MASTER OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING

Computer Vision-Based Affordable High-Density Traffic Monitoring System

Department of Electrical and Electronic Engineering
Islamic University of Technology (IUT)
Board Bazar, Gazipur-1704, Bangladesh
June, 2023.

Computer Vision-Based Affordable High-Density Traffic Monitoring System

A thesis submitted in partial fulfilment of

the requirements for degree of

Master of Science

in

Electrical and Electronic Engineering

by

Mohammed Yasin Arafat

Student Number: 171021010

Academic Year: 2017-2018

to the

Department of Electrical and Electronic Engineering

Islamic University of Technology

Board bazar, Gazipur, Bangladesh

June, 2023

Computer Vision-Based Affordable High-Density Traffic Monitoring System

by

Mohammed Yasin Arafat

Student Number: 171021010

Academic Year: 2017-2018

A thesis submitted in partial fulfilment of the requirements for degree of

Master of Science

in

Electrical and Electronic Engineering

to the

Department of Electrical and Electronic Engineering

Islamic University of Technology

Boardbazar, Gazipur, Bangladesh

June, 2023

Department of Electrical and Electronic Engineering

ISLAMIC UNIVERSITY OF TECHNOLOGY

DECLARATION OF CANDIDATE

It is hereby declared that, this thesis report or any part of it has not been submitted elsewhere for the award of any Degree or Diploma.

-------------------------------	-------------------------------
Prof. Dr. Md. Fokhrul Islam	Mohammed Yasin Arafat
Supervisor and Professor	Student No: 171021010
Department of Electrical and Electronic Engineering	Academic Year: 2017-2018
Islamic University of Technology (IUT)	
Gazipur-1704, Bangladesh	

CERTIFICATE OF APPROVAL

The thesis titled “Computer Vision Based Affordable High-Density Traffic Monitoring System” submitted by Mohammed Yasin Arafat, Student No. 171021010 of Academic Year 2017-2018 has been found as satisfactory and accepted as partial fulfillment of the requirement for the Degree of Master of Science in Electrical and Electronic Engineering on 09 June 2023.

Board of Examiners

ACKOWLEDGEMENT

I would like to convey my heartiest gratitude to my supervisor Dr. Md. Fokhrul Islam, Professor, Electrical and Electronic Engineering Department, Islamic University of Technology (IUT), Gazipur, for the encouragement and valuable advice provided to me during the completion of my thesis. I couldn’t have completed my research without his continuous guidance and support.

I would like to thank all the faculty members and staff of the Department of EEE, IUT for their inspiration and help. I would like to thank my parents for their blessings and my family and friends for having faith in me throughout the times of this thesis.

Finally, I would like to express my unparalleled gratitude to Almighty Allah for divine blessing without which it would not have been possible to complete this thesis successfully.

Mohammed Yasin Arafat June 2023

ABSTRACT

Vehicle detection and traffic monitoring system play a significant role in many areas of transportation infrastructure and to reduce traffic congestion. In this research, an affordable computer vision-based amazingly simple method has been proposed to detect and track vehicles, and also monitor high traffic density using CCTV cameras. Traffic monitoring system's infrastructure is expensive to build and implement. An affordable traffic monitoring system has been proposed which when implemented will be able to autonomize traffic monitoring system consequently reduce traffic congestion. Still image and traffic footage dataset were collected and input into the model. Vehicles were detected by YOLOv3 algorithm which was modified to detect 5 classes of vehicles only. Traffic density was estimated with respect to time and the whole model was implemented in Raspberry Pi. A series of experiments were conducted to validate the accuracy and affordability of the model. Highest model accuracy obtained from vehicle detection from still image was 92%, vehicle detection from traffic footage was 88%, traffic density estimation was 88%. When compared with existing model, the proposed model was 3 times more affordable. Variations in lighting and angles of camera position in real life can affect the vehicle detection and identification process which will provide further scopes of work. This research was aimed to contribute a little to pave the pathways which will help traffic monitoring and ultimately reduce traffic congestion.

Contents

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ABBREVIATIONS

CV	Computer Vision
AI	Artificial Intelligence
YOLO	You Only Look Once
CNN	Convolutional Neural Network
IOU	Intersection Over Union
NMS	Non-Max Suppression
COCO	Common Objects in Context
BB	Bounding Box
ML	Machine Learning
IDE	Integrated Development Environment
FPS	Frames Per Second
GPS	Global Positing System
RPI	Raspberry Pi
PC	Personal Computer

CHAPTER I
INTRODUCTION

1.1 Introduction

A Traffic Monitoring system is a system that automatically and continually monitors the no. of vehicles and speed of the vehicles and hence leading to the detection of traffic density monitoring. A dependable and robust Traffic Monitoring System is a crucial need to improve traffic control and vehicle traffic management [1]. Various systems are being studied and proposed, most of which use sensors and critical technology to detect the vehicles and speed, which infrastructure is quite expensive to implement. To ensure road safety, it is imperative to regularly monitor the traffic and manage it well. The chances of any accident happening is more at high traffic density area and hence immediate measures need to be taken to save precious lives. Nonetheless, controlling and monitoring the traffic is a challenging task and it requires great human effort. Typically, officials use CCTV cameras to see live footage of the traffic but it is not self-intelligent and requires a mass number of human and their efforts. The accident rate is increasing in a tremendous way worldwide. One is the reasons for these road accidents is unreliable traffic monitoring systems [2].

1.2 Background

In this modern era, the number of vehicles is in hike every day and hence leading to traffic congestion.

Fig.1.1 shows a traffic congested condition in a highway of Dhaka, Bangladesh, where a traffic monitoring system can be implemented as currently there is no traffic monitoring system placed on that location.

Fig. 1.1: Traffic condition to implement a traffic monitoring system [3]

For developing countries like Bangladesh, traffic jam is a major concern for the development of the country. A huge amount of time from daily life is wasted every day due to traffic congestion. Traffic congestion, especially in Dhaka, has become a nightmare for the daily commuters. It has turned into a major concern now which undermines the devolvement of a city and ultimately of the whole country. According to World Bank report, in the previous 10 years, the mean speed of traffic in Dhaka decreased from 21 km/h to 7 km/h. And by 2035, the mean speed might go down as low as 4 km/h, which will be less than the walking speed.

Fig.1.2 illustrates an example of how chaotic traffic situation in Dhaka can be. If the megacities around the world are considered, all incur traffic congestion at certain hours but in Dhaka the traffic situation is full of improper management.

Fig. 1.2: Traffic in Dhaka, Bangladesh [3]

Another research, which was conducted by the Brac Institute, stated that due to traffic jam in Dhaka, 5000,000 hours are ruined each day which damages the country by USD11.4 billion every year. According to a report by “The Daily Star”, in 2022, from January to April 1841 persons died due to road accidents, and 5477 persons were wounded.

As depicted in Fig. 1.3. it is not only in Dhaka that the traffic rules are violated, but the whole country is packed with rules violating drivers and walking passersby. These drives and passersby have little respect for the country’s traffic system.

Fig. 1.3: Traffic rule violation at Chattrogram, Bangladesh [3]

Despite everyone’s effort to keep the traffic situation at a tolerable range, the situation is all the same.

1.3 Problem Statement

The problem that this study addresses is the lack of a proper traffic monitoring system. This could be due to the vast amount of traffic monitoring being done by humans and also traffic monitoring system infrastructure is expensive to build.

Due to the lack of implementation of a proper traffic monitoring system, traffic congestion is unbearable, especially in developing countries like Bangladesh. This traffic congestion kills a valuable amount of time from everyday life and impacts the living standard and also costs huge monetary value.

In this research, a computer vision-based affordable high-density traffic monitoring model has been proposed, which can be used in real-time and take traffic input from a CCTV surveillance camera to detect, track and estimate vehicle traffic density at a specific region.

1.4 Scope of the Research

The study is to propose an affordable way of traffic monitoring that will reduce traffic congestion, especially at a junction, traffic signal. The vehicle categories considered are “car, bus, motorcycle, truck, cycle” only for vehicle detection and traffic density estimation. The approximate cost estimation provided is only for one city/area in Chattogram, Bangladesh and the proposed model is implemented in Raspberry Pi to validate its affordability.

1.5 Objectives

- To propose an affordable model for high-density vehicle monitoring
- To detect the vehicles using Computer Vision and track traffic density
- To calculate the approx. cost estimation for an area for the proposed model
- To implement the model in Raspberry Pi

1.6 Research Methodology

Traffic monitoring is a need everywhere. Autonomous systems have mostly been centralized in Western countries. In Bangladesh, especially in Dhaka, traffic jams are intolerable.

The proposed model offers an affordable way in which high-density traffic can be monitored using computer vision which is a part of AI. It can incorporate affordable CCTV video surveillance cameras (moving images) as input to the system. Still images and traffic footage datasets will be collected and loaded into the model and will be run in python using Open CV (AI). Input data will be tested and Computer Vision (AI) will be initiated to check the model accuracy on the collected dataset. Vehicle detection by computer vision and tracking will be completed. The number of vehicles present in a specific region is detected and tracked for 30 seconds. This loop repeated every 30 seconds. In this way, the traffic density in a certain region was estimated. Finally, the computer vision-based model has been implemented in Raspberry pi.

i. Dataset Collection: Still images and traffic footage datasets were collected and run in Python using Open CV. This dataset was used to detect bicycles, cars, motorbikes, buses, and trucks only.

ii. Data Validation: Computer vision (AI) will be initiated and random images of vehicles will be input into the model for detection. Different images of bicycles, cars, motorbikes, buses, and trucks were input into the model to check how the model is performing for the detection of the mentioned vehicles.

iii. Detection of vehicles: Vehicle detection from video surveillance which can be put live in real time (digital cameras were used instead of CCTV). YOLOv3 algorithm was modified to detect the mentioned classes of vehicles.

iv. Real-time Vehicles Density Estimation

- Tracking of the vehicles
- Traffic density calculation

Fig. 1.4 is the proposed methodology flowchart. The YOLOv3 algorithm was modified to detect and output vehicles only. The whole model was implemented in Raspberry Pi for edge implementation.

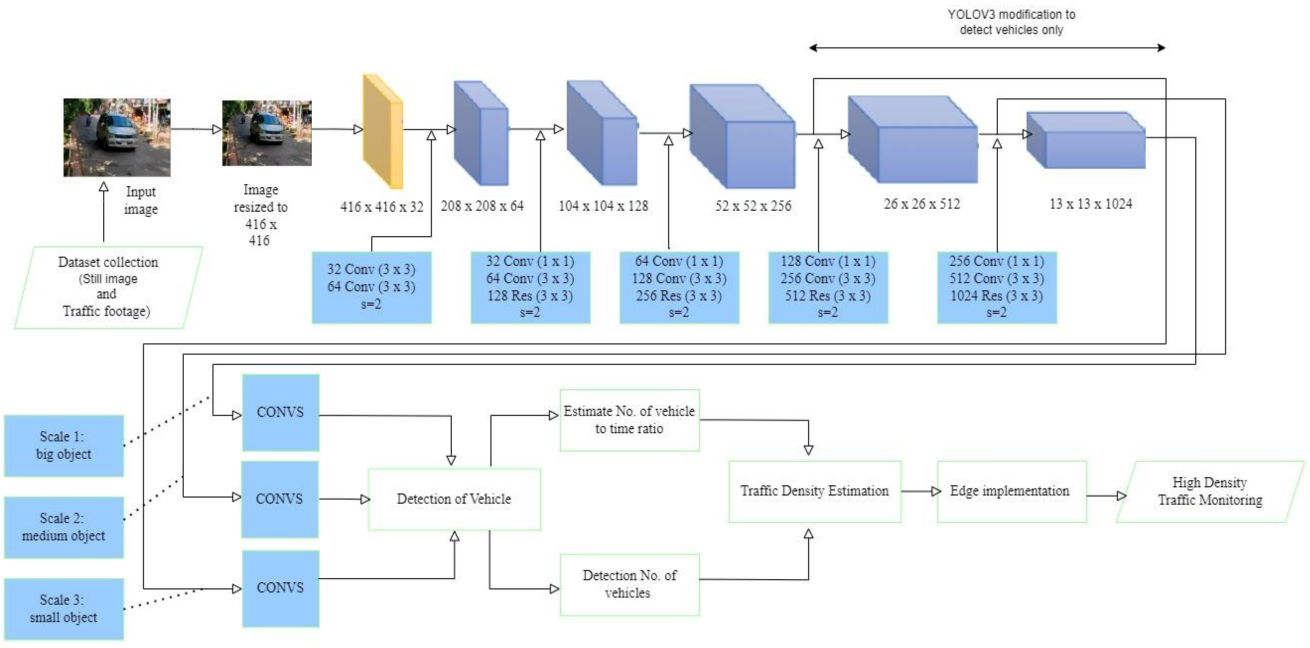

YOLOV3 modification to detect vehicles only
Input image
Image resized to 416 x 416
416 x 416 x 32
208 x 208 x 64
104 x 104 x 128
52 x 52 x 256
26 x 26 x 512
13 x 13 x 1024
Dataset collection (Still image and Traffic footage)
32 Conv (3 x 3) 64 Conv (3 x 3) s=2
32 Conv (1 x 1) 64 Conv (3 x 3) 128 Res (3 x 3) s=2
64 Conv (1 x 1) 128 Conv (3 x 3) 256 Res (3 x 3) s=2
128 Conv (1 x 1) 256 Conv (3 x 3) 512 Res (3 x 3) s=2
256 Conv (1 x 1) 512 Conv (3 x 3) 1024 Res (3 x 3) s=2
Scale 1: big object
Scale 2: medium object
Scale 3: small object
CONVS
CONVS
CONVS
Detection of Vehicle
Estimate No. of vehicle to time ratio
Detection No. of vehicles
Traffic Density Estimation
Edge implementation
High Density Traffic Monitoring

Fig.1.4: Methodology flowchart

Vehicle detection, vehicle tracking, and traffic estimation, all will be computed in real-time. The time cost for traffic density estimation is approximately 1 second. This is the initial time the model takes when started.

Fig.1.5. shows the basic architecture of the hardware for the proposed model. The raspberry pi implementation is included in the “Computer vision-based model (Traffic Monitoring)” block in Fig. 1.5

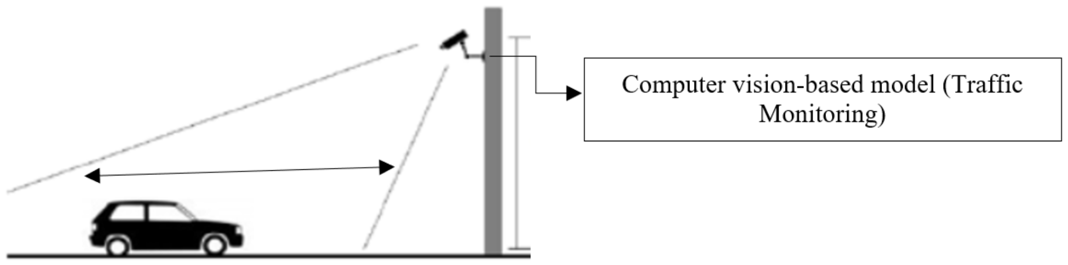


Fig. :1.5: Hardware configuration for the proposed model

In Fig. 1.5, the hardware configuration for the proposed model is shown. As seen from the figure the architecture consists of CCTV cameras connected with computers in which the computer vision-based model is running. This shows that the model is very affordable. Also, if considering Bangladesh’s scenario, there are already CCTV cameras installed in many places such as traffic junctions, and important places. Also, these CCTV cameras can be used to connect with the model and hence for those new places, there will not be any need to install new CCTV cameras. Hence the affordability will be further increased. Below list of the relevant deployed devices:

- CCTV camera
- A computer connected to a CCTV camera
- Raspberry pi (version 3b to above)

v. Implementation of the Model in Raspberry pi:

To implement the model in Raspberry pi, a Raspberry pi 3b or above version is required. The implementation can be done by installing a few dependencies on the Pi.

Fig. 1.6 shows a physical image of a Raspberry Pi (mini-computer). The mini-computer can connect to mobile phones over the WIFI network or via Bluetooth. This can be used instead of PC.

Fig. :1.6: Physical image of a Raspberry Pi (mini-computer)

1.7 Organization of the Thesis

The book is organized in the following chapters

Chapter I: Introduction

Chapter II: Literature Review

Chapter III: Computer Vision and Detection of Vehicle

Chapter IV: FPS Detection, Estimation of Traffic Density, and Raspberry Pi Implementation

Chapter V: Results and Discussions

Chapter VI: Conclusion and Further Work

1.8 Summary

This chapter introduces the Traffic Monitoring system, a background scenario of traffic congestion, the purpose of the study, the methods that will be used to achieve the mentioned objectives.

CHAPTER II
LITERATURE REVIEW

2.1 Introduction

In this chapter, the contemporary works on the traffic monitoring system, the different techniques of the traffic monitoring system, advantages, and disadvantages of the various traffic monitoring system are going to be stated.

2.2 Contemporary Work

Some road monitoring systems consisting of cameras can be positioned inside the vehicles to provide information and aid the driver. Some smart vehicles incorporate vision-based systems for critical functions like overtaking maneuvers in a state when detected on a collision course. Another research field includes using surveillance cameras and still cameras [7] to detect vehicle license plates. Night-time vehicle detection can be done through the thresholding method and cross-correlation method [8] by detecting the vehicle's headlight at night, merging vehicle parts [9] and light detection [10] and also by connected component labeling [11], light refection [12]. Incorporating edge nodes, edge computing [13] also can be used. In many places, traffic cameras have been installed so that the violators can be fined to implement the traffic rules. Some traffic management systems can be automated to identify traffic offenses and act accordingly and also take a snippet to act accordingly if an offense is committed [14]. Also, some highway traffic surveillance systems are composed of cellular equipment [15]. Some non-invasive vehicle density monitoring system are based on the ultrasonic devices placed alongside the road. The limitation of the existing systems is that it is not cost effective and hence cannot be implemented in a wide scale to monitor the traffic and hence cannot make substantial contribution to reduction in traffic congestion, the proposed model in this research can mitigate this gap.

This research proposes an affordable way of traffic monitoring system that can be used especially when the traffic density is high which will give a visualization of traffic density at a specific region or a junction. Most of the techniques out there use sensors to detect vehicles whose infrastructure is too expensive to build. In this research, a simple video surveillance camera was used to take the input in real-time (the actual scenario used a digital camera instead). The vehicles will be using computer vision to detect the vehicles, track the vehicles and calculate traffic density which can be used by the traffic administration for better traffic management. Also, can be used by road users to select the best way for travelling. YOLOv3 [16] algorithm is used to detect and track each vehicle and finally count the number of vehicles per sec. Which finally will lead to the detection of vehicle density at a certain region. Computer vision is the main tool to help us better process and understand millions of data from video surveillance cameras which can be utilized more proficiently.

2.3 Different Techniques of a Traffic Monitoring System

There are various ways for traffic monitoring systems. Each of the methods has its advantage and disadvantage. With the advancement in technology and research, modern traffic monitoring systems are improving quite considerably.

2.3.1 Types of Traffic Monitoring Techniques

As mentioned earlier, there are various ways for traffic monitoring. The traffic monitoring system has been a very interesting topic of research. As a result, different techniques have been incorporated into different methods of traffic monitoring. A traffic monitoring system can be categorized as below.

- Invasive
- Non-invasive
- Off-road
- Sensor-Based

2.3.2 Advantages and Disadvantages of Various Traffic Monitoring Systems

There are both advantages and disadvantages to every method mentioned above. Table 2.1 summarizes the benefits and disadvantages of the mentioned methods.

Table 2.1: Advantages and disadvantages of existing Traffic Monitoring System

SL	Method	Advantage	Disadvantage
1	Invasive	Not affected by weather conditions, accurate	The pavement needs to be cut for installation & maintenance, costly
2	Non-Invasive	Vehicle velocity and location can be accurately measured.	The environmental situation can have an impact on performance, costly
3	Off-Road	More area coverage, high accuracy	Real-time data cannot get, costly
4	Sensor Based	High accuracy	High energy usage, restriction on capacity on a particular sensor, costly

The main challenge with all the existing Traffic Monitoring Systems is that they are way too expensive to build and consequently their maintenance is also quite costly.

The below points summarize little descriptions of the existing Traffic Monitoring Systems:

- Invasive: Inductive coil, different type of probes is used under the pavement and also devices installed for weighing the vehicles.
- Non-invasive: Consists of mainly image processing, electromagnetic waves, etc. The devices can be set overhead along the roads.
- Off-road: Consists of mainly Global Positioning system (GPS), cellular phones, etc. This type of method does not need to cut the pavement or set along the road.
- Sensor-Consolidation: Different types of sensors will be combined and used in this method.

As it can be seen in Fig. 2.1 the invasive methodology requires lots of labor charges, and sometimes existing road's surface needs to be cut.

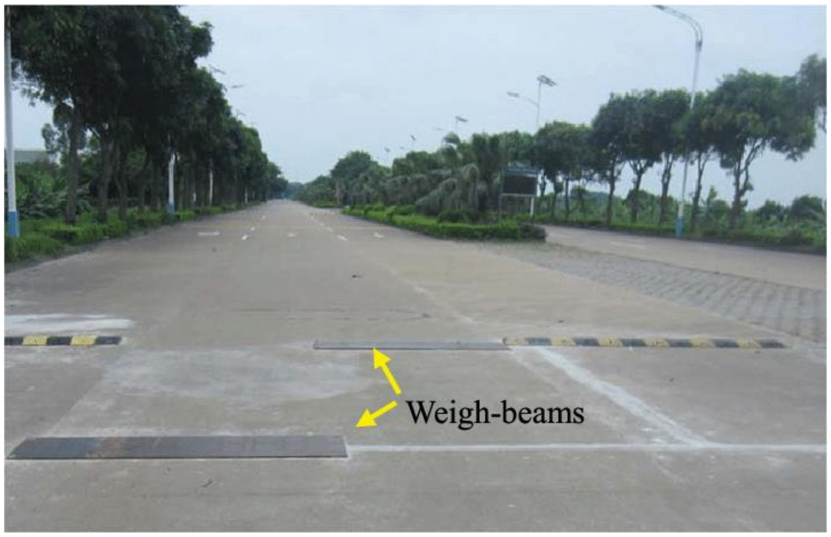


Fig. 2.1: Invasive: Weight beams installed into the pavement [17]

Fig. 2.2 shows ultrasonic wave devices installed overhead on the roads for traffic monitoring which falls under the non-invasive category. Traffic density will be monitored only on the sensing range.

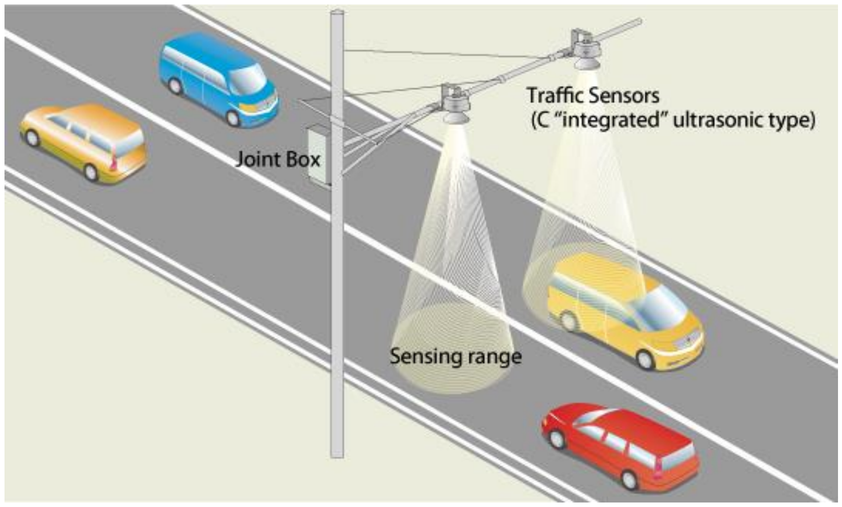


Fig. 2.2: Non-Invasive: Ultrasonic wave device set overhead [17]

Fig. 2.3 shows how the GPS works for traffic monitoring, and Fig. 2.4 shows a GPS satellite revolving around the planet. This falls under the off-road category. This type of system is very costly.

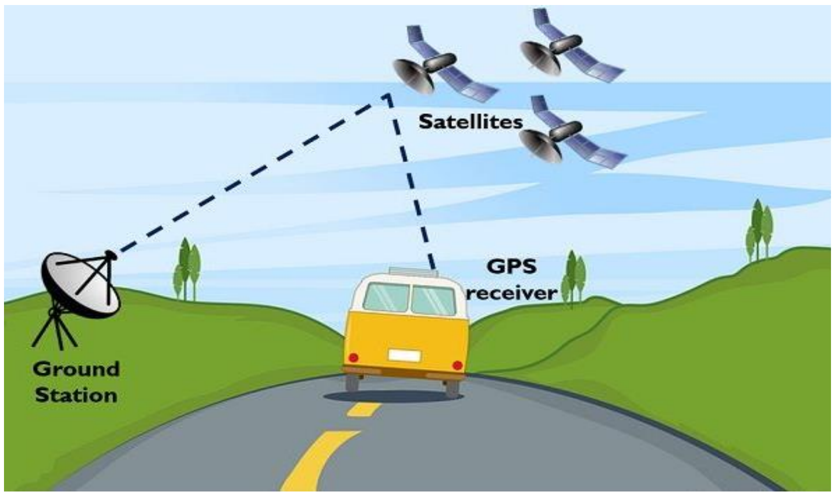


Fig. 2.3: Off-Road: GPS function is in function [17]

Fig. 2.4: Off-Road: GPS satellite revolving around the earth [17]

Fig 2.5 shows 3 different sensors/devices for traffic monitoring which can be combined for sensor-consolidation. This sensor consolidation method aims to break the restrictions of the above methods.

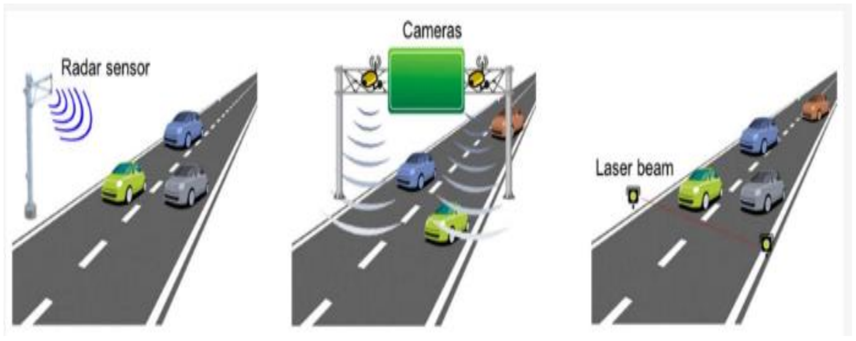


Fig. 2.5: Sensor-Consolidation: different sensors which can combine for sensor-consolidation [17]

2.4 Summary

It is seen that most of the existing traffic monitoring system infrastructures are expensive to implement. In this research, a computer vision based, affordable, and easy to implement high-density traffic monitoring system was proposed. The model uses simple video surveillance (CCTV) camera to take input and uses computer vision (CV) to detect, track the vehicles and monitor the density of the traffic. The model was tested with different inputs of road surveillance video from different roads in Bangladesh. Digital cameras were used instead of CCTV for the flexibility of the research.

CHAPTER III

PROPOSED MODEL AND EXPERIMENTAL STUDIES

3.1 Introduction

Computer vision is deeply connected to this research for the detection of vehicles. Computer vision is the part of artificial intelligence that has been used in this study. The proposed model will use CCTV camera traffic footage as input, will detect the vehicles using computer vision, and will estimate traffic density. This model can run on personal computer. The whole model will be implemented in Raspberry Pi to validate its affordability.   Still images and traffic video footage dataset was collected and input to the model. Using the proposed model vehicle detection was complete from both still image and traffic footage. In the next step, traffic density was estimated by the model which is given in Chapter IV. The algorithm of the model was implemented in Python.

3.2 Computer Vision

It empowers computers and systems to extract relevant data from digital pictures, moving images, and other optical aid. It performs similarly to mankind's eyesight and can train machines to perform various functions (object detection, moving objects, etc.). Incorporates cameras, data, and algorithms. Also, used in industries (Energy & Utilities to Manufacturing & Automotive). Due to progress in artificial intelligence and progress in neurological web and deep learning, computer vision has made significant progress in science in recent years [4].

Fig. 3.1 shows a simple image of a computer when it is used for computer vision. The key factor behind the progress in artificial intelligence is that the amount of data produce every day can be used to train and make computer vision one step ahead each time.

Fig. 3.1: Computer Vision [18]

Computer vision is controlling dominantly complicated challenging tasks [5]. The particular sort of neurological network that accomplish this is known as a convolutional neural network, CNN. CNN divides the picture down into a short cluster of pixels called a filter. A matrix of pixels makes a filter. A network does a series of calculations on these pixels comparing them against pixels that the network is looking for. It is computer vision, which trains computers to understand images and videos and extract info from them.

Fig 3.2 shows an image of the road that the computer perceives, it is how the road looks like to computers when computer vision is used.

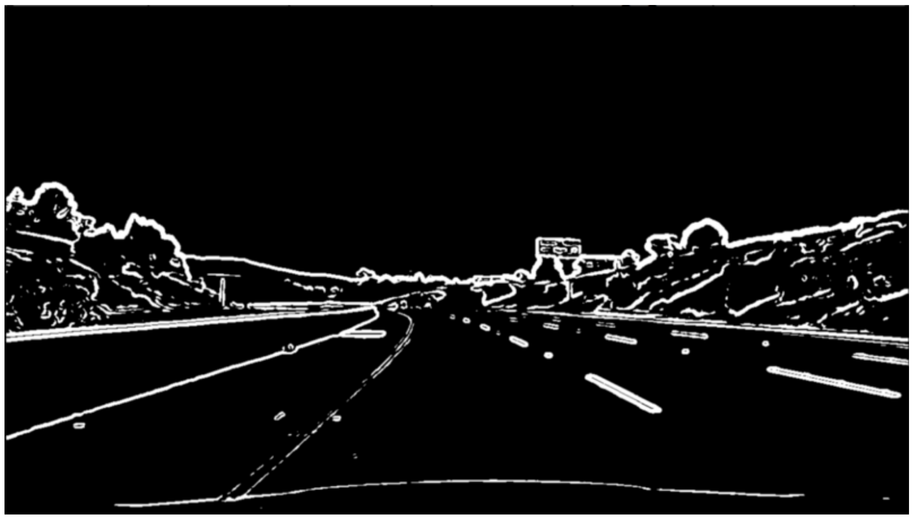


Fig. 3.2: Computer vision highway [18]

Without CV, AI cannot achieve much in the field of being self-autonomous. CV’s main objective is to identify particular objects from still and moving images and act accordingly. CV ranges from biometrics, and image processing to augmented reality. The 3 breakthroughs in computer vision are cheaper GPU units, big data, and much improved algorithms. Better visuals in moving images, games, etc. were required and comparatively cheaper GPU helped us move to a higher level. For computer vision to learn, it needs data, literally lots of data. In this era, data is being used for almost everything and these data are shared online. Big data helped us to solve this issue, it helps computer vision to grow. As a result of better education and keen towards learning, many improved algorithms are being developed every day which in turn is a big milestone in the breakthrough of computer vision.

For example, to detect a face, it has to extract the features of a face, which can be called patterns, and finally need to compare and make the decision that these features are exactly the features of a face. Consequently, the face will be detected. Through better algorithms, these operations faster can be performed faster.

The computer vision system is intricate. It works much like the human eye and brain but much quicker and more correctly.

As shown in Fig. 3.3 a very important application of computer vision is autonomous cars. Which are being improved day by day. Self-driving can detect passersby, obstacles, and other cars with the help of computer vision.

Fig. 3.3: Self driving car using computer vision [18]

Computer vision also plays an important role among medical professionals who uses CT scan and other imaging techniques. To perform in all of these applications, computer vision needs to be trained with an enormous no. of images. The main objective of computer vision is to comprehend the visual universe. In the future, it is believed that computer vision will be very useful in nearly every application that uses AI. Not before long, it was a dream that the computer would judge and act like a human.

Fig. 3.4 shows how computer vision visuals the surrounding. Now the dream is a reality and computer vision enabled computers to see everything surrounding them.

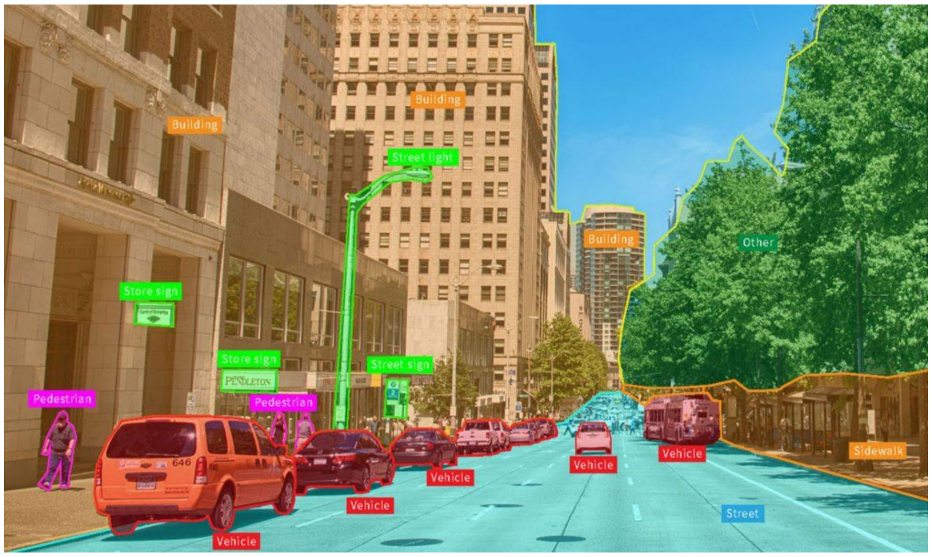


Fig. 3.4: Computer Vision surrounding visual [18]

Fig. 3.5 shows a simple comparison between Mankind Vision and Computer Vision System. Computer vision can also be narrated as educating computers to operate an image at the pixel stage and fully comprehend it.

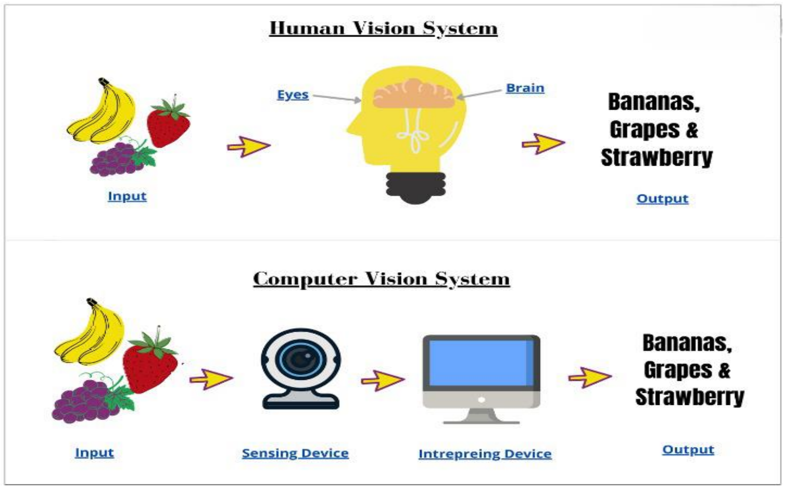
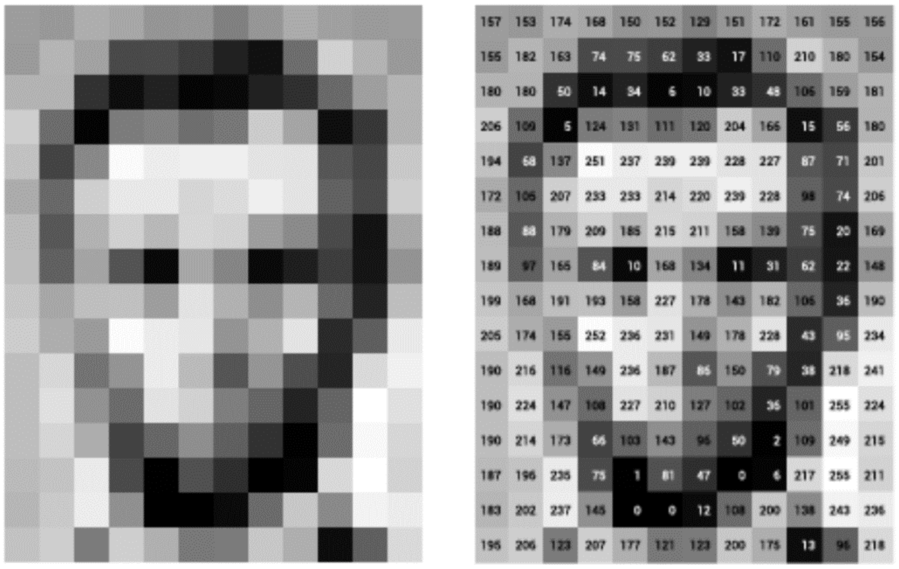


Fig. 3.5: Human Vision System Vs Computer Vision System [19]

As can be seen in fig. 3.6 most of the computer vision algorithms are on the basis of pattern recognition. To state this in simple terms, computers need to be trained on vast amounts of data. Computers will analyze the images, mark labels where appropriate on them, and finally find out patterns corresponding to them. Hence, for instance, if thousands of images of cows have been input, the computer will process them, recognize the patterns corresponding to them, and finally will be able to detect the cow in the image every time.



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Fig. 3.6: Computer Vision pattern recognition [19]

3.3 Python

Python as programming language was used to implement the algorithm. Python is a simple to understand programming language.

For example, the below syntax simply prints “Hello World”.

syntax: print ('Hello World')

The syntax is easy to use in python. Nowadays, python is widely being adopted as the global programming language [6].

Fig. 3.7 shows the top reasons to choose python as a programming language which is essential for Machine Learning. Also, errors can be easily mitigated as many people are using python. The implementation time in python is comparatively low.

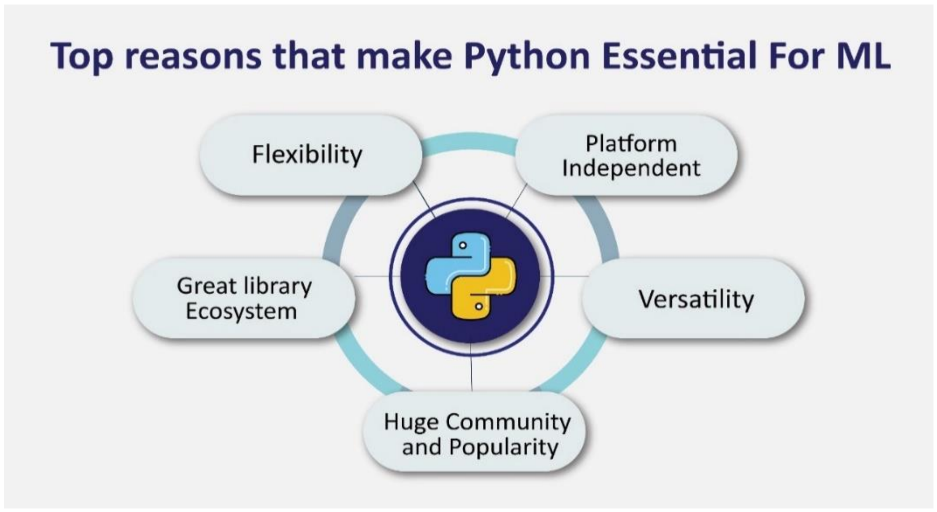

Top reasons that make Python Essential For ML
Flexibility
Platform Independent
Great library Ecosystem
Versatility
Huge Community and Popularity

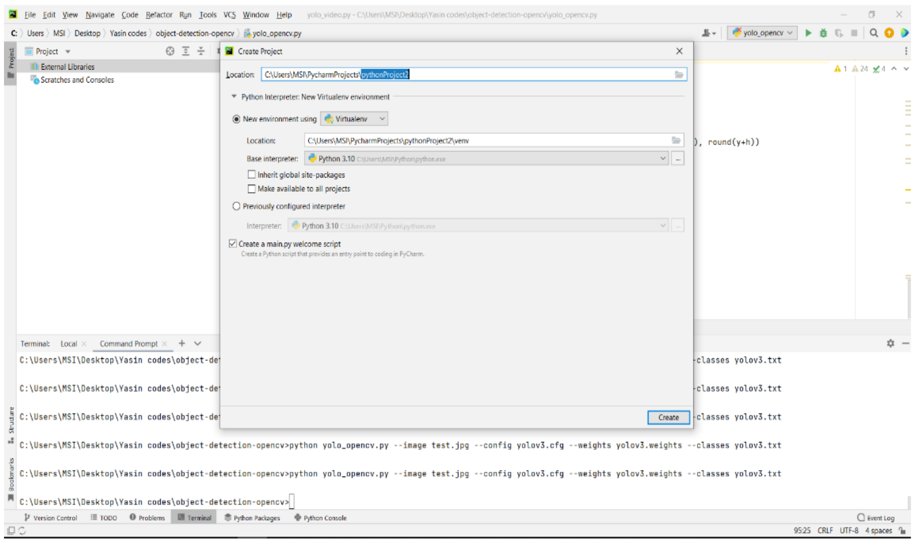
Fig. 3.7: Top reasons to choose python for ML [19]

A few of the Integrated Development Environments (IDE) through which python can be used are given in table 3.1.

Table 3.1: IDE in which Python can be used

SL	IDE Name	Platform
1	Programiz	Python
2	PyCharm	
3	IDLE	
4	Sublime Text 3	
5	Atom	
6	Thonny	
7	Visual Studio Code	
8	Vim	
9	Spyder	

Fig. 3.8 shows an interface of PyCharm IDE. PyCharm was used as the IDE interface for using Python to implement the computer vision-based model.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
yolo_video.py - C:\Users\MSI\Desktop\Yasin codes\object-detection-opencv\yolo_opencv.py
C: Users MSI Desktop Yasin codes object-detection-opencv yolo_opencv.py
yolo_opencv
Project
External Libraries
Scratches and Consoles
Create Project
Create
Terminal: Local Command Prompt
Version Control TODO Problems Terminal Python Packages Python Console Event Log
95:25 CRLF UTF-8 4 spaces

Location: C:\Users\MSI\PycharmProjects\pythonProject2

Python Interpreter: New Virtualenv environment

New environment using: Virtualenv

Location: C:\Users\MSI\PycharmProjects\pythonProject2\venv

Base interpreter: Python 3.10 C:\Users\MSI\Python\python.exe

Inherit global site-packages [ ]

Make available to all projects [ ]

Previously configured interpreter

Interpreter: Python 3.10 C:\Users\MSI\Python\python.exe

[✓] Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.

```
), round(y+h))
```

```
C:\Users\MSI\Desktop\Yasin codes\object-de                                      -classes yolov3.txt

C:\Users\MSI\Desktop\Yasin codes\object-de                                      -classes yolov3.txt

C:\Users\MSI\Desktop\Yasin codes\object-de                                      -classes yolov3.txt

C:\Users\MSI\Desktop\Yasin codes\object-detection-opencv>python yolo_opencv.py --image test.jpg --config yolov3.cfg --weights yolov3.weights --classes yolov3.txt

C:\Users\MSI\Desktop\Yasin codes\object-detection-opencv>python yolo_opencv.py --image test.jpg --config yolov3.cfg --weights yolov3.weights --classes yolov3.txt

C:\Users\MSI\Desktop\Yasin codes\object-detection-opencv>
```

Fig. 3.8 PyCharm IDE interface

Any other IDE could be used that is available to use for python programming. But, PyCharm was selected due to the following reasons below:

- User friendly interface
- Used widely
- Debugging is easier

3.4 Dataset Collection

Still images and traffic footage of motorcycles, buses, trucks, cars, and cycles were collected. YOLOv3 weights are used for the vehicle detection model. These weights were trained on the Common Object in Context, COCO dataset [20] to detect 80 different classes (bicycle, car, person, aero plane, motorbike, bus, train, boat, truck, etc.). It was trained with the support of the darknet framework [25] using the COCO dataset as it is an entrenched framework for feature extraction which can detect these 80 different classes.

These weights were used to detect 5 categories of vehicles. They are motorcycles, buses, trucks, cars, and cycles. The mentioned classes of the vehicles are components of the 80 different classes of the weight file.

3.4.1 COCO Dataset List

Table 3.2 provides a list of items in the COCO dataset. As mentioned earlier, w5 classes or categories of vehicles have been used. All of these 5 categories of vehicles will be detected by the model.

Table 3.2: COCO dataset different classes names

SL	Class Name	SL	Class Name	SL	Class Name	SL	Class Name
1	person	1	elephant	1	wine glass	1	dining table
2	bicycle	2	bear	2	cup	2	toilet
3	car	3	zebra	3	fork	3	Tv monitor
4	motorbike	4	giraffe	4	knife	4	laptop
5	aero plane	5	backpack	5	spoon	5	mouse
6	bus	6	umbrella	6	bowl	6	remote
7	train	7	handbag	7	banana	7	keyboard
8	truck	8	tie	8	apple	8	cell phone
9	boat	9	suitcase	9	sandwich	9	microwave
10	traffic light	10	frisbee	10	orange	10	oven
11	fire hydrant	11	skis	11	broccoli	11	toaster
12	stop sign	12	snowboard	12	carrot	12	sink
13	parking meter	13	sports ball	13	hot dog	13	refrigerator
14	bench	14	kite	14	pizza	14	book
15	bird	15	baseball bat	15	donut	15	clock
16	cat	16	baseball glove	16	cake	16	vase
17	dog	17	skateboard	17	chair	17	scissors
18	horse	18	surfboard	18	sofa	18	teddy bear
19	sheep	19	tennis racket	19	Potted plant	19	hair drier
20	cow	20	bottle	20	bed	20	toothbrush

3.5 YOLOv3 Modification

The proposed model detected the mentioned 5 classes of vehicles only. For that it was needed to modify YOLOv3. YOLOv3 was modified accordingly so that only 5 classes of vehicles will be detected and tracked.

To do that it was required to remove person from the detection. This person can be driver or any passerby on the road. This was accomplished by modifying the algorithm. "Person" is having index (0) in COCO dataset.

To remove "Person" from the output detection, the algorithm was modified such that it will output bounding boxes for the mentioned classes of vehicles but will not output any bounding box for "Person". Hence, index (0) bounding boxes were removed by applying the argument "draw detection boxes" "if classIDs[i]=0 ".

Hence, the output of the model will be only the mentioned vehicle classes but will not include any person.

3.6 Vehicle Detection from Still Image

In this section, offline images of the 5 different classes of vehicles which are motorcycle, bus, truck, car, and cycle were collected. These collected images will be input into the model to test the detection accuracy of the mentioned vehicles. The accuracy obtained from the vehicle detection from still image will be plotted against the original input data. A total of 400 images and 80 different images for each class have been collected.

3.6.1 Data Tabulation

Table 3.3 shows the image dataset. The input images were collected from different road conditions and various places.

Table 3.3 Still image dataset for vehicle detection

SL	Vehicle Class	No. images
1	Car	80
2	Bus	80
3	Motorcycle	80
4	Truck	80
5	Cycle	80
	Grand Total	400

3.6.2 Vehicle Image Input to the Model

Fig. 3.9 shows a few of the input images of vehicle class Car that was input to the model offline. After input into the model, the model will detect them and show bounding boxes as per our modified YOLOv3 algorithm.

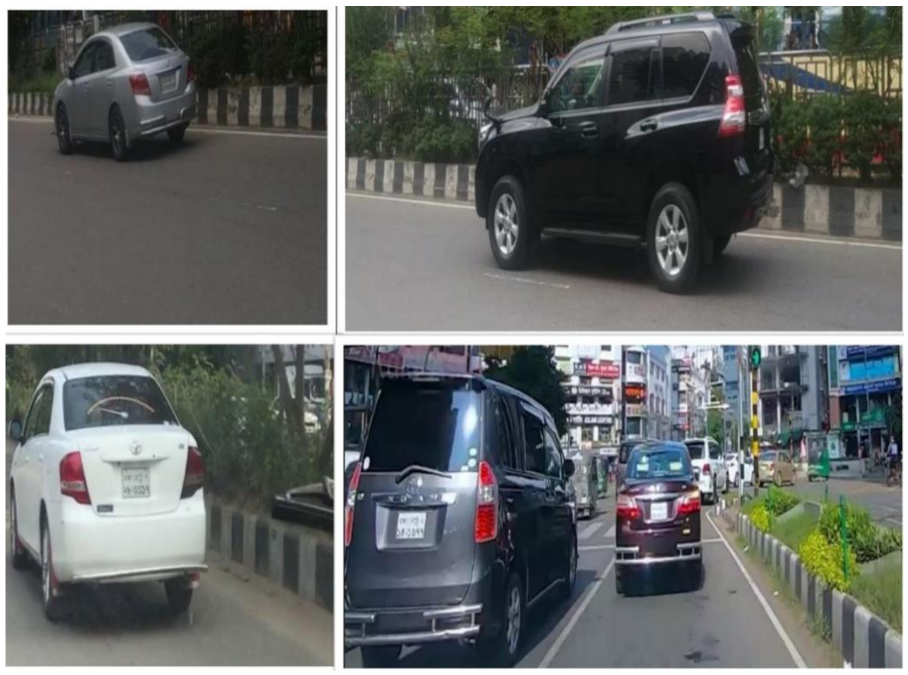


Fig. 3.9: Car input images

Fig. 3.10 shows a few of the different input images of the Bus that was input to the offline model. After input into the model, the model will follow the same process as it followed for the detection of car.

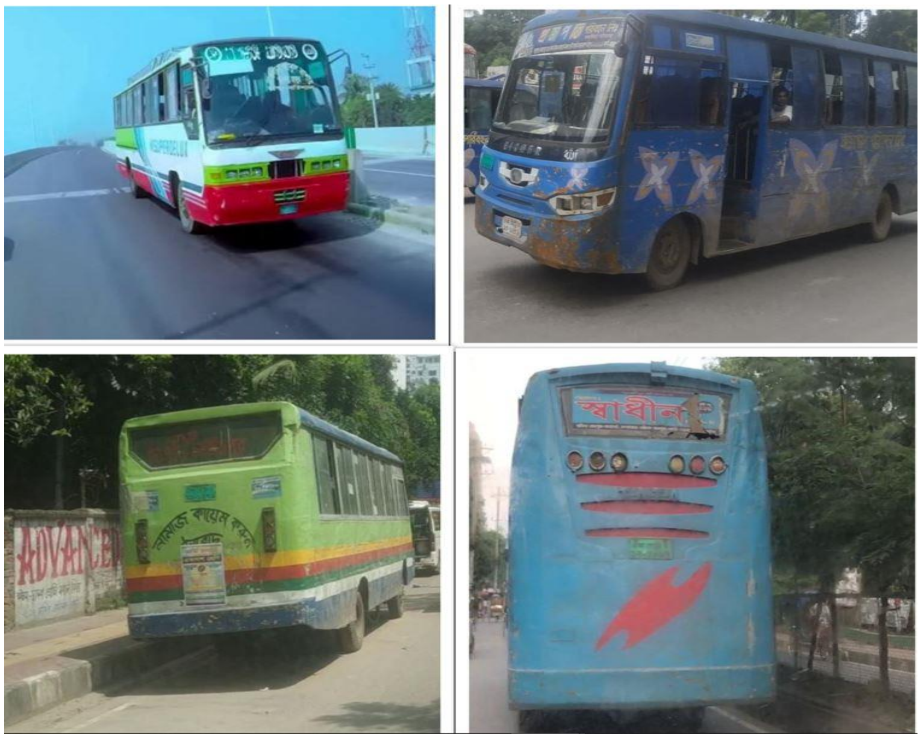


Fig. 3.10: Bus input images

Fig. 3.11 shows a few of the different input images of the Motorcycle that were captured from various roads and input into the offline model to output the detection.

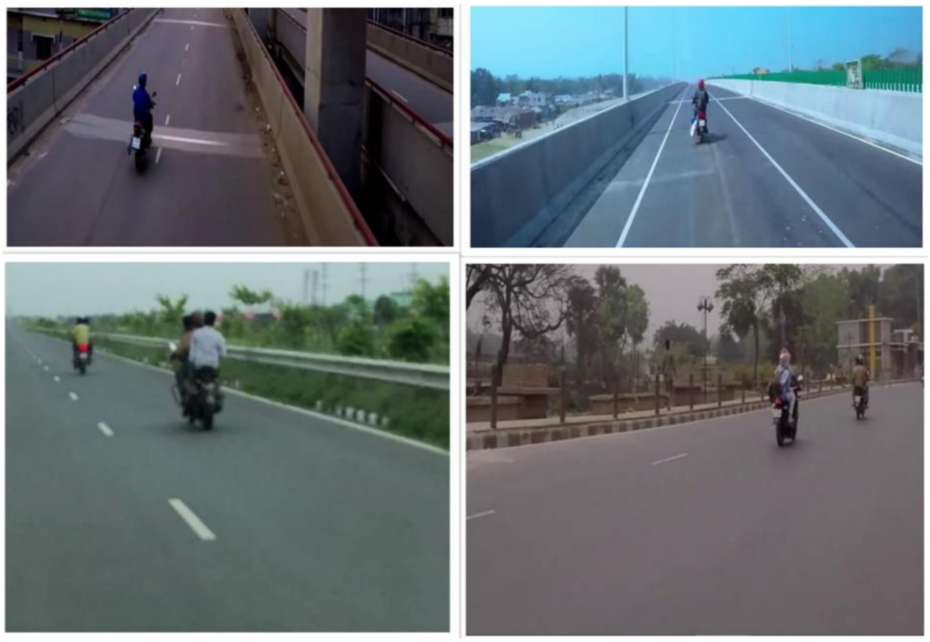


Fig. 3.11: Motorcycle input images

Fig. 3.12 shows a few of the different input images of the Truck that was input to the offline model. After the detection is complete, model with output bounding box for the detection of truck.

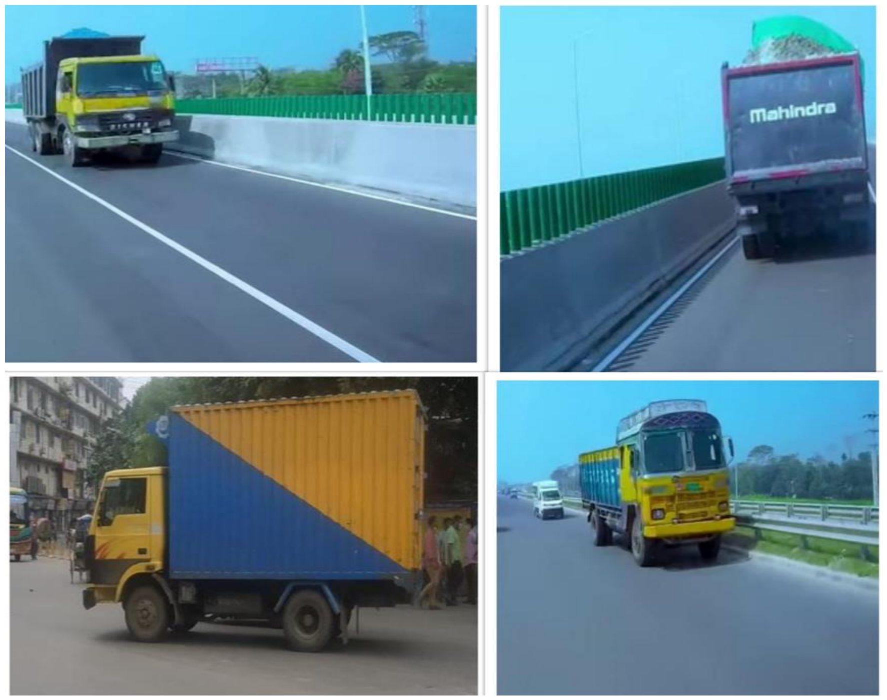


Fig. 3.12: Truck input images

Fig. 3.13 shows a few of the different input images of Bicycle that was input to the offline model. As mentioned earlier, bicycle is also included as a class of vehicle in the model.

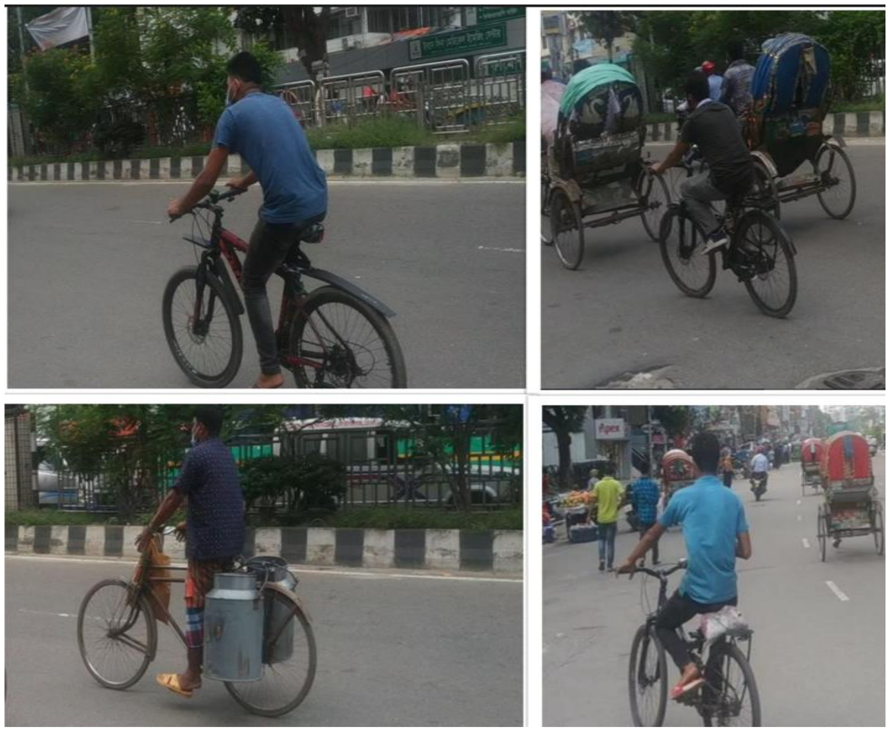


Fig. 3.13: Bicycle input images

3.7 Detection of a Vehicle from Traffic Footage

YOLOv3 object detection algorithm [22] was used to detect the vehicles which include bicycle, car, motorbike, bus, and truck. YOLO can detect multiple objects. YOLO algorithm is so quick [23] that it is now considered a basic way to identify objects in the field of computer vision. In 2015 YOLO algorithms were invented. Also, vehicle detection can be done by merging hue and dept pictures [24]. YOLOv3 is also used for other models like the detection of laborers who do not wear helmets from video images [25], detection of UAV for anti-UAV [26], road crack [27], smoking objective detection [28], advancement in item detection [29].

3.7.1 YOLOv3 Architecture

Two main tasks will need to be performed in vehicle detection, locating the vehicles in the surveillance footage and then classifying these vehicles. For and ideal situation, real-time feed from surveillance CCTV cameras that are installed at a specific degree elevation should be used, but a digital camera was used for the flexibility of the research.

YOLOv3 applies a single neural network (NN) to the whole network. It splits the image/frames into grid cells. YOLOv3 architecture has convolutional layers to outcome a detection that was predicted after passing through the features learned onto a regressor or classifier. The features are coordinates (centers, width, height) of the bounding boxes, vehicle names, sizes of the boxes, etc. Size of prediction map same as the size of feature map before it as machine-learning algorithm YOLO3 uses 1x1 convolution.

In prediction map, every cell will predict a specific number of bounding boxes. The cell, which has the ground truth box of object of interest is the cell that will ultimately be designated for vehicle prediction. In YOLOv3, it has 3 anchor or bounding boxes, hence the vehicle prediction will result in 3 bounding boxes. Each image frame will be down sampled by 3 scales (32, 16, 8). Hence, each frame will be down sampled to 13 x13, 26 x 26, and 52 x 52 if the frame size is 416 x 416. And the consequent feature map will be of equal spatial dimension.

The dept of the kernel is calculated by

$$Kernel\ dept = b \times (5 + c) \qquad (1)$$

Where,

b=3, number of bounding boxes (BB)

(5+c) = number of attributes for each bounding box

c=80, trained on COCO dataset

The attributes are:

t_x and t_y = center coordinates of the bounding boxes

t_w and t_h = width and height coordinates of the bounding boxes

p_0 = objectness score (the probability of a vehicle present inside the BB)

$p_{1\sim}p_c$= the probability of each class

Every feature map outcome by detection kernels at three different layers (84, 92 and 106) has one more dimension depts that integrate 255 attributes of the bounding boxes for COCO dataset. Hence the shapes of these feature maps will be as 13x13x255, 26x26x255 and 52x52x255.

In YOLOv3, each cell outcome 3 bounding boxes (BB), each cell will predict a vehicle through one of its bounding boxes if the center of the vehicle belongs to the receptive field of this cell.

To define bounding boxes, YOLOv3 uses predefined bounding boxes known as anchors or priors. The coordinates of the predicted bounding boxes are computed by these anchor boxes.

Three anchor boxes at each scale and hence 9 anchor boxes Fig. 3.14 are used. It shows that, at each scale, each cell of the feature map will predict 3 bounding boxes.

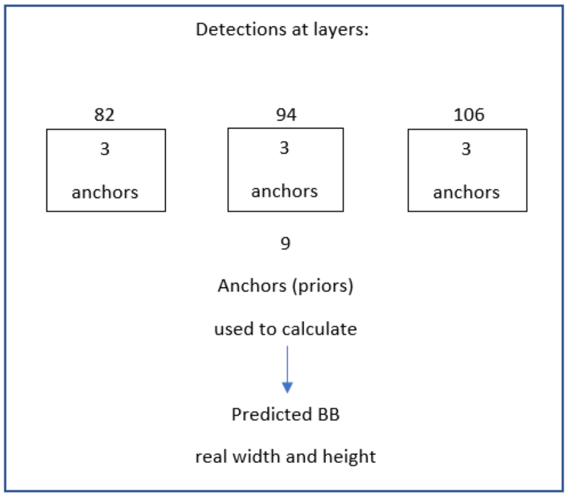


Fig: 3.14 Anchor boxes of YOLOv3

Maximum probability score is taken to determine which classes of the vehicles are from the classes (motorcycle, bus, truck, car, cycle). The calculation for maximum probability is applied for 13x13 cells across 3 BB. Hence, at each of the 3 scales, the number of BB will be scale-1: 507, scale-2: 2,028, scale-3: 8,112 making a total of 10,647 BB and will be further filtered out with Non-Max Suppression (NMS) to output the BB with the highest probability.

To predict BB, YOLOv3 will use offset to the predefined anchors also known as log-space transform, shown in Fig. 3.15. To define the coordinates of the center of the BB, the output will be passed through a sigmoid function.

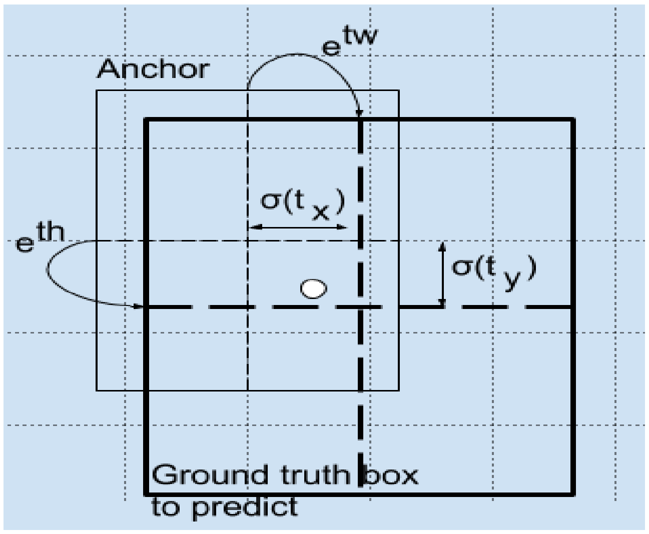


Fig.: 3.15 Log space transform and sigmoid functions

The x-coordinate of the center of BB [14] is

$$(b_x) = \sigma(t_x) + c_x \tag{2}$$

The y-coordinate of the center of BB [14] is

$$(b_y) = c(t_y) + c_y \tag{3}$$

The width of the BB [16] is

$$(b_w) = p_w e^{t_w} \tag{4}$$

The width of the BB [14] is

$$(b_h) = p_h e^{t_h} \tag{5}$$

Where,

b_x, b_y, b_w, b_h- center, width, the height of the predicted bounding box (BB)

t_x, t_y, t_w, t_h- are the outputs of the network

c_x, c_y- cells top left corner of the anchor box

p_w, p_h- anchor’s width, height

To detect if any vehicle is present in the bounding box, objectness score, $p_{0,}$ (the probability of the vehicle present inside a bounding box) is required.

The objectness score, p_0 is

$$(P_0) = P_{object} \times IOU = \sigma(t_0) \tag{6}$$

Where,

P_{object} = predicted probability

Fig. 3.16 shows intersection over union. This IOU, intersection over union is between BB$_2$ and ground truth BB$_1$.

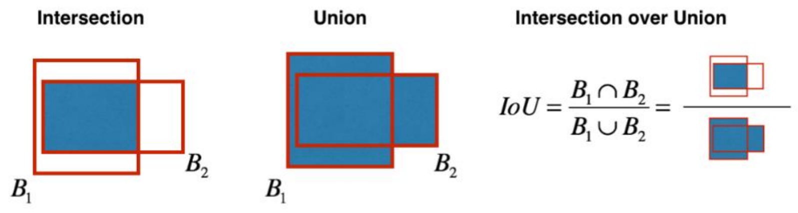
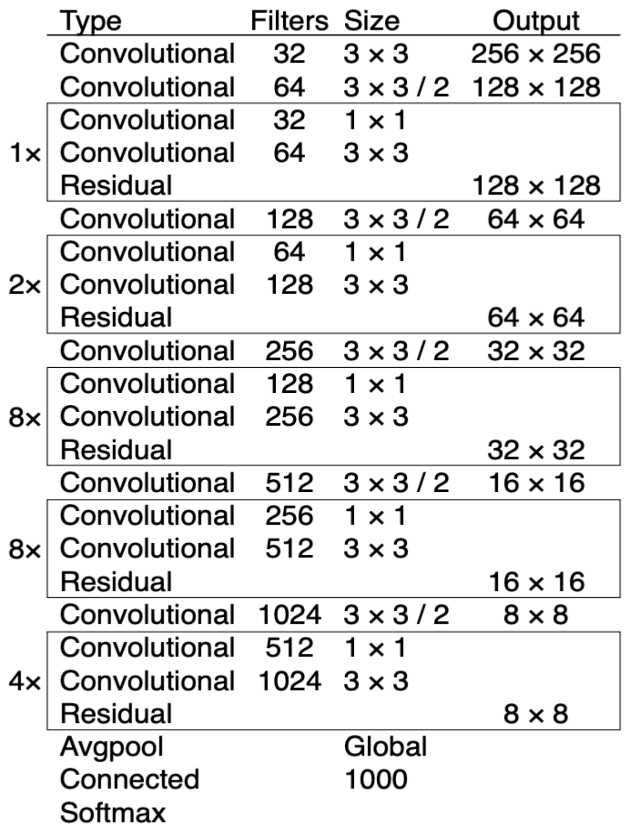


Fig. 3.16: Intersection over union

YOLO v3 is based on the Darknet-53 [21] network architecture. Fig. 3.17 shows the architecture of the framework of YOLO v3.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
1×	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
2×	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
	Convolutional	128	1 × 1	
8×	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
8×	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
4×	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 3.17: YOLO v3 network framework

Darknet-53 [28] is defined as a neural network which is convolutional and it has a depth of 53 layers.

Fig 3.18 shows the framework of YOLOv3 along with Darknet-53 which is used as the backbone. The network is already trained on millions of images from the database of ImageNet.

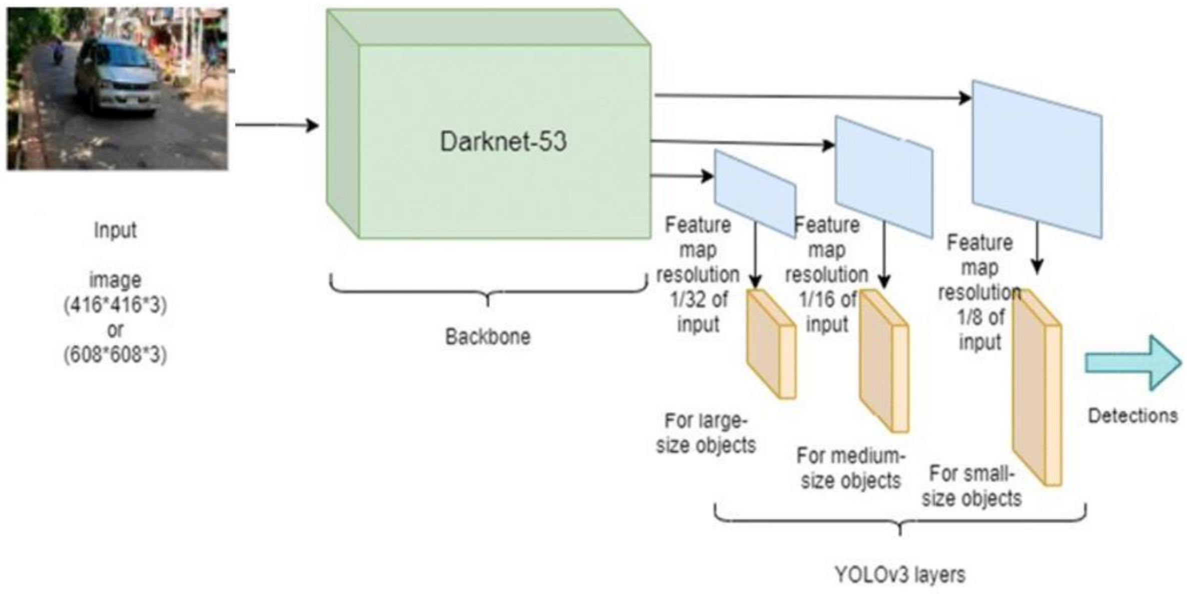


Fig. 3.18: YOLOv3 along with Darknet-53

As already mentioned previously, YOLOv3 has 3 scales, which are 32,16, and 8. As the frame size of the input image on fig. 3.18 is 416 x 416, the image will be down sampled by these 3 scales. After down sampling, it will be 13x13, 26 x26, 52 x52.

Fig. 3.18, it can be seen that the resolution of feature map for these 1/32, 1/16 and 1/8 input. All of these are called YOLOv3 layers [30]. The below points summarizes the usage of the 3 scales on the basis of the size of detection:

- 1/32: service for the detection of large-size objects,
- 1/16: service for the detection of medium-size objects
- 1/8: service for the detection of small-size objects

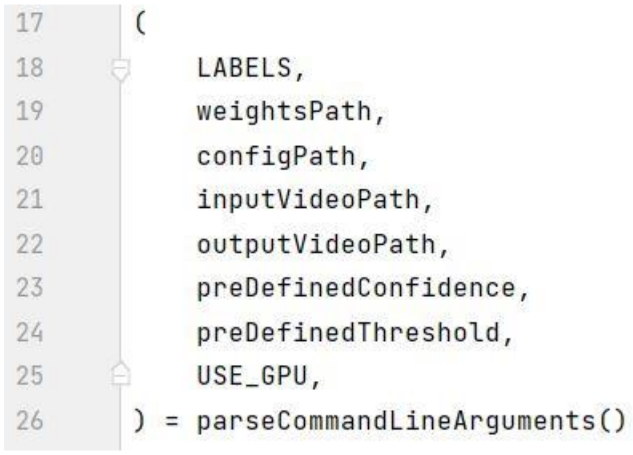
YOLO v3 is good at detecting small vehicles and also can detect vehicles when there is low lighting condition [32].

3.7.2 Vehicle Detection

Traffic video footage is fed to the input the vehicle detection model. In this case, high resolution mobile camera was used to the video of the traffic rather than CCTV surveillance footage. The width and height of the video is converted to 416 x 416 as default by the following python command below:

inputWidth, inputHeight = 416, 416

The weights file on the COCO data is imported to the model and the configuration file is imported [33]. The input video path is set to the model. The confidence value and threshold value are defined in Fig. 3.19.

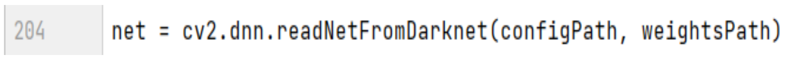


```
(
    LABELS,
    weightsPath,
    configPath,
    inputVideoPath,
    outputVideoPath,
    preDefinedConfidence,
    preDefinedThreshold,
    USE_GPU,
) = parseCommandLineArguments()
```

Fig. 3.19: Weight file, configuration file, video input path declaration

GPU in CPU can also be used, in that case, the flag needs to be changed and that provision in the model.

Darknet framework is loaded into the model in which the weights file was trained on the COCO dataset. Fig. 3.20 shows the loading of the darknet into the model algorithm.



```
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
```

Fig. 3.20: Darknet framework loaded

If the confidence value of the detected vehicle is more than the predefined confidence, the vehicle will be detected.

Fig 3.21 shows the video image before vehicle detection and fig 3.22 illustrates the output of the model after vehicle detection. It can be seen that only the mentioned vehicle classes are detected and output by the model.

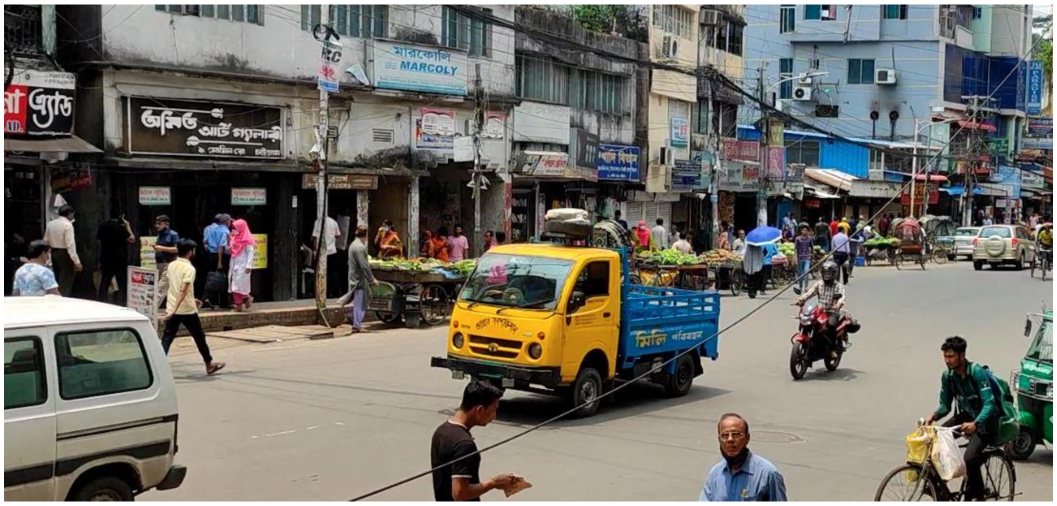


Fig.3.21: Traffic scenario before vehicle detection

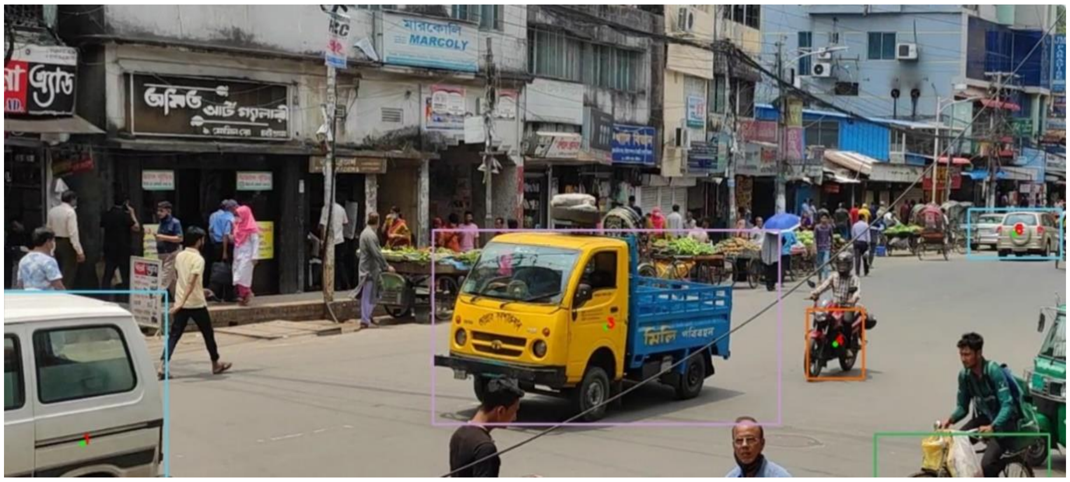


Fig.3.22: Model output after vehicle detection

Different colors of the BB have been used to represent the detection of the different types of vehicles (bicycle, car, motorbike, bus, truck) in the model using NumPy different color generator.

Fig.3.23 shows the NumPy color generator function used in the model. Fig 3.24 shows the different classes of vehicles are given different colors bounding boxes.

```
29    np.random.
```

Fig.3.23: NumPy color generator used in model

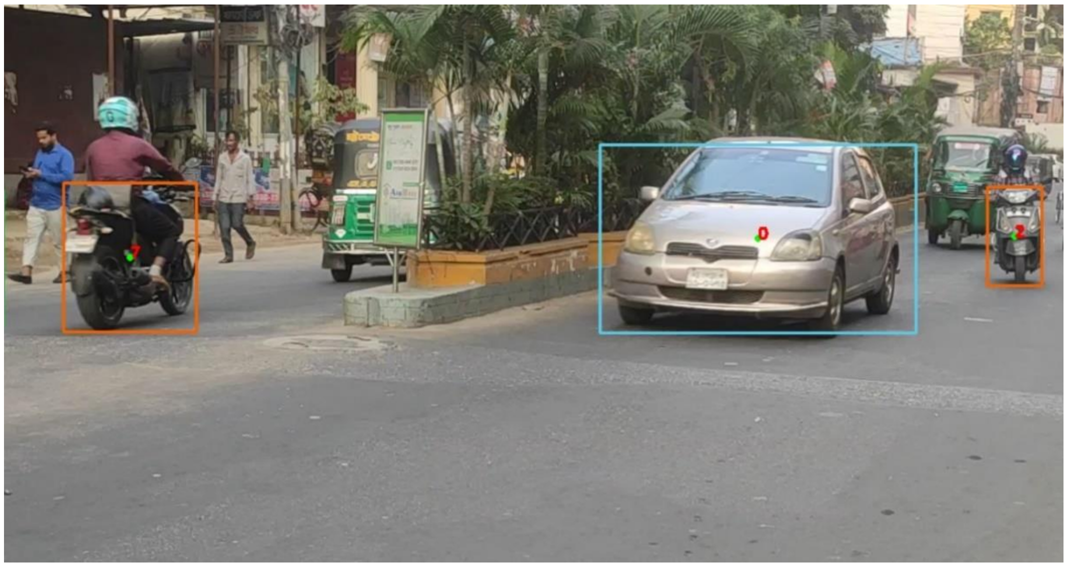


Fig. 3.24: Different class vehicles given different color BB

In Fig.3.25, it can be seen that the color of the BB drawn on the detected vehicles is of different colors for motorcycles and cars. Once the detection is made, detection boxes will be drawn with a green dot at the center:

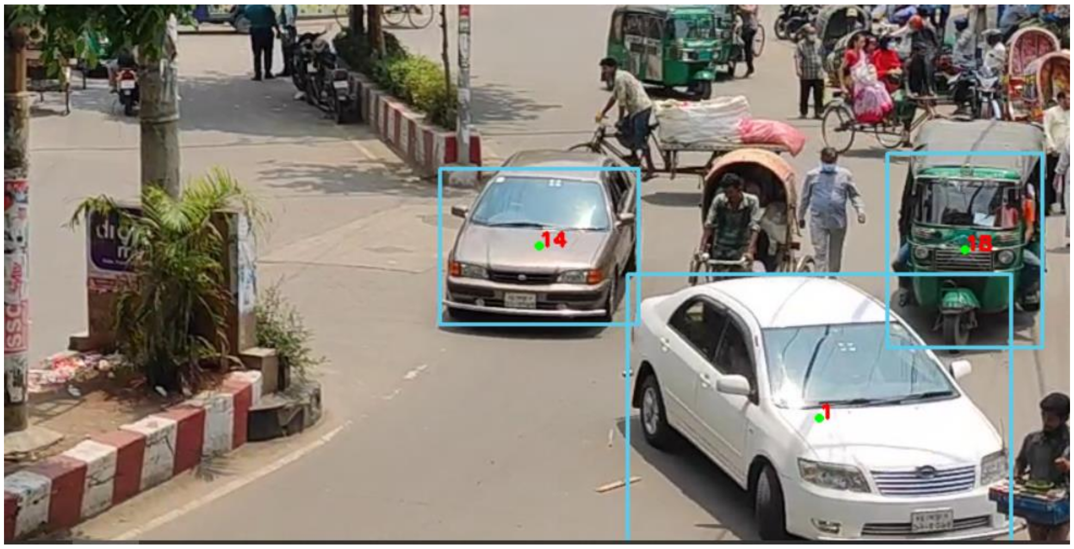


Fig. 3.25: Green dot at the center of each detection

If the current detection box is present in the previous frame, it is already detected and no detection box is drawn.

Fig. 3.26-Fig.3.28 show some of the detections during night time. Also, the model works during night time [34]. After detection, rectangular bounding box is put around each detection.

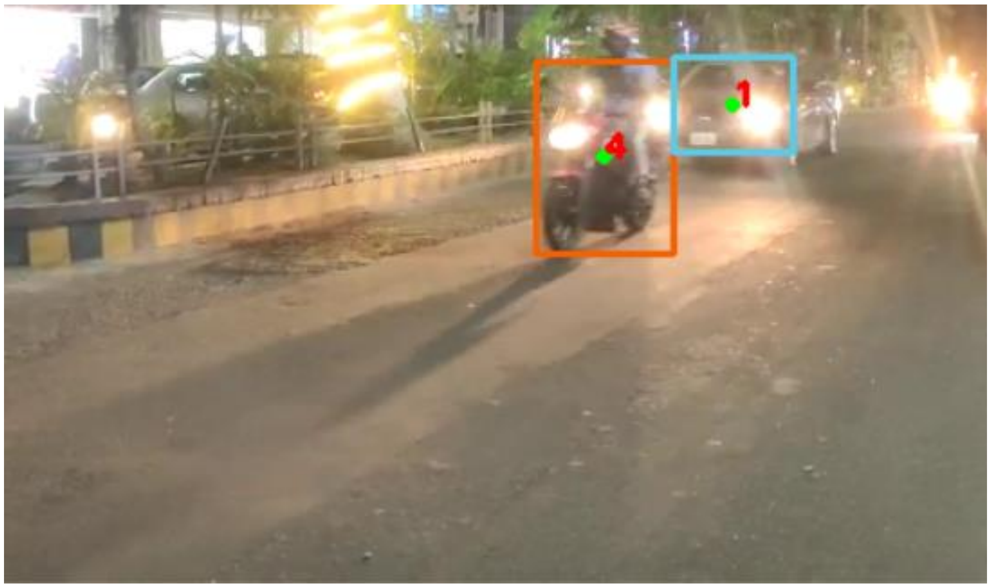


Fig. 3.26: Night time detection part (a)

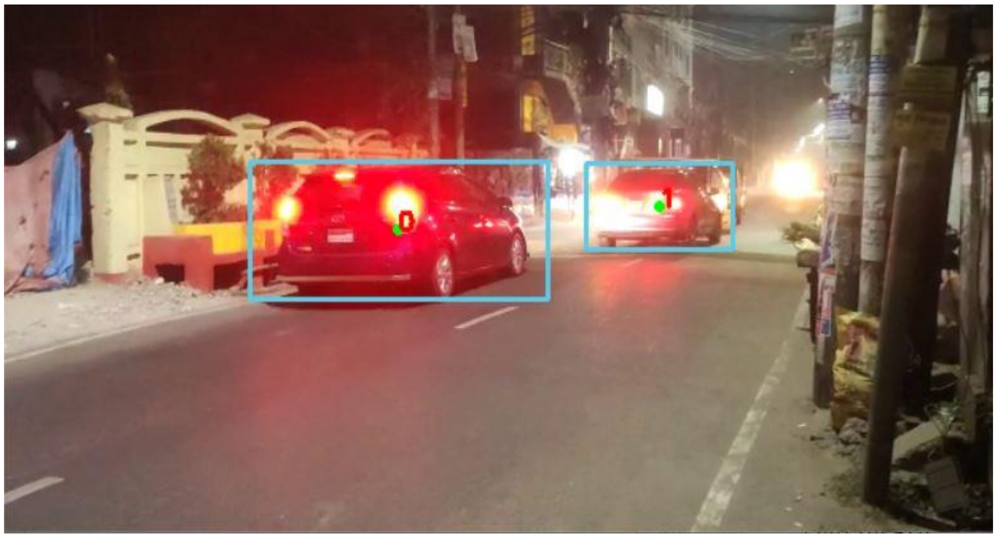


Fig. 3.27: Night time detection part (b)

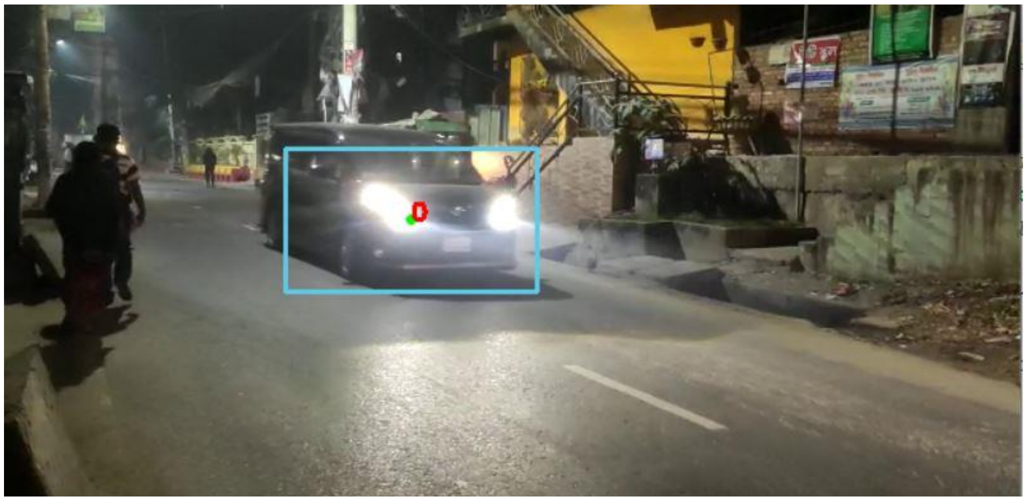


Fig. 3.28: Night time detection part (c)

3.7 Summary

The algorithm for vehicle detection using computer vision is implemented in Python. And PyCharm IDE is used. Vehicle detection is performed by YOLOv3 which incorporates computer vision. Vehicle detection is tested both during daytime and nighttime.

CHAPTER IV

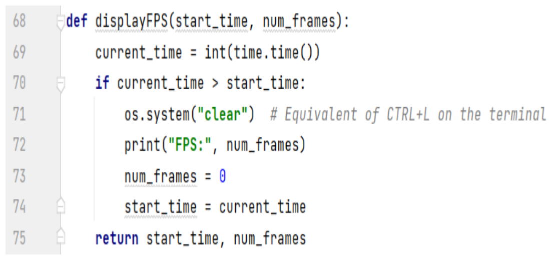
ESTIMATION OF TRAFFIC DENSITY

4.1 Introduction

In this chapter, how the detection of frames per second (FPS) is done from the computer vision-based model has been described and also the process for estimating traffic density at a specific region.

4.2 Detection of FPS from the Model

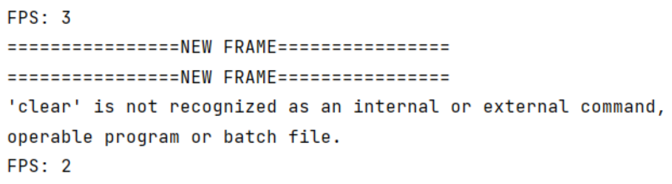
The model was set to calculate the frames per second (FPS) of the output traffic videos where the detection of the vehicles is seen and also the Traffic density Estimation (provided section 4.3). The model will calculate the FPS of the output and also will display the FPS. Fig. 4.1 shows the algorithm used to calculate the FPS in the model.



```
def displayFPS(start_time, num_frames):
    current_time = int(time.time())
    if current_time > start_time:
        os.system("clear")  # Equivalent of CTRL+L on the terminal
        print("FPS:", num_frames)
        num_frames = 0
        start_time = current_time
    return start_time, num_frames
```

Fig. 4.1: FPS calculation algorithm (a)

In Fig. 4.2, it shows the output of the FPS of the model from the model.



```
FPS: 3
================NEW FRAME================
================NEW FRAME================
'clear' is not recognized as an internal or external command,
operable program or batch file.
FPS: 2
```

Fig. 4.2: FPS calculation algorithm (b)

4.3 Traffic Density Estimation

To estimate the traffic density at a certain region was one of the aims of this study. Traffic density is stated as how many vehicles present on that region of the road at a specific time. In section 3.6, already the detection method and also the detection of the vehicles were described.

Fig. 4.3(a) and Fig. 4.3(b) show how dense a traffic situation can be at normal times in Dhaka. Also, in section 1.2, the intolerable traffic in Bangladesh, especially in Dhaka was described.

(a)

(b)

Fig. 4.3: Traffic situation in Dhaka

To predict the traffic density at a certain location at a specific interval of time. The model was set to count the no. of vehicles present within 5 points front and back from the middle line that was drawn in the video frame. All of the vehicles present within this region will be counted as per the vehicle detection list provided earlier and will account for the traffic density estimation. The model will count the vehicles and put a consecutive number on the vehicles after detection so that it is not counted again.

Traffic density is estimated as of below with respect to time, given below. Also, the output video is saved.

$$k = \frac{No\ of\ vehicles\ in\ the\ region}{30\ sec\ interval} \tag{7}$$

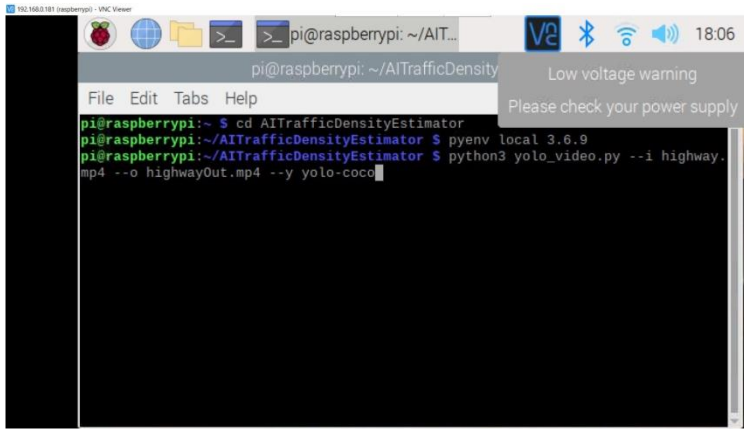
Where K is traffic density.

4.4 Hardware Implementation

The computer vision-based model was implemented in Raspberry pi 3B. This model can run from any version starting from 3B to the newer version. The implementation is done by following steps below:

- Install PUTTY
- Install VNC
- Connect the Raspberry PI with computer over WIFI or Bluetooth
- Install dependencies (Python3, pkg-config)
- Run the model

Fig. 4.4 shows the initiation of the model in Raspberry Pi. Alongside showing the output, the model also saves the output file in a video format.


192.168.0.181 (raspberrypi) - VNC Viewer
pi@raspberrypi: ~/AIT...
18:06
pi@raspberrypi: ~/AITrafficDensity
File Edit Tabs Help
Low voltage warning
Please check your power supply

```
pi@raspberrypi:~ $ cd AITrafficDensityEstimator
pi@raspberrypi:~/AITrafficDensityEstimator $ pyenv local 3.6.9
pi@raspberrypi:~/AITrafficDensityEstimator $ python3 yolo_video.py --i highway.
mp4 --o highwayOut.mp4 --y yolo-coco
```

Fig. 4.4: initiation of the model in Raspberry pi

4.5 Summary

The methodology that was used to estimate the vehicle traffic density is given in section 4.2. This concept and the result for vehicle density estimation are given in section 5.4.

CHAPTER V

RESULTS AND DISCUSSION

5.1 Introduction

This chapter describes all the results that have been obtained so far in this research. All the results are arranged in sub-section wise. The tabulation is given in Table: 5.1 and also results are plotted for analysis and conclusion.

The experimental and edge implementation results are provided in this section. Also, trajectory of the models will be plotted for analysis.

5.2 Vehicle detection from still image results

400 images were collected and input to the model 80 images of each class of vehicles that are going to detect (car, bus, motorcycle, truck, cycle) by the model. The images were taken by digital cameras. The results are tabulated in section 5.2.1 on the next page.

5.2.1 Result tabulation

The no. input images, original no. vehicle, no. of model detected vehicles are put in a table 5.1. Finally, the model accuracy is plotted against the classes/categories of the vehicles.

Table: 5.1: Vehicle detection from still image output

Category	No. input images	Original no. vehicle	No. of model detected vehicle	Model Accuracy on still image
Car	80	190	175	92%
Truck	80	103	94	91%
Bus	80	111	95	86%
Motorcycle	80	167	148	89%
cycle	80	185	168	91%

5.2.2 Vehicle detection from still image output

In Fig. 5.1, a few outputs of the vehicle class “car” after detection from the model is shown. The detections are from the input of still image to the model.

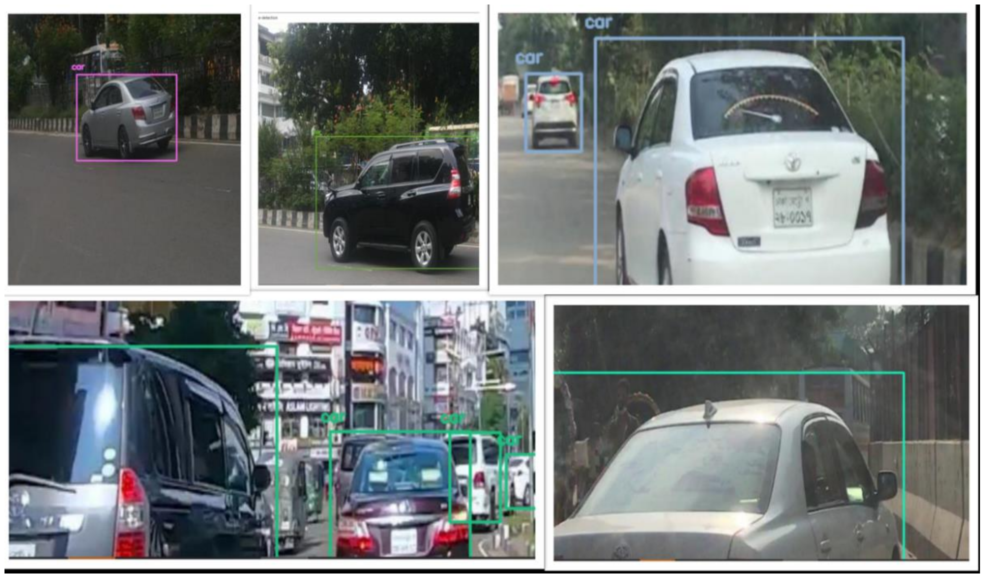


Fig. 5.1: “Car” detection output from the model

It is seen that from the detection of the model that each car within the range is detected and a bounding box, BB, is drawn after detection. Also, the vehicle class name is output by the model after detection.

In Fig. 5.2, a few outputs of the vehicle class “truck” after detection from the model is shown. For detections from still images, also vehicle category is labeled with the bounding box.

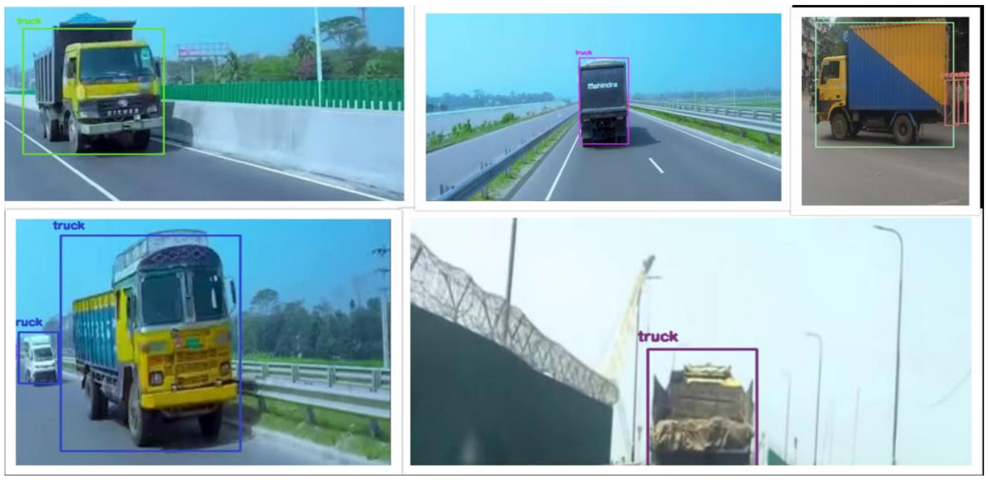


Fig. 5.2: “Truck” detection output from the model

In Fig. 5.3, a few outputs of the vehicle class “bus” after detection from the model is shown. The output is from the input of “bus” category vehicle from various roads.

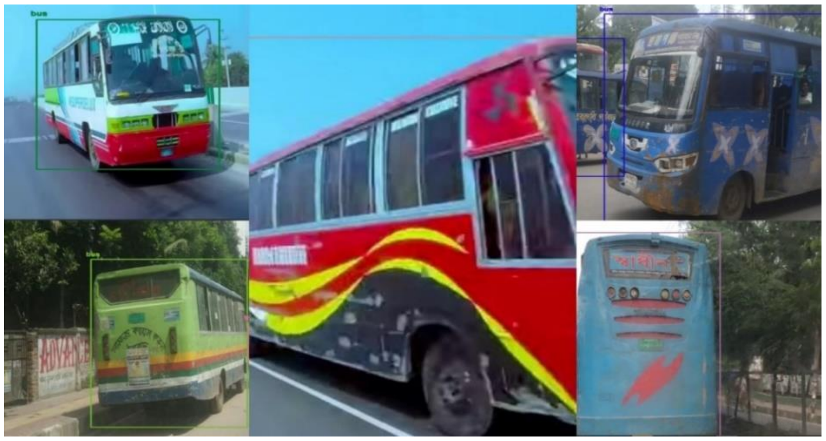


Fig. 5.3: “Bus” detection output from the model

In Fig. 5.4, a few outputs of the vehicle class “motorcycle” after detection from the model is shown. As mentioned previously, 80 images were input in total for each of the category of vehicle.

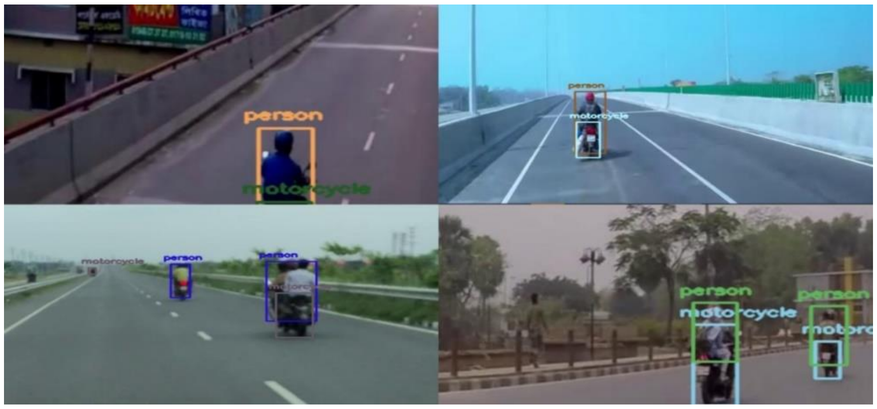


Fig. 5.4: “Motorcycle” detection output from the model

In Fig. 5.5, a few outputs of the vehicle class “bicycle” after detection from the model is shown. The output is shown on the same images that were given as input into the model in section 3.5.2.

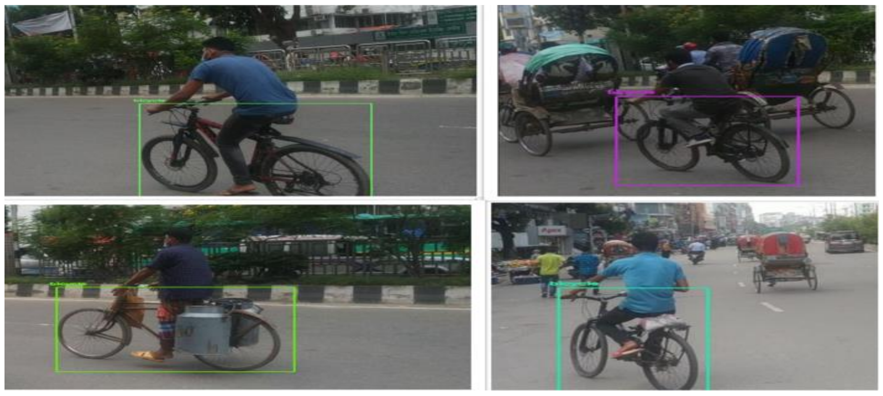


Fig. 5.5: “Bicycle” detection output from the model

The bar chart in Fig.5.6 and line chart in Fig.5.7 is the result of model accuracy plotted against the classes of vehicles for each class and also illustration of the trajectory of model accuracy against the class type of vehicles.

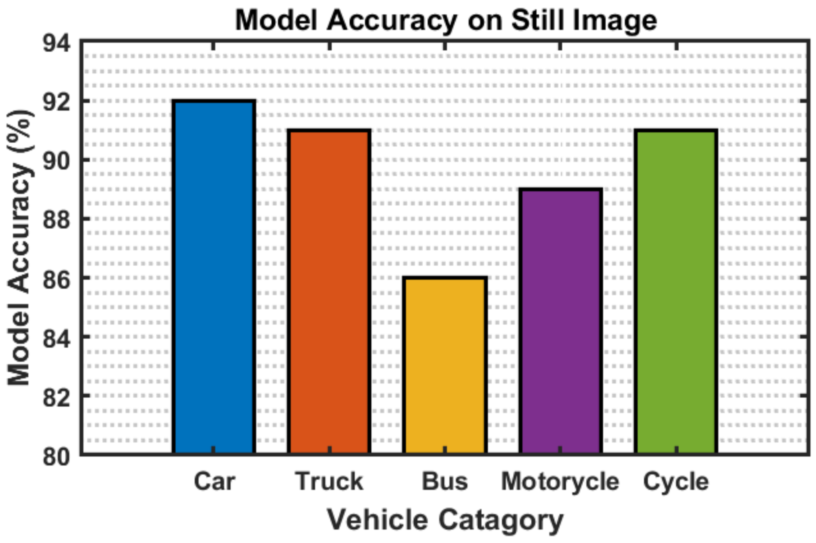


Fig.5.6: Model accuracy Vs vehicle category

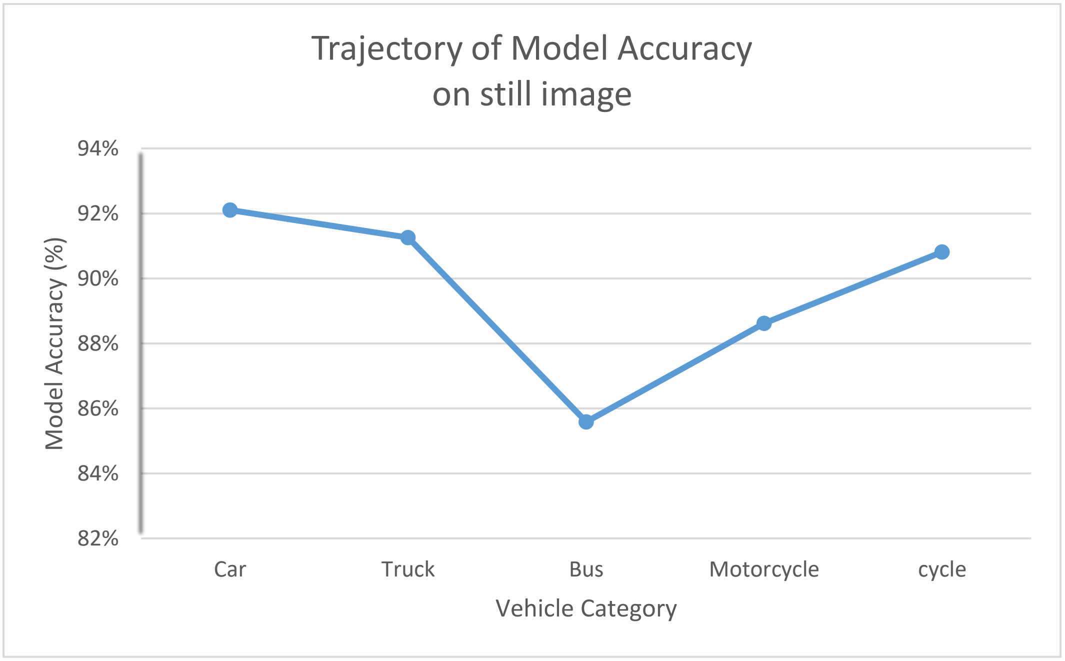


Fig.5.7: Trajectory of model accuracy Vs vehicle category

From the analysis of the bar chart and the trajectory, it can be concluded that the model produced the highest accuracy for the vehicle category “car” which is 92% and the lowest for “bus” which is 86%. This analysis is for still images.

5.3 Vehicle detection results and comparative analysis with real data

Real life traffic videos from different road scenarios were input to the model. Then the model was run. This is where artificial intelligence come to action. The model detected the vehicles as mentioned in the above 5 categories.

Fig. 5.8 shows one input snippet from the real-life traffic video footage and fig. 5.9 shows a snippet of the output from the model. Once the model detects the vehicle, it draws the rectangular bounding box around it and puts a green dot at its center. Real time traffic videos from highly dense roads and also from mediocre traffic roads were captured and input into the model.

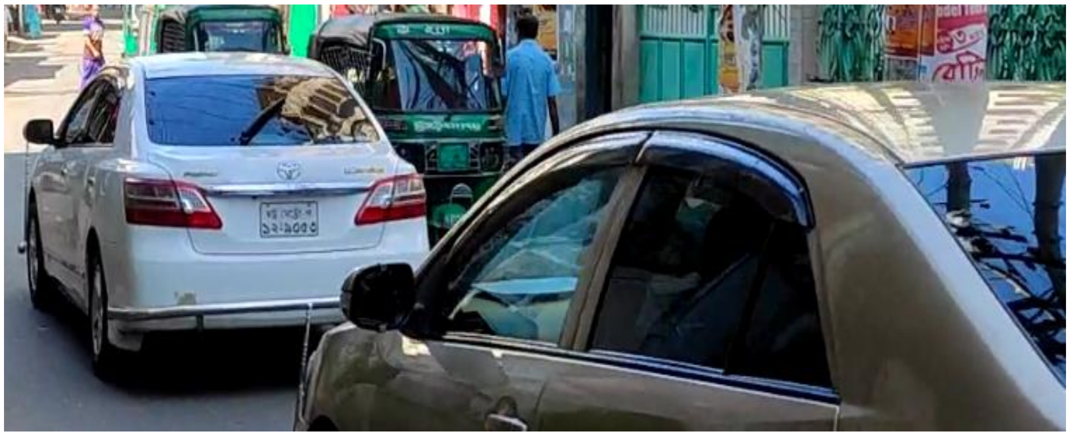


Fig. 5.8: Input to the model for vehicle detection

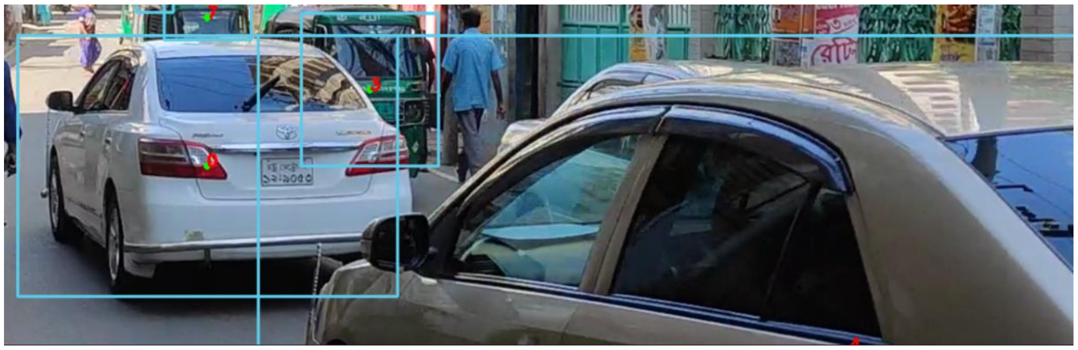


Fig. 5.9: Vehicle detection from the model output

The data for vehicle detection for each of the 5 classes of vehicles were taken and put in table 5.2.

Table 5.2: Model accuracy of different vehicle classes on traffic footage

Vehicle Class	No. of traffic video footage	Original No. of Vehicle	No of model Detected Vehicle	Model Accuracy
Car	5	25	22	88%
Motorcycle	5	33	29	88%
Bus	5	15	12	80%
Truck	5	14	12	86%
Bicycle	5	22	19	86%

Model Accuracy (%) Vs Vehicle Class is plotted in fig. 5.10 for traffic video footage in a bar chart.

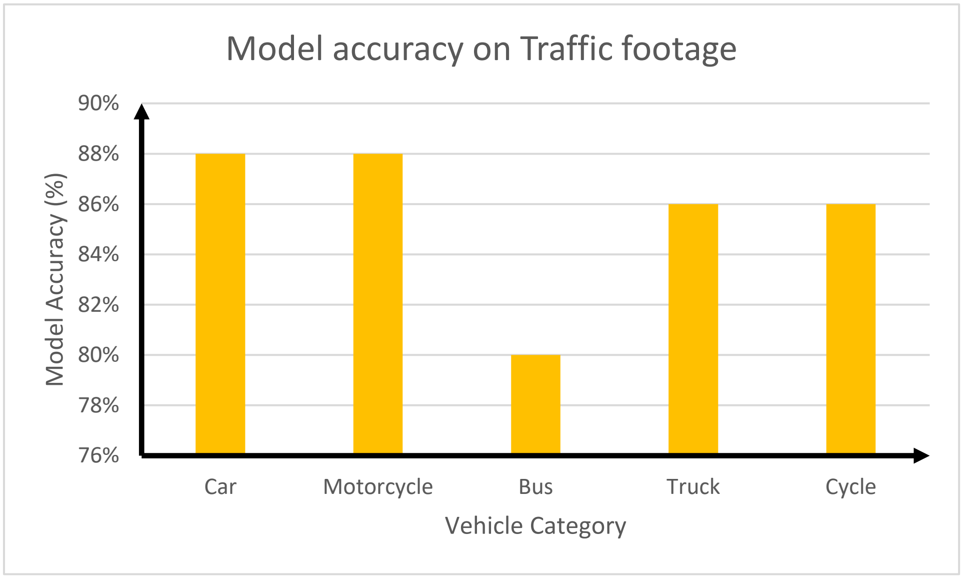


Fig. 5.10: Model Accuracy (%) Vs Vehicle Class (Traffic footage)

Fig. 5.11 illustrates the trajectory for the model accuracy (%) plotted against the vehicle category for traffic video footage.

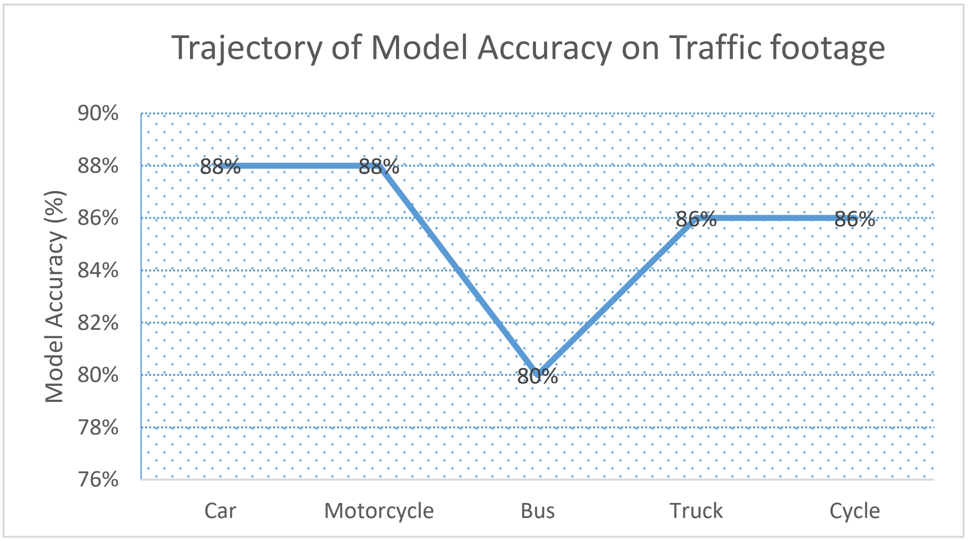


Fig. 5.11: Trajectory for the model accuracy (%) Vs Vehicle Class (Traffic footage)

From the bar chart and trajectory analysis of the model, accuracy plotted against the vehicle category, it can be seen that the vehicle category ‘car’ produces the highest accuracy in the model while live traffic video footage is input to the model.

The vehicle category ‘bus’ has the lowest accuracy produced by the model which is approx. 80%.

The result is similar to the one found previously with the model for accuracy on still images. Table 5.3 shows a comparative analysis between model accuracy on still images and model accuracy on traffic video footage against each vehicle category.

Table 5.3: Comparative data on model accuracy between still images and traffic footage

Vehicle Class	Model Accuracy on still image (%)	Model Accuracy on traffic footage (%)	Difference (%)
Car	92%	88%	4%
Motorcycle	89%	88%	1%
Bus	86%	80%	6%
Truck	91%	86%	5%
Cycle	91%	86%	5%

Fig. 5.12 is a comparison plot of the model accuracy on both still images and traffic surveillance footage.

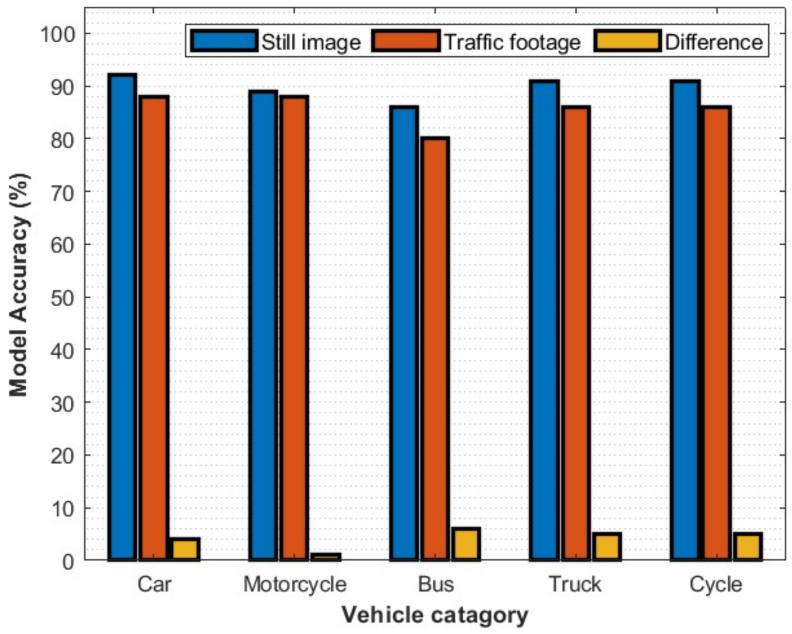


Fig. 5.12: Comparative analysis of model accuracy on still image and traffic footage

Fig. 5.13 illustrates the trajectory when the accuracy for both still image and traffic surveillance footage is compared.

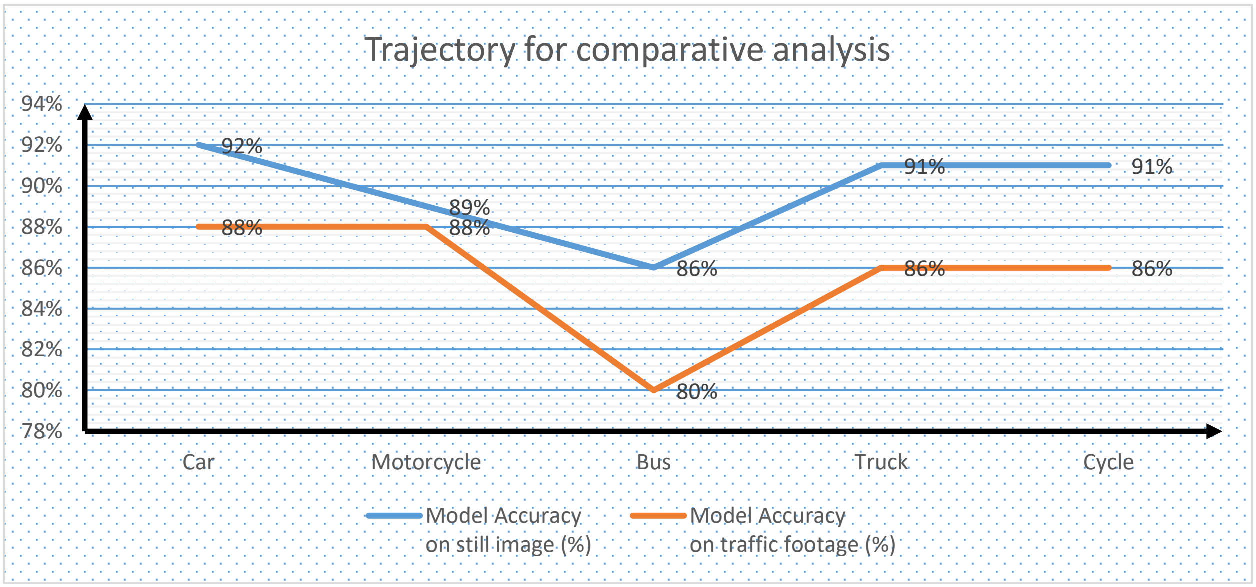


Fig. 5.13: Trajectory analysis of model accuracy on still image and traffic footage

From the trajectory analysis, it can be concluded that the vehicle category “car” has the highest accuracy in both still images and traffic footage. While the vehicle category “bus” has the lowest accuracy both in still images and traffic footage.

5.4 Traffic Density Estimation Results

Various traffic situation videos were input into model and the model detected, tracked, and estimated the traffic density, K as mentioned for the list of vehicles. Below input/output data for 5 different traffic conditions is given in table 5.4.

Table 5.4: Model accuracy results on different traffic situations

Traffic Situation	Original no. vehicles	AI detected traffic density (K)	Model Accuracy (%)	Category
Situation 1	11	9	82%	Highway
Situation 2	15	12	80%	Junction
Situation 3	7	6	86%	Street
Situation 4	12	10	83%	Busy Street
Situation 5	8	7	88%	City Road

Model accuracy Vs road Category for Traffic Density (K) is plotted below in fig. 5.14 in a bar chart.

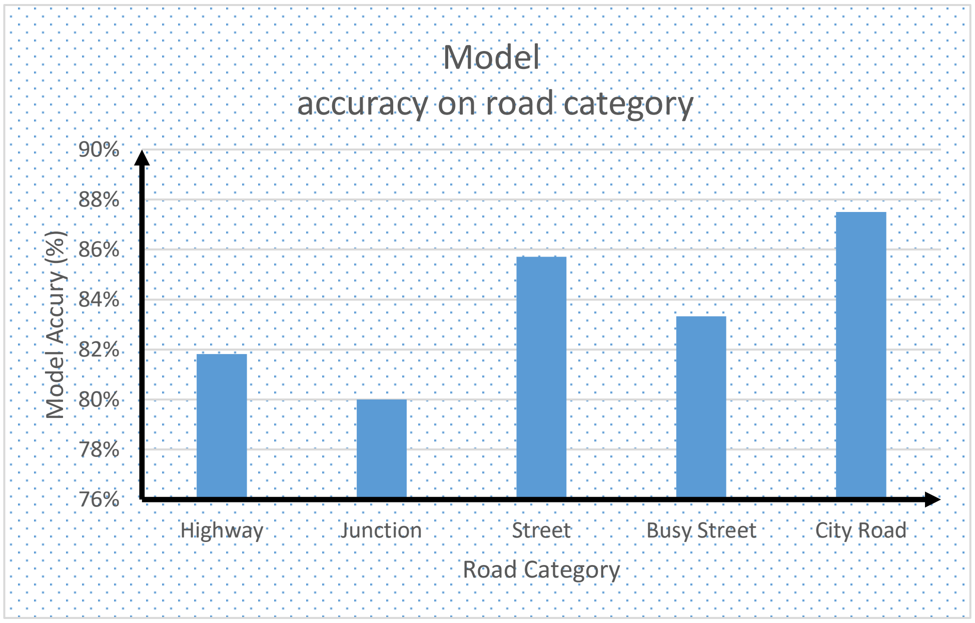


Fig.5.14: Model accuracy (%) Vs road category for Traffic Density (K)

Fig. 5.15 illustrates the trajectory of the model accuracy (%) on different road categories.

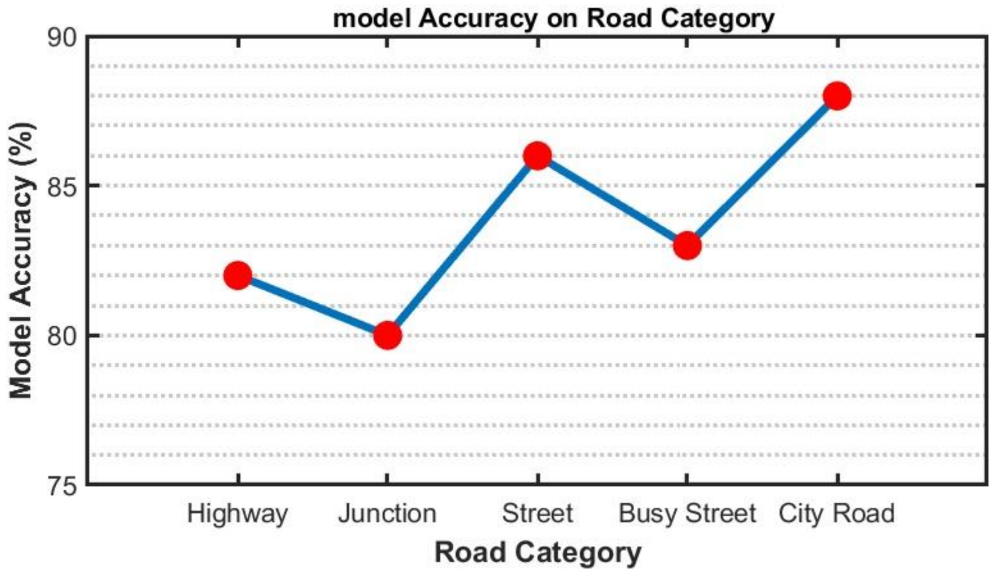


Fig. 5.15: Trajectory model accuracy (%) Vs road category for Traffic Density (K)

From the trajectory analysis, can be concluded that the model accuracy is highest in City Road conditions and lowest at a junction. For the City Road, the accuracy is 88% and for the Junction, the accuracy is 80%.

Fig. 5.16 and Fig. 5.17 show the snippet of traffic density estimation results in various situations. The output shows that each of the vehicles for those 5 categories the model is built upon, getting detected, being tracked, and finally the estimated vehicle density is being calculated and produced at the top left corner of the output screen.

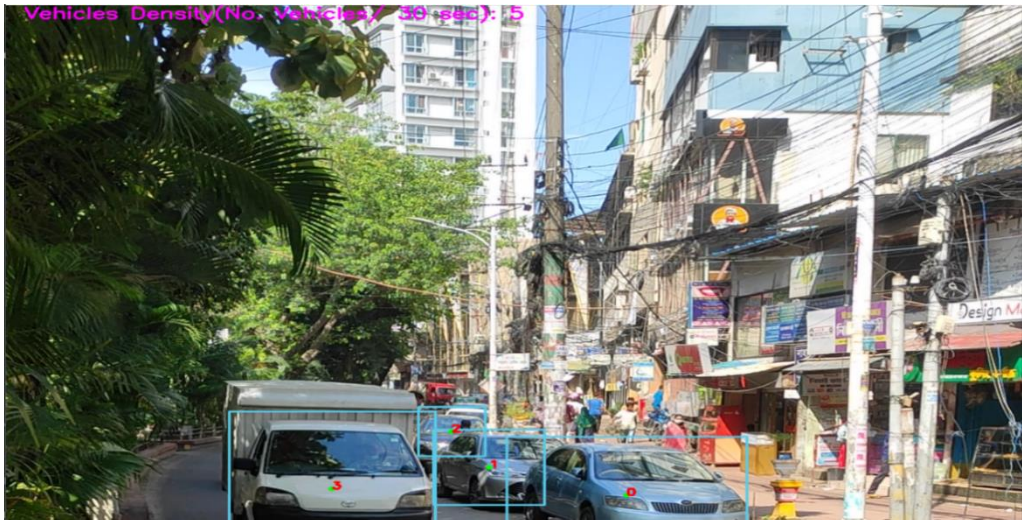


Fig.5.16: Traffic Density estimation at a city road

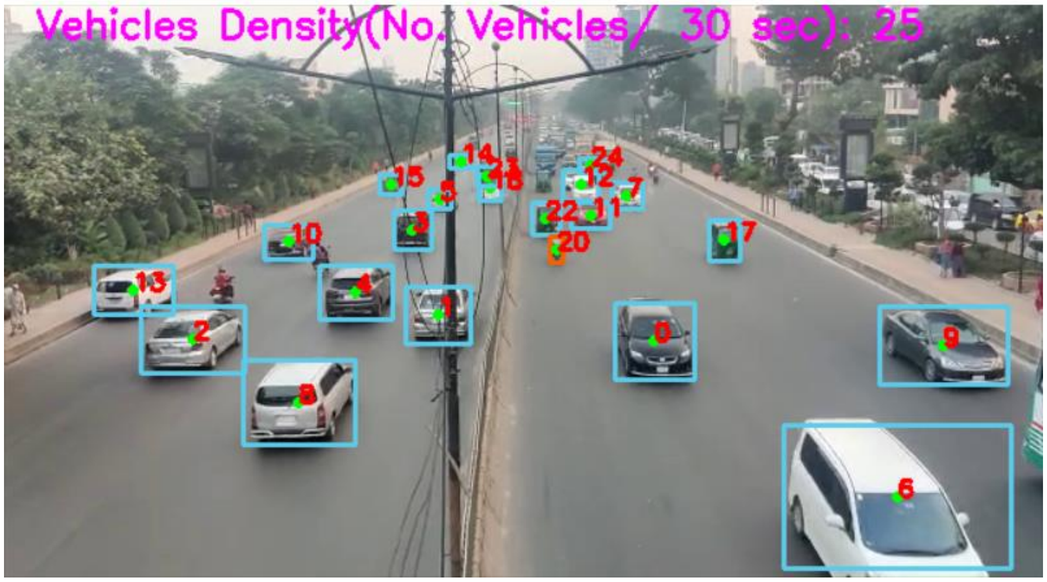
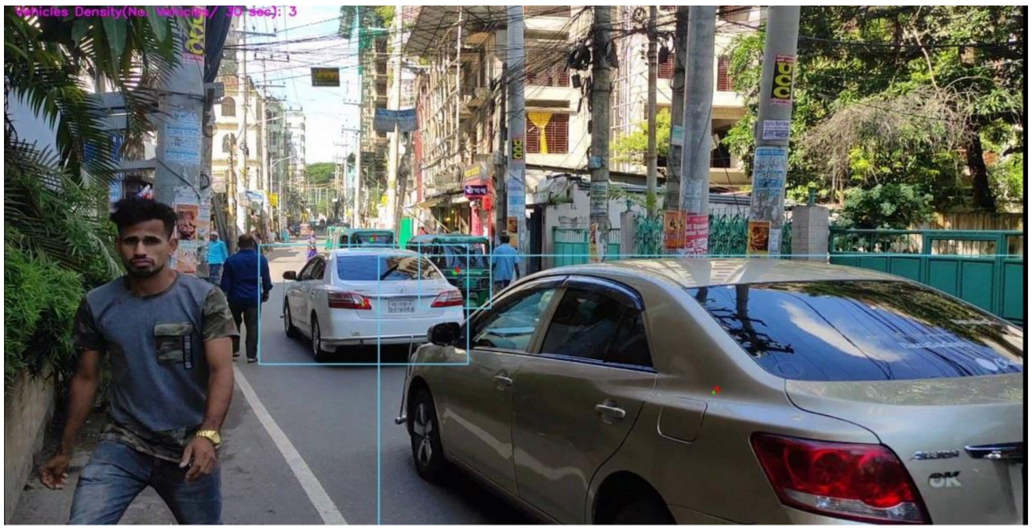


Fig. 5.17: Traffic Density estimation at a highway

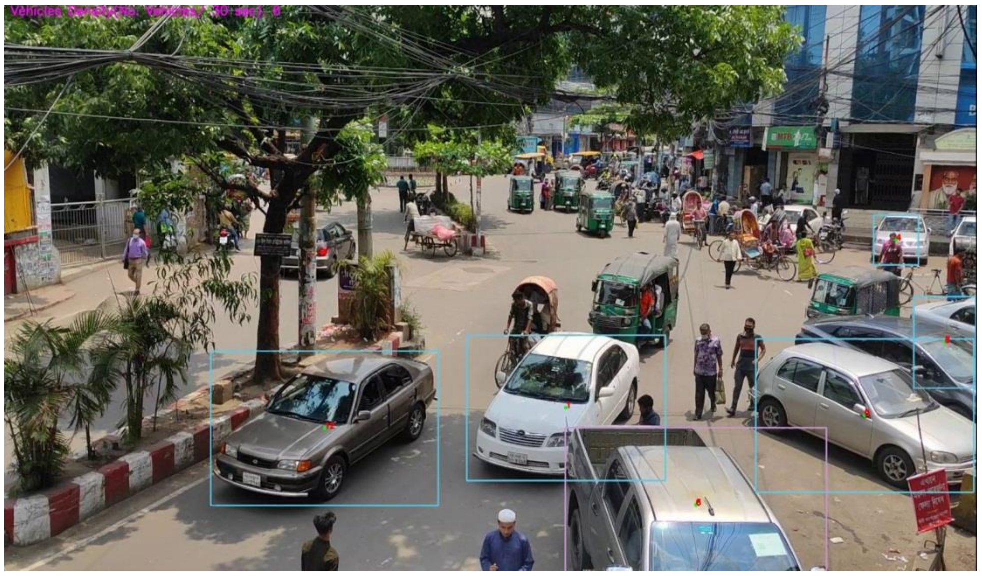
As mentioned previously, the time cost for traffic density estimation is approximately 1 second by the model. This is when the model was started, and the result of the traffic density estimation was viewed on the display approximately after 1 second of the start of the model.

5.5 Raspberry Pi Implementation Results

Fig. 5.18 (a) & Fig. 5.18 (b) below were a few of the outputs from the model when it was implemented in Raspberry Pi. As discussed in section 4.4, the model was run in the Raspberry Pi.



(a)



(b)

Fig. 5.18 (a) (b): Output of the model from Raspberry Pi

5.6 Cost estimation of the model

Below is the approximate cost estimation for the computer vision-based model. 3 different scenarios were considered for the cost estimation. Table 5.5 lists the scenario of different categories.

Table 5.5: List of different scenarios for cost estimation

SL.	Scenario category
Scenario 1	Considering new computer (CPU) & monitor
Scenario 2	Considering computer (Rpi) & Wi-Fi connection
Scenario 3	Considering computer already present in station

Table 5.6 provides the approximate cost estimation at a junction or traffic monitoring area for scenario 1 considering a new computer (CPU) & monitor.

Table 5.6: Scenario1, cost estimation considering new computer (CPU) & monitor

SL.	Item List	Cost / (BDT)
1	Close circuit (CCTV) Camera	2,500.00
2	CCTV Camera Cables	500.00
3	Computer: PC (CPU)	20,000.00
4	Monitor	10,000.00
5	Computer accessories	1,500.00
	Total cost	34,500.00

Table 5.7 provides the approximate cost estimation at a junction or traffic monitoring area for scenario 2 considering raspberry pi as a computer and the built-in Wi-Fi feature of raspberry pi.

Table 5.7: Scenario2, cost estimation considering raspberry pi and Wi-Fi connection

SL.	Item List	Cost / (BDT)
1	Close circuit (CCTV) Camera	2,500.00
2	Cables	500.00
3	Raspberry pi 3	5,000.00
	Total cost	8,000.00

Table 5.8 provides the approximate cost estimation at a junction or traffic monitoring area for scenario 3 considering a computer already present in the station.

Table 5.8: Scenario3, cost estimation considering computer already present in the station

SL.	Item List	Cost / (BDT)
1	Close circuit (CCTV) Camera	2,500.00
2	Cables	500.00
	Total cost	3,000.00

Table 5.9 shows the cost estimation of the 3 different scenarios at a junction or traffic monitoring area

Table 5.9: Cost estimation of the 3 different scenarios

SL.	Cost estimation / (BDT)
Scenario 1	34,500.00
Scenario 2	8,000.00
Scenario 3	3,000.00

Fig. 5.19 plots the cost estimation for 3 different scenarios in a bar chart.

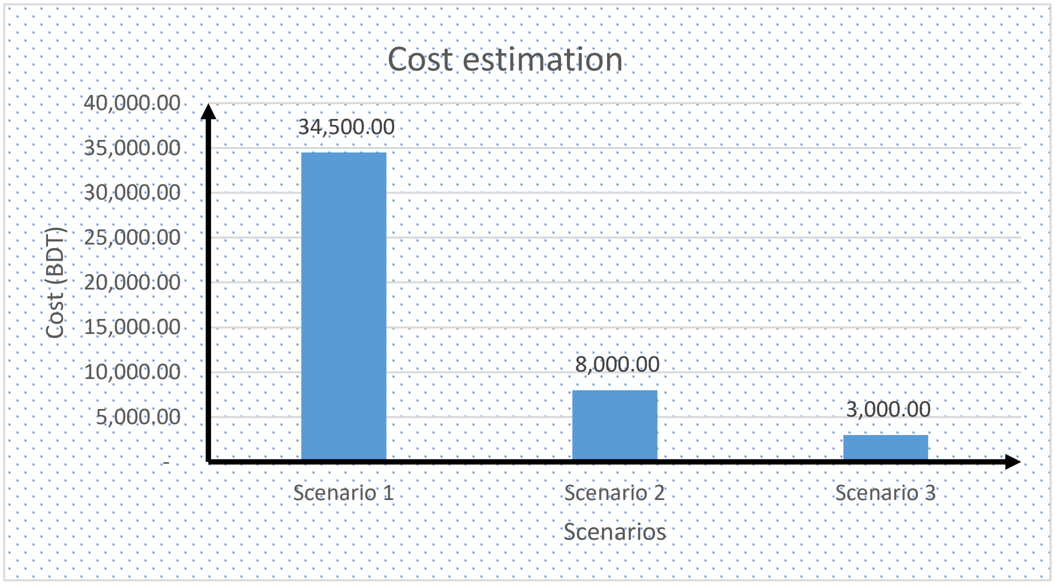


Fig. 5.19: Cost estimation of 3 scenarios for 1 junction of surveillance area

From the plot analysis above, can be seen that for scenario 3, the cost is estimated to lowest.

According to the data from "Chattogram Development Authority", there are 18 prime junctions in the city of Anderkillah, Chattogram. The cost estimation of Anderkillah City was surveyed computed in Table 5.10.

Table 5.10: Cost estimation of Anderkillah city

SL.	Cost estimation / (BDT)	No. of prime junctions	Total cost estimation (BDT)
Scenario 1	34,500.00	18	621,000.00
Scenario 2	8,000.00	18	144,000.00
Scenario 3	3,000.00	18	54,000.00

Fig. 5.20 plots the cost estimation for Anderkillah city, Chattogram considering the prime junctions.

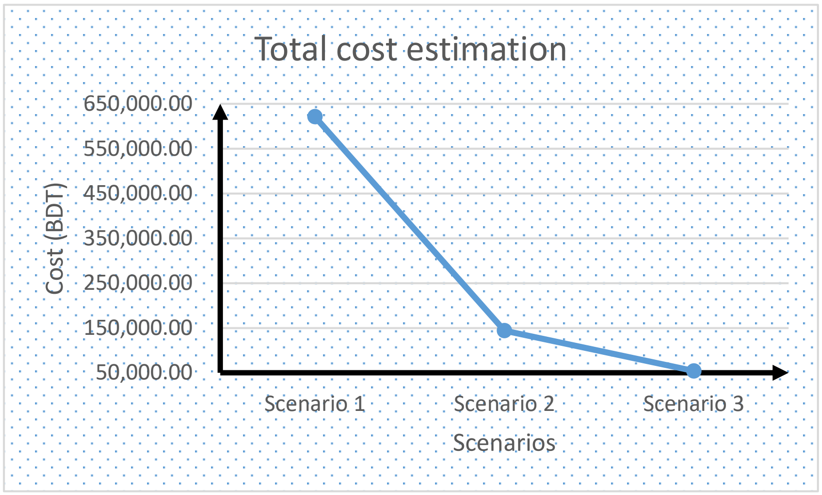


Fig. 5.20: Plot of cost estimation of Anderkillah city, Chattogram

From the results of the plot, it can be seen that the total estimated cost of the proposed model is the lowest (54,000.00 BDT) for Anderkillah city which is scenario 3. Also, have found that among these 18 prime locations, we considered, 75% of the prime locations already have a computer to which the model can be connected.

Cost estimation of the proposed model was compared with an existing model as below. Table 5.11 shows the cost estimation of the existing non-invasive type traffic density motoring system using ultrasonic devices to monitor traffic density at a region.

Table 5.11: Cost estimation of the existing model based on ultrasonic devices

SL.	Item List	Cost/BDT
1	Ultrasonic Device (6)	25,000
2	Cables	500
3	Ultrasonic Device Post	15,000
4	Joint Box	5,000
5	Model Device (Motherboard integrated)	50,000
6	Monitor	10,000
7	Model Device accessories	1,500
	Total cost	107,000

Anderkillah city, cost of the existing model = 107,000 x 18 = 1,926,000 BDT. Fig. 5.21 plots the comparison of the cost estimation between the proposed model and the existing model considering 3 scenarios given in Table 5.5.

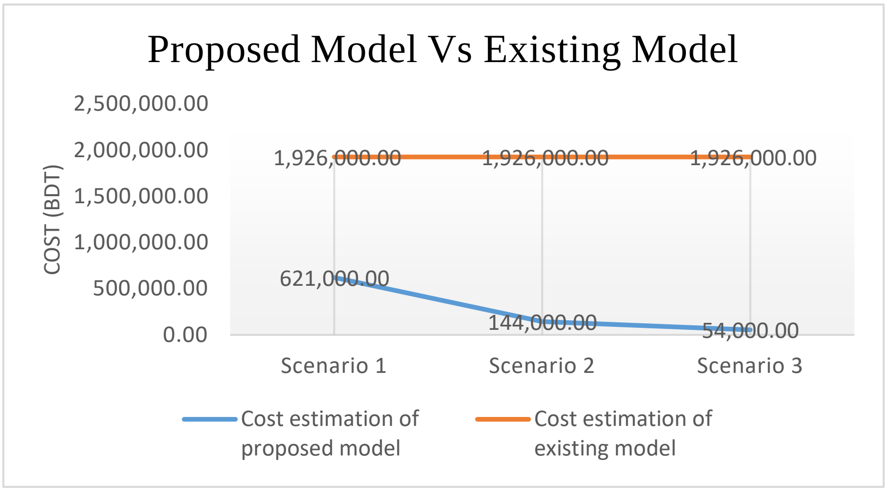


Fig. 5.21: Comparison of the cost estimation between the proposed model and the existing model

5.7 Discussion and Comparative Study

From the vehicle detection of still image result and vehicle detection of traffic footage result, it was found that model accuracy is highest for vehicle category "Car" and lowest for vehicle category "Bus". This could be due to the angle of camera position, speed of vehicle in the footage, lighting condition and camera resolution.

Also, it was found from the analysis of the result of vehicle detection that model accuracy for small vehicles such as "Car" and "Bicycle" comparatively have better model accuracy than large vehicle category such as "Bus", "Truck" etc.

The comparative analysis of the model accuracy result between detection from still image and detection from traffic footage showed that vehicle category "Bus" to have highest difference in accuracy.

It was found from the result of traffic density estimation analysis by the model that model accuracy was highest for city road condition where usually the vehicles move at an average speed. In the junction region, model accuracy was found to be lowest among the other traffic situation, which is believed to be increased if industrial grade CCTV camera could be used instead of the conventional one. When the cost estimation of the proposed model was compared with the existing model, proposed model was 3 times more affordable for scenario 1.

5.8 Summary

The model was tested for vehicle detections (for the mentioned vehicle classes) on still images and also in various traffic situation video footages. In all the cases, it was found that the model accuracy is highest for the car category.

Finally, the model estimated the Traffic Density (K) in various road traffic situations and the results were collected which was the ultimate goal of this research. The model also detected and tracked each class of vehicle. All the images and video surveillance of the traffic were captured with a mobile camera instead of a CCTV camera as mentioned earlier. Also, when CCTV cameras are used (in many junctions and traffic surveillance areas there are already CCTV installed for security purposes), the model can use this CCTV surveillance camera and hence this model of traffic density estimation and traffic monitoring will be more affordable.

The computer vision-based model was implemented Raspberry Pi 3b, which is a minicomputer. And the output of the model from the raspberry pi 3b was shown.

Also, from the plot of fig. 4.17, can be concluded that for scenario 3, the cost estimation is lowest for a city in Chattogram for example Anderkillah city which was only approximately 54,000.00 BDT. This denotes that the proposed model will be very affordable.

As it was found that the time cost for traffic density estimation is approximately 1 second, which is very negligible in terms of traffic scenario, can be stated that the model computes real time calculation for vehicle detection, vehicle tracking and estimating traffic density.

In the model, the frames per second (FPS) were 1,2,3 etc. If high-capacity GPU, were used, could get higher FPS from the model.

The proposed model will make a substantial contribution to detect, track and estimate vehicle traffic density in real time and can be used by traffic administration, security farms, private farms etc. Also, when connected with the recommended future works in section 6.2 it can lead to a novel smart traffic management system.

CHAPTER VI
CONCLUSIONS AND FURTHER WORK

6.1 Conclusion

The proposed traffic density estimation model can be used by traffic controller such that only one traffic controller can monitor the traffic and traffic density hence can coordinate the traffic light signal instead of three or four traffic controllers. Thus, these surplus traffic controllers can be deployed to other traffic regions where there is need for traffic monitoring and control and can use the proposed model. Consequently, more traffic region will be under autonomous traffic monitoring and will result in reduction in traffic congestion. The traffic controller at a junction can monitor in which road the vehicle density is more and can green signal to release the vehicles in that road. Hence, vehicles at the junction on a specific road will not be stuck for a long time leading to reduction of traffic congestion at that junction.

In this research, an affordable computer vision-based high-density traffic monitoring system was emphasized which may link up to various aspects of the Traffic Management system including utilization of this model by the traffic police administration. Still images and various traffic video footages were collected and tested with the model for vehicle detection, vehicle tracking, and traffic density estimation. When the model was used with Raspberry pi, the cost further reduces. The goal of this research was achieved which was to provide an affordable way that will be computer vision-based for high-density traffic monitoring systems. In this research, digital camera footage of traffic was used instead of CCTV camera footage. Hence the proposed model will make a substantial contribution to the traffic monitoring system if properly utilized.

This simple affordable infrastructure can be utilized for many purposes. The aim was to contribute a little to pave the pathways by affordable means which will help for traffic monitoring and ultimately reduce traffic congestion.

It is true that making an autonomous computer vision based traffic monitoring system will make traffic monitoring more robust and will contribute to a more hassle-free traffic situation. Parallelly, it is also necessary to increase public awareness of road safety, and traffic rules, which will ultimately lead to better traffic situation.

6.2 Future Scope of Work

For future work, the model can be trained for rickshaw, auto rickshaw, CNG, etc. which will make it more rigorous. Also, this model can be adopted by Traffic Police Department to better enhance the traffic monitoring system in Bangladesh. The real time traffic data can be incorporated through an app which will help the drivers to navigate more wisely and ultimately lead to less traffic congestion in different regions.

A prototype can be developed, where drivers will also get more information on the traffic density on their surroundings and will be able to drive more carefully and which will decrease the number of accidents.

CPU was used for the computations and model running, but if high powered GPU can be used, the output rate (FPS) will be more.

In comparative study between the result of the still image and traffic footage, it was found that vehicle category “Bus” has highest difference in accuracy. Further work can be done to mitigate the gap in the accuracy.

Also, along the proposed model, sensors can be installed in different vehicles and they can co-ordinate with each other vehicle, sharing traffic information to each other and also to the traffic controller. Thus, saving valuable time and money. By using the model, the traffic controller can co-ordinate traffic signals at the crossing properly which will reduce traffic congestion and enhance traffic monitoring at the crossing. Also, the system can be integrated so that traffic signals which are next to next can coordinate with one another and trigger on/off accordingly for better traffic control. Drone aid from next generation drones can help to enhance the traffic monitoring system even better. This drone aid system can be incorporated with the proposed computer vision-based high-density traffic monitoring system. All of these small systems could combine to make a smart intelligent traffic monitoring system that will increase road safety, security and advance the existing traffic monitoring and controlling infrastructure.

Reference:

[1] M. S. H. Onim, M. I. Akash, M. Haque and R. I. Hafiz, "Traffic Surveillance using Vehicle License Plate Detection and Recognition in Bangladesh," 2020 11th International Conference on Electrical and Computer Engineering (ICECE), 2020, pp. 121-124, doi: 10.1109/ICECE51571.2020.9393109.

[2] Huansheng Song, Haoxiang Li ang, Huaiyu Li, Zhe Dai and Xu Yum, "Vision-based vehicle detection and counting system using deep learning in highway scenes", in Journal of Song European Transport Research Review (2019), pp. 1

[3] https://www.nytimes.com/2016/09/23/t-magazine/travel/dhaka-bangladesh-traffic.html

[4] J. Lin et al., "From computer vision to short text understanding: Applying similar approaches into different disciplines," in Journal of Intelligent and Converged Networks, vol. 3, no. 2, pp. 161-172, June 2022, doi: 10.23919/ICN.2022.0010.

[5] K. M. Pai, K. B. A. Shenoy and M. M. M. Pai, "A Computer Vision Based Behavioral Study and Fish Counting in a Controlled Environment," in Journal IEEE Access, vol. 10, pp. 87778-87786, 2022, doi: 10.1109/ACCESS.2022.3197887

[6] A. S. Mohammed Shariff, R. Bhatia, R. Kuma and S. Jha, "Vehicle Number Plate Detection Using Python and Open CV," 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2021, pp. 525-529, doi: 10.1109/ICACITE51222.2021.9404556.

[7] A. Kumar and S. P. Panda, "A Survey: How Python Pitches in IT-World," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), 2019, pp. 248-251, doi: 10.1109/COMITCon.2019.8862251.

[8] A. Zaarane, I. Slimani, W. Al Okaishi, I. Atouf and A. Hamdoun, "An automated night-time vehicle detection system for driving assistance based on cross-correlation," 2019 International Conference on Systems of Collaboration Big Data, Internet of Things & Security (SysCoBIoTS), 2019, pp. 1-5, doi: 10.1109/SysCoBIoTS48768.2019.9028038.

[9] X. Zhang, B. Story and D. Rajan, "Night Time Vehicle Detection and Tracking by Fusing Vehicle Parts From Multiple Cameras," in Journal IEEE Transactions on Intelligent

Transportation Systems, vol. 23, no. 7, pp. 8136-8156, July 2022, doi: 10.1109/TITS.2021.3076406.

[10] H. Kuang, L. Chen, F. Gu, J. Chen, L. Chan and H. Yan, "Combining Region-of-Interest Extraction and Image Enhancement for Nighttime Vehicle Detection," in IEEE Intelligent Systems, vol. 31, no. 3, pp. 57-65, May-June 2016, doi: 10.1109/MIS.2016.17

[11] X. -Z. Chen, K. -K. Liao, Y. -L. Chen, C. -W. Yu and C. Wang, "A vision-based nighttime surrounding vehicle detection system," 2018 7th International Symposium on Next Generation Electronics (ISNE), 2018, pp. 1-3, doi: 10.1109/ISNE.2018.8394717.

[12] L. Ewecker, E. Asan and S. Roos, "Detecting vehicles in the dark in urban environments - A human benchmark," 2022 IEEE Intelligent Vehicles Symposium (IV), 2022, pp. 1145-1151, doi: 10.1109/IV51971.2022.9827013.

[13] G. Liu et al., "Smart Traffic Monitoring System Using Computer Vision and Edge Computing," in Journal IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 8, pp. 12027-12038, Aug. 2022, doi: 10.1109/TITS.2021.3109481.

[14] Vedant Singh, "Intelligent Traffic Management System," International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-3, September 2019, pp.4

[15] Lewandowski, Marcin & Płaczek, Bartlomiej & Bernaś, Marcin & Szymała, Piotr. (2018). Road Traffic Monitoring System Based on Mobile Devices and Bluetooth Low Energy Beacons. In journal of Wireless Communications and Mobile Computing. 2018. 1-12. 10.1155/2018/3251598.

[16] Dr. Suwarna Gothane, “A Practice for Object Detection Using YOLO Algorithm”, 2021 International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp. 1,4.

[17] https://www.kyosan.co.jp/english/product/traffic04.html

[18] https://appen.com/blog/computer-vision-vs-machine-vision

[19] https://www.datarobot.com/blog/introduction-to-computer-vision-what-it-is-and-how-it-works

[20] N Dewantoro, “YOLO Algorithm Accuracy Analysis in Detecting Amount of Vehicles at the Intersection,” IOP Conf. Series: Earth and Environmental Science 426 (2020) 012164, doi:10.1088/1755-1315/426/1/012164

[21] Vedant Singh, “Intelligent Traffic Management System,” International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-3, September 2019

[22] Redmon, Joseph & Farhadi, Ali. (2018), “YOLOv3: An Incremental Improvement,” Available: arXiv:1804.02767v1 [cs.CV].

[23] Q. Xu, R. Lin, H. Yue, H. Huang, Y. Yang and Z. Yao, "Research on Small Target Detection in Driving Scenarios Based on Improved Yolo Network," in Journal of IEEE Access, vol. 8, pp. 27574-27583, 2020, doi: 10.1109/ACCESS.2020.2966328.

[24] W. Dong, Z. Yang, W. Ling, Z. Yonghui, L. Ting and Q. Xiaoliang, "Research on vehicle detection algorithm based on convolutional neural network and combining color and depth images," 2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE), 2019, pp. 274-277, doi: 10.1109/ICISCAE48440.2019.221634.

[25] J. Hu, X. Gao, H. Wu and S. Gao, "Detection of Workers Without the Helments in Videos Based on YOLO V3," 2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2019, pp. 1-4, doi: 10.1109/CISP-BMEI48845.2019.8966045.

[26] Y. Hu, X. Wu, G. Zheng and X. Liu, "Object Detection of UAV for Anti-UAV Based on Improved YOLO v3," 2019 Chinese Control Conference (CCC), 2019, pp. 8386-8390, doi: 10.23919/ChiCC.2019.8865525.

[27] M. Nie and C. Wang, "Pavement Crack Detection based on yolo v3," 2019 2nd International Conference on Safety Produce Informatization (IICSPI), 2019, pp. 327-330, doi: 10.1109/IICSPI48186.2019.9095956.

[28] J. Liao and J. Zou, "Smoking target detection based on Yolo V3," 2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE), 2020, pp. 2241-2244, doi: 10.1109/ICMCCE51767.2020.00486.

[29] R. B. Diwate, A. Zagade, M. R. Khodaskar and V. R. Dange, "Optimization in Object Detection Model using YOLO.v3," 2022 International Conference on Emerging Smart Computing and Informatics (ESCI), 2022, pp. 1-4, doi: 10.1109/ESCI53509.2022.9758381.

[30] Y. Li, Q. Wang and R. Liu, "Research on YOLOv3 pedestrian detection algorithm based on channel attention mechanism," 2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI), 2021, pp. 229-232, doi: 10.1109/CEI52496.2021.9574546.

[31] Handalage, Upulie & Kuganandamurthy, Lakshini. (2021). “Real-Time Object Detection Using YOLO: A Review.” 10.13140/RG.2.2.24367.66723.

[32] W. Fang, L. Wang and P. Ren, "Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments," in Journal IEEE Access, vol. 8, pp. 1935-1944, 2020, doi: 10.1109/ACCESS.2019.2961959.

[33] M. Mahmud, M. S. Islam, A. Ahmed, M. Younis, and F.-S. Choa "Cross-Medium Photoacoustic Communications: Challenges, and State of the Art," Sensors, 22(11), pp. 4224, June 2022.

[34] Y. Miao, F. Liu, T. Hou, L. Liu and Y. Liu, "A Nighttime Vehicle Detection Method Based on YOLO v3," 2020 Chinese Automation Congress (CAC), 2020, pp. 6617-6621, doi: 10.1109/CAC51589.2020.9326819.