

**VISION-BASED REAL-TIME  
TRAFFIC CONTROL USING  
ARTIFICIAL NEURAL  
NETWORK ON  
GENERAL-PURPOSE  
EMBEDDED HARDWARE**

Hithanadura Kavindu Gimhan Zoysa

(188470C)

Dissertation submitted in partial fulfillment of the requirements for the degree  
Master of Science in Electronic and Automation

Department of Electronic and Telecommunication Engineering

University of Moratuwa

Sri Lanka

August 2020

## DECLARATION

---

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

Date:

The above candidate has carried out research for the Masters Dissertation under my supervision.

Name of the supervisor: Prof. Rohan Munasinghe

Signature of the Supervisor:

Date:

## Abstract

In urban cities, traffic management of intersections is a substantially challenging problem. Inappropriate traffic control leads to waste of fuel, time, and productivity of nations. Though the traffic signals are used to control traffic, it often causes problems due to the pre-programmed timing being not appropriate for the actual traffic intensity at the intersection. Traffic intensity determination based on statistical methods only gives the average intensities expected at any given time. However, to control traffic effectively, the knowledge of real-time traffic intensity is a must-have. In this project, vision-based technology and artificial intelligence (AI) are used to estimate traffic in real-time and control the traffic in order to reduce the traffic congestion. General-purpose electronic hardware has been used for in-situ image processing based on edge-detection methods. A Neural Network (NN) was trained to infer traffic intensity in each image in real-time using a scale of 1(very low) to 5 (very high). A Trained AI unit, which takes approximately 4 seconds to process each image and estimate traffic intensity was tested on the road where it recorded a 90% acceptance rate. In order to control the traffic, a ratio-based method and a reinforcement learning (RL)-based method was used. The performance of these methods are compared with a pre-programmed traffic controller.

***Keywords***-traffic sensing, traffic control, neural network, reinforcement learning

## ACKNOWLEDGMENTS

---

First, I wish to express my sincere appreciation to my supervisor, Professor Rohan Munasinghe. I was privileged to work under his guidance. I thank him for his support, encouragement and patience made my success reality. Without his persistent help, the goal of this project would not have been realized.

Then, I would like to extend my gratitude to Doctor Chamira Edussooriya for his guidance and the support I had over the past two years as the coordinator of my MSc program. Also, my appreciation goes to the staff members and all of my friends who participated in the MSc program in the Department of Electronic and Telecommunication Engineering for their invaluable support.

This work was supported by World Bank AHEAD ITS project and Innovation Quotient Pvt Ltd.

# TABLE OF CONTENTS

---

<b>Declaration</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Acknowledgments</b>	<b>iii</b>
<b>Table of Contents</b>	<b>vi</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>ix</b>
<b>List of abbreviations</b>	<b>x</b>
<b>List of Appendices</b>	<b>xi</b>
<b>1 INTRODUCTION</b>	<b>1</b>
<b>2 DESIGN OF TRAFFIC INTENSITY ESTIMATION</b>	<b>5</b>
2.1 Overall design . . . . .	5
2.2 Edge Detection for traffic intensity estimation . . . . .	7

2.3	Estimation of traffic using a neural network . . . . .	13
<b>3</b>	<b>TEST TRAFFIC INTENSITY ESTIMATION METHOD</b>	<b>20</b>
3.1	Traffic intensity Estimation using sidebar images . . . . .	20
3.2	Traffic intensity estimation using crossbar images . . . . .	20
<b>4</b>	<b>DESIGN OF TRAFFIC CONTROLLER</b>	<b>27</b>
4.1	Simulator . . . . .	27
4.2	Evaluators . . . . .	28
4.3	Basic Configurations . . . . .	28
4.4	Real world configurations . . . . .	30
4.5	Ratio-based traffic controller . . . . .	31
4.5.1	Methodology . . . . .	31
4.5.2	Results . . . . .	31
4.5.3	Discussion . . . . .	31
4.6	RL-based traffic controller . . . . .	32
4.6.1	Methodology . . . . .	32
4.6.2	Results . . . . .	33
4.6.3	Discussion . . . . .	33
<b>5</b>	<b>CONCLUSIONS AND RECOMMENDATIONS</b>	<b>37</b>



## LIST OF FIGURES

---

2.1	Overall design of traffic intensity estimation . . . . .	6
2.2	Sidebar image and crossbar image . . . . .	7
2.3	Three zones of the sidebar image and crossbar image . . . . .	8
2.4	Edges count . . . . .	9
2.5	Lane marks that are detected as edges . . . . .	10
2.6	Zero traffic image . . . . .	11
2.7	An image after removing erroneous edges . . . . .	12
2.8	Flow of detecting and counting the edges . . . . .	12
2.9	Example images to explain the annotation criteria . . . . .	14
2.10	NN used for real-time traffic intensity estimation . . . . .	14
2.11	ReLU activation function . . . . .	16
2.12	Flow of training the NN . . . . .	17
3.1	Traffic intensity estimation using sidebar images . . . . .	21
3.2	Traffic intensity estimation using crossbar images . . . . .	24
4.1	The two way isolated intersection . . . . .	29



4.2	Map of the junction used to obtain phase timing . . . . .	30
4.3	Real world phases . . . . .	30
4.4	Performance of ratio-based controller . . . . .	34
4.5	Flow of the RL-based controller . . . . .	35
4.6	Performance of RL-based controller . . . . .	36

## LIST OF TABLES

---

2.1	Criterion to Remove Erroneous Pixels . . . . .	8
2.2	Traffic intensity of sidebar image annotated by an expert . . . . .	18
2.3	Traffic intensity of crossbar image annotated by an expert . . . . .	19
3.1	Accuracy of the sidebar estimation . . . . .	21
3.2	Compare expected values and estimated values for sidebar images	22
3.3	Accuracy of the crossbar estimation . . . . .	24
3.4	Compare expected values and estimated values for crossbar imaeges	25

## LIST OF ABBREVIATIONS

---

Abbreviation	Description
AI	Artificial Intelligence
NN	Neural Network
RL	Reinforcement Learning
GPS	Global Positioning System
YOLO	you-only-look-once
ReLU	Rectified Linear Unit
SUMO	Simulation of Urban Mobility
TraCI	Traffic Control Interface

## LIST OF APPENDICES

---

Appendix	Description	Page
RELATED CODES	Python codes related to performance	40

## INTRODUCTION

---

In most parts of the world, particularly in urban cities, vehicular traffic increases steadily because of the increase in population, lack of effective traffic management systems, careless driving styles, unplanned urban cities, etc. Since the past few decades, governments, scientists, and researchers have been focusing on different kinds of traffic management and control systems. However, traffic congestion is a main problem for urban cities all over the world [1].

Despite the constant development and technology introduction traffic management and control has not been able to cope up with the increasing traffic. Inappropriate traffic control at intersections has been found increasingly frequent in urban cities in the developing world where variation of traffic intensity is large and difficult to be modelled statistically. As a result of inappropriate signal control, the average speed of vehicles drops leading to huge losses in fuel and waste of time [2]. Also it can lead to drivers becoming frustrated and engaging in road rage. Consequently, such countries will have to face issues such as urban air pollution, health issues, and long term effects on the economy and overall development. In this view, effective traffic control has become one of the critical issues for those countries.

During the last decade, technologies including vision-based methods have been introduced to sense traffic and learning-based methods to control traffic [3]. Vision-based systems could be very effective because they could provide real time traffic information, which is key to effective traffic control. In fact, the major

reason for inappropriate traffic control at intersections is the pre-programmed static timing, which opens and closes the lanes without knowing the actual traffic situation at that moment.

In this project, we focus on sensing and controlling the traffic in an isolated two way intersection. What is being implemented in traffic signal control at intersections is a static time schedule, which is in effect for a certain period of the day. For other times of the day, there are other schedules, which are all static and customized to different times of the day. The approach in this method is to break the day into a number of time slots based on traffic flow variation during the day assuming that the traffic flow remains unchanged within such periods [4]. This assumption is often found incorrect. Yet due to the absence of a better solution, it is being practiced despite its inherent imperfections. The researchers mainly focus on the two aspects of traffic management and control as mentioned below.

- Traffic sensing
- Traffic controlling

Most of the research papers are written focusing on these two aspects. According to that, we have divided our project into these two aspects.

Before Emerging vision-based methods were used for traffic sensing, traffic was detected using loop detectors [5]- [6], which are designed combining analog and digital circuits. In some researches, Global Positioning System (GPS) data [7] is used to analyse queue length and control delays. Also, in some cases, they have used a sensor network to detect vehicles [8].

After researchers move to vision-based methods, they have introduced different kinds of technologies to sense and control traffic. The you-only-look-once (YOLO) [9] detectors are used to count the number of vehicles because accuracy of the YOLO detector is very high. However, it is computationally as well as cost-wise a

very expensive algorithm [10]. Also there are several robust algorithms [11] that have been introduced to count vehicles.

One objective of this research is to devise a method to estimate accurate traffic intensity in real time. It is not expected to have the millisecond level real time execution because comparatively phase time is large. Traffic images taken using a general-purpose miniature camera are processed in-situ on general -purpose electronic hardware that runs artificial intelligence inside to estimate traffic intensity with reasonable accuracy in real time. The AI entity used in this research is a neural network (NN). Traffic intensity is estimated based on edge-detection and supervised learning of the NN using a set of annotated traffic images. Major attention was given to the accuracy of traffic intensity estimate and the time taken to derive that estimate. All the technologies used in this project should be implemented in low cost. Because for single intersection, at least there should be 4 units and for long run we need to develop more units. It a major objective to make the traffic sensing unit financially affordable for mass scale production and deployment on roads. In order to minimize the cost, we had to stop increasing accuracy and decreasing execution time at the explained level.

Considering the second aspect, which is controlling the traffic, at the very beginning researchers had been trying to develop mathematical models [12], [13] and [14] for traffic control systems. However, with the development of machine learning and AI algorithms, researchers used those theories to build traffic controllers. After that, they used supervised learning models [15] such as conventional neural networks, deep neural etc. At the present, researchers use unsupervised learning methods such as reinforcement learning [16], [17].

In this research, we have implemented and simulated two types of controllers to control the traffic in an isolated two way junction. Each controller is compared with a conventional pre-programmed traffic management system.

- Ratio-based traffic controller

This controller measures the traffic of four incoming roads. At the beginning of each green phase, the green phase time is calculated according to the ratio of the traffic of four incoming roads.

- Reinforcement learning-based traffic controller

This controller maintains a q-table to decide the action and the q-table is updated according to the Q Learning algorithm [18]. After each action, the current state changes to a new state and there is a corresponding reward to each action. The current state, new state, and reward is used to calculate the new q value according to the Q Learning algorithm.



## DESIGN OF TRAFFIC INTENSITY ESTIMATION

---

### 2.1 Overall design

In this project, our main purpose is to control the traffic signals in an isolated two-way intersection in order to reduce traffic. At the first phase, the traffic needs to be measured and at the second the traffic signals should be controlled using the measured traffic values. This section discusses the method, which is used to estimate the traffic. Figure 2.1 shows the overall design flow of estimation of the traffic intensity. Vision-based methods and conventional neural network theories are used to implement the model to estimate the traffic. First, a trained neural network is required which can be fed with the information, which is extracted from the image and it should be able to give the traffic intensity level. Then, in the other part, the information should be extracted from the image captured real time and those information should be fed to the neural network. Chapter 4 discusses the methods, which are used to control the traffic.

One of the major requirements in using vision-based methods for traffic estimation is to make it fast enough for real time implementation. Nevertheless, sophisticated, fast vision-based methods are also not appropriate because they are not financially viable for wide-scale implementation. Therefore, general-purpose electronic hardware gets attention as a prospective means of a solution. To make such hardware actually acceptable, it is essential to stick to vision processing algorithms that are reasonably simple and executable with sufficient speed on

such hardware. Our vision-based system runs on each image it captures. Each image is processed to extract the key information, which are fed to the neural network for traffic intensity estimation. In this process, information extraction is the most time-consuming aspect if sophisticated algorithms are used. In this regard, counting the number of vehicles in an image is a very high cost computationally, and generally not implementable for general-purpose hardware. On the other hand, well-established edge-detection methods are not only fast but also implementable on general-purpose hardware. Therefore, this research attempts to develop a vision-based edge detection method for real time traffic intensity estimation using artificial intelligence.

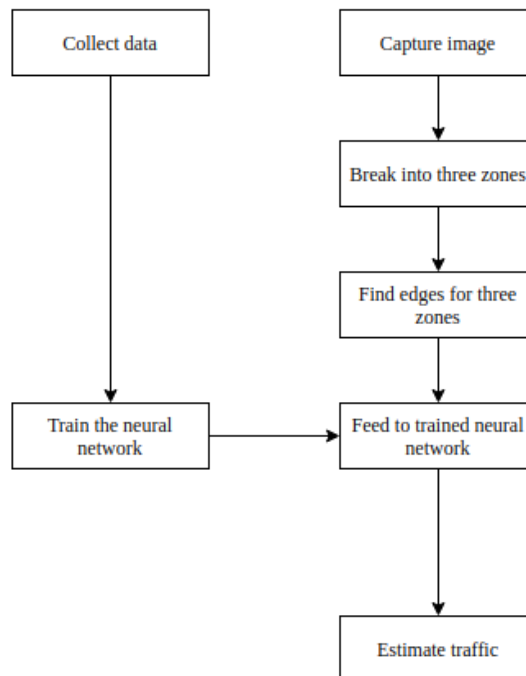


Figure 2.1: Overall design of traffic intensity estimation

Under the image processing part, the image captured in real time is divided into three zones to find the edges count in each zone. Then, the edges count is fed to the trained neural network and the neural network gives the estimated traffic as the output. Thus, the overall design can be divided into two parts as mentioned below.

- Edge detection and counting for traffic intensity estimation
- Traffic estimation using a neural network

## 2.2 Edge Detection for traffic intensity estimation

When there is huge traffic in the road, more edges can be found in an image. Also there is not a huge traffic, edges count is low. Therefore, it should be able to build a relationship between edges count and traffic intensity. This section discusses how to detect and count edges for traffic intensity estimation. The traffic intensity estimation method used in this project will be tested for two types of images, which are taken from two angles. Those are a sidebar image and a crossbar image (Figure 2.2).



Figure 2.2: Sidebar image and crossbar image

In order to capture the traffic intensity distribution from the intersection along the road to about 100m, three zones were defined as near, mid, and far. Due to the vehicles present in each zone, the edges on the image are identified separately and the pixel count of the edges are calculated. Edges are detected using the Canny edge detector [19], which is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. The three zones defined for each angle are shown in Figure 2.3. After dividing the original image into three



Figure 2.3: Three zones of the sidebar image and crossbar image

zones, the number of edges can be calculated for three zones. Example images are shown in Figure 2.4.

After the initial training, it was realized that the neural network finds it difficult to recognize traffic level 1 (lowest) but always produces level 2. It was found that the features shown in Figure 2.5 such as lane marks on the road that were erroneously detected as vehicle edges caused this issue. Usually, such features are obscured by the vehicles when there are many vehicles in the near field (traffic levels 2 and above), but when there aren't many vehicles (traffic level 1), these features appear and erroneously get counted. Therefore, a new algorithm shown in Table 2.1 was incorporated to remove these marks from the image before counting vehicle edges.

Table 2.1: Criterion to Remove Erroneous Pixels

Pixel from the image where there are no vehicles	Pixel from real time image where there are vehicles	Decision
black	black	black
black	white	black/white
white	white	black
white	black	black

To remove the non-vehicular edges, the real time image is compared with a zero-traffic image as shown in Figure. 2.6. Then, each image is compared with

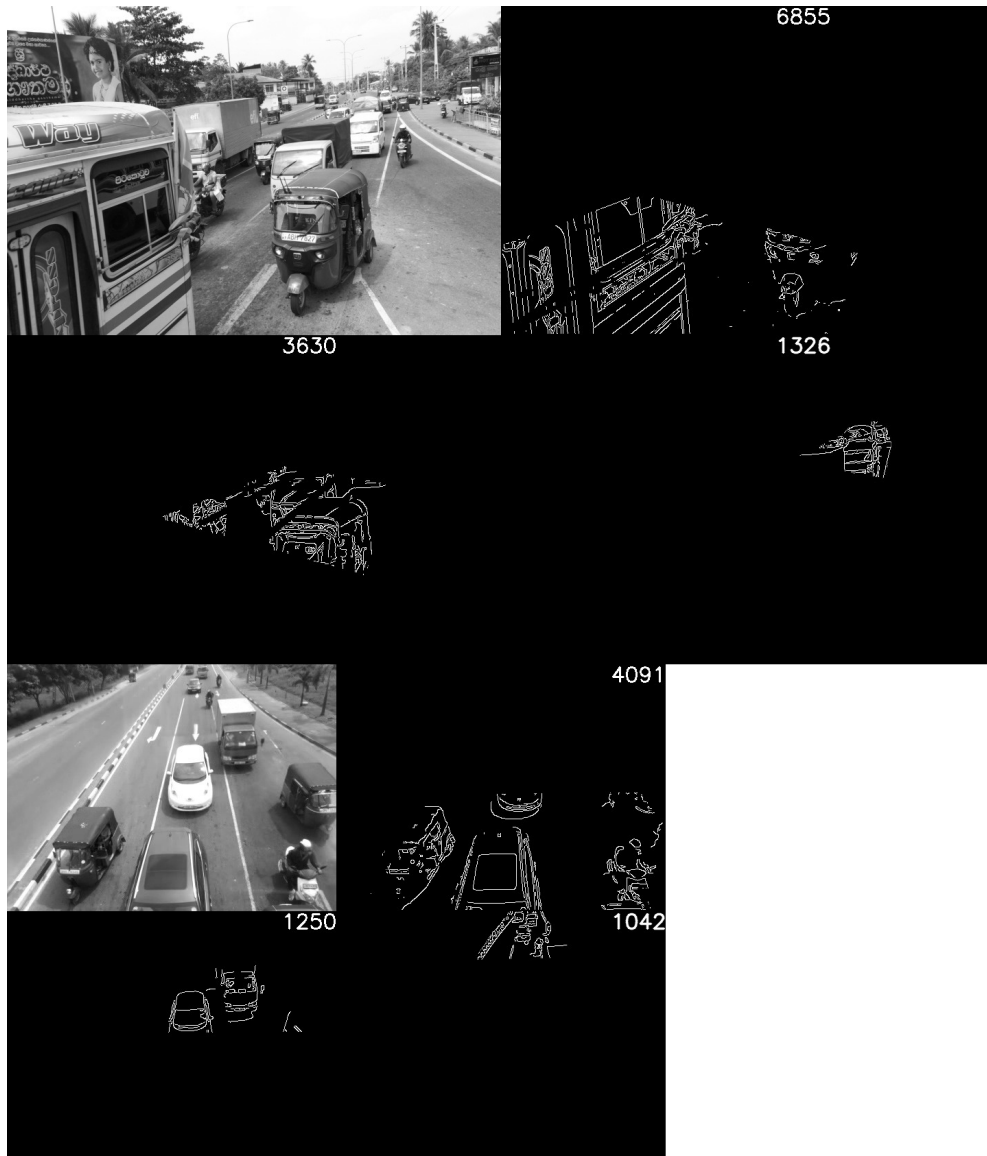


Figure 2.4: Edges count

the zero-traffic image pixel by pixel according to Table 2.1 and only the pixels that result in black/white as in the second row are unable to be directly counted as a vehicular or non-vehicular edge.

The pixels that are white on the image where there are no vehicles are surely due to erroneous detection so they have to be neglected (the decision is black). However, if a pixel, which is black in the image where there are no vehicles, becomes white in the real time image, it can be due to a vehicle edge or non-vehicular edge (the decision can be white or black). The reason can be explained

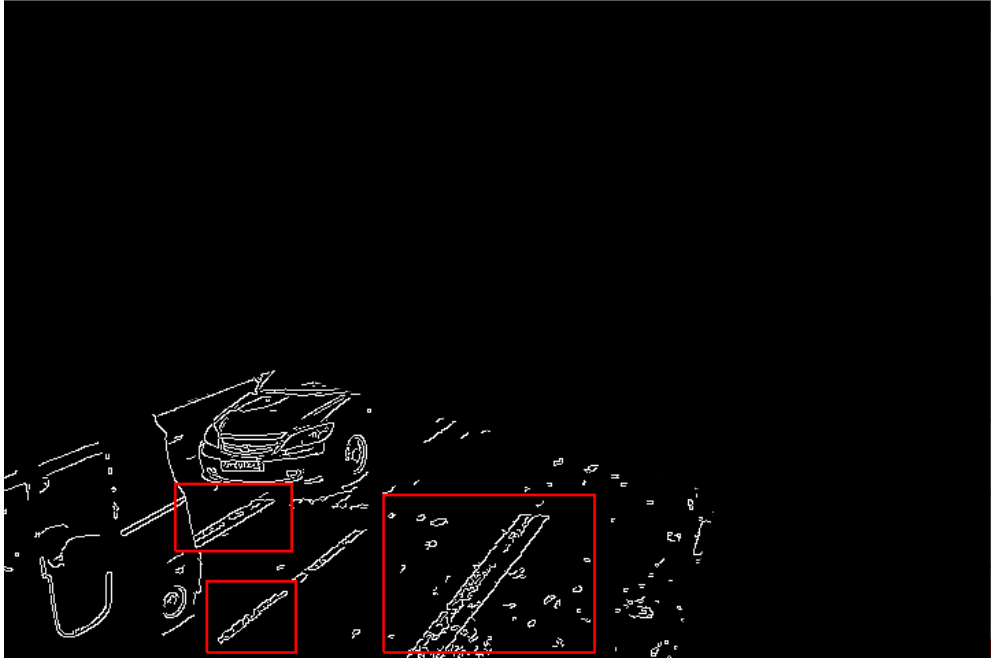


Figure 2.5: Lane marks that are detected as edges

as follows. If the pixel is white in the real time image, it can be a vehicular edge or non-vehicular edge. If the corresponding pixel in the image where there are no vehicles is black (corresponds to the second row in Table 2.1), it cannot be directly determined that the decision is black. However, in this scenario, the corresponding pixel in the image where there are no vehicles can be a non-vehicular pixel (while) because of the low intensity and the small displacement of the image and it is not shown in the image as a white pixel.

In order to classify these situations accurately, an algorithm shown in Algorithm 1 is executed for each pixel. According to the algorithm, a small kernel is considered and checked whether there is any white pixel within that kernel in the image where there is no vehicle. If at least a single white pixel is found, the decision is considered as black.

When the above algorithm is executed in Raspberry-pi, critical performance issues were faced. Since the algorithm iterates over the image several times, it increases the execution time and reduces the performance.



Figure 2.6: Zero traffic image

---

**Algorithm 1** Removing Erroneous Edges

---

**Require:** : zero traffic image, real time image

**Ensure:** : edges count

```
for position in zero_traffic_image do  
     $h = position[0]$   
     $w = position[1]$   
    if ( $zero\_traffic\_image[h, w] == 0$  &  $real\_time\_image[h, w] == 255$ ) then  
        create a kernel in zero_traffic_image  
        if (at least one pixel is white in kernel) then  
             $edge\_count+ = 1$ ;  
        end if  
    end if  
end for
```

---

Since the real time execution is mandatory, it is essential to overcome the performance issue. Therefore, the code needs to be changed not to iterate over the image several times. For that, first the position of images are obtained into an array and iterated over that array, which contains the positions.

According to this method, the performance of finding the edges gets increased. Since this performance is not enough to achieve the real time traffic estimation goal, it had to be improved the performance further. Therefore, the solution was to change the code to execute as multi processes in each core. Now, the algorithm

runs simultaneously in each core for different areas in the image. In this way, it was able to increase the performance to get the edges counted. Figure 2.7 shows how this criterion removes erroneously detected edges from an image. Figure. 2.8 depicts the full flow of edge detection and counting.



Figure 2.7: An image after removing erroneous edges

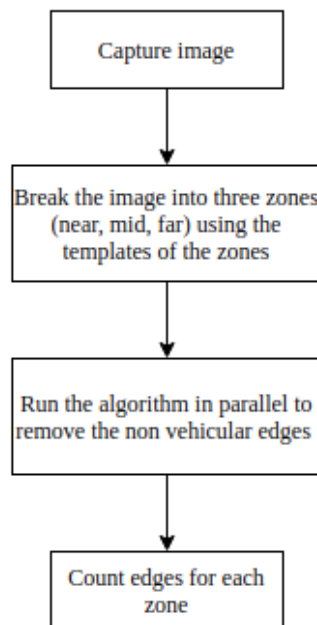


Figure 2.8: Flow of detecting and counting the edges



### 2.3 Estimation of traffic using a neural network

The reason to use a neural network to determine the traffic intensity is based on the observation of a non-linear relationship between the edge count and traffic intensity. A well-trained NN can estimate the output of a non-linear relationship with high accuracy. In the first part of the process, a Raspberry Pi electronic development board was used to take a set of traffic images using a Raspberry Pi camera. These images are visually inspected and annotated by an expert for traffic intensity.

Under the annotation process, it does not mainly focus on the number of vehicles. For example, let's consider two scenarios. The first one is that the considered area of the road is filled with light vehicles such as motor bicycles, cars, etc. The second one is that the considered area of the road is filled with heavy vehicles such as buses, lorries etc.

The first case is shown in the left side image of Figure 2.9 and the second case is shown in the right side image of Figure 2.9. Number of vehicles in the first case is greater than the second case. However, in the first case, those large numbers of vehicles do not affect the traffic compared with the second case. The reason is, in the first case, the mobility of the vehicles is greater than the second case. Thus, the first case is annotated with a lower traffic level than the second case. According to the explanation, the traffic level of the left side image is annotated as 3 and the traffic level of the right side image is annotated as 4. Like this, under the annotation process, experts mainly concern about the reason, which can increase the traffic. Example images are shown in Table 2.2 and 2.3.

A conventional neural network is trained using these training images of which the traffic intensities are known. The neural network training takes place in the lab on a high-speed computer. Once the NN is trained, the process on the left is started where the trained NN is copied from the computer to the Raspberry



Figure 2.9: Example images to explain the annotation criteria

Pi board, which is brought onto the road intersection to capture incoming traffic and evaluate traffic intensity in real time.

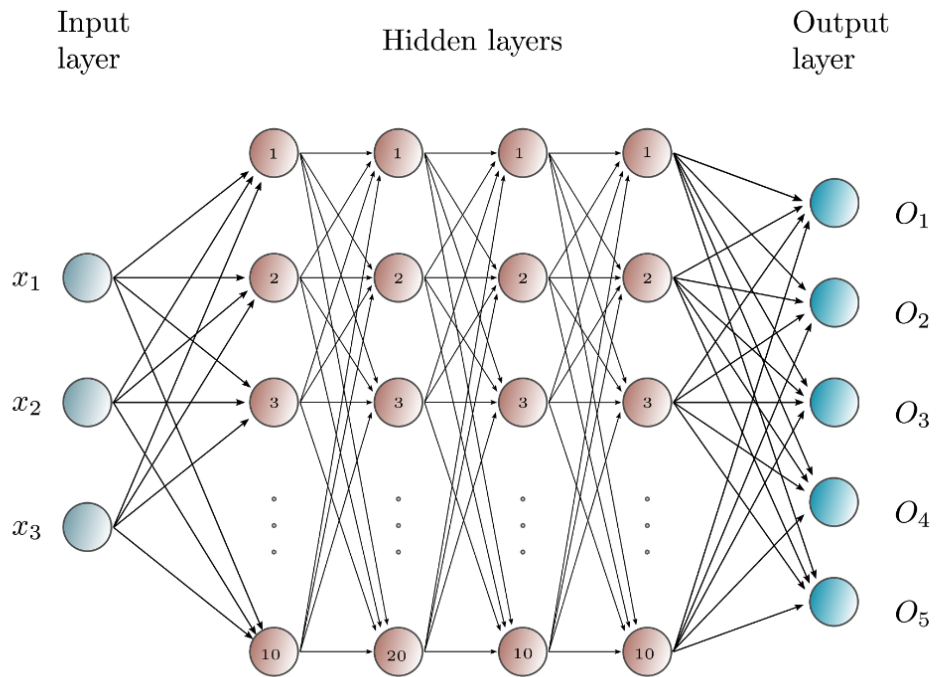


Figure 2.10: NN used for real-time traffic intensity estimation

The number of pixels on all the edges of each zone of an image is counted as input information to the NN. The NN shown in Figure 2.10 consists of three input neurons to receive the edge pixel counts of the three zones of an image. The output of the deep neural network consists of five neurons. Each neuron

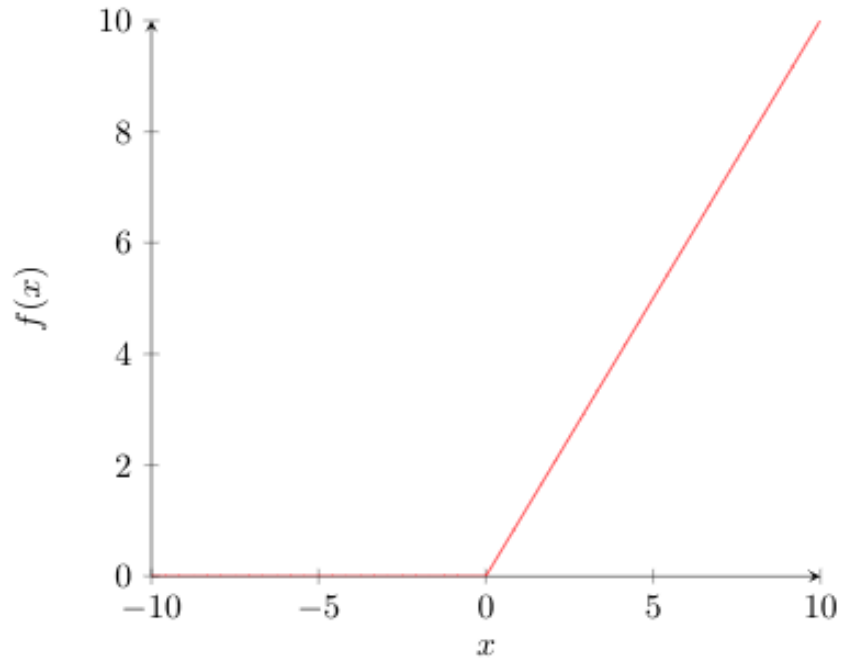
represents the traffic intensity from one to five. The average of those five neurons is considered as the traffic intensity.

There are four hidden layers in the network and each hidden layer contains 10, 20, 10, and 10 neurons from input to output. In the NN, Softmax cross-entropy loss is used as loss function and the equation is shown in 2.1. The activation function used in NN is Rectified Linear Unit (ReLU) and the formula and graph is shown in Figure 2.11. It is the most commonly used activation function in deep neural networks. Initially sigmoid activation function is used for this neural network. After having several tests, it was realized that ReLU activation function gives more accurate results than sigmoid activation function. There are several reasons for that such as,

1. It does not contain complicated math, and thereby, it does not take a huge computational cost. Also, the model takes less time to train or run.
2. It converges faster. Since the linearity of the function, the slope does not saturate when input gets large. It does not have the vanishing gradient problem suffered by other activation functions like sigmoid or tanh.

$$J = \frac{1}{N} \sum_{i=1}^{i=N} y_i \log(\hat{y}_i) \quad (2.1)$$

Tensorflow was used to implement the NN. Training images are annotated by the expert on a scale of 1 (very low) to 5 (very high). 80% of the training images were used for training and the rest 20% was used for accuracy evaluation. Figure 2.12 shows the training process, which is a recursive algorithm that runs through a repeating loop until the error reduces to a given value. In each loop, it uses the 20% of the images put aside initially for error calculation. The general-purpose embedded platform, Raspberry Pi does not have enough computational power to train a neural network and neither there is a need for this implementation because the neural network is trained once upfront. Therefore, training the network offline



$$f(x) = \max(0, x)$$

Figure 2.11: ReLU activation function

is more appropriate. Moreover, this method makes it convenient to duplicate the trained neural network on multiple units.

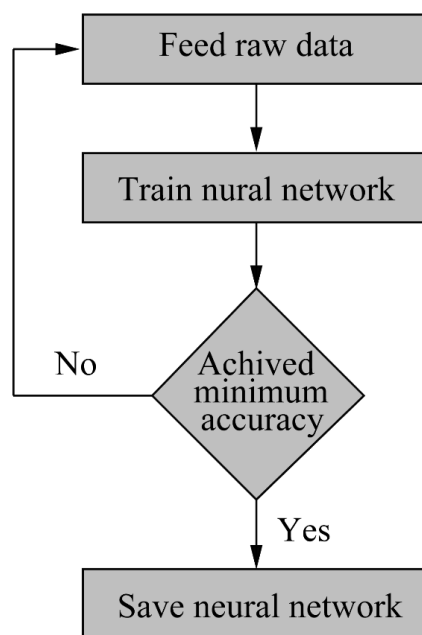


Figure 2.12: Flow of training the NN

Table 2.2: Traffic intensity of sidebar image annotated by an expert











Image	Traffic level determined by the expert
	1
	2
	3
	4
	5

Table 2.3: Traffic intensity of crossbar image annotated by an expert

Image	Traffic level determined by the expert
	1
	2
	3
	4
	5

## TEST TRAFFIC INTENSITY ESTIMATION METHOD

---

This chapter discusses how the used method was tested for crossbar images and sidebar images. Also, the test results are presented for each type of image.

### 3.1 Traffic intensity Estimation using sidebar images

In Figure 3.1, the flow is depicted for an image captured in the sidebar. It shows all the steps discussed in chapter 2. Table 3.2 shows the example images taken from the sidebar and the comparison between expected traffic level and estimated traffic. Also Table 3.1 gives the rate of accuracy of the estimation done by the NN. According to the results table, it can achieve 64% of perfect estimation and 30% of one step difference estimation. It can be considered that the accuracy is approximately 94% as one step difference estimation can be added to the perfect estimation.

### 3.2 Traffic intensity estimation using crossbar images

This section discusses how traffic estimation method is applied to the images taken from the crossbar of the road. The flow of estimating the traffic level for the crossbar images is shown in Figure 3.2. Table 3.4 shows the example images taken from the crossbar and traffic level annotated by an expert. Also Table



3.3 gives the rate of accuracy of the estimation done by the NN. According to the results table, it can achieve 66% of perfect estimation and 29% of one step difference estimation. It can be considered that the accuracy is approximately 95%.

Table 3.1: Accuracy of the sidebar estimation

Perfect judgement	64%
+1 or -1 step difference	30%

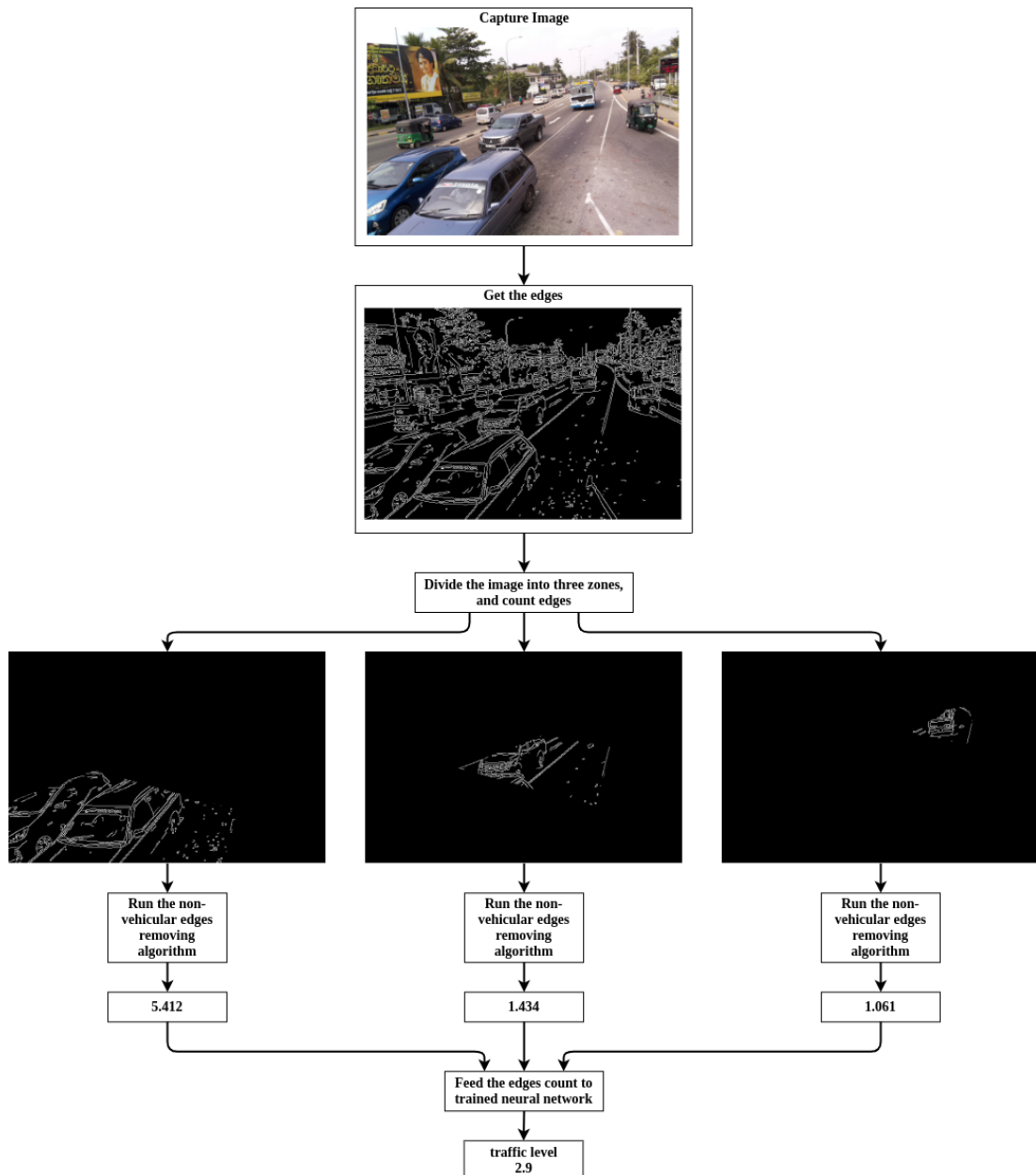







Figure 3.1: Traffic intensity estimation using sidebar images

Table 3.2: Compare expected values and estimated values for sidebar images

Real-time sidebar image	Expected traffic intensity	Estimated traffic intensity
	5	3.9
	4	3.9
	3	2.9
	2	1.9
	1	1.0






Real-time sidebar image	Expected traffic intensity	Estimated traffic intensity
	5	3.9
	4	3.9
	3	2.9
	2	2.0
	1	1.3

Table 3.3: Accuracy of the crossbar estimation

Perfect judgement	66%
+1 or -1 step difference	29%

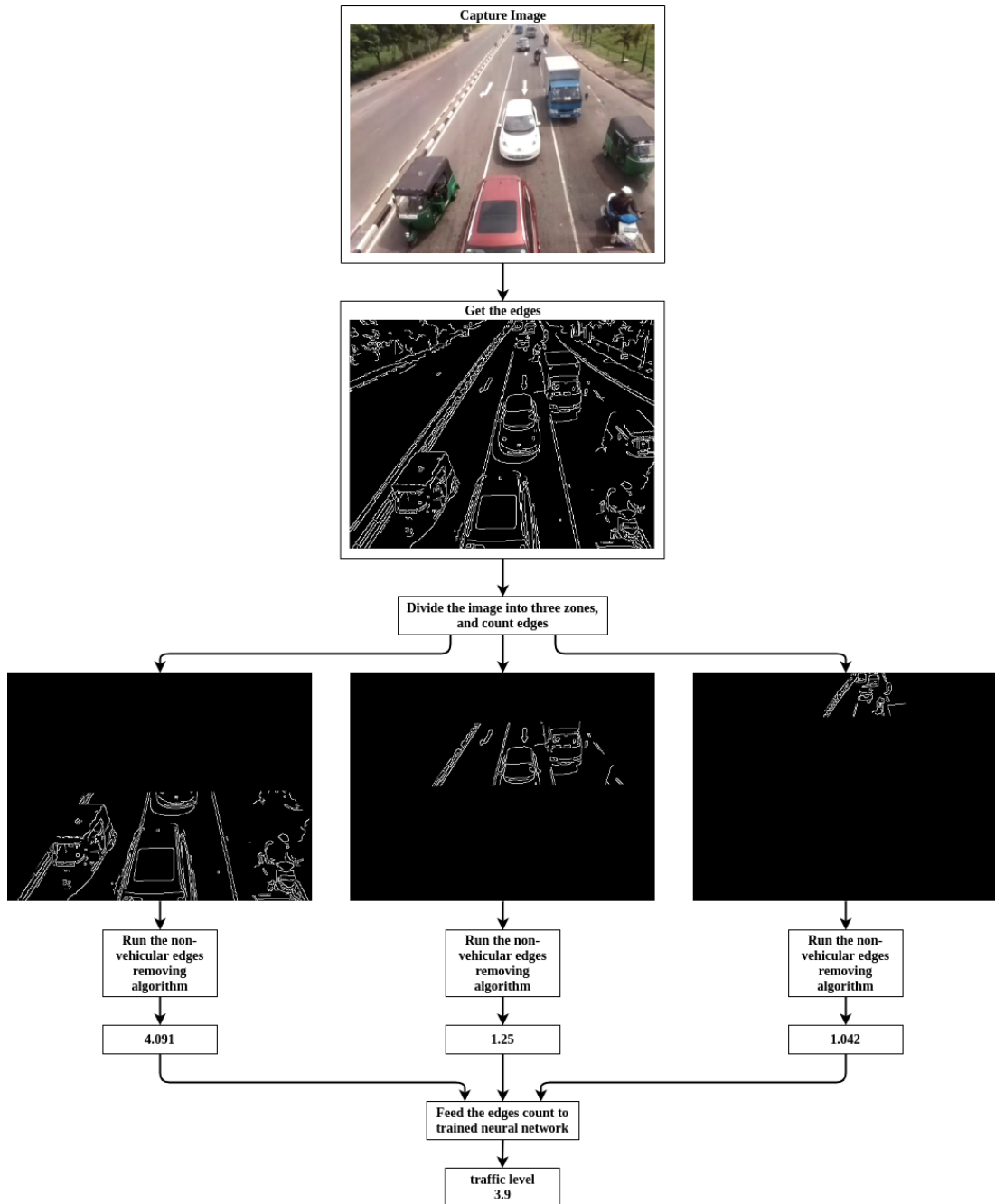
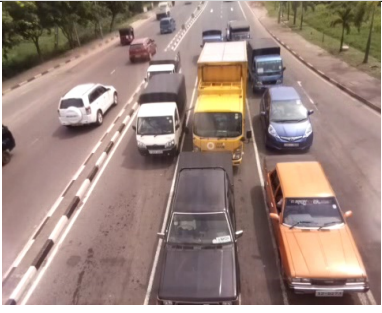











Figure 3.2: Traffic intensity estimation using crossbar images

Table 3.4: Compare expected values and estimated values for crossbar images

Real-time crossbar image	Expected traffic intensity	Estimated traffic intensity
	5	3.9
	4	3.9
	3	2.9
	2	1.9
	1	1.0

Real-time crossbar image	Expected traffic intensity	Estimated traffic intensity
	5	3.9
	4	3.9
	3	2.9
	2	1.9
	1	1.0

## DESIGN OF TRAFFIC CONTROLLER

---

The second part of this project is about controlling the traffic in a two-way isolated intersection in order to reduce traffic. At the very beginning of the start of these types of research, researchers mainly focused on implementing mathematical models to represent the traffic and control it. However, with the emergence of learning-based methods, researchers moved to use them. At the beginning of that period, they used supervised learning methods. When considering the traffic controlling, the input-output dataset cannot be given as the action cannot be predicted. Due to this drawback, researchers moved to unsupervised learning methods such as reinforcement learning (RL). This project uses two types of controllers as the ratio-based traffic controller and RL-based traffic controller. Sections 4.5 and 4.6 are allocated to discuss each of these controllers.

### 4.1 Simulator

Since it is hard to implement and test this concept in the real world, SUMO (Simulation of Urban Mobility) was used as a simulator, which is an open source, highly portable, microscopic, and continuous road traffic simulation package designed to handle large road networks. Also, TraCI (Traffic Control Interface) is used to access SUMO using Python. By using SUMO, a two way isolated junction was modelled and each road of the junction has two lanes. It can be used to pre-define the distribution of the vehicle insertion of each incoming road. It

was tried to produce traffic congestion at least in one incoming road. As an open source simulator, SUMO is easy to manipulate.

## 4.2 Evaluators

Evaluators are required to check the success or the failure of controllers and compare them with existing pre-programmed systems. Most of the researchers use average queue length and average waiting time as evaluators. In this project, average speed and departed vehicle count are used as evaluators.

- Queue length (meters) - The average of the length of all queued vehicles in the incoming road. In reality, the space between the vehicles should be added to the queue length. However, in the simulation it is not added because the space is a constant value.
- Average waiting time (seconds per vehicle) - The average of the waiting time of all the vehicles that wait in the queue. In SUMO, The waiting time of a vehicle is defined as the time spent with a speed below 0.1m/s since the last time it was faster than 0.1m/s.
- Average speed (m/s per vehicle) - The average speed of all the vehicles, which are available in the field.

## 4.3 Basic Configurations

In SUMO, a two way isolated intersection (shown in figure 4.1) was implemented with an 8 phase pre-programmed traffic controller. The minimum time allowed to move vehicles is 20 seconds and 10 seconds are allocated for pedestrians. (Please note that these values can be changed.) In both controllers, the phase duration is changed in different ways. However, in each controller, we made



sure that the time allocated to pedestrians does not change. Thus, the time allowed for pedestrians is not affected by the new controllers. Therefore, the safety of the pedestrians is ensured.

To estimate the traffic in the simulation, it is unable to use the method explained in chapter 2. Therefore, the number of vehicles available in each incoming road is taken as the measurement for the traffic intensity. The distribution of this estimated value should be equivalent to the distribution of the value estimated in chapter 2. In that distribution, the probability of perfect estimation is between 0.6 and 0.7 and the probability of one-step estimation is approximately 0.3. In the simulation, this distribution is replicated with these probabilities.

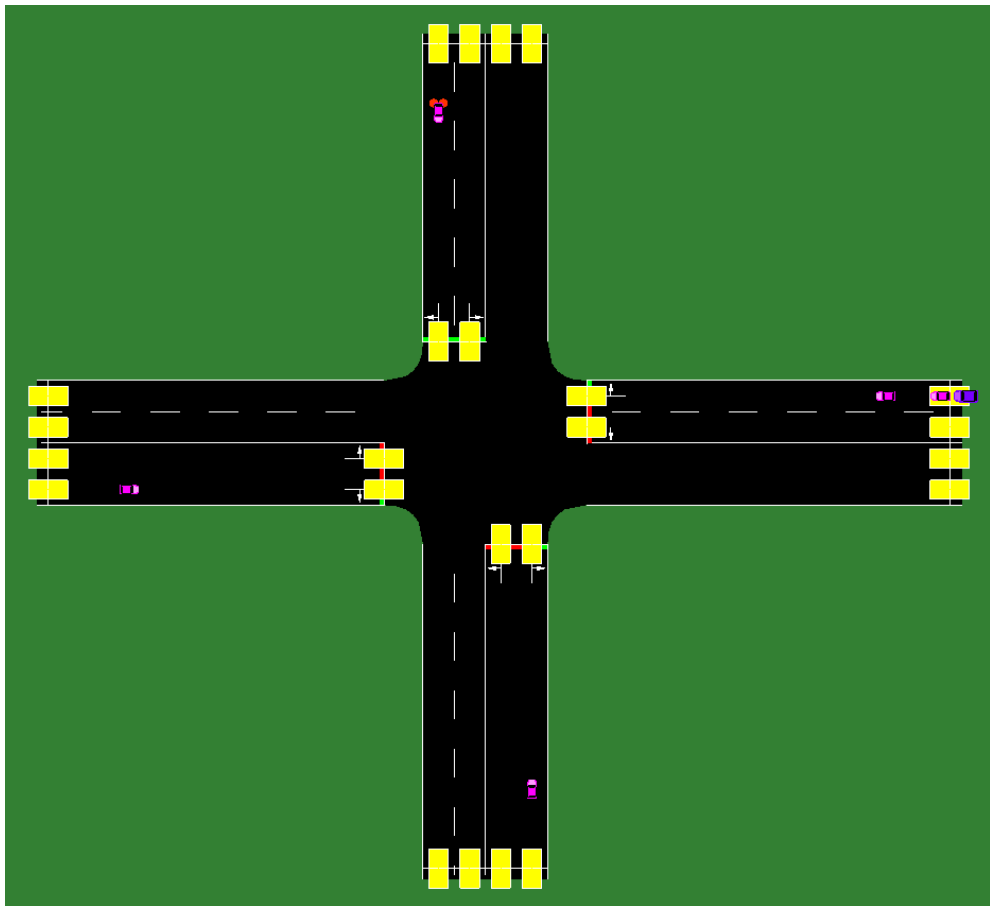


Figure 4.1: The two way isolated intersection

#### 4.4 Real world configurations

Since we need to execute and compare our controlling methods based on real world phase timing, we have selected the isolated intersection shown in figure 4.2 to get the phase timing. There is an eight phase controller in that junction. We have collected the data in the morning peek and it could be seen that for one direction there was a huge traffic. Therefore for that direction, 45 seconds were allowed to the green phase. All other green phases have 15 seconds. Phases are depicted in Fig 4.3.

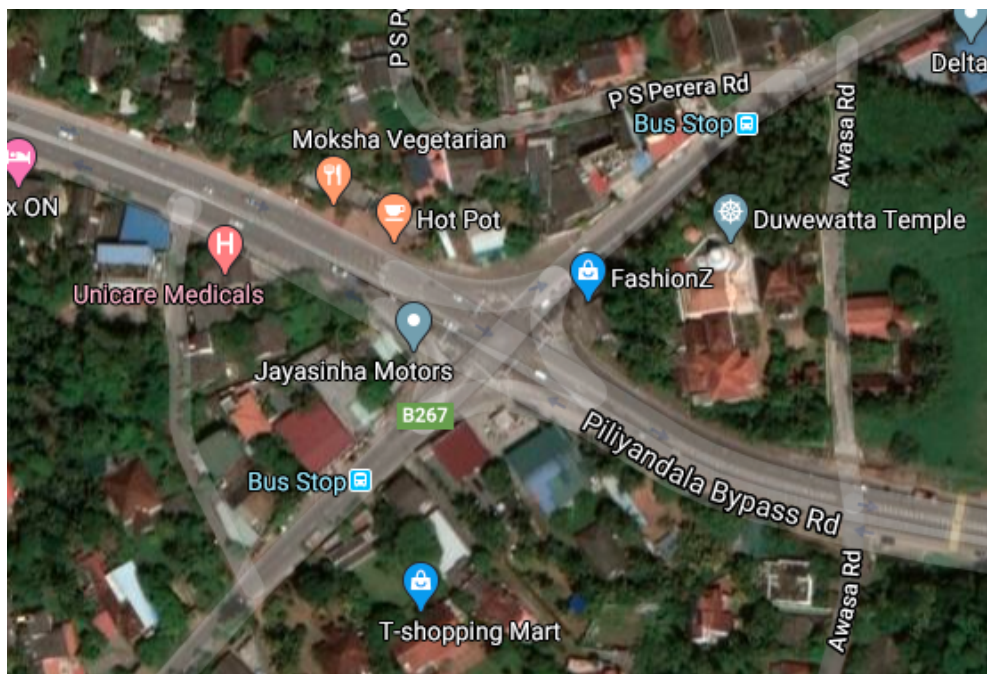


Figure 4.2: Map of the junction used to obtain phase timing

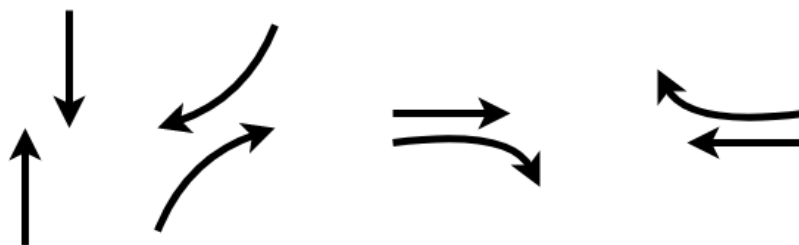


Figure 4.3: Real world phases

## 4.5 Ratio-based traffic controller

### 4.5.1 Methodology

At the beginning of the green phase, traffic intensity of 4 incoming roads are calculated and a new green phase duration is calculated according to the ratio of four traffic values. If the new green phase duration is less than the minimum phase duration, new phase duration is ignored and the minimum phase duration is executed. The method is illustrated in Algorithm 2. Initially ratio based controller starts from the pre-programmed controller mentioned in section 4.4.

---

**Algorithm 2** Calculate green phase time according to the ratio of traffic

---

**Require:** : traffic values

**Ensure:** : update the phase duration if it does not exceed minimum phase time

**while** until all vehicles have entered **do**

**if** new green phase started **then**

$traffic = \text{traffic in four incoming roads}$

$total\_traffic = \sum traffic$        $ratio\_of\_traffic = traffic / total\_traffic$

$phase\_duration = ratio\_of\_traffic * TOTAL\_TIME$

**if**  $phase\_duration > MINIMUM\_PHASE\_DURATION$  **then**

            update the phase duration

**end if**

**end if**

**end while**

---

### 4.5.2 Results

The figure 4.4 depicts the comparison between the ratio-based controller and static controller.

### 4.5.3 Discussion

According to these results, after the 500 th second, average queue length and average waiting time is reduced and average speed is increased. Therefore it can be concluded that ratio based controller can be successfully used to control the

traffic. The main advantage of this controller is that, it can be used without any training process. But it should be programmed according to the phase sequence of the junction.

## 4.6 RL-based traffic controller

### 4.6.1 Methodology

In RL, it is mandatory to define the state, action, reward, and the learning algorithm. Each one is described below.

- State - The traffic intensity of incoming roads and the current phase is defined as the state. Therefore, the state consists of five values. First four values indicate the traffic intensity of four incoming roads and the last value indicates the phase number.
- Action - In any RL design, there can be several actions. In this case, two actions are defined. First one is that extending the current phase by 5 seconds and second action is moving to the next phase and executing it.
- Reward - The reduction of queue length after executing the decided action is defined as reward. If the queue length is reduced, the reward is a positive value and if the queue length is increased, then the reward is negative.
- Learning algorithm - Q learning is used as the learning algorithm. For each state and action in the q table has a q value. Q value is updated according to the equation shown in equation 4.1.

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max(Q(s')) - Q(s, a)] \quad (4.1)$$

where:

s current state

- a current action
- s' next state
- r reward

In order to fill the q table with proper values, the simulator should run several times. Each iteration starts by executing an initial action. After the action is executed, the environment obtains a new state and the reward is calculated based on the executed action. At this phase, the q table gets updated. This process repeats for a given number of iterations (Figure 4.5). At the initial state of the evaluation process, it reads the action, which has the maximum q value for a given state and that action is executed.

#### **4.6.2 Results**

The figure 4.6 depicts the comparison between the ratio-based controller and static controller.

#### **4.6.3 Discussion**

According to these results, after the 500 th second, average queue length and average waiting time is reduced and average speed is increased. Like ratio based controller, rl-based controller can be used to control the traffic successfully. The disadvantage of this controller is that, it should be trained before use in the actual environment. Since this controller only depends on current traffic of incoming roads and current phase number, it can be deployed in any junction without doing any changes. This is the main advantage of rl-based controller when it is compared with ratio-based controller.

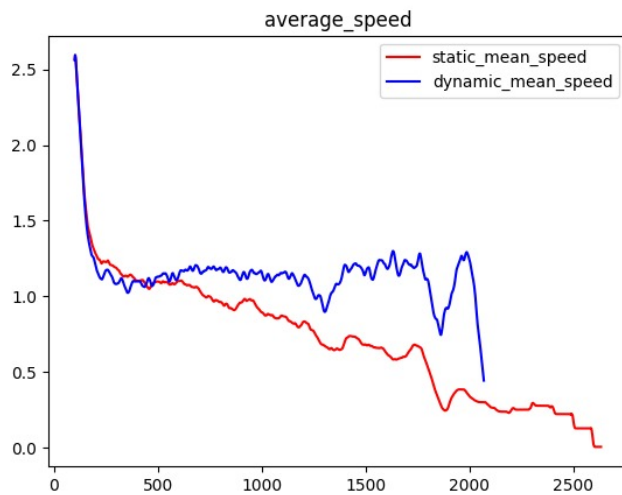
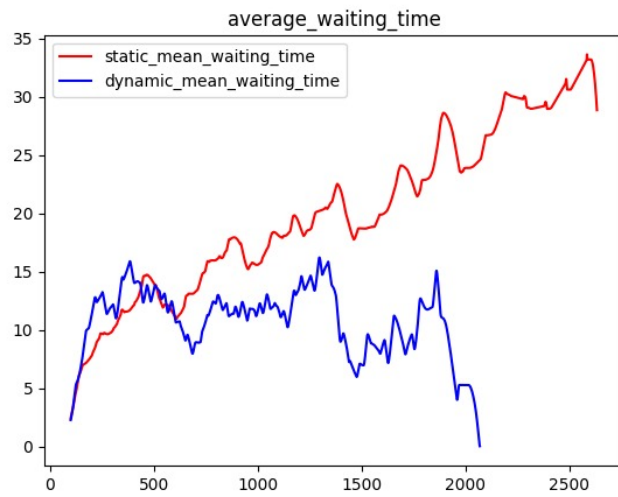
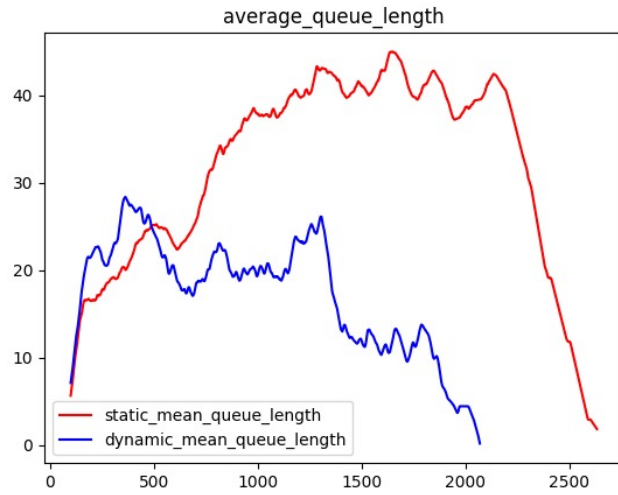


Figure 4.4: Performance of ratio-based controller

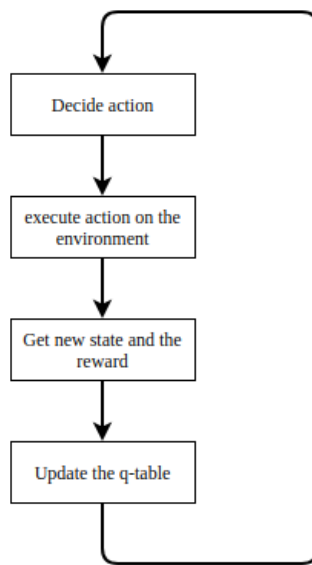


Figure 4.5: Flow of the RL-based controller

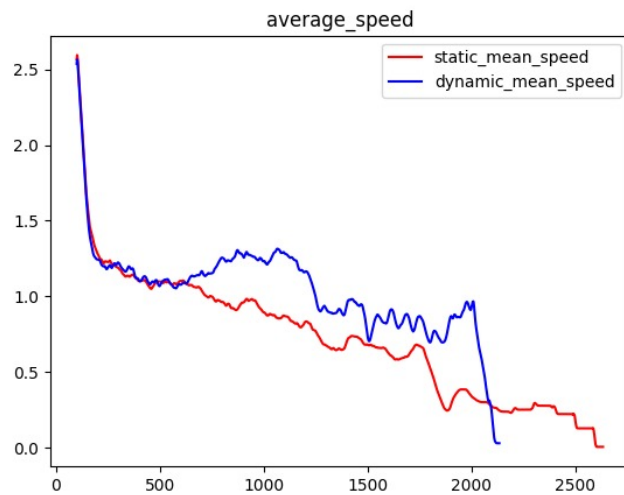
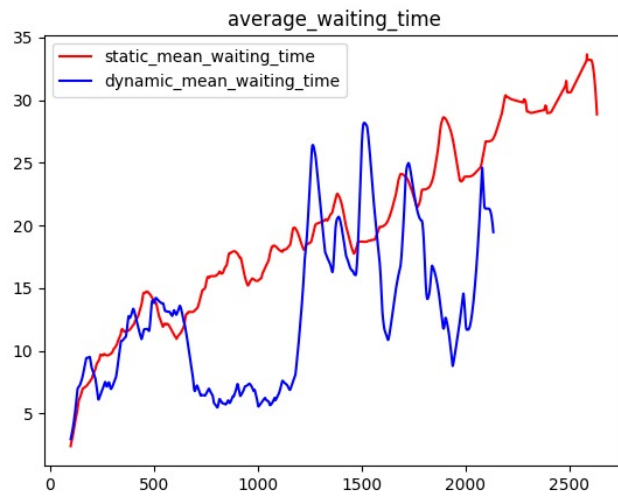
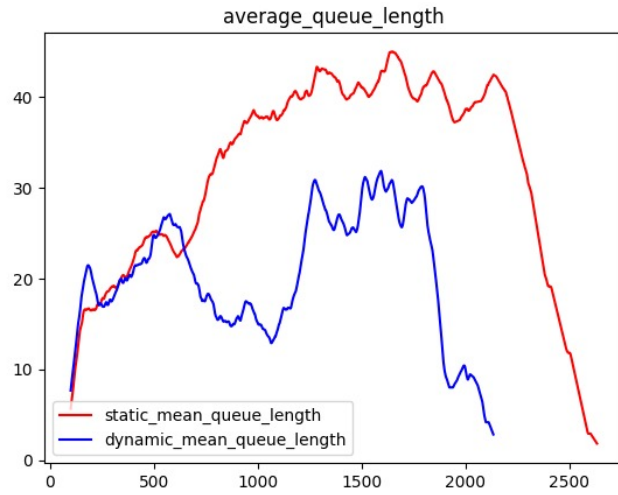


Figure 4.6: Performance of RL-based controller



## CONCLUSIONS AND RECOMMENDATIONS

---

In this research, vision and image processing technologies with artificial intelligence have been utilized to estimate traffic intensity in real time. A deep neural network was trained using a training image set where traffic intensity was specified by an expert on a scale of [1, 5]. Trained NN was tested for a test image set in that 65% of the images were perfectly judged and 30% of the images were judged very closely. This research also restricted to general purpose, low cost hardware to make the findings financially affordable for mass scale deployment. Under the control of traffic in an isolated two way junction we have developed two methods, the first one is a ratio-based method and the other one is a RL-based method. In order to compare the performance of these methods, we used existing real world phase timing in an isolated two way junction. Here we consider the queue length, average waiting time, and average speed as evaluators. When comparing the existing time scheduler with our methods, it can be seen that queue length and average waiting time is reduced and the average speed is reduced.

## BIBLIOGRAPHY

---

- [1] R. Arnott and K. Small, “The economics of traffic congestion,” *American scientist*, vol. 82, no. 5, pp. 446–455, 1994.
- [2] M. Barth and K. Boriboonsomsin, “Real-world carbon dioxide impacts of traffic congestion,” *Transportation Research Record*, vol. 2058, no. 1, pp. 163–171, 2008.
- [3] S. R. E. Datondji, Y. Dupuis, P. Subirats, and P. Vasseur, “A survey of vision-based traffic monitoring of road intersections,” *IEEE transactions on intelligent transportation systems*, vol. 17, no. 10, pp. 2681–2698, 2016.
- [4] L. Brederode, M. Bliemer, and L. Wismans, “Staq: Static traffic assignment with queuing,” in *European Transport Conference*. Citeseer, 2010.
- [5] R. L. Bruce, “Loop detector for traffic signal control,” Feb. 7 1984, uS Patent 4,430,636.
- [6] J. Gajda, R. Sroka, M. Stencel, A. Wajda, and T. Zeglen, “A vehicle classification based on inductive loop detectors,” in *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Re-discovering Measurement in the Age of Informatics (Cat. No.01CH 37188)*, vol. 1, May 2001, pp. 460–464 vol.1.
- [7] L. Tišljarić, T. Erdelić, and T. Carić, “Analysis of intersection queue lengths and level of service using gps data,” in *2018 International Symposium EL-MAR*. IEEE, 2018, pp. 43–46.

- [8] S. Coleri, S. Y. Cheung, and P. Varaiya, “Sensor networks for monitoring traffic,” in *Allerton conference on communication, control and computing*, 2004, pp. 32–40.
- [9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [10] J. Lin and M. Sun, “A yolo-based traffic counting system,” in *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, Nov 2018, pp. 82–85.
- [11] J. F. Song, A. N. Bai, and R. Xue, “A reliable counting vehicles method in traffic flow monitoring,” in *2011 4th International Congress on Image and Signal Processing*, vol. 1, Oct 2011, pp. 522–524.
- [12] T.-H. Chang and J.-T. Lin, “Optimal signal timing for an oversaturated intersection,” *Transportation Research Part B: Methodological*, vol. 34, no. 6, pp. 471–491, 2000.
- [13] P. B. Mirchandani and N. Zou, “Queuing models for analysis of traffic adaptive signal control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 50–59, 2007.
- [14] Q. He, K. L. Head, and J. Ding, “Multi-modal traffic signal control with priority, signal actuation and coordination,” *Transportation research part C: emerging technologies*, vol. 46, pp. 65–82, 2014.
- [15] D. Jia and Z. Chen, “Traffic signal control optimization based on fuzzy neural network,” in *Proceedings of 2012 International Conference on Measurement, Information and Control*, vol. 2. IEEE, 2012, pp. 1015–1018.
- [16] S. Mikami and Y. Kakazu, “Genetic reinforcement learning for cooperative traffic signal control,” in *Proceedings of the First IEEE Conference on Evolu-*

*tionary Computation. IEEE World Congress on Computational Intelligence.*  
IEEE, 1994, pp. 223–228.

- [17] B. Abdulhai, R. Pringle, and G. J. Karakoulas, “Reinforcement learning for true adaptive traffic signal control,” *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.
- [18] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [19] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.