driving.

1.2 Background and Motivation

According to Malaymail.com (2019), 281,527 road accidents had been recorded in Malaysia within the first six months of 2019, which had been increase by 2.5 per cent compare to last year. Besides Malaysia, according to Who.int (2018), about 1.35 million people die due to road traffic crashes annually. And one of the factors that cause road accident was distracted driving, whereby the distraction caused by smart phone is the most critical part among all driving distraction. But the distracted driving is difficult to track since most of the driver will not admit that they are performing secondary task (Lawrence, 2018).

Human action recognition is a computer vision technique that has the ability to identify, recognize, and classify an action done by a human. Human action recognition from video is one of the most popular subjects of research nowadays due to the emergence of deep learning. Normally, a frame of video which is the image will be extracted out to obtain its meaningful features such as body, head and hand movement. After that, those features will be used to perform some kind of classification to determine the action performed by human based on the features.

Basically, human action recognition technique had become one of the most popular research that is interested in many system designer and researcher. It is because nowadays, design a system that implements to the camera which could help to monitor human activities is far more efficient than hiring some people to monitor through close-circuit television (CCTV). In this era, high-quality camera and computer hardware are very popular around this world with low cost, hence it is easy to get even for low salary worker and implement with system to bring better quality of life. Human life quality can be improved by implementing many convenience systems in many areas. Human action detection work had been explored in the past few years within video. For example, Wang et al (2016) propose RGB difference and warped optical flow modalities for the human action recognition.

Nowadays, most people will own their personal vehicle and get to the road. However, due to the improvement of the technologies, smartphone plays an important role in our daily life whereby people keep interact with their mobile phone even though during driving. The

Chapter 1: Introduction

interaction with smartphone during driving information as secondary task which might draw driver attention away from driving primary task, hence it increases the number of traffic accident. According to Motus (2018) stated that the number of traffic accident had increased 12.3 per cent due to smartphone working during driving.

However, the development of Advanced Driver Assistance system (ADAS) had shown potential to help people to avoid from getting themselves involved in unnecessary traffic accident by assist driver in different kind of ways. This system not only can provide vital information to driver but also can monitor driver fatigue and distraction of driver to give alert.

Nowadays, most of the vehicle provide more useful features to the driver to reduce their workload on driving such as a built-in Wi-Fi hotspot and camera that can show outside the vehicle. However, most of the vehicle does not implement the system that can recognize driver action and give alert to the secondary task performed. In most of the system even through automated vehicle, this feature does not include as part of the vehicle function. Hence, monitoring a driver's action is important to make sure that they are always ready for the driving task. In these few years, most of the system developed is more concern on the driver's fatigue by monitoring their face. Thus, a driver's action monitoring system was required to make sure that not only the face but the driver secondary task is also a point to give monitor.

1.3 Objectives

The main objective of this project is to design and develop a real-time driver action monitoring system as it will recognize driver's tasks performed and give alert to driver when the tasks performed are considered as secondary tasks to decrease the car accident.

This project's objectives can be divided into following sub-objectives:

1. To record down driver action in real-time through the camera.
 - The camera should be able to keep taking the frame of driver clearly to observe the driver activity.
2. To do the classification of driver's action accurately in real-time.
 - The system should straight do the recognition of driver's action from the frame taken from the camera with high accuracy.
 - The system should be able to classify driver secondary tasks perform such as texting on smartphone, answering phone calls, eating, adjusting radio player, and makeup.
3. To give alert to the driver when tasks performed is dangerous.
 - The system should give alert to driver by buzzing the buzzer when the secondary tasks performed is too long or is consider dangerous that will influence to the traffic safety.

1.4 Report Organization

There are seven chapters included in this report, whereby introduction is the first chapter, followed by literature review, system design, system analysis, implementation, system testing and also conclusion. Chapter 1 which is introduction of the proposed project. The problem domain that is related on this topic, some background information and motivation towards this project, as well as the objectives that is going to be achieve in this project was included in chapter one. While chapter 2 consist of all literature reviews regarding on the previous works that are similar or may be helpful for this project. Next, chapter 3 will present the design of the proposed system. It will briefly describe in details how to rebuild the system by explain in details the system design, hardware and software requirement as well as the method to setup the system.

Besides that, chapter 4 will describe some analysis regarding to the system for example like setting up the camera in different location and train different model to do the analysis for selection the model that proposed in chapter three. Chapter 5 will consist of implementation, where the system is conducted inside the vehicle and the process and output from each block of system design will be explained. The purpose of this chapter is to shown clearly the output that is going to be obtain from each block step-by-step by reaching the final output.

Chapter 6 which will conduct some system testing for this proposed system to test out its accuracy and performance. Lastly, the last chapter which is conclusion will summarize of all over the project and further development that can be made.

CHAPTER 2 LITERATURE REVIEW

2.1 Driver Action Recognition using Convolutional Neural Network

The ever-growing traffic density nowadays caused number of road accident increase a lot. This paper proposed a convolutional neural network (CNN) architecture to represent and recognize driver action, whereby this architecture aims to build a high-level feature representation from low-level input, whereby the high-level feature will be extracted from raw input image (Yan et al., 2016). This proposed approach was evaluated on SEU dataset as well as Driving-Posture-atNight and Driving-Posture-inReal datasets that created purposely using UWISH UC-H7225 infrared camera to solve different road conditions. The architecture of this proposed paper is shown in Figure 2.1 below.

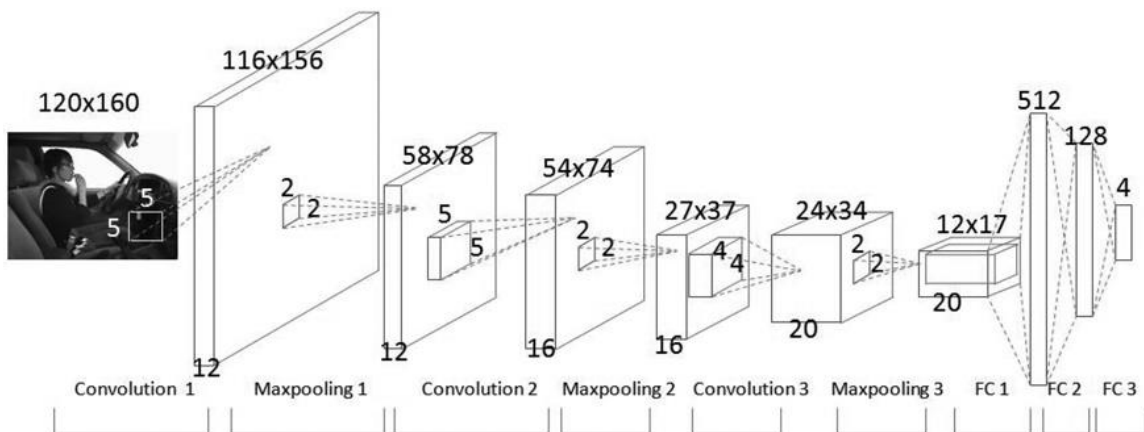


Figure 2.1: Architecture of Proposed CNN Network Contains Three Stages (Yan et al., 2016)

The CNN architecture consists three convolution stages with three fully connected layers that end up with softmax layer to classify four different classes. The final results obtained an accuracy of 99.47% on SEU dataset, 99.3% on Driving-Posture-atNight and 95.77% on Driving-Posture-inReal dataset. This prove that the overall results obtained were better than approaches that using hand-coded features.

2.2 Skin-Like Regions

In another paper, skin-like regions from an image are extracted by Gaussian Mixture Model (GMM) and pass to a CNN model which is called region-convolutional neural network (R*CNN) to classify driver's action (Yan et al., 2016). In the beginning, the full image will be pass to GMM which had been trained with different skin samples. After that, skin-like regions will be generated and pass as the second region for the conventional R*CNN which are more informative compared with selective search in R*CNN. Lastly, the R*CNN will process the first region which is the full image together with second region to do the classification. Figure 2.2 shows the flow of image going through these processes.

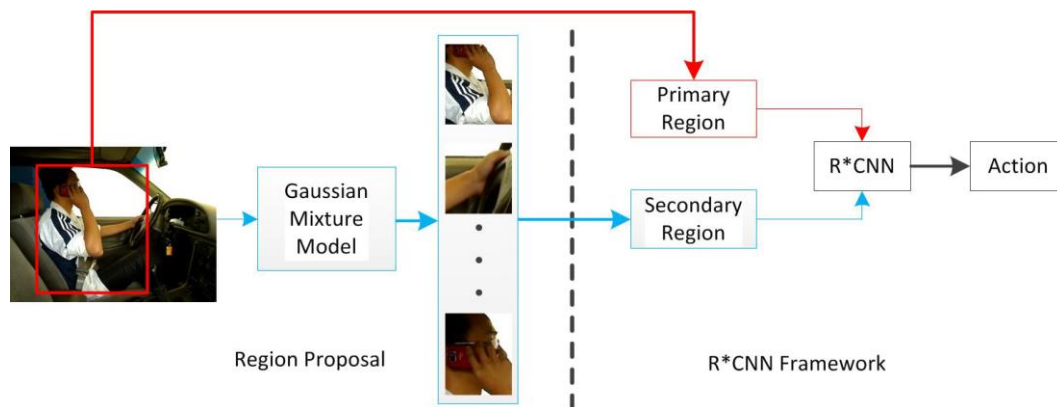


Figure 2.2: Implementation of Gaussian Mixture Model for Skin Modeling (Yan et al., 2016)

2.3 Merge Multiple Layer Output in CNN

Figure 2.3 shows the trainable deep framework of DedistractedNet that proposed by Pang et al (2018) used to recognize the driver behaviors from an image which directly profiles the features from the input image based on CNN. DedistractedNet consists of five sets of convolutional layer, after performing each layer, the subsequent max-pooling layers will be conducted to offer invariance to decrease the resolution of activation maps. Feature maps from convolutional layer and max-pooling layers will be associate to train the DedistractedNet.

The output from P3 and P4 will merge together with output from C2 and C4 to become a new input to 5th convolution layer which this method can further improve the accuracy. At last, after output from P5, first fully connected layer will have 512 neurons whereby at the last stage which is Softmax will classify the driver action in one out of 8 classes.

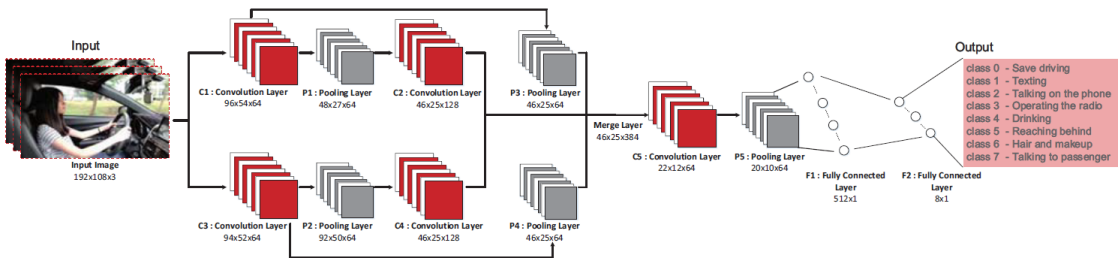


Figure 2.3: Framework of the proposed DedistractedNet (Pang et al., 2018)

2.4 Six-Axis Motion Processor

Apart from that, Zhang et al (2019) propose a method to recognize vehicle driving behavior based on six-axis motion processor. Since the sample size is small and easy to achieve overfitting, hence a joint data augmentation (JDA) scheme is proposed with a multi-view convolutional neural network model (MV-CNN). The experimental system includes data acquisition and receiver, data augmentation, and deep neural network model structure. In the data acquisition module, it uses the MPU-6050 which is a six-axis sensor to transfer the data through Wi-Fi. In the data augmentation processing, multi-axis weighted fusion algorithm, background noise fusion algorithm, and random cropping algorithm that come together to form the JDA are processed.

In the last part, the features will go into MV-CNN for the recognition of driving behavior. First, the feature map is sliced from different views which consist of H, W, and C that stand for the number of rows, columns, and channels respectively. Down and Up module split into H slices while Right and Left split into W slices. The feature maps with size of H x W x C extracted from four views are next concatenated with the original input to compile a new feature map having H x W x 5C size. Then the new feature maps will pass through the CNN, flatten it into vector, fully connected layer (FC), as well as the SoftMax function to do the classification. Figure 2.4 shows the structure of MV-CNN.

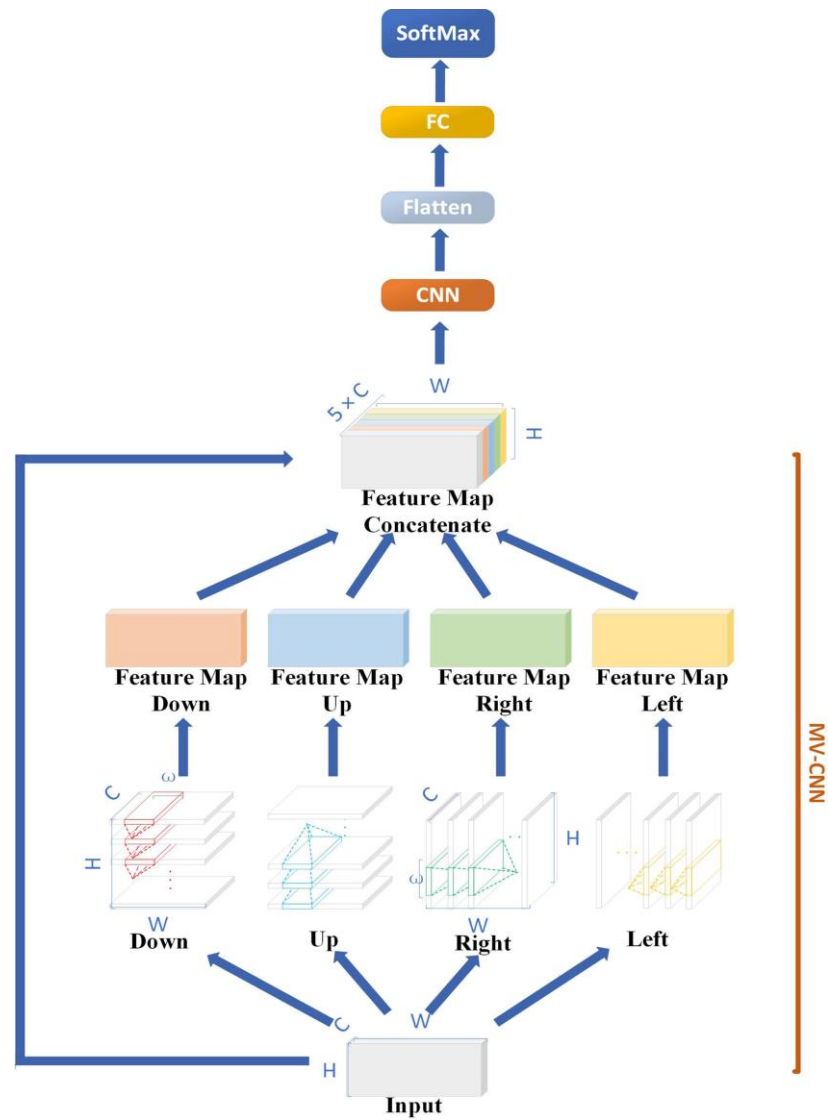


Figure 2.4: Structure of MV-CNN (Zhang et al., 2019)

2.5 Action Detection Using Pose Estimation

Huang et al (2018) proposed a paper by computing temporal pose features with a 3D CNN model to action recognition from videos. The pose estimation result will be generated from multi-person pose estimator, then some pre-processing procedure is proposed to utilize the multi-channel human joint position maps. After that, they the extent of optical flow stacking CNNs to human-joint-part stacking CNNs to better capture both spatial and temporal clues. Figure 2.5 shows the flow diagram of pose-based 3D CNN model.

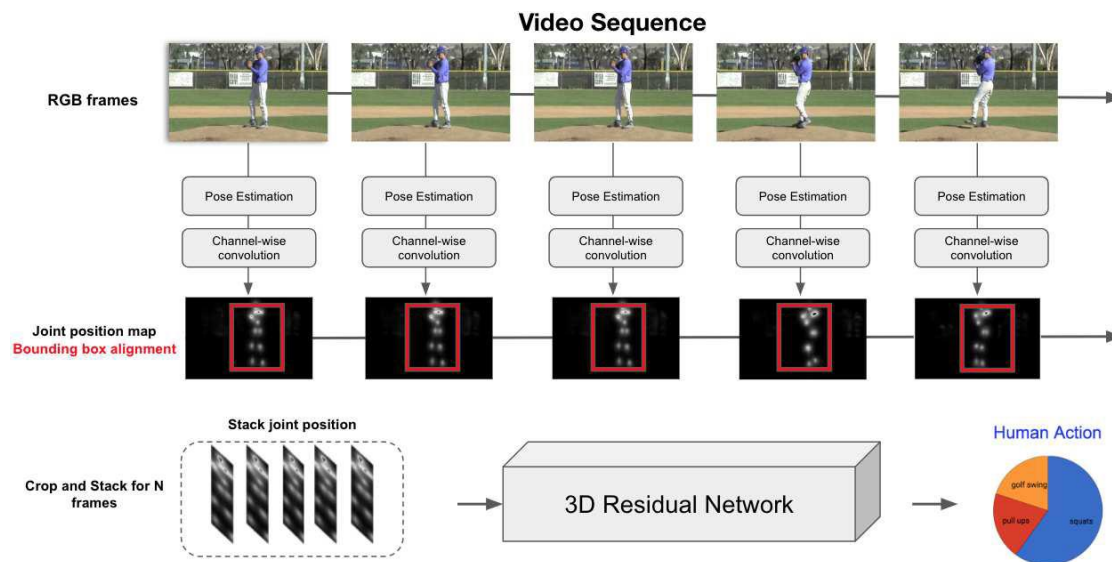
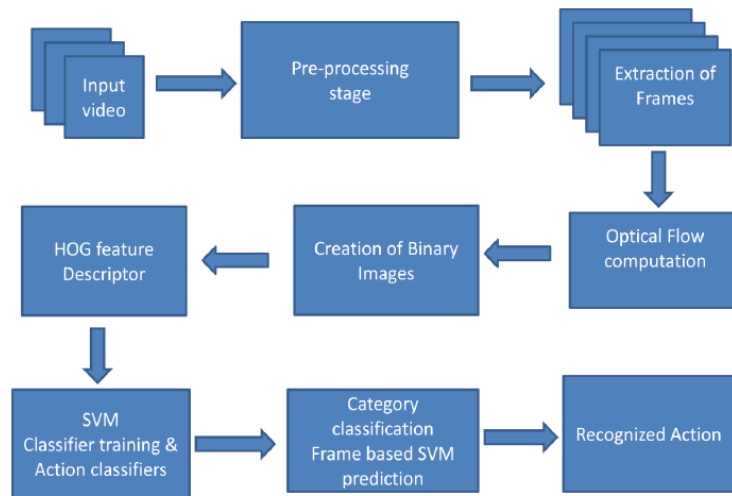


Figure 2.5: Flow diagram of pose-based 3D CNN model (Huang et al., 2018)

2.6 Optical Flow and SVM Classifier

Jagadeesh and Chandrashekar (2016) propose a method to recognize the human action inside the video whereby the model is trained using Kungliga Tekniska Hogskolan (KTH) dataset. At the beginning, the frame will be extracted from video and follow by Lucas-Kanade for optical flow computation. Optical flow is used to describe how an object or person between two consecutive frames from a video change due to the motion between scene and the camera. The next step is converting the data obtained from optical flow into binary image to allow histogram of oriented gradient (HOG) descriptor to process the next step which is feature extraction. The features will encode local shape information from regions inside an image which can be used for classification. Lastly, the extracted features will fit into the trained support vector machine (SVM) model for classification of action. Figure 2.6 show the block diagram of the proposed framework.



*Figure 2.6: Block Diagram of Human Action Recognition using Optical Flow and SVM
(Jagadeesh and Chandrashekar, 2016)*

2.7 SURF Key Points

In this paper, an approach is proposed to deal with the distracted driving tasks using the speeded up robust features (SURF) to detect the key points, follow by histogram of gradient (HOG) to extract the features and k nearest neighbor (KNN) to do the classification using the extracted features (Jegham et al., 2018). The flow of the proposed framework is shown in Figure 2.7.

First of all, the safe driving frame will first be captured and keep to compare with later frame. After that, the frame contains the driver next action will be acquired. The SURF key points will then be detected from the first safe driving frame and the next frame capture from camera. Then, the common key points of both frames will be detected by comparing the similarity. After getting the common key points, it will then eliminate those points from the distracted driving frame which left only the points that will be consider as distracted action key points.

Now it will filter the key points such that strongest metrics points will be remain and other will eliminate as noise. The body part segmentation will be done based on the filtered key points. From the segmented image, histogram of oriented gradient (HOG) will be applied to extract the features and the extracted features will be fit into KNN model for classification.

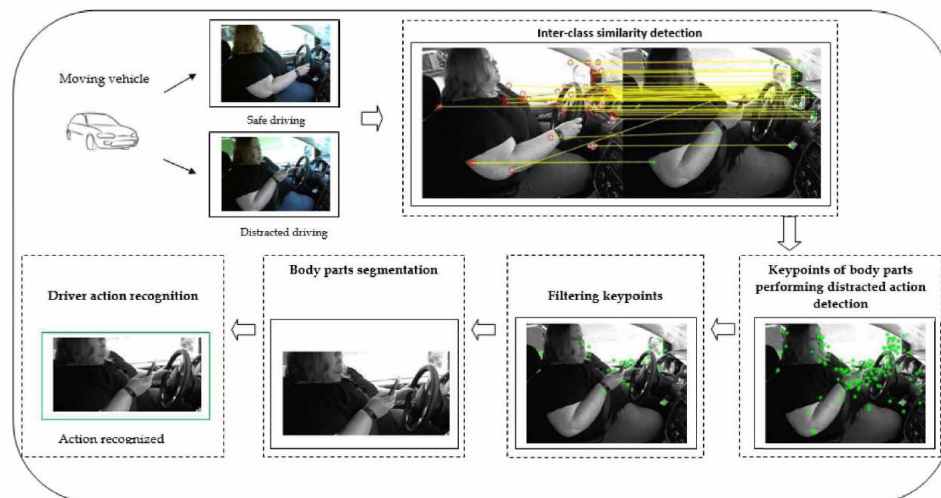


Figure 2.7: Flow of Driver Action Classification using SURF Key Points (Jegham et al., 2018)

2.8 Skeleton Motion History

This paper will propose a system to classify human action in three parts (Phyo et al., 2019). In the first part, the system first generates skeleton image whereby the human parts are generated using Microsoft Kinect V2 sensor. The sensor will first generate 25 body parts, using the part, the system can link all the parts together to form pairs which describe as skeleton image.

After that, the system will take 4 frames before and after the current frame, which sum up all together having 9 frames to perform binary OR-operation to create Skl MHI. All frames will generate the skeleton image first before perform the operation. After that, the skeleton region will be extracted and normalized into 62 x 62 size. Figure 2.8a show the flow of creating the Skl MHI.

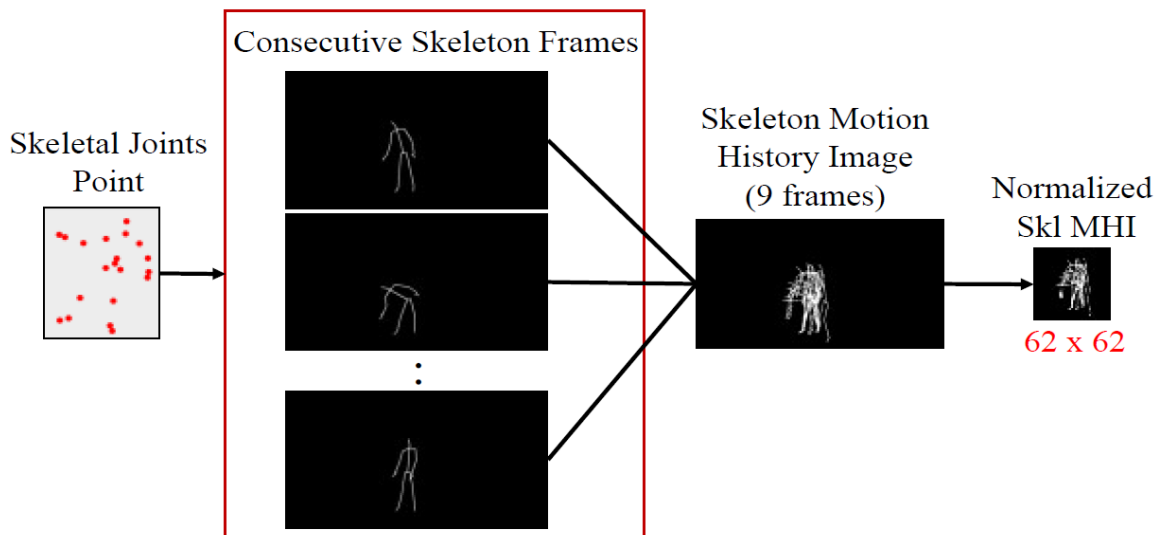


Figure 2.8a: Flow of Create Skl MHI (Phyo et al., 2019)

After getting the Skl MHI, it will be used to do the prediction using deep learning model called 2D-DCNN whereby the model architecture is shown in Figure 2.8b. The parameter and output of all hidden layers is shown in Table 2.8. This CNN model achieves classification accuracy of 97.82%, which is consider performs very well.

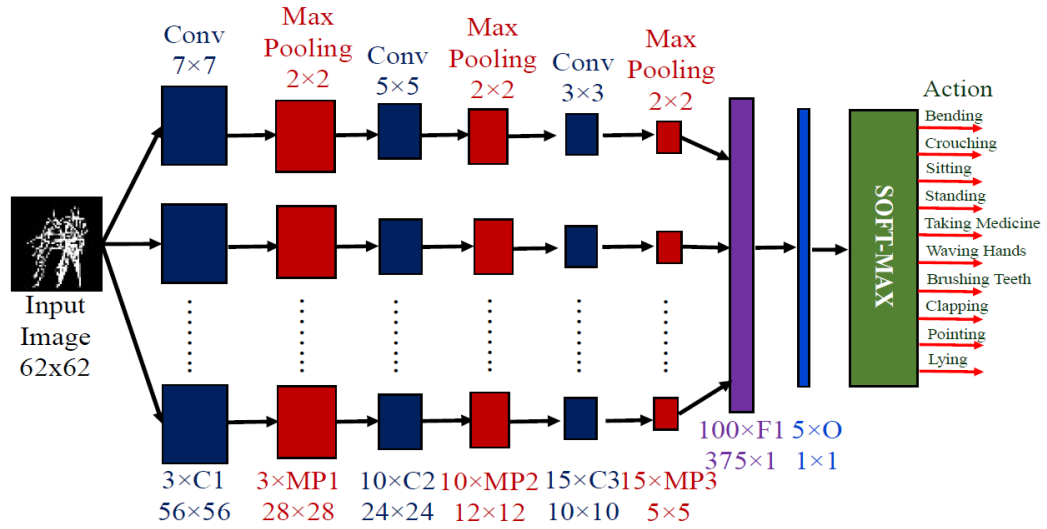


Figure 2.8b: Model Architecture of 2D-DCNN (Cho Nilar, Thi Thi and Tin, 2019)

Table 2.8: Parameter and Output of Hidden Layers (Cho Nilar, Thi Thi and Tin, 2019)

Layer	Filter Size	Filter Type	Feature Maps
C1	7X7	Gabor	3
MP1	2X2	-	3
C2	5X5	Gaussian	10
MP2	2X2	-	10
C3	3X3	Gaussian	15
MP3	2X2	-	15

2.9 Human Pose Estimation

OpenPose framework which is a multi-person pose estimation will be used to help for generate the human pose frame (Cao et al., 2017). The architecture for the proposed model is shown in Figure 2.9a. First of all, it will go through the features extraction layers which is the VGG-19. After that, it will split the next part of network into two parts whereby these two branches will predict different things. The first branch will predict a set of 18 confidence maps, since the model will estimate 18 different part of the human body based on the COCO dataset, hence it will come out with 18 different maps which representing different part on the particular human body, it is called heatmap. The outcome from the second branch will be 38 different output which represent the degree of association which is metrics that give information about position and orientation of pairs, it is called part affinity fields (PAF). To simplify this branch, it will show that which two parts can be combine to become a pair. This network is having multiple stages so each stage will refine the output come from previous stage.

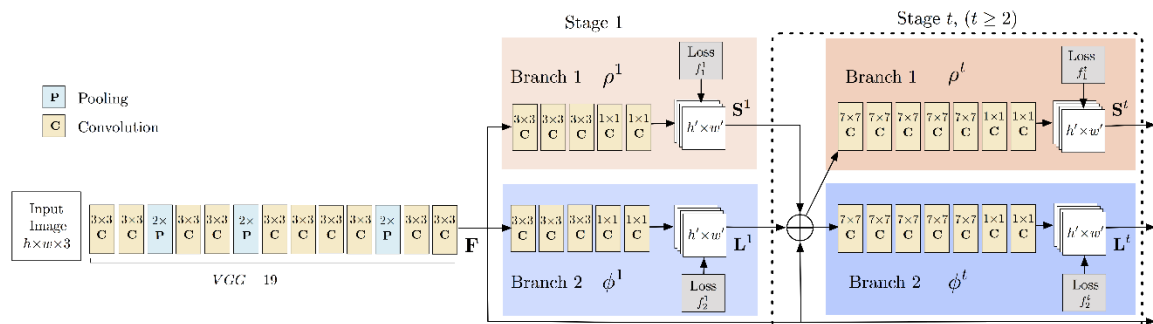


Figure 2.9a: OpenPose Model Architecture (Cao et al., 2017)

The next step is extracting the body parts which is the local maximums in the heatmaps generated from branch 1. Non-maximum suppression (NMS) algorithm will be used to find those peaks which is the body part. After getting all the body parts, the next step is to connect the parts to form pairs.

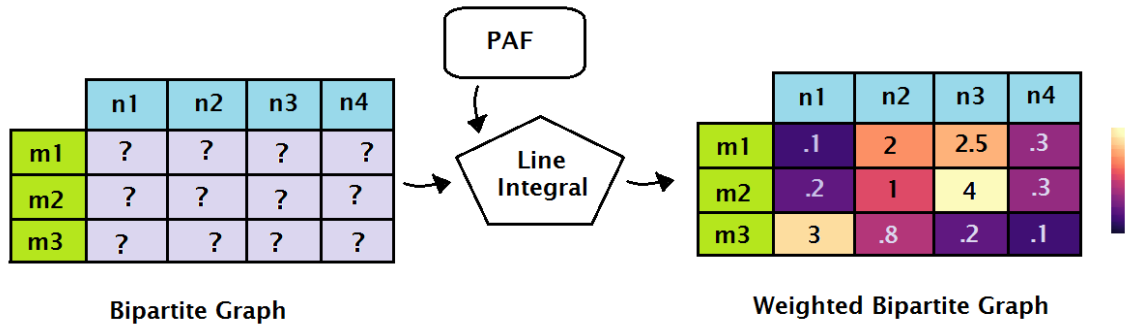


Figure 2.9b: Line Integral Algorithm

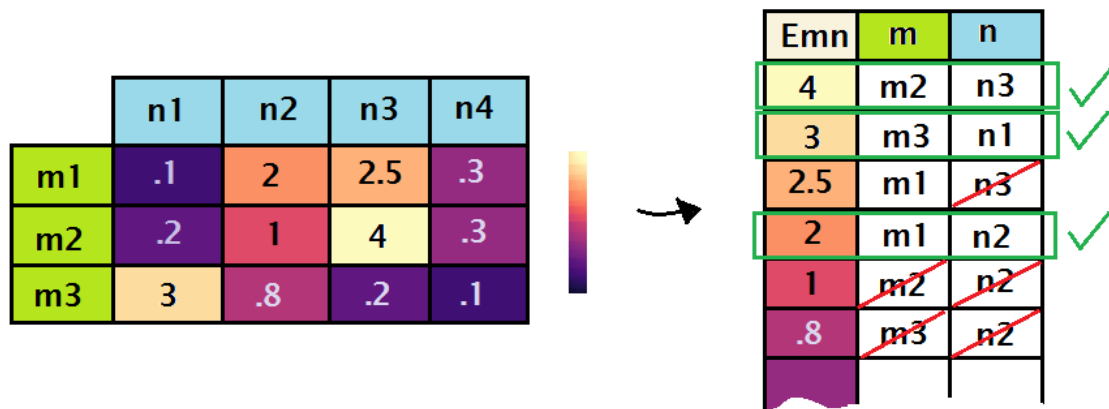


Figure 2.9c: Assignment Algorithm

PAF generated from branch 2 will help to find which pair will give correct estimation. Line integral will be used to find the connection that have the highest value. Here it will create a weighted bipartite graph that display all possible pairs between each two parts from each humans and record down the score for each connection as shown in Figure 2.9b. For the connection have the highest value which means that pair have the strongest bond, which that pair will be selected and eliminate others. To achieve this, sort each connection by score from maximum to minimum, greedy approach will be applied to find maximum for each of the pair as shown in Figure 2.9c.

The last step will get all different pairs which can be used together to create the human skeleton. Consider that each connection belongs to different human, then in the merge will try to see if both pair having one common vertices then it will be considered as same human and merge both set of humans into one human and remove another one. The merging algorithm is shown below.


```

if  $H_1 \cap H_2 \neq \emptyset$ 
then
 $H_1 = H_1 \cup H_2$ 
delete( $H_2$ )
    
```

After finishing all the step stated, the output will be the collection of human sets whereby each set consisted of detected key points coordination. The final step is to link all the key points with a line based on the key points pair for example like shoulder will link to elbow. The human pose is generated based on the connection of key points on a new blank frame. Figure 2.9d show the overall pipeline of OpenPose.

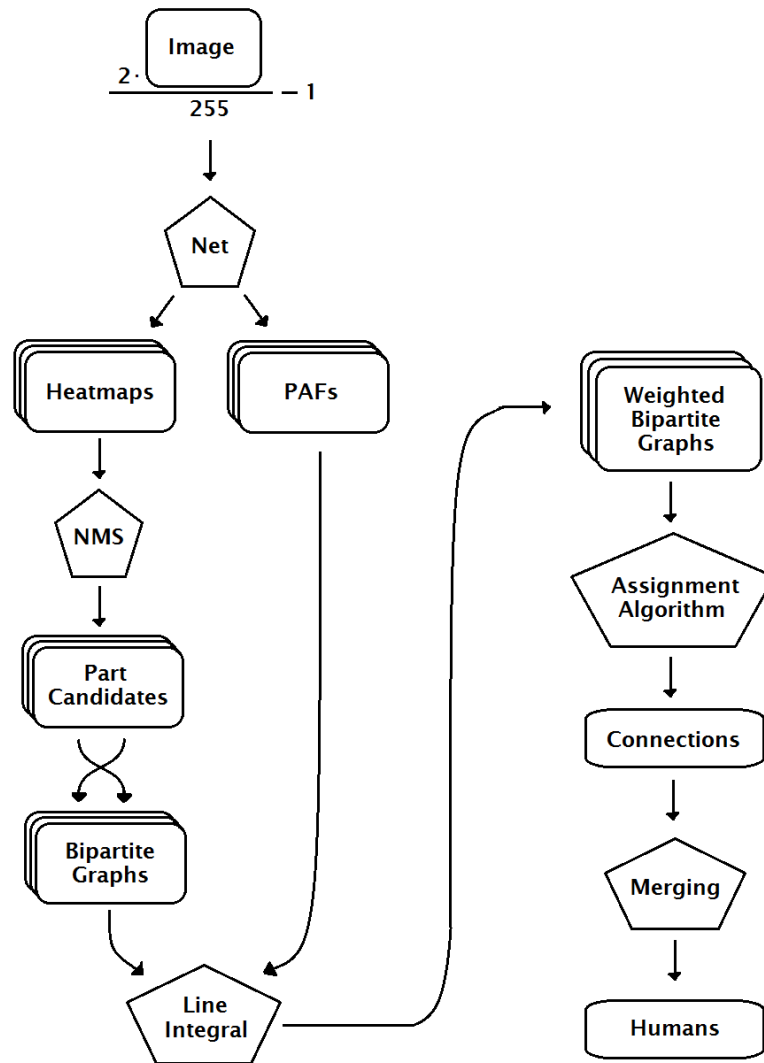


Figure 2.9d: OpenPose Pipeline (Cao et al., 2017)

2.10 Human Action Detection Using Raspberry Pi

In this paper, two different stages of tasks of approach for implementation of activity recognition using 3D-Convolutional Neural Network (3D-CNN) (D'Sa et al., 2019). In first stage, the work for activity recognition will be done and for second stage, the work will be further extended into Raspberry Pi. First of all, background subtraction using static reference image with no moving objects to identify the moving objects in foregrounds. Next, it will be pass to 3D-CNN network to make classification of human activity. Figure 2.10a show the architecture for activity recognition system.

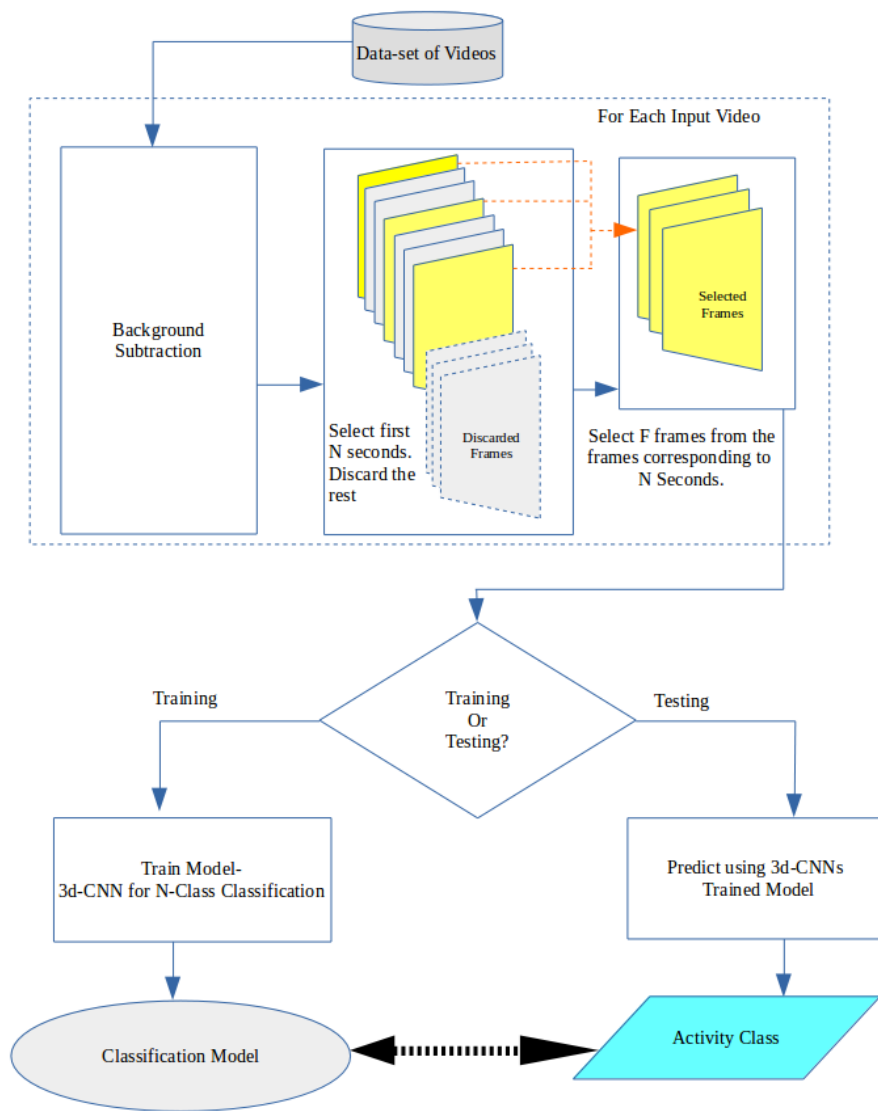


Figure 2.10a: System Architecture for Activity Recognition System (D'Sa et al., 2019)

The next stage is to extend the implementation by applied it on Raspberry Pi that can acts as mobile device for the task. The video of actions will be recorded using a camera module with specific time length, after that the input will be processed such that background will be subtracted. After that, few frames will be concatenated to be input for 3D-CNN network. Figure 2.10b show the framework for activity recognition in Raspberry Pi.

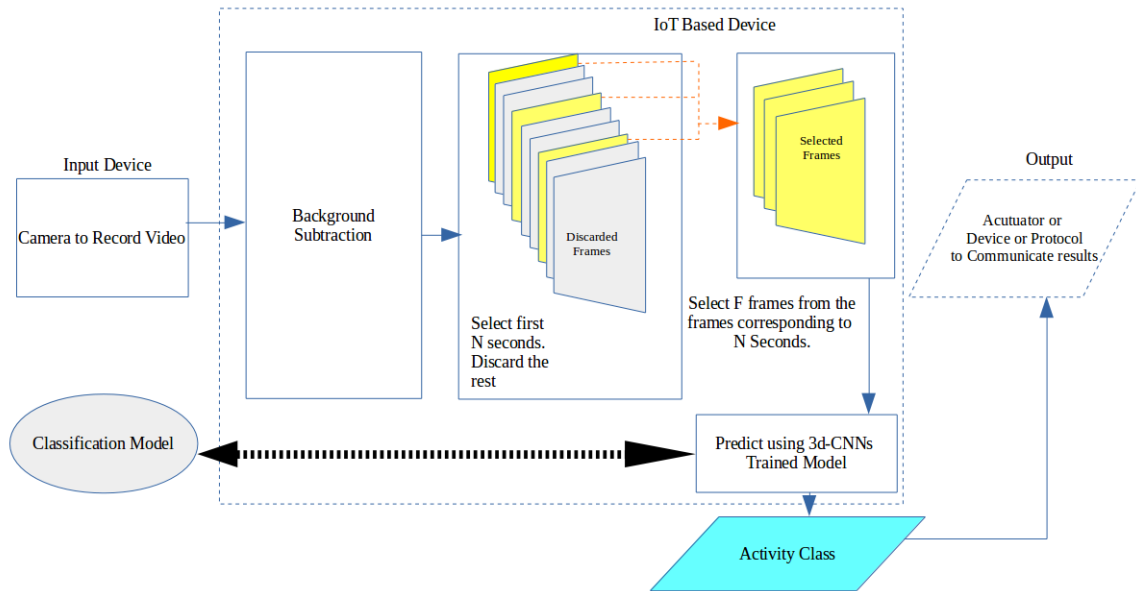


Figure 2.10b: Raspberry Pi Based Framework for Activity Recognition (D'Sa et al., 2019)

Chapter 3: SYSTEM DESIGN

3.1 Proposed System Framework

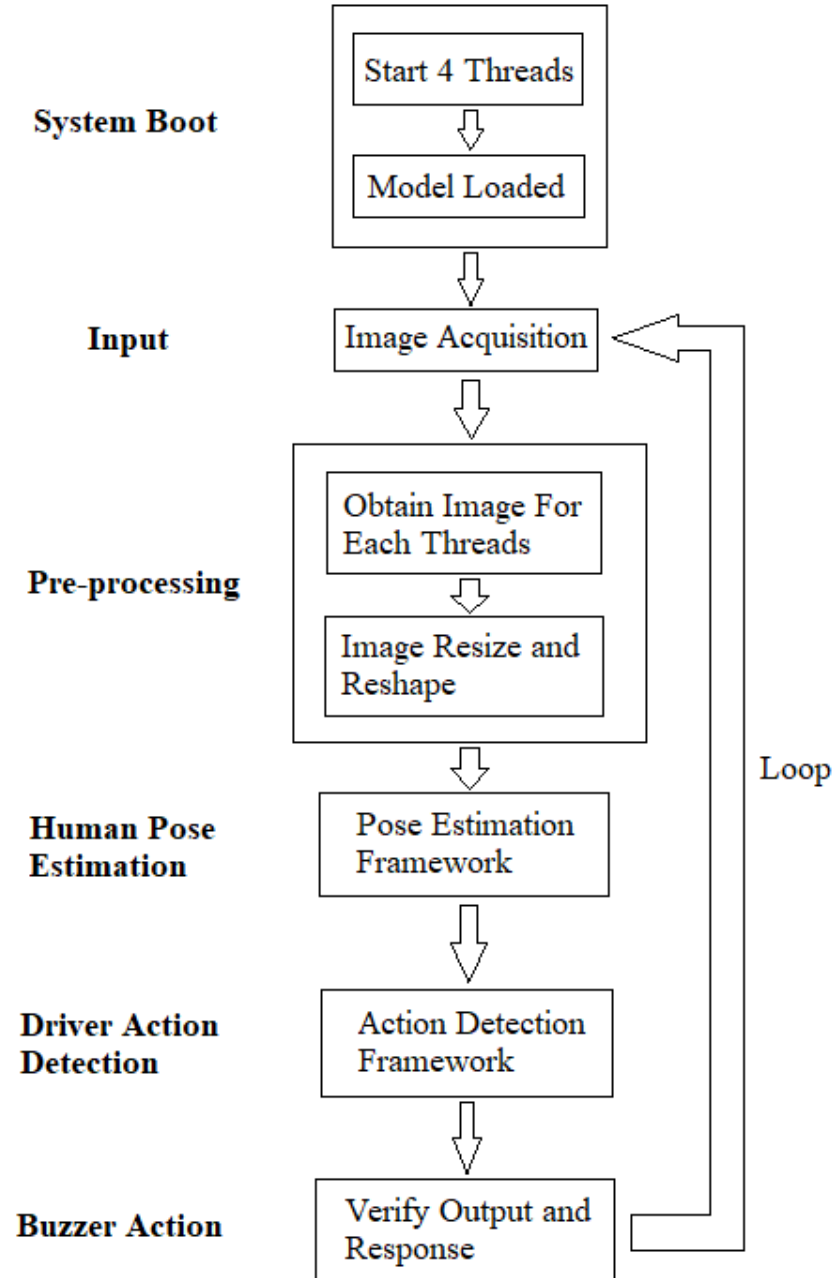


Figure 3.1: Flow Diagram of Proposed Framework

Chapter 3: System Design

This framework needs to be done by using Raspberry Pi development board loaded with Raspbian-OS, the device will be setting up inside the car with the Pi Camera, power supply is provided from car. This framework system is built using Python languages. The framework started when the Raspberry Pi is booted. First of all, 4 threads will be loaded whereby each of them will carry different models used to generate human pose frame. The image acquisition will be obtained from the Raspberry Pi Camera using the PiCamera module. After that, the image will undergo pre-processing part, whereby all the thread will obtain the same image for their individual process. The image will be resized and reshaped in order to fit into the pose generation model.

The pose estimation framework will detect human body key points and link all the points on a blank image to produce a human pose frame. The next part will be action detection framework, whereby at this stage the human pose frame collected from previous stage will be used as input into a CNN classification model. This model consists some convolutional, batch normalization, rectified linear unit (ReLU), max pooling and lastly fully connected with Softmax classification. The output from this model is a numpy array which consisted of 10 items whereby each item represents the score for each class of action, use it to get the predicted action class. The last step of this framework is to verify whether the action is considered as normal driving or not. If driver currently performing secondary tasks, the alert will be given from buzzer. The framework will get another frame from the PiCamera after everything had been done and the process will continue goes on. Figure 3.1 show the summary of the proposed framework.

3.1.1 Step 1: System Boot

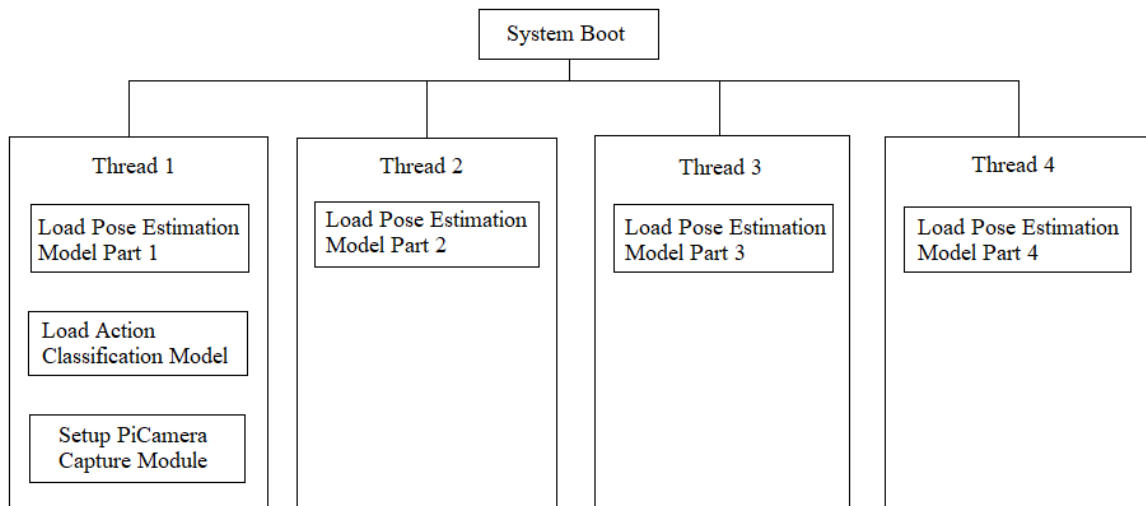


Figure 3.1.1: System Boot

When the system is being booted in the Raspberry Pi operating system, 4 threads will be setup and each of them will loaded with their respective pose estimation model. The purpose of doing this is because Raspberry Pi processor having 4 cores, and the Raspberry Pi does not have GPU. Hence it might be difficult for this development board to run a deep learning model in faster speed. That is the reason to load different model in different core whereby each model with smaller size can used to classify certain work using CPU. Although this method might not as fast as use GPU, but it can solve the problem of time consuming on a single CPU.

Thread 1 will be used as main function for this framework, where the action classification and PiCamera module will be loaded and setup at. After all the threads had been setup properly, thread 1 will started to read a frame from PiCamera module. Figure 3.1.1 show the flow of system boot.

3.1.2 Step 2: Image Acquisition

This section will obtain the video input from Pi Camera, make sure to enable to camera support in the Raspberry Pi to use the camera. The Pi Camera will first be set up inside the car, the camera will keep on capturing the video of driver action. Unlike OpenCV, Pi Camera require picamera library to obtain the frame from camera. First the camera needs to be initialized and grab a reference to camera capture via PiRGBArray. After that, the image can grab from the camera, the image can obtain via the camera capture's array function eg. cameraCapture.array.

3.1.3 Step 3: Pre-processing

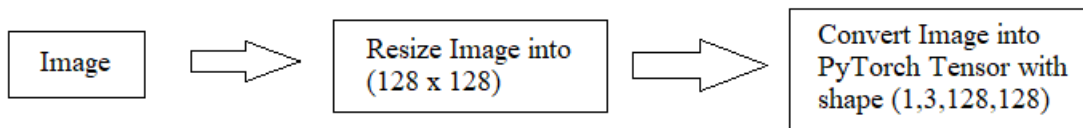


Figure 3.1.3: Image Pre-processing

This section is where the image acquired by each threads and input into their respective model. After thread 1 obtained the image, it will put the image into a global queue. Thread 2-4 will acquire the images from queue when queue is not empty.

Figure above show the step for image pre-processing. After the image was obtained, it will first be resized into shape of (128, 128, 3) which having width and height of 128 and channel of 3. After that, since the model is trained using PyTorch, the image in numpy array type had to be convert into PyTorch tensor. Firstly, use from_numpy function from PyTorch to convert the numpy array into PyTorch tensor. After that, permute (2, 0, 1) to change the shape into (3, 128, 128) which move the channel in front. Last step was unsqueeze function to convert the input into (1, 3, 128, 128) to be able to input into model.

3.1.4 Step 4: Human Pose Estimation

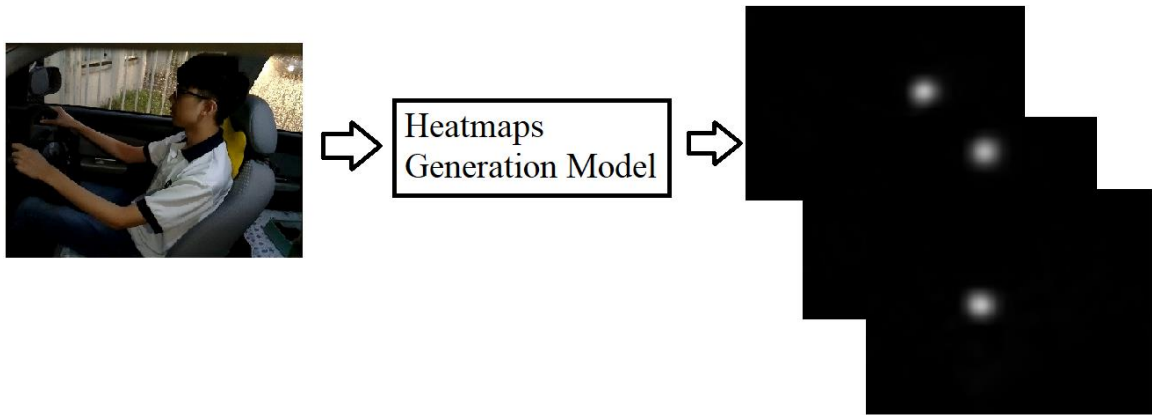


Figure 3.1.4a: Heatmap Generation Model

Figure 3.1.4a above show the overview of heatmap generation model. When a processed image is input into the model, heatmaps being the activation maps will be produced from the convolutional layer. A basic CNN model consists of multiple convolutional layers, with activation after each convolution, batch normalization layers, flatten in order for fully connected layers. However, for this project, flatten and fully connected layer is removed. Instead, a convolutional layer will be the last layer in order to produce the heatmaps. Figures 3.1.4b to 3.1.4e below show the architecture of model and details of each layers.

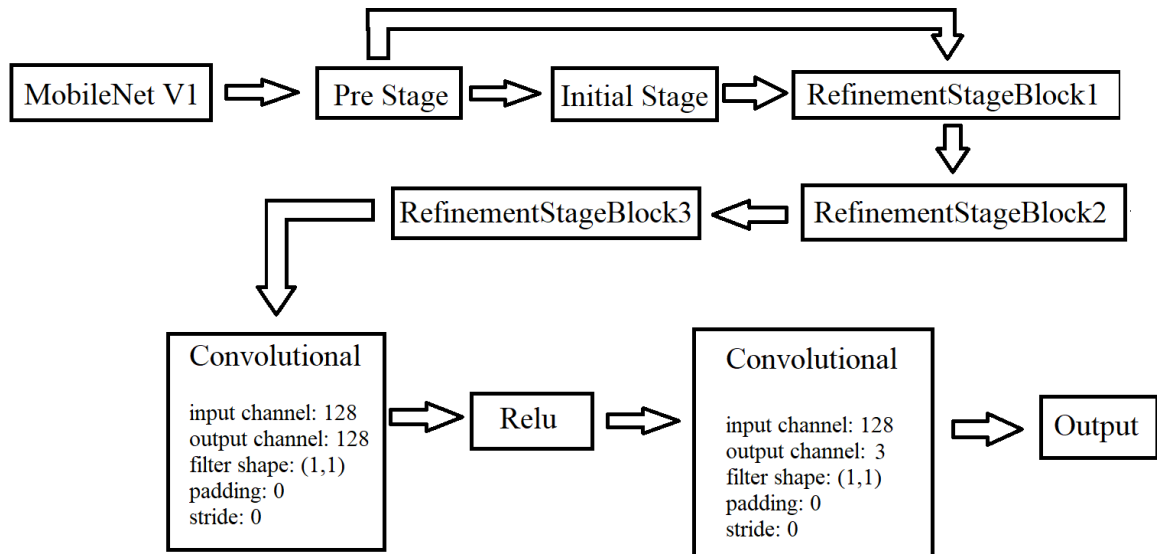


Figure 3.1.4b: Overview of Heatmap Generation Model

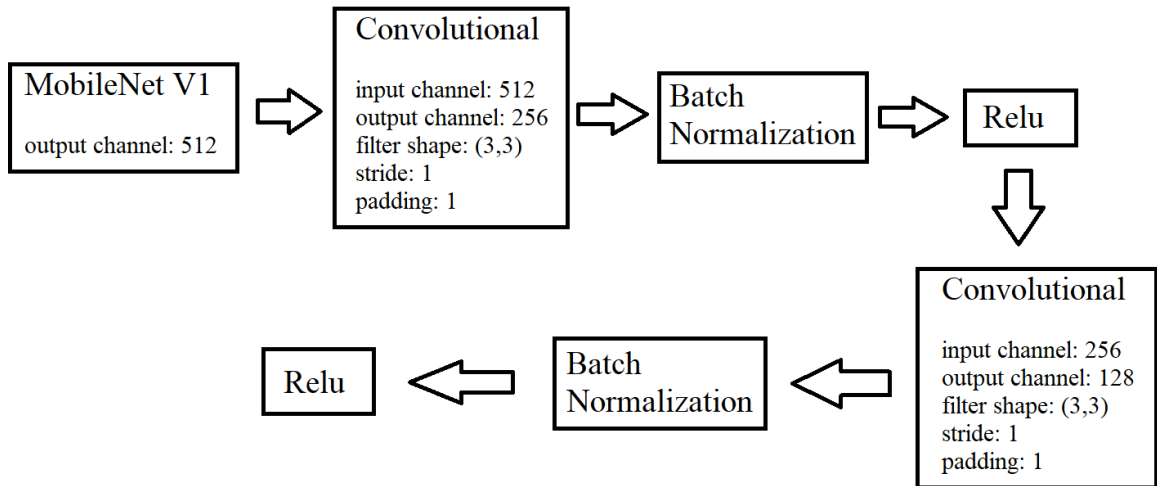


Figure 3.1.4c: Architecture of Pre Stage

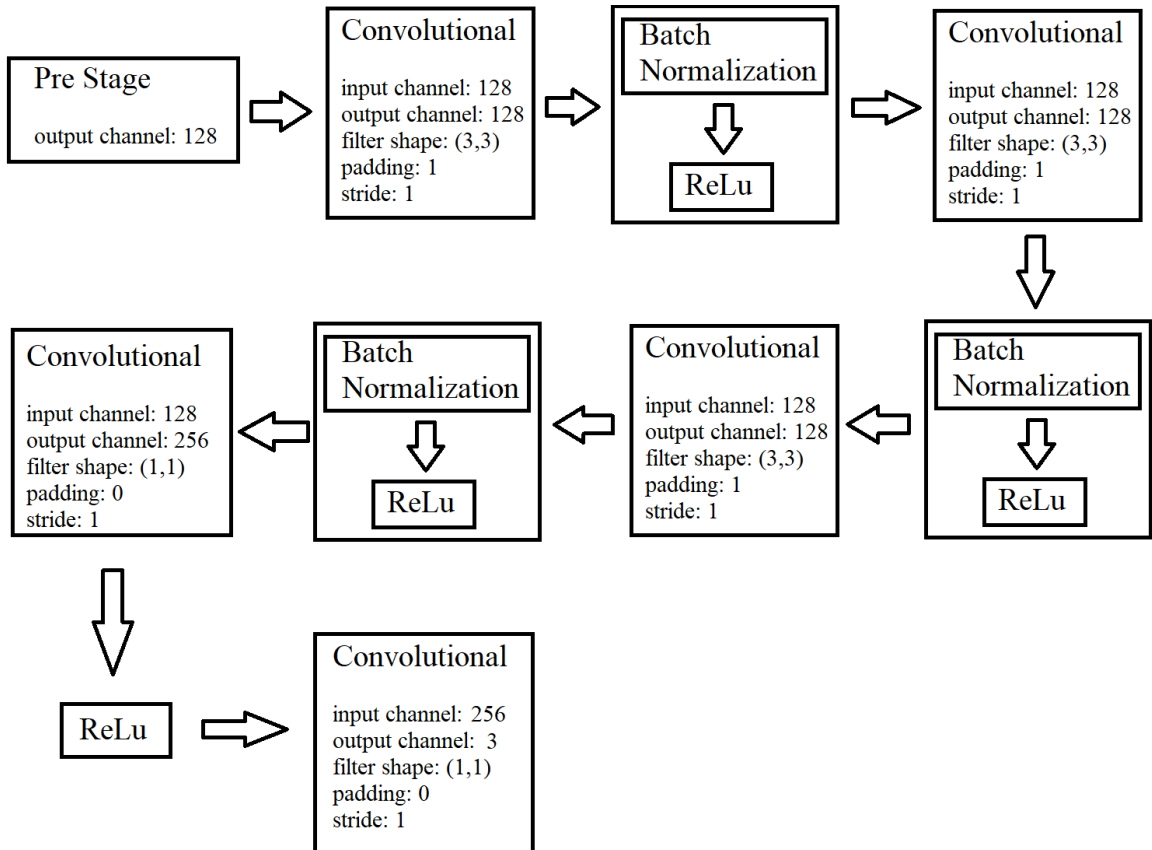


Figure 3.1.4d: Architecture of Initial Stage

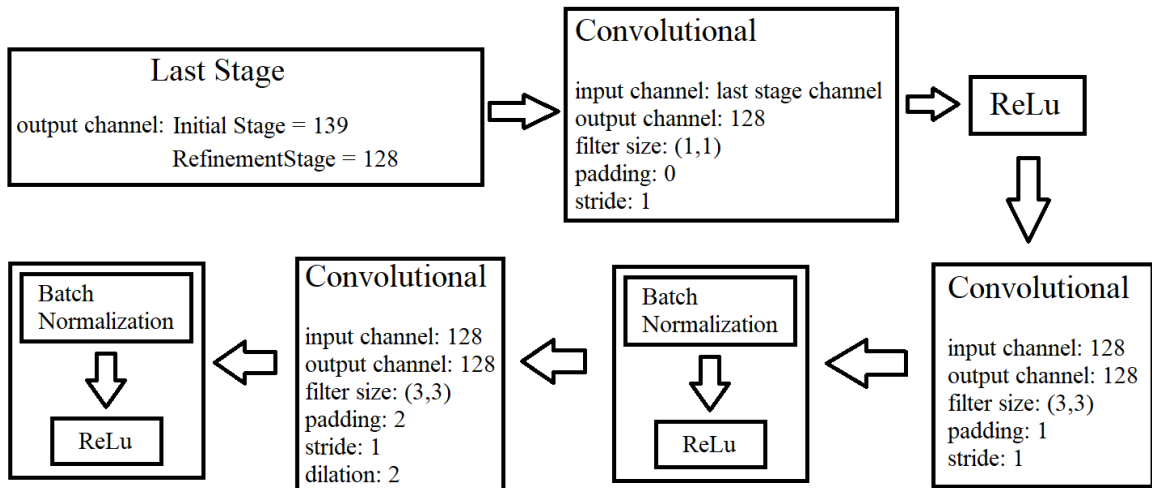


Figure 3.1.4e: Architecture of Refinement Stage

Figure 3.1.4b show the overview architecture of heatmap generation model. This model is trained using looking into person (LIP) dataset that provide many human images and the annotation is a list of human key points coordinate in that particular image. Where during training, the annotation will be converted into heatmaps in order to compute loss with output activation maps from the model. The input shape used at here is (1, 3, 128, 128). First of all, the input will go into the MobileNet V1 for feature extraction. At here the transfer learning method is applied by using the pretrained parameters of MobileNet V1. The MobileNet V1 will extract the features which produce deep feature maps, the output from MobileNet V1 is feature maps with shape of (-1, 512, 16, 16), next it will go into pre stage. The details of pre stage architecture is shown in Figure 3.1.4c. The purpose of this stage is to reduce the channel size slowly to prevent too large channel size that will cause the model to run slowly. The output from this stage having the shape of (-1, 128, 16, 16).

Next is the initial stage where the detail of the architecture was shown in Figure 3.1.4d, there will be three convolutional layers for producing better feature maps. After that, next convolutional layer will used to increase the number of channels. It is because after going through this convolutional layer, next one will straight produce an output of shape with (-1, 3, 16, 16). Which mean last convolutional will produce 3 heatmaps which represent

human key points, hence more channel can ensure the produced heatmaps having better quality.

After the initial stage, it will continue follow by refinement stage. The refinement stage will consist three RefinementStageBlock where the architecture is shown in Figure 3.1.4e. The purpose of refinement stage is used to produce a better heatmaps, the architecture of these block is shown above. The RefinementStageBlock purpose to further refine the features maps, with last output should be shape with (-1, 128, 16, 16).

Lastly, two convolutional layers with filter size of (1,1) will be used to create the heatmaps. First convolutional layer was just making the feature maps more deep, next layer will straight output it into channel size of 3 which is the human key point heatmaps. Without any flatten and fully connected layers, the feature maps will output to the system for further processing to extract out the key points axes in the next process.

The architecture described above is consider a very small network, which it is very difficult to detect all human key points from a single image. Hence this framework is using 4 threads, whereby each thread will output a feature map of (1, 3, 16, 16). In the output feature map, first and second channel is represented 2 human key points. There is total of 4 different models in 4 different threads, all models are being trained to detect different human key points. Model 1 is used to detect human neck and head; model 2 detect right wrist and right elbow; model 3 detect right shoulder and left shoulder; model 4 detect left elbow and left wrist.

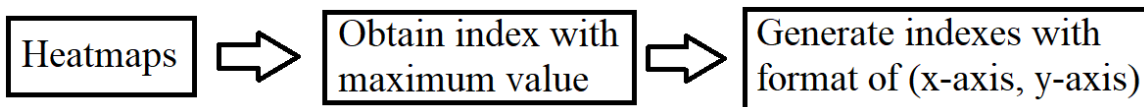


Figure 3.1.4f: Generate Key Points Indexes

The next step is to generate human key points axes from the output heatmaps. The summary of this part is shown in Figure 3.1.4f. For each heatmap, which is a type of numpy array will first be used to obtain the index with maximum value. The function used at here would be argmax from numpy, where it will return the index of the maximum value lie inside an array.

Chapter 3: System Design

After that, `unravel_index` function from `numpy` will be used, the parameter will be the index return from `argmax` and the shape of heatmap. The purpose of using this function is because the heatmap having a 2-dimensional array, and specific row and column were needed to draw line on image. The return from this function will be (row, column) which is similar to the x-axis and y-axis in a graph.

The last step for generating human pose is draw and connect the key points on the image. Since all key points axes already obtained, `OpenCV` function can be used to draw the key points on the image. After that, based on the pair initialize at the beginning, link all the key points for example like shoulder to elbow, elbow to hand wrist, head to neck and so on. The key points will be drawn on a blank image and this image will be used for classification driver action in the next step.

3.1.5 Step 5: Driver Action Detection

In this step, the driver action is classified based on the human pose by input the human pose frame into the classification model. The classification model architecture was shown in Figure 3.1.5 below. This model built by 3 stages of CNN layers and 1 stages of fully connected layers. The dataset used to train this model is from State Farm Distraction Driver Detection dataset.

First of all in the first CNN stage, the frame will go into the first layer which is convolutional layer. In this layer, 32 filter with kernel size of 3×3 is used. The activation function used at here is ReLu which is called rectified linear unit, the purpose of using this activation function is to identify all negative values and change it into value zero to speed up the training. After the convolutional layer, next layer would be batch normalization to normalizes output of the previous layer to speed up the learning. The activation ReLu will be used after batch normalization. The convolution and batch normalization layers with same parameters will add in again again to extract more information and having a better representation of data. After going through two convolutions layer, max pooling layer will be added to calculates the maximum value in each patch of each feature map and reduce the spatial dimensions.

The CNN layers stated above in stage 1 will repeat by two more times, going into deeper layer of convolutions, the number of filters will increase to 64 and 128. It is because getting into deeper layers will represent more detailed features, hence the number of filters increased. At the end of each stage, a dropout with value of 0.3 will be applied at the end of stage to prevent overfitting except for last stage whereby the dropout value used is 0.5. After that, flatten layer will be added to get output from the convolution layers, flatten the output structure to be used by dense layers for classification.

Lastly, three dense layers will be added, these layers are fully-connected layers which is classification layers. In between each dense layer, a dropout with value of 0.5 and 0.25 will be added respectively to prevent overfit. A batch normalization will be added after first dense layer also.

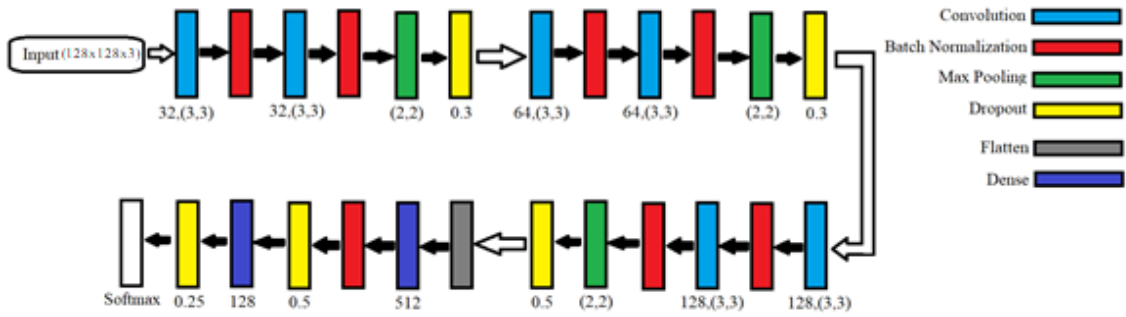


Figure 3.1.5: Model Architecture of Action Classification

3.1.6 Step 6: Buzzer Action

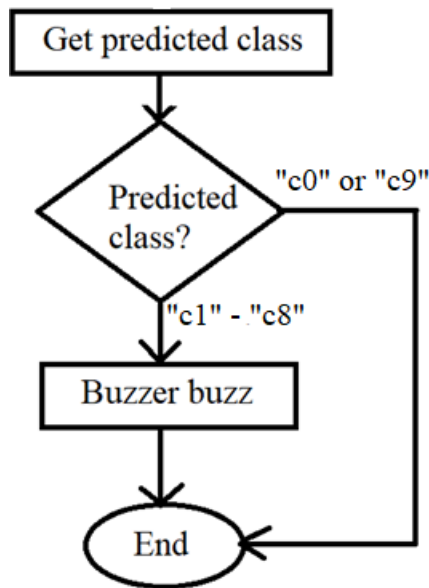


Figure 3.1.6: Buzzer Action

In this last stage, after getting the output from model, the system will verify whether it is considered as dangerous driving action or not. Since the dataset consisted of 10 different class whereby only the first and last class is considered as safe driving, hence if the predicted output not belongs to class 0 or 9, it considers driver is performing dangerous tasks and alert from buzzer will be given. Figure 3.1.6 show the flow chart of how the system going to work after getting the output from classification model.

3.2 System Requirement

This section provides overview of the system hardware and software requirements as well as the libraries needed.

3.2.1 System Hardware Requirement



Figure 3.2.1a: Raspberry Pi 4 Model B Module

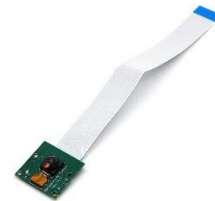


Figure 3.2.1b: Raspberry Pi Camera Module



Figure 3.2.1c: Speaker Buzzer



Figure 3.2.1d: GPS Holder Mount



Figure 3.2.1e: USB Car Charger



Figure 3.2.1f: 2m Type-C Cable

Chapter 3: System Design

The hardware that is going to be used for this framework was shown above from Figure 3.2.1a to Figure 3.2.1f. Raspberry Pi 4 Model B will provide processing power for the proposed framework instead of using laptop inside car, at here the Raspberry Pi 4 Model B with 2GB RAM is selected to prevent memory from running out easily. The Pi Camera will be setting up with Raspberry Pi to capture frame of driver. The speaker buzzer will be setting up with Raspberry Pi to give alert to driver when secondary tasks are performed. Figure 3.2.1g shows the 40-pin general-purpose input/output (GPIO) layout. The buzzer will be connected to Raspberry Pi pin 15 for positive red color which is GPIO 22 and pin 9 for negative black color which is ground.

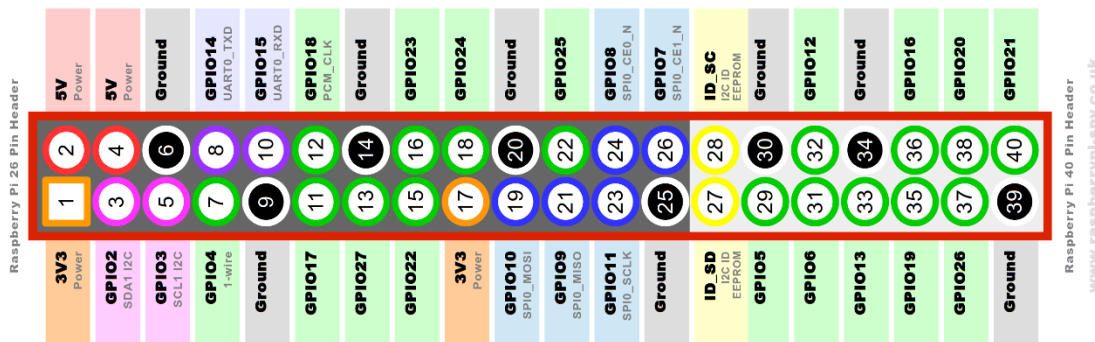


Figure 3.2.1g: Raspberry Pi 40-pins GPIO Layout

Next is the USB car charger, this charger will output 5V 3A power supply to Raspberry Pi which is similar to output of original Raspberry Pi adapter. Finally, a type-c cable with 2m long will connect to the USB car charger to provide the power supply. The reason for choosing 2m long is because 1m of cable are no longer enough to connect from the charger to Raspberry Pi.

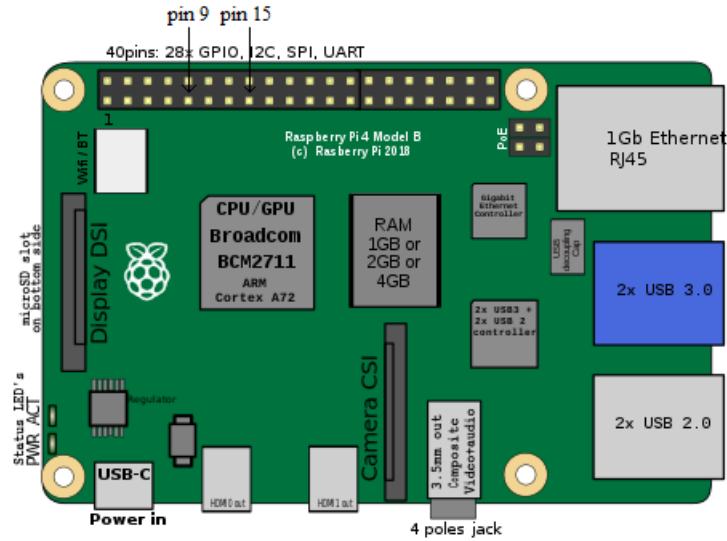


Figure 3.2.1h: Buzzer Location

Figure 3.2.1h show the exact location where the buzzer should be located. This figure clearly shows the Raspberry Pi 4 and the location where the buzzer should connect to it is pin 9 and pin 15 where pin 9 should connect to black color of buzzer wire and pin 15 should be connect to red color buzzer wire.

3.2.2 System Software Requirement



Figure 3.2.2a: Python IDLE



Figure 3.2.2b: OpenCV Library



Figure 3.2.2c: PyTorch Library

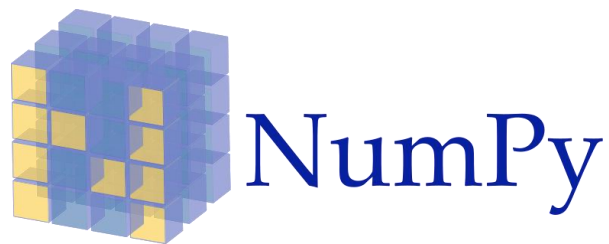


Figure 3.2.2d: Numpy Library



Figure 3.2.2e: Keras Library



Figure 3.2.2f: Tensorflow Library

Figure 3.2.2a to 3.2.2f show the software and libraries required to run this system. Python IDLE is required because this system is written using Python language. OpenCV library is required to perform the processing part for the image such as resize and reshape. Since all the models was built using PyTorch , the library is required to load the models into system. Lastly, Numpy library is used to create an empty blank image in order for the key points to draw on it. Keras library is used to build classification Keras model and Tensorflow is run at Keras backend.

3.3 System Setup

In this part, the configuration and setting of the proposed project will be done in step-by-step walkthrough. The proposed system should be work in Raspberry Pi, the setting up of Raspberry Pi and mounting of this hardware inside vehicle will be explain in details.

First of all, the speaker buzzer shall be connected to Raspberry Pi based on the explanation of GPIO as stated in 3.2.1 with Figure 3.2.1g as reference. The location of GPIO is located beside the ethernet port. Figure 3.3a show the exact location as how the speaker buzzer shall connect in the Raspberry Pi motherboard.



Figure 3.3a: Speaker Buzzer Connect to Raspberry Pi

Next, Raspberry Pi camera module can be installed into Raspberry Pi by connect the camera into CSI camera connector which is located beside the mini HDMI port. Figure 3.3b show the exact location on how the camera module is being installed to the Raspberry Pi motherboard.



Figure 3.3b: Camera module connect to Raspberry Pi

Chapter 3: System Design

After the camera module connected to Raspberry Pi, the setting up of this motherboard is almost done. The following step will describe how the system is going to be setup up inside the vehicle. First part is placing the Raspberry Pi motherboard on the GPS holder as shown in Figure 3.3c below, ensure that the motherboard mounted tightly on the GPS holder for stability.

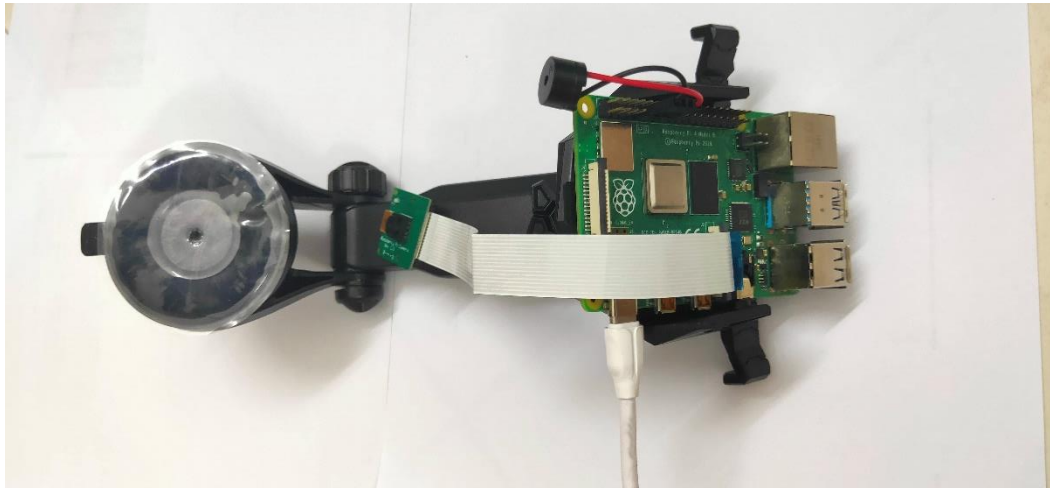


Figure 3.3c: Mount Raspberry Pi on GPS Holder

For the power supply to Raspberry Pi, simply connect the type-c cable to the USB car charger and connect the USB car charger into the cigarette lighter socket inside the vehicle as shown in Figure 3.3d below. Ensure that on the other side of the cable which is the type-c head is connected to the Raspberry Pi power supply socket.



Figure 3.3d: Provide Power Supply to Raspberry Pi

Chapter 3: System Design

The last thing to do is setting up the GPS holder as the exact location shown in Figure 3.3e. Ensure the GPS holder and the camera is stick tightly to the vehicle to prevent any fall. For the reason on why the setting of GPS holder and camera is located at this location, further analysis will be done in following chapters.

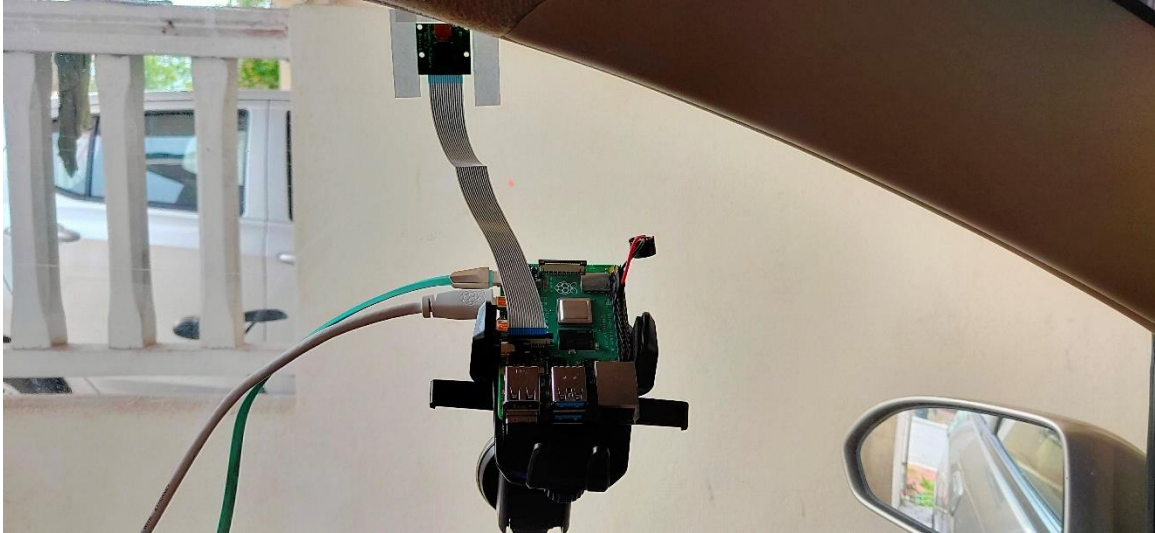


Figure 3.3e: Setting Raspberry Pi inside Vehicle

After the car engine is started, wait around 1 – 2 minutes for the system to totally boot up. It is because the motherboard required to load all models into all its CPU cores which took a lot of times. When the driver is performing dangerous secondary task, the system will trigger the speaker buzzer to give alert to driver. The system will continue run until the user stop the car engine. When the car engine is start again, system will boot up again too.

CHAPTER 4: SYSTEM ANALYSIS

4.1 Analysis Overview

This section is mainly to explain the details for analysis about last chapter such as camera mounting and so on. The reason for this chapter is to ensure the all the method, technique or hardware that selected was go through a deep analysis and not by simply select for no reason.

4.2 Camera Selection Analysis

In this section, different type of camera will go through an analysis to select a better one. Camera is one of the most important tools for this propose framework, hence analysis must be done with different cameras to choose wisely the best and most suitable one for the proposed system.

In this project, Raspberry Pi 4 model B will be selected to provide processing power, hence we can select whether to use smartphome camera, universal serial bus (USB) webcam or Pi camera to capture the image of driver. The comparison between different cameras will be done in order for a better one to be selected.

4.2.1 Pi Camera VS USB Webcam



Figure 4.2.1a: Pi Camera



Figure 4.2.1b: USB Webcam

The main different between Pi Camera and USB webcam is the performance. With Pi camera, since it directly connects to the motherboard instead of using USB, hence it gives better performance and higher frame rate with h.264 video encoding at 1080p30. However, USB webcam have lower frame rate compare to Pi camera. Moreover, Pi camera directly connect to GPU, hence it only gives little impact on central processing unit (CPU), leaving

it available for other processing. However, webcam normally use a lot more on CPU, unless it has built in encoding, but it will be more expensive.

4.2.2 Pi Camera VS Smartphone Camera



Figure 4.2.2a: Pi Camera



Figure 4.2.2b: Phone Camera

First of all, size is the problem. Pi camera having 20 x 25 x 9mm together with weight of 3g, while according to Petrov in 2018, majority of smartphones nowadays having at least 5.5" display, and weight of more than 100g. As such, Pi camera are more easily to be mount on car compare to smartphone based on their size and weight. In addition, smartphone having same problem as USB webcam where they only can connect to Raspberry Pi via USB, while smartphone can connect with Wi-Fi too. However, if the frame needs to go through network before enter the Raspberry Pi processing units, delay will be a trouble since this project focusing on real-time.

Based on the comparison with different kind of camera, we can conclude that Pi Camera are more suitable for this project since performance is the interest for real-time project. Furthermore, Pi Camera have lesser weight which is more suitable to be mount inside vehicle to prevent fall.

4.3 Camera Location Setup and Analysis

In this section, the Pi camera will be setting up at three different location. The output will be shown at below and analysis will be done to determine a suitable location which capture full driver body with proper angle and ensure that all actions can observe clearly.

4.3.1 Camera Setup Location 1

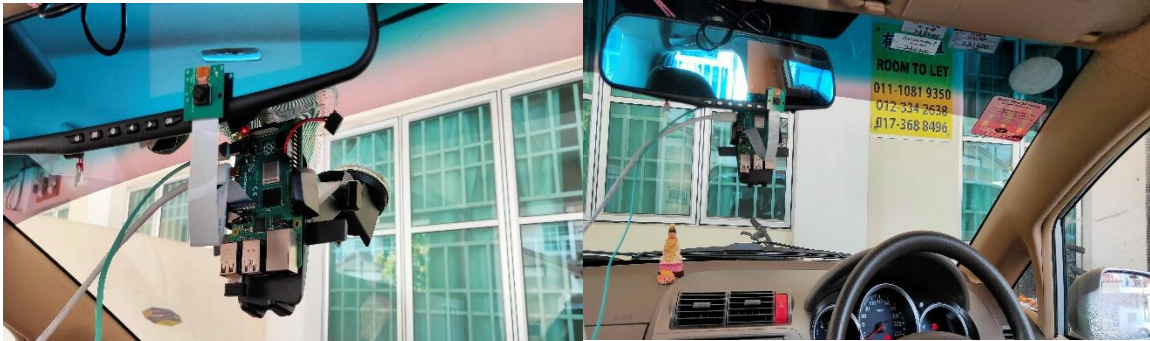


Figure 4.3.1a: Camera Setup Location 1

Figure 4.3.1a show the camera was set up in front of driver with the respective output, where the Pi camera mount on the interior car mirror. In this case, although the camera can mount easily at that location, but the output cannot fully display out driver's whole body. In this angle, we are unable to observed whether driver is playing smartphone or not especially when driver hold their smartphone with right hand. Furthermore, if the driver wishes to pick up the things or object that fell down, their head will probably block the camera and thus unable to capture driver's action.

Hence, we can conclude that in this location, it is not a perfect location for driver's action classification, but it will be useful if the project target on driver's face detection like fatigue detection. Figure 4.3.1b below show the frame that captured by the Pi Camera under this setup location and Figure 4.3.1c show the result obtained after the frame undergoes pose estimation framework.

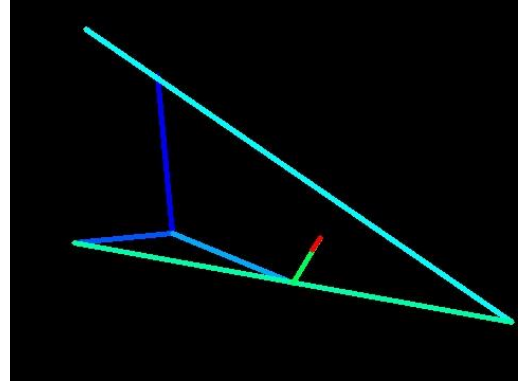


Figure 4.3.1b: Frame Capture from Location 1 *Figure 4.3.1c: Pose Estimate Output*

From the result, it shown that the camera can capture driver face nicely but cannot capture driver body. Hence, the output from human pose estimation framework is a very weird image. Except head, neck, and shoulders, it cannot detect hand hence result in the key points detected for hand gone wrong.

4.3.2 Camera Setup Location 2



Figure 4.3.2a: Camera Setup Location 2

Figure 4.3.2a show the camera was setup beside window. At this location, driver's body can be fully captured, but not totally complete. Apart from that, the surface where the camera is mounted are not flat, which cause the camera unstable and will likely to shake or fall down. In addition, if the co-driver wants to open the door, the Pi Camera had to take down because the hardware is link in between vehicle door and beside window. Therefore,

we can conclude that it is also not a perfect location for setting up the Pi camera. Figure 4.3.2b below show the frame result obtained from camera.



Figure 4.3.2b: Frame Capture from Location 2

4.3.3 Camera Setup Location 3



Figure 4.3.3a: Camera Setup at Location 3

In the location shown in Figure 4.3.3a, the camera is mounted at the window. At this location, driver's whole body can be shown clearly and captured by the Pi camera. Moreover, the camera can stably mount at that location since the surface is flat. This can avoid the problem of shaking or fall down, which can cause inaccurate result. Hence, we can conclude that this is the most suitable location for camera setting. Figure 4.3.3b show the frame captured by Pi Camera under this location setting. Table 4.3 show the summarization of comparing at different location.



Figure 4.3.3b: Frame Capture from Location 3

Table 4.3: Comparison of Camera Setting at Different Location

Location	Stability	Problem
Interior car mirror	Stable	No full body capture
Beside window	Not Stable	Difficulty in open car door.
Below car interior handle	Stable	No problem

4.4 Raspberry Pi Selection Analysis

Table 4.4: Comparison of Different Raspberry Pi Model

Model	CPU	CPU Clock	Number of Cores	RAM
Raspberry Pi 3 A+	Cortex-A53 64-bit	1.4GHz	4	512MB LPDDR2
Raspberry Pi 3 B+	Cortex-A53 64-bit	1.4GHz	4	1GB LPDDR2
Raspberry Pi 4	Cortex-A72 (ARM v8) 64-bit	1.5GHz	4	1/2/4GB LPDDR4

Table 4.4 shows the comparison of different Raspberry Pi Model, whereby the important specifications is highlighted to assist in selection of model. First of all, the CPU of model 3A+ and 3B+ is using Cortex-A53 64-bit while model 4 is using Cortex-A72 (ARM v8) 64-bit, whereby Cortex-A72 will perform better than Cortex-A53, which is better during process in real-time system as compare to Cortex-A53. Although model 4 having CPU clock of 1.5GHz which perform better a bit compare to having 1.4GHz of model 3, which is not much different. Both three models having same number of cores which is 4, which means can use multi-threading inside the process which make the execution of code more efficient.

However, the amount of RAM is a big gap compare with each other. RAM is referred as random-access memory; it allows processes to be store inside to allow processor to process them inside. In short, if the memory is not enough, it is difficult for some process to be execute which required large amount of memory. In Raspberry Pi, LPDDR, which refer as low-power double data rate synchronous, is used for all models. Model 4 having LPDDR4 which have faster speed compare to LPDDR2, hence it had more advantage compare to other models. Furthermore, model 4 can have selection of either 1, 2 or 4GB of RAM which is larger size compare to RAM of both model 3 Raspberry Pi.

In conclusion, overall specification of Raspberry Pi 4 is better than model 3A+ and 3B+. In the proposed system, 2GB RAM of model 4 is selected to ensure that the memory is always enough to prevent run out of memory.

4.5 Human Pose Estimation Analysis

In this section, different types of human pose estimation framework from others people work was applied in Raspberry Pi for detecting human key points. Due to the reason that nearly all human pose estimation was done using either laptop or desktop and the case of using it in Raspberry Pi was not found. Hence, the pretrained model was downloaded and test in the Raspberry Pi. The main reason for not using Raspberry Pi is because for detecting human key points in a single image require a large and complicated model. Hence it results that the processing power must be enough for the model to run. Raspberry Pi does not contain GPU, and its CPU processing power is weaker a lot compare to laptop CPU. This cause that the work to detecting human key points was not done in Raspberry Pi, and in order to do the comparison, the pretrained model had to be self-installed into Raspberry Pi and test it our self.

The models selected and available online to compare with the framework done in this project is OpenPose (Cao, et al., 2017) and OpenPose in MobileNet which had been done by ildoonet, this person successfully convert the OpenPose from Caffe model into tensorflow model run with MobileNet as backend feature extraction. Below table show the performance when applied these models on Raspberry Pi.

Table 4.5a: Comparison of Human Pose Estimation Framework

Framework	Time usage per frame (Raspberry Pi)	Frame per second (Laptop)	Detected key points
OpenPose	~45 s	~4.2 s	18
Ildoonet MobileNet OpenPose	~4.5 s	~0.12 s	18
Pose Generation Model (Proposed Framework)	~2.3 s	~0.89 s	8

Table 4.5a show the comparison of different framework for detecting human pose. As result, the proposed framework for this project could achieve a better result in Raspberry Pi compare to others two framework. However, ildoonet framework is able to achieve a

better performance on laptop. It is because the separable convolutional technique (Chollet, 2017) was applied which reduce the parameters for the CNN model computation. However, OpenPose which is the most popular framework that compute the human pose achieve the slowest time, but the most accurate.

For the key points detection part, both OpenPose and Ildoonet framework could detect 18 key points from a human while the proposed framework for this project can only detect 8 key points which only cover the upper body part. The reason is because for a driver detection system, upper body part is enough to represent the driver action.

The reason that the proposed framework is able to run faster compare those two frameworks is because the size of the graph of CNN network is very small compare to them. The more the output channel in a convolutional layer, the more the parameter required. The formula for getting the parameter required for a convolutional layer is:

$$(N \times M \times L + 1) \times K$$

N and M are the filter size, L is the number of input feature maps and K is the output feature maps. If the input channel is 32, the filter size is 3x3 and the output channel is 64, then this layer is learning 64 different 3x3x32 filters.

The more the parameters, the slower the network. Hence instead of using a large network to compute many key points, the proposed network used a small network to compute 2 key points. There is totally of 4 models running parallel in the CPU of Raspberry Pi which total up can calculate 8 key points with faster speed compare to others. The summary of all the things is shown in Table 4.5b below.

Table 4.5b: Output from Different Conditions

Condition	Time Usage	Accuracy
Large network more key points output	More	Ok
Large network less key points output	More	Good
Small network more key points output	Less	Bad
Small network less key points output	Less	Ok

In conclusion, it can be summarized as if user want to estimate human pose frame using laptop regardless of the time usage, OpenPose could be the choice. If user want to estimate with faster speed, Ildoonet framework is the best choice all the time. However, if the user wishes to solve some problem using certain hardware that lack or without GPU, the proposed framework could be the choice. Keep in mind that the problem that going to solve must only related to upper body part.

Furthermore, both OpenPose and Idoolnet could compute multi-person in a single image whereby the proposed framework could only compute one person. It is because both networks consist of PAF that store information to represent which part is belongs to which part that can detect multiple person and all of their key points. Both of the network contain two different branch that compute human 18 key points and compute 38 PAF, however, the proposed framework only compute single human key points. It is because since there would only one driver hence it is not necessary to apply multi-person human pose estimation framework that might compute even more processing power and increase time usage.

4.6 Classification Models Analysis

In this section, the analysis of different kinds of CNN models with different layer and parameter will be analyze to obtain a better classification model for driver action. The purpose of this section is made sure that the finalize model is having a high accuracy, hence every model will be evaluated to ensure that no overfitting or underfitting happen. The architecture of each model will be described first with their respective outcome. After that, their overall performance will be shown in Table 4.6.

4.6.1 Model 1

In this model, two-stages of CNN is built. For the first stage, two convolutions layer will be used, with activation of ReLu, 32 filters with kernel size of 3 x 3, and a batch normalization layer in between each convolution layer. After finishing the convolution layers, a max pooling of 2 x 2 is used to reduce the spatial dimension of image, and lastly a dropout layer with argument of 0.2 to prevent overfit.

At the next stage, same layer as stated in first stage is used, but having 64 filters for each convolution layer. It is because after go through the pooling layer, the dimension of image had decreased hence more filters is needed. After finish the convolutional stages, fully connect layers will be added. A dense with class of 512 is added with ReLu activation first, follow by a dropout of 0.2. Another dense with argument of 128 with ReLu activation is added again with a dropout of 0.2. Lastly, a Softmax with 10 class is added.

Figure 4.6.1a show the accuracy and loss of model being trained with epoch of 10. It shows that the test accuracy is higher than train accuracy, which suppose not to happen. Figure 4.6.1b will show the architecture of the model with their respective argument for each layers and Figure 4.6.1c will show more detail version of model architecture by showing the input and output from the model layer by layer.

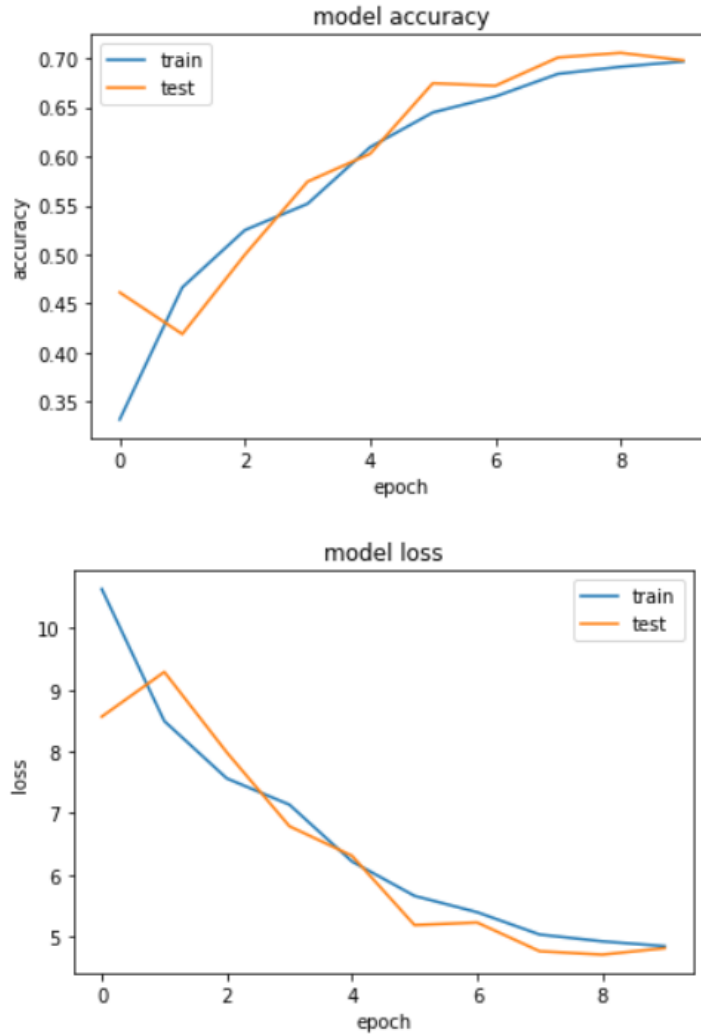


Figure 4.6.1a: Outcome of Evaluation of Model 1

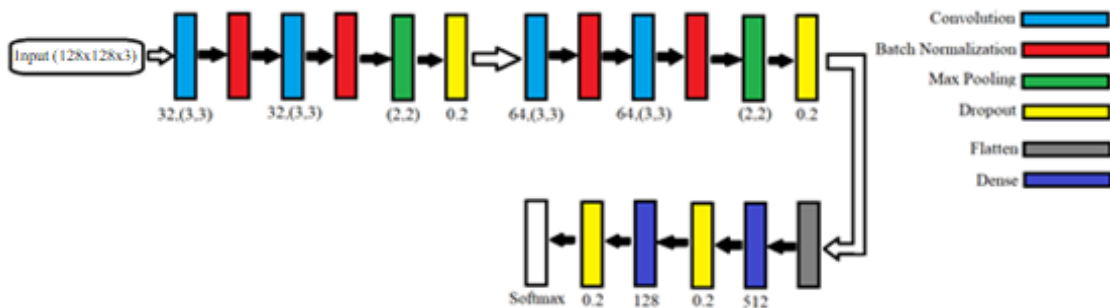


Figure 4.6.1b: Architecture of Model 1

Chapter 4: System Analysis

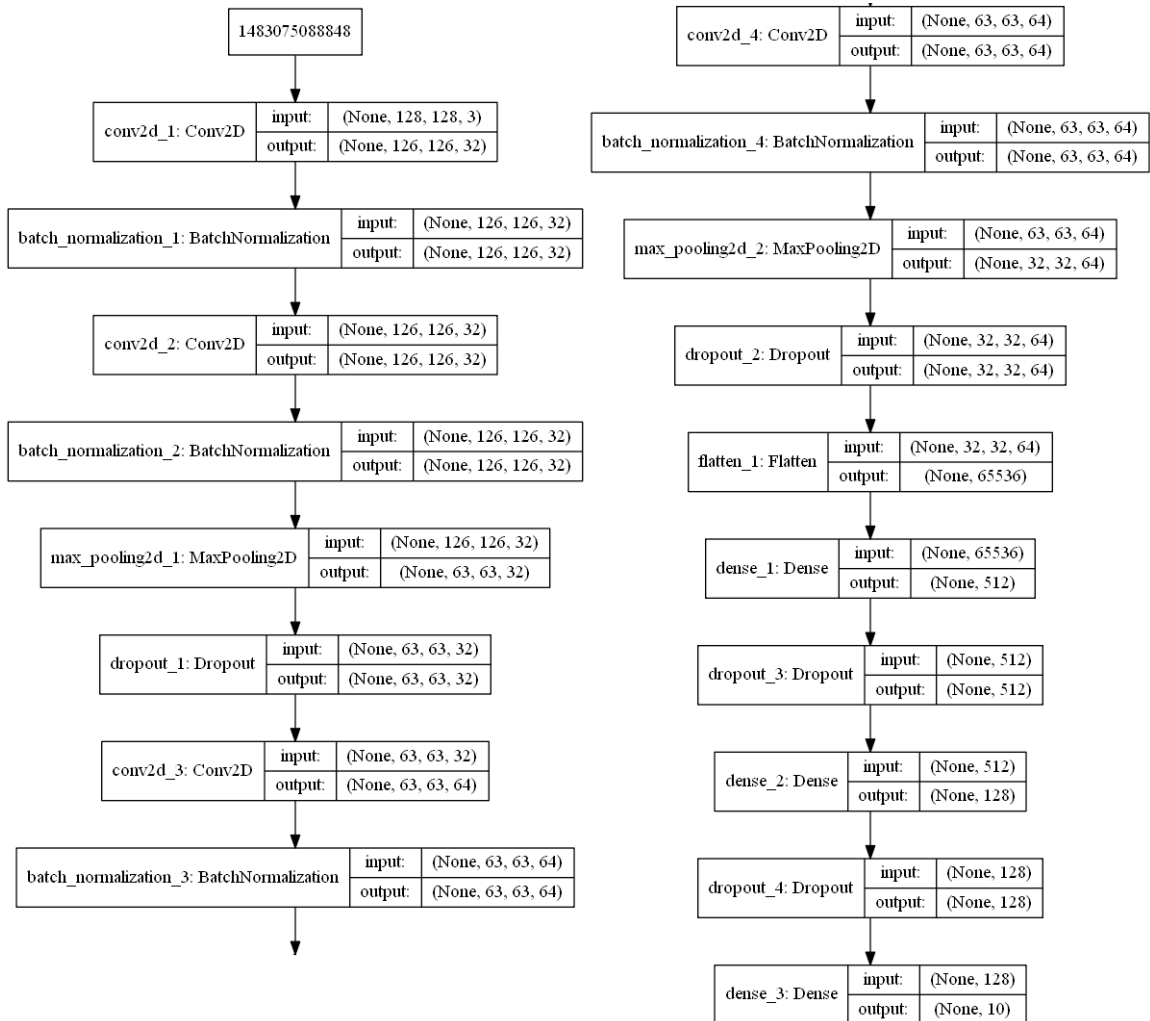


Figure 4.6.1c: Details of Model 1

4.6.2 Model 2

This time, another stage of convolutional layers which is similar to previous stages is added after the CNN layers of model 1. This time will try to make sure that the training accuracy is better than the testing accuracy. At the third stages, the convolutional layers will use 128 filters with 3 x 3 kernel size and activation of ReLu. Since after go through the two-stages CNN, the spatial dimension of image had been decreased from 128 x 128 to 32 x 32, hence the number of filters need to increase also. The fully connected layer will be similar to model 1.

Figure 4.6.2a show the output after model 2 had been trained using the architecture as described above. After go through the evaluation, it shows that the train accuracy and test accuracy are very bad and not even reach 10%. Furthermore, the loss is not even less than 1 which is very high. Figure 4.6.2b will show the model architecture with their respective argument for each layers and Figure 4.6.2c show more details about the architecture.

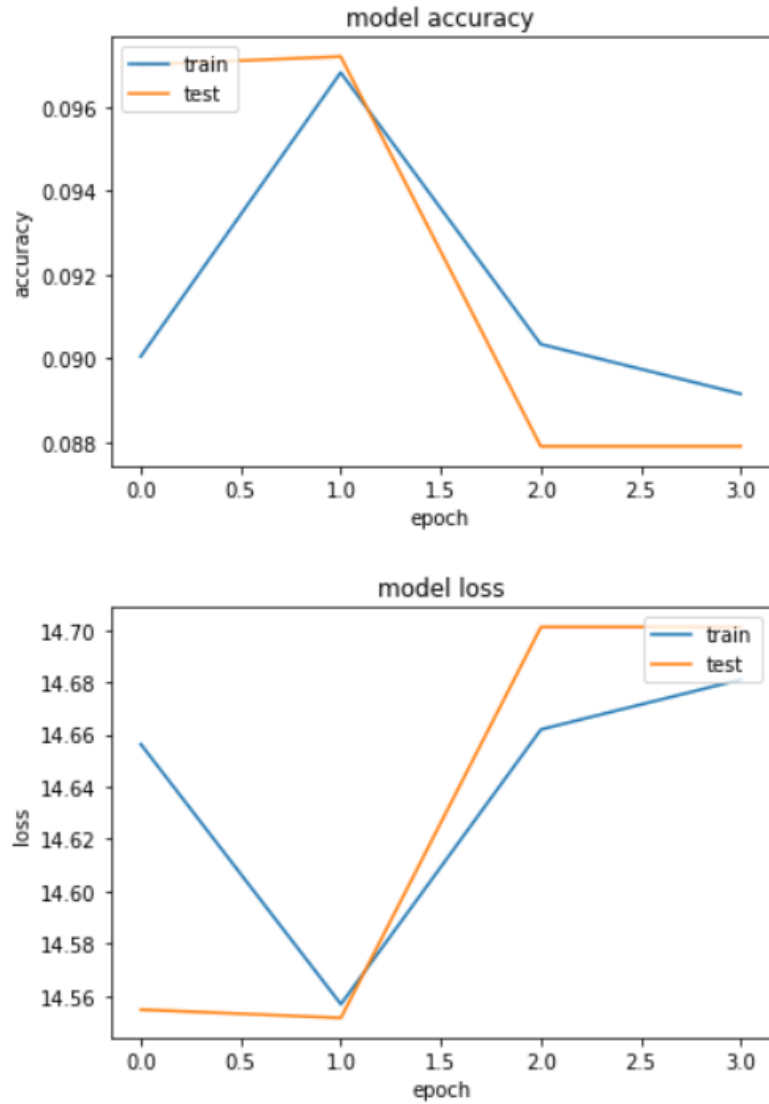


Figure 4.6.2a: Outcome of Evaluation of Model 2

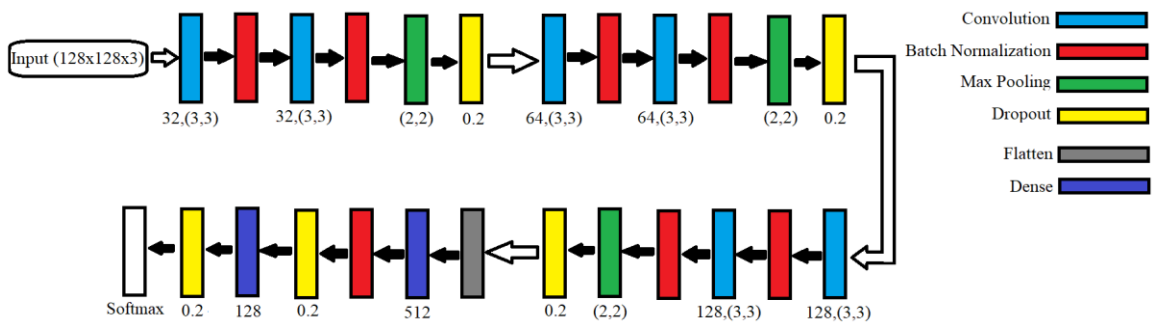


Figure 4.6.2b: Architecture of Model 2

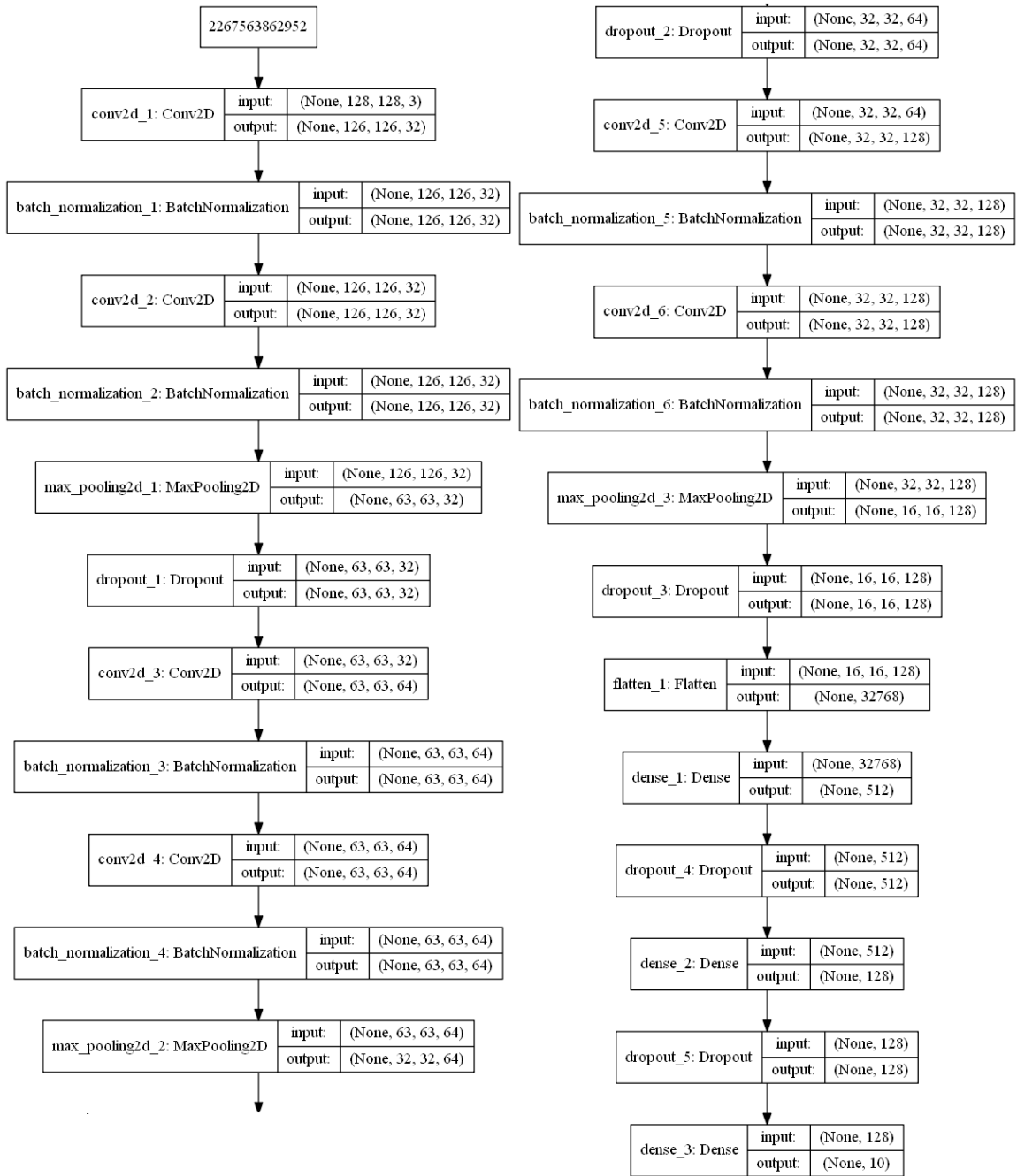


Figure 4.6.2c: Details of Model 2

4.6.3 Model 3

In this part, to ensure that the model could extracted a more and better quality of features, the size of input was increased to 128x128. To avoid the problem of overfitting, the dropout value will increase also. It is because dropout layer can used as regularization by setting some input units to zero, which could help to prevent from overfitting. By using same architecture as mentioned in model 2, but this time the dropout value for first and second stages set as 0.3. It is because in the input layer, if the dropping too much input data might affect the training, hence the dropout value tries to make it as low as possible at early stage. In the third stage, the dropout value will be set into 0.5. In the fully connected layer, after first dense layer will use a dropout value of 0.5 and second dropout will be 0.25.

Figure 4.6.3a show the output after changing the value of dropout in each layer. For the model loss, both training loss and testing loss having almost similar value. Which means, this model currently does not encounter overfitting issues or underfitting issues. Although the training accuracy lower than the testing accuracy a bit, but the different is not big which still can be acceptable. It might due to the test dataset is too simplify. Figure 4.6.3b will show the architecture of model 3 and Figure 4.6.3c will show the details of model 3.

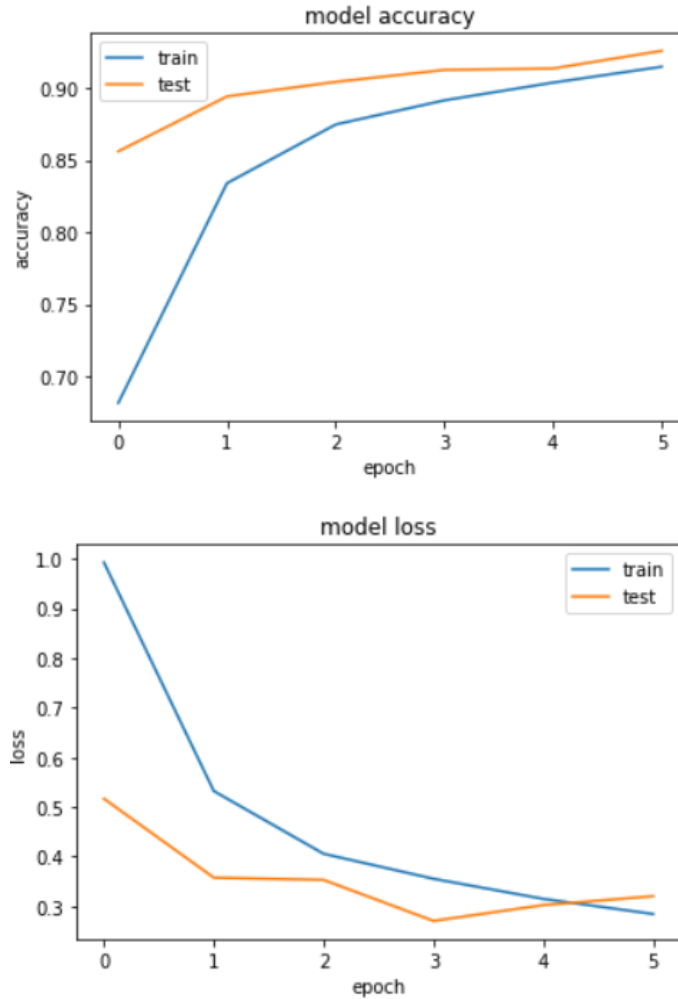


Figure 4.6.3a: Outcome of Evaluation of Model 3

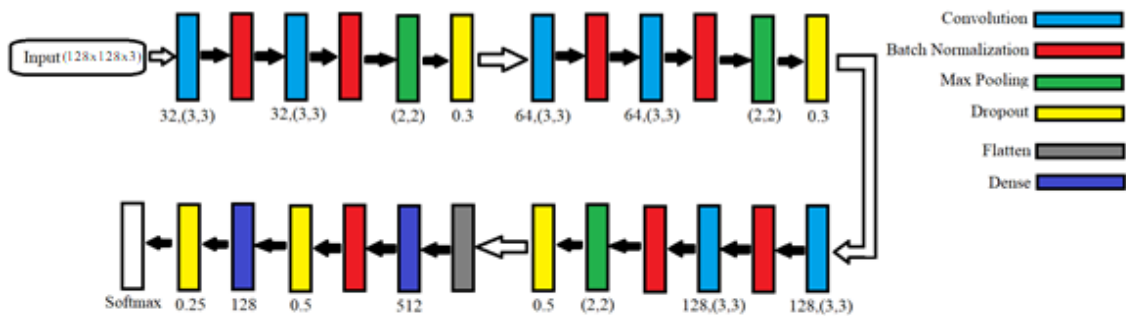


Figure 4.6.3b: Architecture of Model 3

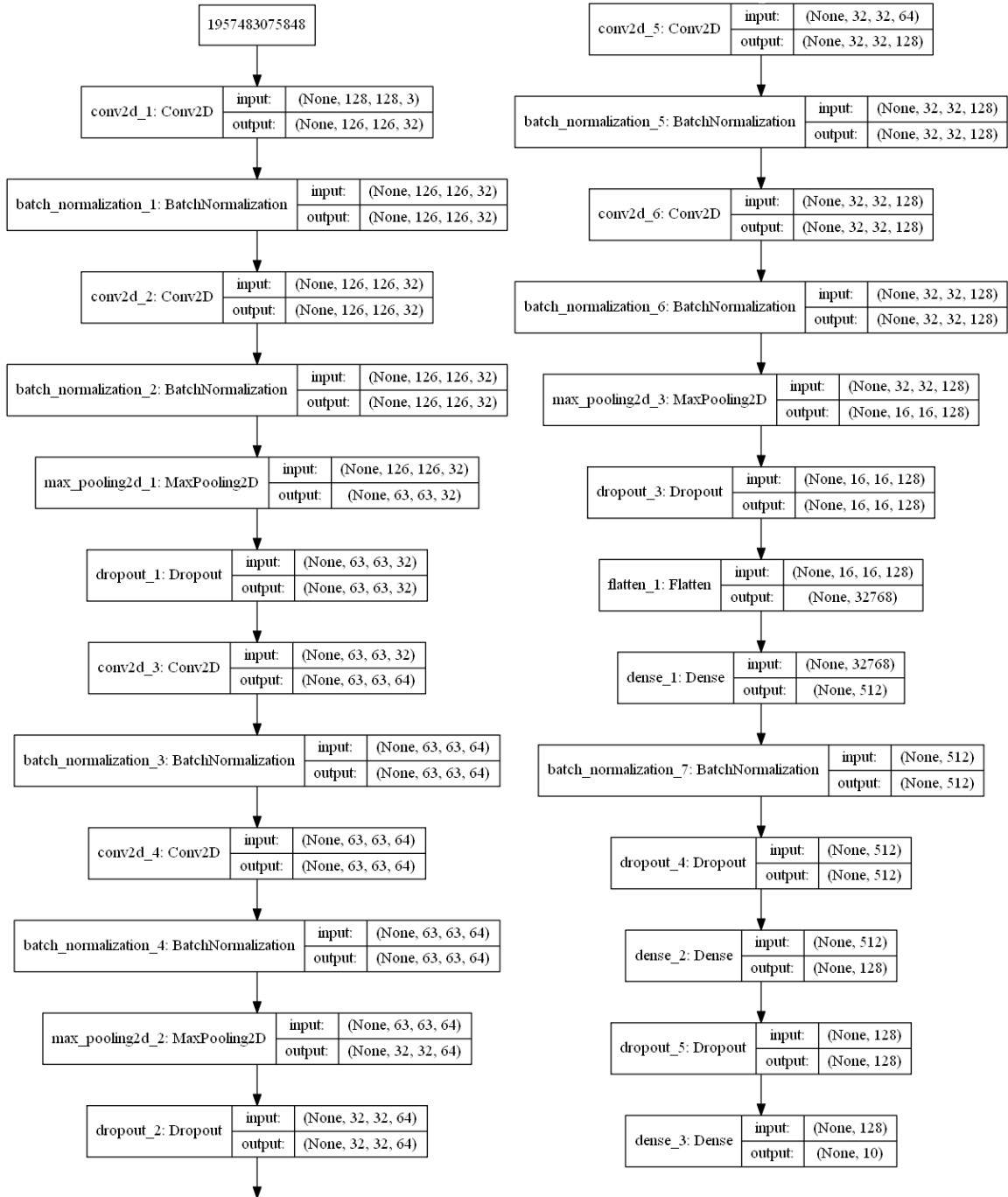


Figure 4.6.3c: Details of Model 3

Table 4.6: Result from each Models

Model	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss	Logarithmic Loss	RMSE
Model 1	0.6969	0.6979	4.8502	4.8172	10.2368	0.2453
Model 2	0.0892	0.0879	14.6810	14.7012	31.5025	0.4271
Model 3	0.9147	0.9257	0.2840	0.3199	0.3216	0.1092

According to the outcome of evaluation of three different models, the summarize output is shown in Table 4.6. Table 4.6 show that by comparing model 1 and model 2, model 1 overall is better than model 1 since it having lesser logarithmic loss. Logarithmic loss, also refer as log loss, is a classification metric to evaluate a model's performance. In short, the log loss value must try to be minimized to obtain a better model. At here, the log loss of model 3 is lower compare to model 1 and model 2, which shows that it has a better prediction. Moreover, the validation loss of model 3 is just higher a bit than its training loss, which is just right. However, model 1 and model 2 shown had some problem in the network architecture since the loss is relatively higher than average. Normally the desire loss obtained should in between 0 and 1 but model 1 and model 2 get a very high loss which means these two models had some problems.

Meanwhile, model 3 having better accuracy compare to model 1 and model 2, and it does not face any overfitting and underfitting issues, which indicated that overall, it is better one. Moreover, its log loss is the lesser among all the models. For the root mean square error (RMSE) of three of the, model 3 having the lowest RMSE hence it will have a lower error since RMSE is a standard way to compute an error of a model in predicting quantitative data. Therefore, model 3 will be selected as the classification model for this system.

CHAPTER 5: IMPLEMENTATION

5.1 Implementation Overview

In this section, the step-by-step and output from the proposed system will be explain and shown. From frame acquisition to buzzer action, everything should be work fine and output should be correct.

5.2 Image Acquisition

Real-time driver action detection framework will be started when the car engine started. Before that, the Raspberry Pi will be mounted on the window inside the car as shown in Figure 5.2a to Figure 5.2c. This position is able to capture driver body movement clearly and very suitable position that won't cause any accident such as fall down happen on Raspberry Pi. The Raspberry Pi is held using a GPS holder.

When the system is start running, all models will be loaded and the frame will be started to be acquired by the Pi Camera. The frame capture by the Raspberry Pi 5MP camera module which connected to the motherboard power by car adapter. The camera will start to capture frame with shape of 480x360 pixels and 30 frame per second.

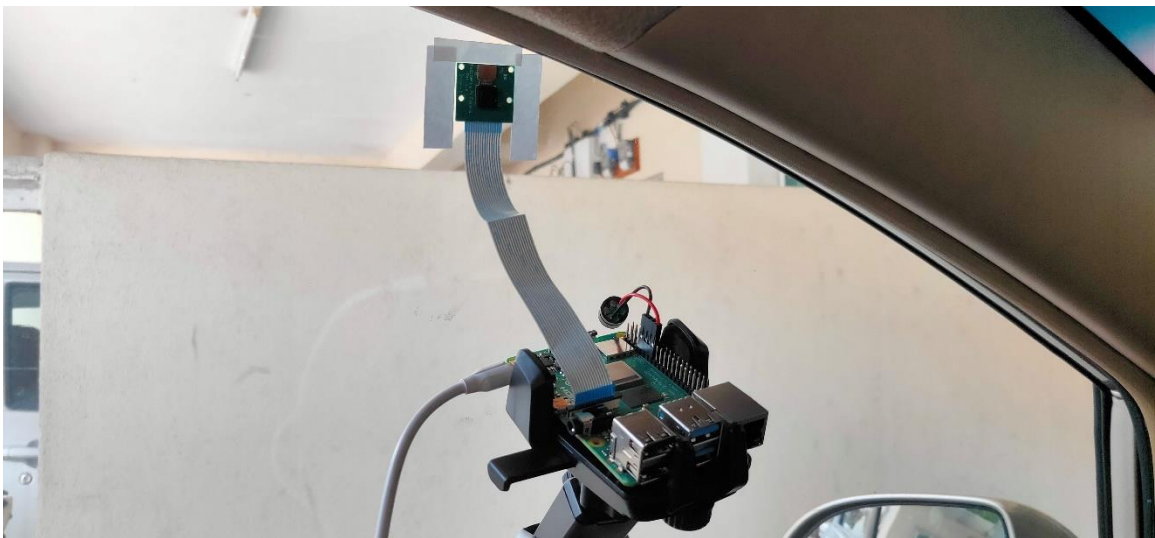


Figure 5.2: Frame Acquisition



Figure 5.2b: Mounting Camera View 1

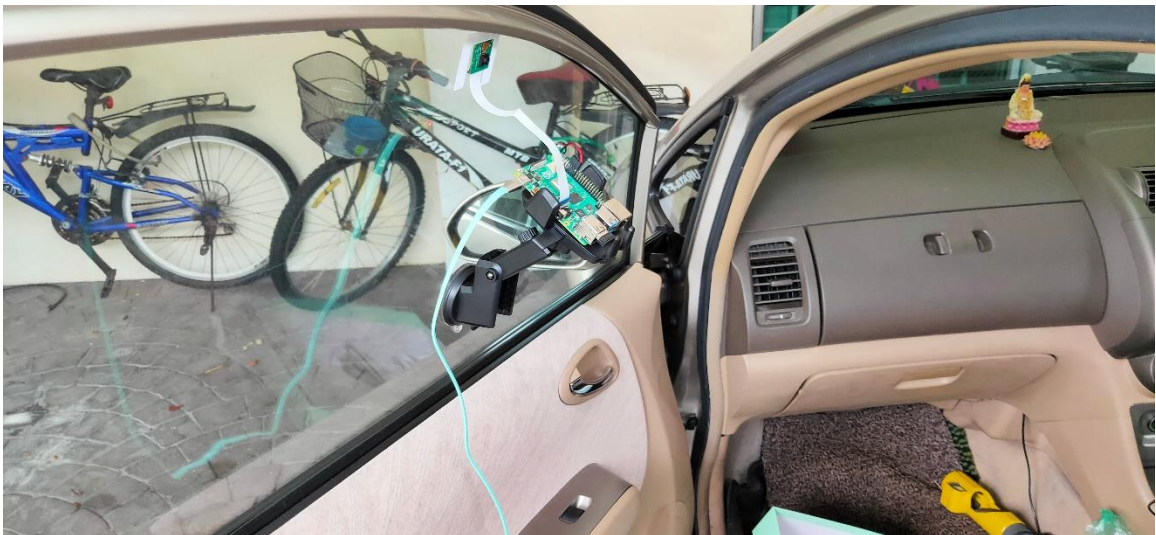


Figure 5.2c: Mounting Camera View 2

5.3 Image Pre-processing

After frame is obtained, the image frame will be given to all other threads for further process. Assume that image obtained from camera is shown in Figure 5.3, it will first undergoes resize and reshape into $(-1, 3, 128, 128)$ which is able to input into human pose estimation model.



Figure 5.3: Frame Captured by Camera

5.4 Human Pose Estimation

After the frame is being resized and reshaped, it will then generate human pose frame which first start by compute the human key points heatmaps using CNN network proposed. Since this system focus on compute out 8 human key points which represent upper body part, 8 different heatmaps output will be generated from CNN model. By technically, these heatmaps is called feature maps, or also called activation maps which is output from convolutional layer inside CNN model. Figure 5.4a to Figure 5.4h show 8 heatmaps that is produced from CNN model whereby the input is the frame captured from camera that shown in Figure 5.3. The heatmap is match with the body part as shown in Figure 5.4i.

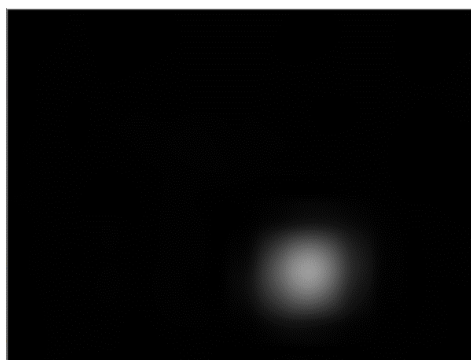


Figure 5.4a: Left Elbow Heatmap

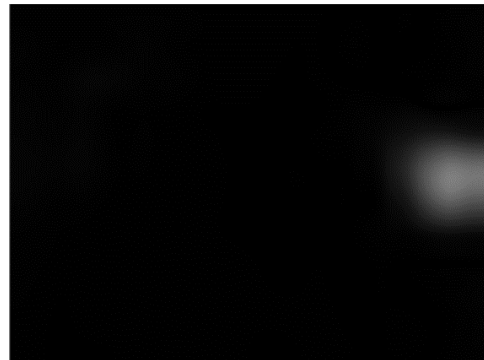


Figure 5.4b: Left Shoulder Heatmap

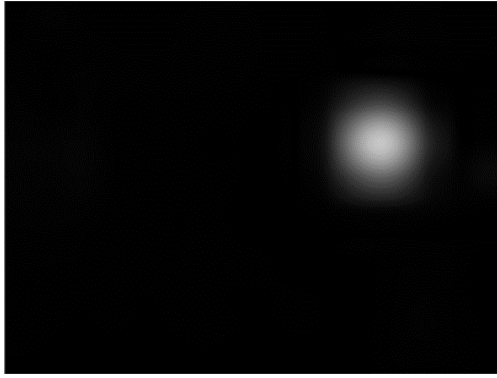


Figure 5.4c: Right Shoulder Heatmap



Figure 5.4d: Right Elbow Heatmap



Figure 5.4e: Right Wrist Heatmap

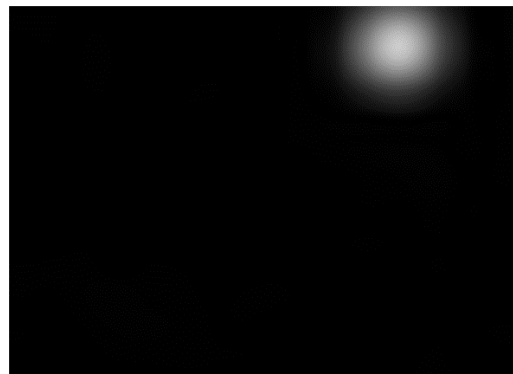


Figure 5.4f: Head Heatmap

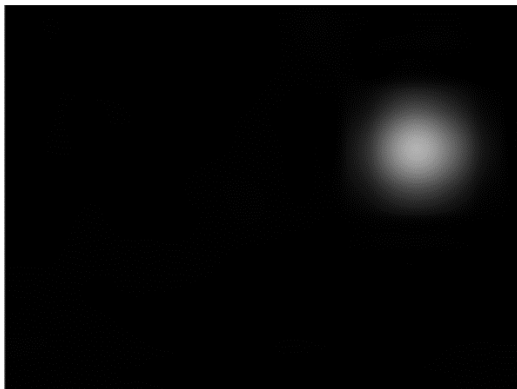


Figure 5.4g: Neck Heatmap

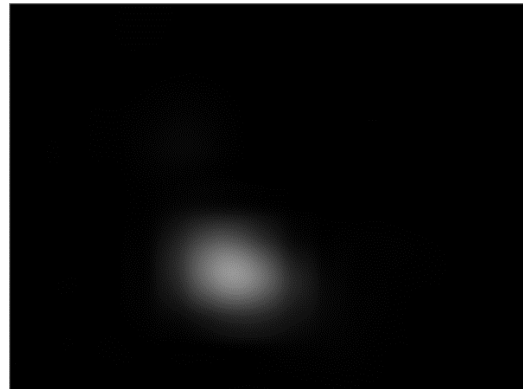


Figure 5.4h: Right Wrist Heatmap

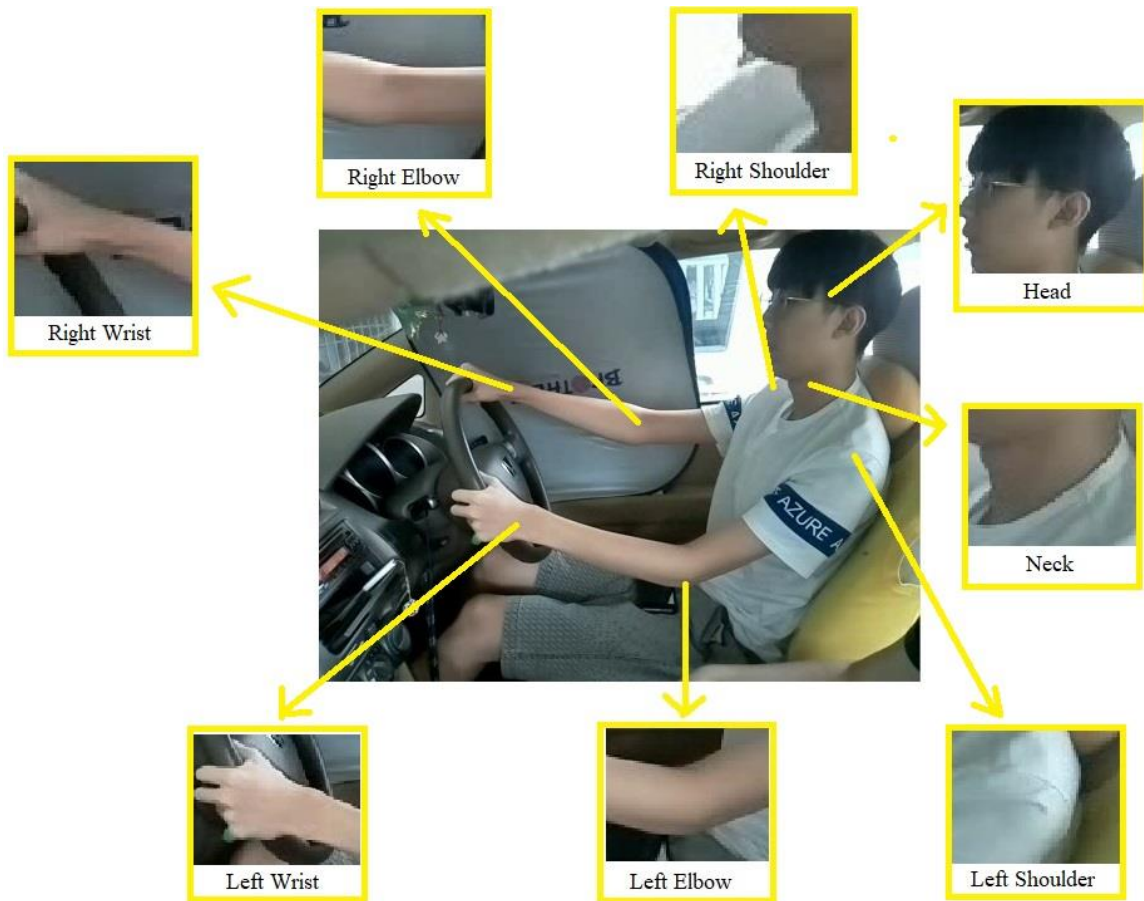


Figure 5.4i: Body Part Location

By comparing the human body part location in Figure 5.4i to the heatmaps in Figure 5.4a to Figure 5.4h, the white color spot for each heatmaps represent the location for that particular part.

After the heatmaps is generated, it will be used to calculate human key points exact location by coordinate using numpy function. For every heatmap, numpy will first obtain the index with maximum value using argmax function. Next, unravel_index function from numpy will be used to return x-axis and y-axis for the specific key points inside the image.

After getting all key points exact location, OpenCV library is used to draw line that connect the key points with different colour that represent different body part in a new blank image. The output for this part is shown in Figure 5.4b below.

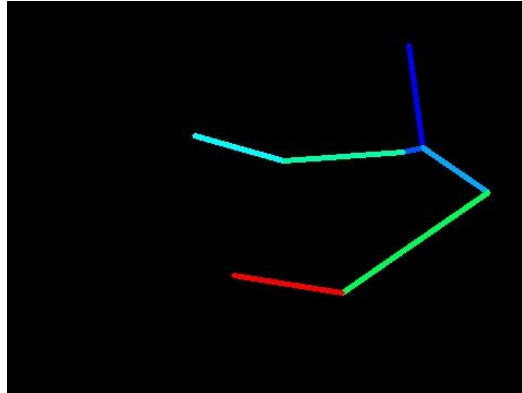


Figure 5.4j: Generate Human Pose Frame

5.5 Driver Action Detection

After the human pose frame is computed, the next step will be passing this frame into the classification CNN model that classify driver current action.



Figure 5.5: Predicted Output

5.6 Buzzer Alert Notification

In the final step, alert from buzzer will be given if the system classified user as doing dangerous secondary tasks. The purpose of this alert is to inform driver to go back to main driving task. After done give the alert, the system will continue grab another frame from camera module and loop again all the work to keep the task in real-time.

CHAPTER 6: SYSTEM TESTING**6.1 Verification Plan**

Before system testing is being conduct, some verification plans are required to be done first at the beginning. Verification plan is act as a checklist which list down everything the system should able be perform and must be able to complete. The list of verification plans is shown in Table 6.1 below.

Table 6.1: Verification Plan

Test	Test Plan	Expected Output
1	Read frame from Pi Camera after turn on car engine	Frame successfully loaded
2	Generate human pose frame	Successfully generate pose frame
3	Classify driver normal driving	Classify driver action correctly
4	Classify driver playing smartphone on left hand	Classify driver action correctly
5	Classify driver playing smartphone on right hand	Classify driver action correctly
6	Classify driver listening to phone call on right hand	Classify driver action correctly
7	Classify driver listening to phone call on left hand	Classify driver action correctly
8	Classify driver adjusting radio	Classify driver action correctly
9	Classify driver eating or drinking	Classify driver action correctly
10	Classify driver talking with passengers	Classify driver action correctly
11	Classify driver reaching behind	Classify driver action correctly
12	Classify driver makeup	Classify driver action correctly
13	Buzzer buzz when action is dangerous driving	Buzzer buzz successfully and correctly


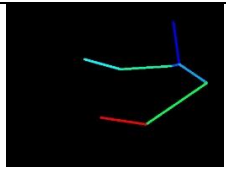

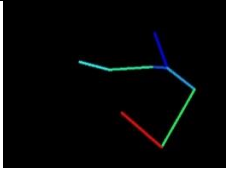

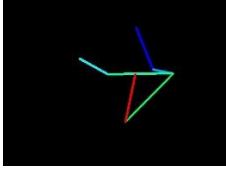
6.2 Short Clip Testing Overview

In this section, some experiment was carried out to ensure that the system can run properly. The experiment was done by driver performing different task inside the vehicle, all tasks performed will be saved into video and all frame will be extracted to do the classification. Below section consisted of different condition for the driver to perform the action inside the same vehicle and table to show the result obtained. The table first column is the class of action performed, second column is the number frame extracted, third column is the first image obtained as sample, fourth column is the output from image in third column after passing through human pose estimation framework, fifth column is the accuracy of successfully classification.

6.2.1 Experiment 1 – Short Sleeve

For this experiment part, driver wearing short sleeve shirt is undergoes the experiment. The purpose of doing this experiment is to ensure that the pose estimation framework is able to compute out driver pose frame when driver is wearing short sleeve. Since driver elbow is visible, hence it must ensure that the result is accurate. The result is shown below:

Table 6.2.1: Detection Results 1

Class	Frame	Total Frame	Human Pose Estimation	Accuracy of Successful Classification
Safe Driving		66		65 / 66
Texting – Left		58		41 / 58
Talking to Phone – Left		97		70 / 97

Chapter 6: System Testing


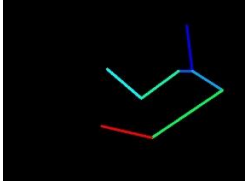

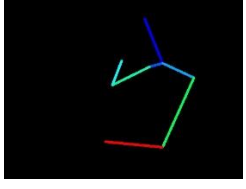



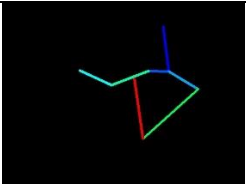

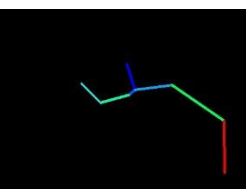

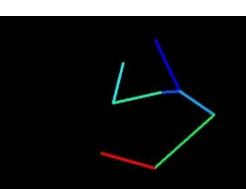

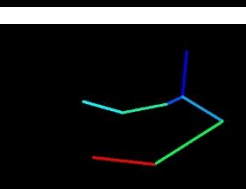
Texting - Right		40		35 / 40
Talking to Phone - Right		126		83 / 126
Operating the Radio		49		35 / 49
Drinking / Eating		58		51 / 58
Reaching Behind		74		63 / 74
Hair and Makeup		63		57 / 63
Talking to Passenger		133		130 / 133


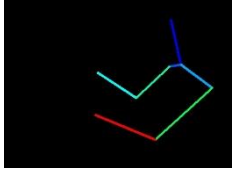

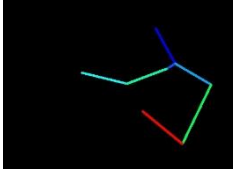



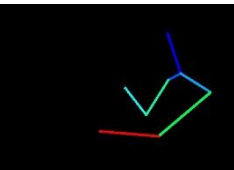


Table 6.2.1 show the results obtained and it shows that the pose estimation framework is able to detect human key points when driver is wearing short sleeve. The accuracy obtained is quite good, but the action like talking to phone on right hand side accuracy is not as good

as other because it keeps detect it as hair and makeup. On the other hand, operating radio also keep classified as safe driving. This case happened because the human pose frame is very similar to each other which cause the CNN classification model to classify wrongly.

6.2.2 Experiment 2 – Long Sleeve

In this experiment, driver will wear long sleeve and check if the pose estimation framework is able to detect driver hand properly or not. It is to ensure the system can compute human pose frame accurately even though human elbow is not visible and being covered by long sleeve.

Table 6.2.2: Detection Results 2

Class	Frame	Total Frame	Human Pose Estimation	Accuracy of Successful Classification
Safe Driving		73		60 / 73
Texting – Left		73		65 / 73
Talking to Phone – Left		98		91 / 98
Texting – Right		80		74 / 80
Talking to Phone – Right		86		76 / 86


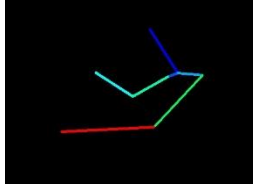

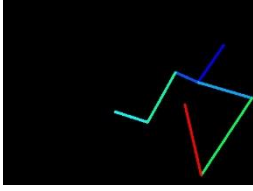

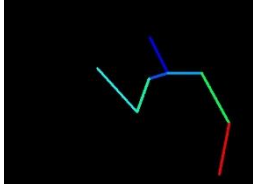

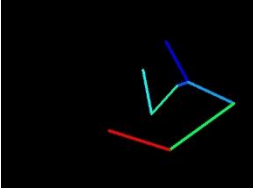

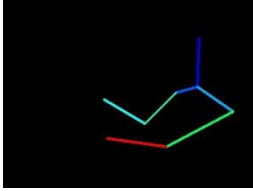
Operating the Radio		92		84 / 92
Drinking / Eating		129		94 / 129
Reaching Behind		52		48 / 52
Hair and Makeup		86		72 / 86
Talking to Passenger		98		79 / 98

Table 6.2.2 above show the results obtained and it shows that the pose estimation framework is still able to detect human key points even though driver is wearing long sleeve shirt. The accuracy obtained is quite good, but this time the model had a bad performance when classifying safe driving and talking to passenger. It is due to the similarity of pose frame generated by human pose estimation framework is very high hence cause model to misclassified on each other.

6.3 Live Based Testing

In this section, the classification will be done in live based. The total time took to perform this experiment is 4 minutes and 33 seconds, where the driver will perform all the driver actions inside the vehicle and the system will do the classification. Unlike the section in 6.2 that only do classification in short clip video that took out all the frames, this section will only perform the classification on a certain frame after previous frame is done. Which means, during the processing on a frame, the camera will only grab a new frame on current driver action for process. Figure 6.3a shows frames that were captured by Raspberry Pi during live classification.



Figure 6.3a: Frames Obtained from Raspberry Pi

In Raspberry Pi, the total frames that can processed within 4 minutes and 33 seconds is 35, where the average time required to compute one frame is 2.25 seconds. The reason for taking so long time to process one frame is because the human pose estimate framework. If the framework contains too less convolution layer which might speed up the framework but the detection on human body part will be very worst.

Table 6.3: Confusion Matrix on Classification Results

		True / Actual										
Predicted	Class	Safe Driving	Texting - Left	Phone - Left	Texting - Right	Phone - Right	Operating Radio	Drinking / Eating	Reaching Behind	Hair and Makeup	Talking to Passenger	
	Safe Driving	6	0	0	0	0	0	0	0	0	0	3
	Texting - Left	0	4	0	0	0	0	1	0	0	0	0
	Phone - Right	0	0	2	0	0	0	0	0	0	0	0
	Texting - Right	0	0	0	1	0	0	0	0	1	0	0
	Phone - Right	0	0	0	0	1	0	0	0	0	0	0
	Operating Radio	0	0	0	0	0	2	0	0	0	0	0
	Drinking / Eating	0	1	0	0	0	0	1	0	0	0	0
	Reaching Behind	0	0	0	0	0	0	0	2	0	0	0
	Hair and Makeup	0	0	0	1	1	0	0	0	0	1	0
	Talking to Passenger	5	0	0	0	0	0	0	0	0	0	2

Table 6.3 shows the confusion matrix that is produced after did the classification on all 35 frames. It shown that safe driving and talking to passenger is very unstable. It is because the pose frame generated from these two actions is very similar hence causing this two actions keep misclassified into each other's. But since talking to passenger is not considered as dangerous tasks during driving, hence buzzer won't make alert which make no different for driver during driver with the system operating. Moreover, texting to the phone on right hand and hair make up also keep misclassified due to the similarity of pose frame too. In addition, there was also contain similarity in between texting on left hand side and eating because both of these actions left hand position is almost similar.

CHAPTER 7: CONCLUSION

Malaysia had become one of the countries with highest death rates on the road. According to LUM (2019), Malaysia had the highest accident rate in Asia countries, just behind Thailand and Vietnam. The death rate due to traffic accident was increasing every year nowadays. Hence, Malaysia death rate which was related to the traffic must be resolve by reduce it in order for Malaysia to become a better and more successful country.

One of the reasons that cause traffic accidents was due to the secondary task performed by the driver. Due to the emerged of high technology nowadays, drivers like to enjoy themselves inside their smartphone, which draw away their attention from primary driving task. Other than using smartphone, talking to passengers, eating, reaching behind, and make up also the factors that leads to driver not paying attention. According to study done by Choudhary and Velaga (2019), driver face more traffic problem especially while doing texting task. Hence, a driver action detection system as an advanced driver assistance system come over to help the driver by alert the driver if secondary tasks is performed during driving. The purpose of this system is to decrease the traffic accident rate on the roadside. According to Hafetz et al. (2010), the driver being less likely to engage in traffic accident if not using smart phone while driving.

The objective of this project is to develop a real-time driver monitoring system which are able to monitor driver action in real-time by using computer vision technique. The system developed was able to capture the frame of driver activity, after that it will generate a human pose frame which link together human key points to represent as more meaningful input image for classification model compare to original image. Then the system is able to classify the driver current action and determine whether the driver is performing secondary tasks or not. Hence, it is able to verify whether to give alert to the driver. This system will be implemented on Raspberry Pi which run with Raspbian operating system, the program will be written in Python in (.py) file format.

This designed system is able to classify 10 different classes of driver actions as the dataset is obtain from State Farm. The classes include safe driving, texting on left hand, texting on right hand, talk to the phone on right hand, talk to the phone on left hand, operating the radio, drinking or eating, reaching behind, makeup, and talking to passengers. Except for

Chapter 7: Conclusion

the first class and last class which is safe driving and talking to passengers, other classes are consider as secondary task which might increase the traffic accident rate. Hence, the buzzer which connected to the Raspberry Pi will buzz when dangerous act is performed during driving.

For future enhancement and development, perhaps others mini motherboard which is similar to Raspberry Pi for example like Nano Jetson could be replaced for this system. It is because Nano Jetson contain GPU which belongs to Nvidia that had cuda device that can support any trained model to be run on GPU that can achieve faster speed during live process. In this case, instead of only using one frame for classification, few frames could be used for tracking driver behavior in order to further understand driver action. Since GPU provide powerful processing power for deep learning model, hence time taken to process one frame should be very fast which allowed system to do the tracking unlike Raspberry Pi that use CPU which compute slower that might took around 7 or even more seconds for tracking where this will failed to be a live processing framework. Besides that, night vision camera can be another pick to be implement into the system too. It can assist the driver even during the night time to ensure the camera could capture driver movement properly for better classification.

Current designed system is using only vision-based technique to perform driver action detection. Perhaps in future the system could also implement vehicle-based measure for example like steering wheel detection to detect whether driver hands are on steering or not. Moreover, by integrate with other's frameworks such as driver drowsiness detection, the system may be improved by further reduce car accidents rate happen all around the world.

In addition, it is suggested to add in extra datasets that could able to detect when driver is perform reverse parking. For current framework, when driver is performing reverse parking and during the time when head is turn to back, the model might classify it as reaching behind hence alert will be given. Hopefully in future some enhance could be done for example like integrate the system with vehicle system to pause the system when driver is performing reverse parking or included dataset where driving is performing reverse parking.

REFERENCE

- B, J. and M Patil, C., 2018. Video Based Human Activity Detection, Recognition and Classification of actions using SVM. *Transactions on Machine Learning and Artificial Intelligence*, 6(6).
- Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S. and Sheikh, Y. (2019). OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp.1-1.
- Cdc.gov. (2018). *Distracted Driving | Motor Vehicle Safety | CDC Injury Center*. [online] Available at: <https://www.cdc.gov/motorvehiclesafety/distracted_driving/>.
- Chollet, F., 2017. Xception: Deep Learning with Depthwise Separable Convolutions. In: *Computer Vision and Pattern Recognition (CVPR)*. Honolulu: IEEE.
- Cho Nilar, P., Thi Thi, Z. and Pyke, T., 2020. Skeleton Motion History based Human Action Recognition Using Deep Learning. In: *6th Global Conference on Consumer Electronics (GCCE 2017)*. Las Vegas: IEEE.
- Choudhary, P. and Velaga, N. (2019). A comparative analysis of risk associated with eating, drinking and texting during driving at unsignalised intersections. *Transportation Research Part F: Traffic Psychology and Behaviour*, 63, pp.295-308.
- D'Sa, Ashwin & Prasad, B. (2019). An IoT Based Framework For Activity Recognition Using Deep Learning Technique.
- Hafetz, J., Jacobsohn, L., García-España, J., Curry, A. and Winston, F. (2010). Adolescent drivers' perceptions of the advantages and disadvantages of abstention from in-vehicle cell phone use. *Accident Analysis & Prevention*, 42(6), pp.1570-1576.
- Huang Y., Lai SH., Tai SH. (2019) Human Action Recognition Based on Temporal Pose CNN and Multi-dimensional Fusion. In: Leal-Taixé L., Roth S. (eds) *Computer Vision – ECCV 2018 Workshops*. ECCV 2018. Lecture Notes in Computer Science, vol 11130. Springer, Cham.

- Jegham, I., Ben Khalifa, A., Alouani, I. and Ali Mahjoub, M. (2018). Safe Driving: Driver Action Recognition using SURF Keypoints. In: *2018 30th International Conference on Microelectronics (ICM)*. Tunisia: IEEE, pp.60-63.
- Lawrence, N. (2018). *Distracted driving, cellphones seen as factors in pedestrian deaths*. [online] The Star Online. Available at: <<https://www.thestar.com.my/tech/tech-news/2018/07/02/distracted-driving-cellphones-seen-as-factors-in-pedestrian-deaths>>.
- Lee, J., Lee, J., Bärgrman, J., Lee, J. and Reimer, B. (2018). How safe is tuning a radio?: using the radio tuning task as a benchmark for distracted driving. *Accident Analysis & Prevention*, 110, pp.29-37.
- LUM, D. (2019). *We have the third highest death rate from road accidents*. [online] The Star Online. Available at: <<https://www.thestar.com.my/lifestyle/health/2019/05/14/we-have-the-third-highest-death-rate-from-road-accidents>>.
- Malaymail.com., 2019. *Traffic police: More than 280,000 road accidents nationwide in first half 2019 Malay Mail*. [online] Available at: <<https://www.malaymail.com/news/malaysia/2019/07/17/bukit-aman-more-than-280000-road-accidents-nationwide-in-first-half-2019/1772104>>.
- Motus (2018). *Car Accidents Increase 12.3 Percent with the Rise of the Always-Connected Mobile Workforce, Finds New Motus Distracted Driving Report*. Boston: Business Wire.
- Pang, Y., Syu, S., Huang, Y. and Chen, B., 2020. An Advanced Deep Framework for Recognition of Distracted Driving Behaviors. In: *7th Global Conference on Consumer Electronics (GCCE 2018)*. Las Vegas: IEEE.
- Rolison, J., Regev, S., Moutari, S. and Feeney, A. (2018). What are the factors that contribute to road accidents? An assessment of law enforcement views, ordinary drivers' opinions, and road accident records. *Accident Analysis & Prevention*, 115, pp.11-24.

- Wang, L., Wang, Z., Xiong, Y. and Qiao, Y., 2016. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In: *The 14th European Conference on Computer Vision (ECCV)*. Amsterdam.
- Who.int. (2018). *Road traffic injuries*. [online] Available at: <<https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>>.
- Yan, C., Coenen, F., & Zhang, B. (2016). *Driving posture recognition by convolutional neural networks*. *IET Computer Vision*, 10(2), 103–114. doi:10.1049/iet-cvi.2015.0175
- Yan, S., Teng, Y., S.Smith, J. and Zhang, B. (2016). Driver Behavior Recognition Based on Deep Convolutional Neural Networks. In: *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. Changsha: IEEE, pp.636-641.
- Zhang, Y., Li, J., Guo, Y., Xu, C., Bao, J. and Song, Y. (2019). Vehicle Driving Behavior Recognition Based on Multi-View Convolutional Neural Network with Joint Data Augmentation. *IEEE Transactions on Vehicular Technology*, 68(5), pp.4223-4234.

POSTER

ACTION DETECTION SYSTEM FOR ALERTING DRIVER USING COMPUTER VISION

Khoo Chia Hong

Supervisor: Dr. Lau Phooi Yee

Problem Statement



Nowadays, secondary tasks performed by drivers in the middle of driving are the predominant cause that leads to traffic accidents. The technologies and facilities, such as smartphones or drive-thru services were created for customers' convenience, which gives the chance for drivers to contact or eating when they were driving. Therefore, activities such as playing mobile phone, using social applications, adjusting radio, or eating are the major causes which will leads to the traffic accidents.

Objectives



Main objective: To design and develop a real-time driver action monitoring system using Raspberry Pi that can mount inside vehicle, which will alert the driver when performing secondary tasks to decrease the car accident.

Sub-objectives:

1. To record the driver action in real-time through camera.
2. To do the classification of driver's action accurately in real-time.
3. To give alert to the driver when tasks performed is dangerous.

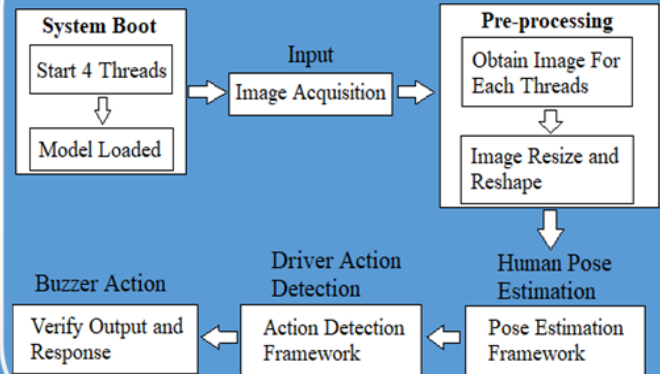
Conclusions

In conclusion, secondary tasks is very dangerous if performed by drivers during driving. This project is developed for detecting driver's action and thus giving alert through the buzzer connected on Raspberry Pi when they perform secondary tasks.

Project Scope

- ◆ Develop action detection system in secondary tasks performed for alerting driver using computer vision.
- ◆ By tracking human body landmarks movement.
- ◆ Implement system in Raspberry Pi with parallel processing technique.

Methodology



Results



Left picture show the input obtained from Pi Camera. Right picture show the output after generated the human pose frame and classify it. Buzzer will trigger if the action is not safe driving. Human pose is generated with parallel processing technique and 8 key points is detected and linked together.

FYP2

ORIGINALITY REPORT

11%

SIMILARITY INDEX

4%

INTERNET SOURCES

7%

PUBLICATIONS

7%

STUDENT PAPERS

PRIMARY SOURCES

1	Yong Zhang, Junjie Li, Yaohua Guo, Chaonan Xu, Jie Bao, Yunpeng Song. "Vehicle Driving Behavior Recognition Based on Multi-View Convolutional Neural Network With Joint Data Augmentation", IEEE Transactions on Vehicular Technology, 2019 Publication	1%
2	"Computer Vision – ECCV 2018 Workshops", Springer Nature, 2019 Publication	1%
3	Submitted to KDU College Sdn Bhd Student Paper	1%
4	link.springer.com Internet Source	<1%
5	Submitted to Brickfields Asia College Student Paper	<1%
6	Coenen, Frans, Bailing Zhang, and Chao Yan. "Driving posture recognition by convolutional neural networks", IET Computer Vision, 2016. Publication	<1%

7	Submitted to The Hong Kong Polytechnic University Student Paper	<1%
8	Submitted to University of Central Lancashire Student Paper	<1%
9	Submitted to University of Wales, Bangor Student Paper	<1%
10	Submitted to University of New South Wales Student Paper	<1%
11	Submitted to University of Hull Student Paper	<1%
12	Submitted to Study Group Australia Student Paper	<1%
13	dl.gi.de Internet Source	<1%
14	Submitted to Monash University Student Paper	<1%
15	export.arxiv.org Internet Source	<1%
16	www.iitr.ernet.in Internet Source	<1%
17	Shilpa Gite, Himanshu Agrawal. "Early Prediction of Driver's Action Using Deep Neural Networks", International Journal of Information	<1%

Retrieval Research, 2019

Publication

18

Alejandro Hernandez Ruiz, Lorenzo Porzi, Samuel Rota Bulò, Francesc Moreno-Noguer. "3D CNNs on Distance Matrices for Human Action Recognition", Proceedings of the 2017 ACM on Multimedia Conference - MM '17, 2017

Publication

<1%

19

Deepika Sirohi, Neeraj Kumar, Prashant Singh Rana. "Convolutional neural networks for 5G-enabled Intelligent Transportation System : A systematic review", Computer Communications, 2020

Publication

<1%

20

"Advances in Information and Communication", Springer Science and Business Media LLC, 2020

Publication

<1%

21

Shiyang Yan, Yuxuan Teng, Jeremy S. Smith, Bailing Zhang. "Driver behavior recognition based on deep convolutional neural networks", 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016

Publication

<1%

22

www.businesswire.com

Internet Source

<1%

- 23 Imen JEGHAM, Anouar BEN KHALIFA, Ihsen ALOUANI, Mohamed Ali MAHJOUB. "Safe Driving : Driver Action Recognition using SURF Keypoints", 2018 30th International Conference on Microelectronics (ICM), 2018
Publication <1%
-
- 24 Submitted to Phuket International Academy Day School
Student Paper <1%
-
- 25 Yu-Ting Pang, Sin-Wun Syu, Yi-Chi Huang, Bo-Hao Chen. "An Advanced Deep Framework for Recognition of Distracted Driving Behaviors", 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE), 2018
Publication <1%
-
- 26 hdl.handle.net
Internet Source <1%
-
- 27 journals.scholarpublishing.org
Internet Source <1%
-
- 28 Cho Nilar Phyo, Thi Thi Zin, Pyke Tin. "Skeleton motion history based human action recognition using deep learning", 2017 IEEE 6th Global Conference on Consumer Electronics (GCCE), 2017
Publication <1%
-
- 29 Submitted to University of Utah

<1%

30

"Biometric Recognition", Springer Science and Business Media LLC, 2017

Publication

<1%

31

Jun Guo, Sunwoo Kim, Henk Wymeersch, Walid Saad, Wei Chen. "Guest Editorial: Introduction to the Special Section on Machine Learning-Based Internet of Vehicles: Theory, Methodology, and Applications", IEEE Transactions on Vehicular Technology, 2019

Publication

<1%

32

"Computer Analysis of Images and Patterns", Springer Science and Business Media LLC, 2019

Publication

<1%

33

Utku Aydonat, Shane O'Connell, Davor Capalija, Andrew C. Ling, Gordon R. Chiu. "An OpenCL™ Deep Learning Accelerator on Arria 10", Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '17, 2017

Publication

<1%

34

Submitted to University of Surrey

Student Paper

<1%

35

www.palgrave.com

Internet Source

<1%

36

Submitted to KYUNG HEE UNIVERSITY

Student Paper

<1%

37

Submitted to Universiti Sains Malaysia

Student Paper

<1%

38

news.wellrx.com

Internet Source

<1%

39

Submitted to The University of Manchester

Student Paper

<1%

40

Submitted to University of Exeter

Student Paper

<1%

41

Fang Wu, Jigang Wang, Jiqiang Liu, Wei Wang.
"Vulnerability detection with deep learning",
2017 3rd IEEE International Conference on
Computer and Communications (ICCC), 2017

Publication

<1%

42

"Medical Image Understanding and Analysis",
Springer Science and Business Media LLC,
2017

Publication

<1%

43

Submitted to Segi University College

Student Paper

<1%

44

Chao Yan, Bailing Zhang, Frans Coenen. "Multi-
attributes gait identification by convolutional
neural networks", 2015 8th International

<1%

Congress on Image and Signal Processing (CISP), 2015

Publication

-
- | | | |
|----|---|-----|
| 45 | Imen Jegham, Anouar Ben Khalifa, Ihsen Alouani, Mohamed Ali Mahjoub. "Vision-based human action recognition: An overview and real world challenges", Forensic Science International: Digital Investigation, 2020
Publication | <1% |
| 46 | Submitted to University of Derby
Student Paper | <1% |
| 47 | Submitted to Manchester Metropolitan University
Student Paper | <1% |
| 48 | Submitted to Nanyang Technological University, Singapore
Student Paper | <1% |
| 49 | Submitted to University of Westminster
Student Paper | <1% |
| 50 | Submitted to International Islamic University Malaysia
Student Paper | <1% |
| 51 | Submitted to CTI Education Group
Student Paper | <1% |
| 52 | "Computer Vision – ECCV 2016", Springer Nature, 2016 | <1% |

53	pdfs.semanticscholar.org Internet Source	<1 %
54	Submitted to Coventry University Student Paper	<1 %
55	www.vitlokens.se Internet Source	<1 %
56	Submitted to University of Portsmouth Student Paper	<1 %
57	Submitted to Multimedia University Student Paper	<1 %
58	Submitted to University of Northumbria at Newcastle Student Paper	<1 %
59	Submitted to Universiti Teknologi MARA Student Paper	<1 %
60	www.plainsandeasterneis.com Internet Source	<1 %
61	Submitted to Universiteit van Amsterdam Student Paper	<1 %
62	S. M. Masudur Rahman Al Arif, Karen Knapp, Greg Slabaugh. "Fully automatic cervical vertebrae segmentation framework for X-ray images", Computer Methods and Programs in	<1 %

Biomedicine, 2018

Publication

63

www.mdpi.com

Internet Source

<1%

64

Submitted to Universiti Tenaga Nasional

Student Paper

<1%

65

"Recent Advances in Information and Communication Technology 2020", Springer Science and Business Media LLC, 2020

Publication

<1%

66

ikee.lib.auth.gr

Internet Source

<1%

67

Submitted to The University of Buckingham

Student Paper

<1%

68

Submitted to University of Nottingham

Student Paper

<1%

69

"Hybrid Artificial Intelligent Systems", Springer Science and Business Media LLC, 2019

Publication

<1%

70

Submitted to Xianjiaotong-Liverpool University

Student Paper

<1%

71

Submitted to University of East London

Student Paper

<1%

72

Submitted to SAE Institute, London

Student Paper

<1%

73

Submitted to University of Bahrain

Student Paper

<1%

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



**FACULTY OF INFORMATION AND COMMUNICATION
TECHNOLOGY**

Full Name(s) of Candidate(s)	Khoo Chia Hong
ID Number(s)	16ACB01757
Programme / Course	Bachelor of Computer Science (HONS)
Title of Final Year Project	Action Detection System For Alerting Driver Using Computer Vision

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u>11</u> % Similarity by source Internet Sources: <u>4</u> % Publications: <u>7</u> % Student Papers: <u>7</u> %	
Number of individual sources listed of more than 3% similarity: <u>None</u>	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

 Signature of Supervisor
 Name: Lau Phooi Yee, PhD
 Date: 20 April 2020

 Signature of Co-Supervisor
 Name: _____
 Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION & COMMUNICATION
TECHNOLOGY (KAMPAR CAMPUS)
CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	16ACB01757
Student Name	Khoo Chia Hong
Supervisor Name	Dr. Lau Phooi Yee

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Front Cover
√	Signed Report Status Declaration Form
√	Title Page
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p> <p style="text-align: center;"><i>Khoo Chia Hong</i></p> <p>(Signature of Student) Date: 20 April 2020</p>	<p>Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.</p> <p style="text-align: center;"><i>JM.</i></p> <p>(Signature of Supervisor) Date: 20 April 2020</p>
--	---