Smart Traffic Control System

(STCS)

By

Muhammad Bilal Khalid

Wishal Arshad

Sahar Fatima

Hamza Jahangir

Supervised by:

Dr. Alina Mirza

Submitted to the faculty of Department of Electrical Engineering,
Military College of Signals, National University of Sciences and Technology, Islamabad,
in partial fulfillment for the requirements of B.E Degree in Electrical (Telecom) Engineering.

June 2021

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

"Smart Traffic Control System"

is carried out by

Muhammad Bilal Khalid, Wishal Arshad, Sahar Fatima, Hamza Jahangir

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Electrical (Telecom.) Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor
Asst. Prof. Dr. Alina Mirza
Department of EE, MCS

Date: _____

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues and most of all supervisor, Dr. Alina Mirza and co-supervisor, Dr. Mir Yasir Umair, this could not have been possible without your guidance.

The group members, who through all adversities worked steadfastly.

Plagiarism Certificate (Turnitin Report)

This thesis has 10% similarity index. Turnitin report endorsed by Supervisor is attached.

Muhammad Bilal Khalid

208701

Wishal Arshad

240882

Sahar Fatima

240936

Hamza Jahangir

237096

Signature of Supervisor

ABSTRACT

Traffic congestion is one of the major issues in the present-day world. The existing traffic signal control works either on a fixed time delay basis irrespective of the vehicle density, or uses sensor based detection of vehicles. With the rapid increase in vehicle usage, such a system can prove to be inefficient causing a lot of losses. The aim of our project is to help reduce traffic congestion problem in Pakistan. It is specially designed for the infrastructure of Islamabad. It uses latest machine learning and image processing techniques to detect vehicles efficiently on the road and determine the vehicle count. The information is then used as an input to help decide the traffic flow time for each road around a traffic signal. In addition, emergency vehicles like local Pakistani ambulances will also be detected and prioritized, saving numerous innocent lives. The final output is shown using a Graphical User Interface (GUI) that depicts a traffic signal for a 4-junction road.

Table of Contents

List of Figures

Chapter 1: Introduction

Professional skills based on scientific experimental knowledge and science of numbers fashion a new branch of science known as Engineering. This branch of science is extremely important to society and its elements in many ways, especially the developmental research in the engineering domain that is boosting the standard of living for mankind.

Engineering is not only limited to the smaller research areas, but it covers entire industrial setups from visual constructions on some software to on-site practical work. Not only this, but this field of science also defines safety policies and evaluation measures. Engineers, who are practicing engineering principles, use their domain knowledge to invent and develop products that solve problems of the society. Either it is an issue of conveyance, medicine, astrology, atmosphere, or entertainment.

With the advancement in engineering knowledge, automation has found a respectful place in the present research interests. Automation is a product of engineering research. It is the process in which machines are learned according to the existing events and are programmed in a way to make decisions in the future either by prediction or by the previous trends. Formerly, mechanization is generally considered as a human labor replacement by the machines. Whereas automation is a general integration of machines into a delf governed system.

Automation has completely revolutionized research interests. Where it has been introduced, it has completely changed the on-ground realities. It is no shame to say that it has almost affected every domain of our lives.

The present-day problems demand automatic solutions that are efficient. Traffic administration problems such as traffic congestion should also be dealt with solutions based on latest technology.

1.1 Overview

Today's world is a world of digitalization. The growing tech field and exponential development in the fields of transport, medicine, metropolitan cities have become the most influential developments in the lives of mankind. Traffic administration is also a technical problem that can be dealt with effectively using engineering principles. World's population is increasing and so is the number of vehicles. Even the engineering principles and systems are insufficient to administer this large number of vehicles. It requires a lot of energy, time, and cost. A recent survey has indicated this fact that an average person spends almost 120 to 180 days of his entire life waiting for the green signal at any signal stop.[1] The survey also threw light on the fact that the recent traffic facilities are inadequate to control the distribution of traffic. Such inadequacy is also causing many traffic violations, like a red signal violation at some signal stop, speeding, etc, roadside accidents, and extreme traffic jams.

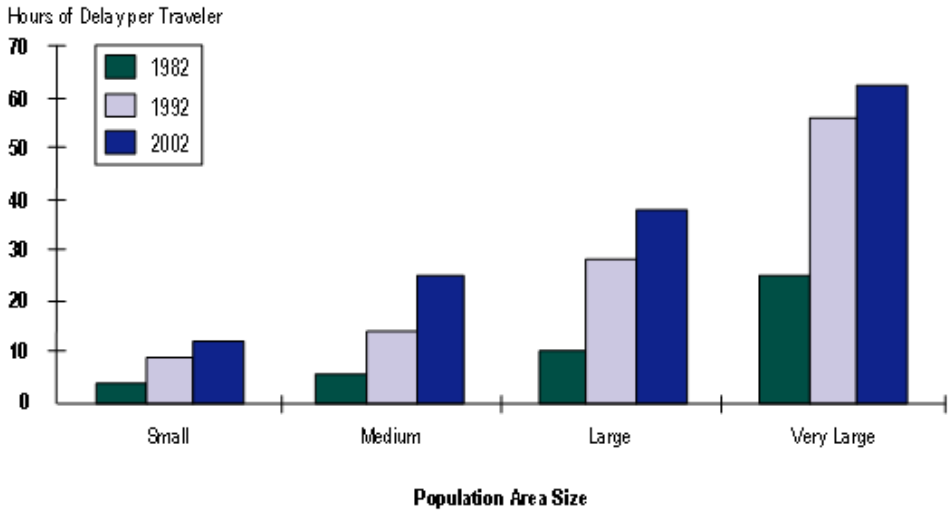


Figure 1: Congestion Grown in U.S. Cities over the Past 20 Years [1]

It is the need of the hour to fashion some policy or mechanism to avoid such huge traffic jams. Hence in our proposed system of traffic control, we not just focus on the signal timings but also encounter the traffic density to get the best results.

1.2 Problem Statement

Pakistan is a third world underdeveloped country. For traffic administration, traditionally traffic light system is used, but it has many limitations that lead to ever-increasing traffic congestion. Following are some highlights of the existing traffic control system.

1. Manual operation of traffic signals by traffic wardens increases the demand for manpower.
2. Empty roads are nothing to do with the traffic signal period that leads to time waste.
3. Large traffic congestion due to non-uniform flow of traffic about any junction.
4. Any sensors installed for traffic administration are mostly active. They must be re-installed, whenever the roads are reconstructed.
5. Increased human irritability is another cause of traffic violations.

1.3 Proposed Solution

The major goal of our proposed solution is to continuously monitor the traffic flow and vehicle density for all the roads around a junction in Islamabad, Pakistan. Traditional open-loop traffic signal systems must be replaced by smart closed-loop traffic signal systems. The proposed closed-loop traffic signal system is capable of learning and predicting the live traffic density. Another important feature of our proposed system is to detect emergency vehicles like ambulances or fire brigade trucks and give priority to them.

1.4 Working Principle

The project mainly works on the principles of image processing amalgamated with machine learning algorithms. The project is divided into different modulus and every module is inter-woven with the next module. The list of modules is as under:

- Datasets and annotations
- Dataset training and processing
- Output extraction
- Decision based upon Output
- Integration
- GUI presentation

1.4.1 Datasets and annotations:

The integral part of the project is the preparation of datasets. The dataset comprises of images of various type of vehicles present on the road: cars, buses, trucks, motorcycle and, ambulances. Ambulance dataset is of pivotal important in our project.

1.4.1.1 COCO Dataset:

Common Objects in Context (COCO) is a labeled dataset comprising of vast sets of common life objects, mainly 80 different types. [2] The objects of interest in COCO data set for this project are traffic vehicles, the proposed project extracts 6 such objects (person, car, motorcycle, bus, truck, fire hydrant).

1.4.1.2 Custom Ambulance Dataset:

This project uses a custom build dataset of Pakistani ambulances for its use. The images are gathered, filtered and annotated to obtain coordinates of the object of interest.

1.4.2 Dataset training and processing:

The prepared dataset is used as input to train object detection models using machine learning.

1.4.2.1 YOLO algorithm:

YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images. [3] Our project uses YOLO (you only look once) algorithm to train the dataset, this prepares object detection model.

1.4.3 Output Extraction:

The outputs are extracted on the basis of objects (person, car, motorcycle, bus, truck, fire hydrant) detected, these objects are counted and stored to keep a record.

1.4.4 Decision based upon Outputs:

The extracted outputs, the count of common vehicles and ambulance on the road, is used in decision making.

1.4.4.1 Density based decision:

The primary decision is based upon the density of the vehicles present on the road: prioritization on the basis of density.

1.4.4.2 Ambulance priority:

The vital decision related to ambulance are based upon the detection of an ambulance: special precedence on the base of detection.

1.4.5 Integration:

The different modules is then integrated in to one stand-alone entity. This stand-alone entity is essential for a compact solution.

1.4.6 GUI presentation:

The visual demonstration of the project is done through the aid of GUI (graphical user interface).

1.4.7 Raspberry-PI:

Raspberry-Pi is a small computer that is helpful in implementing the project physically. This project has a live camera attached with raspberry-pi which process the live feed and produces the desired outputs. [4]

1.5 Objectives

1.5.1 General Objectives:

"To build an innovative state of the art software integrated hardware prototype powered by Machine Learning (ML) and Internet Protocol (IP) techniques, providing a smart administrative tool to reduce the traffic congestion problem."

1.5.2 Academic Objectives:

- Development of a smart and intelligent Traffic administration System
- To implement Machine Learning techniques and simulate the results
- To increase productivity by working in a team
- To design a project that contributes to the welfare of society

1.6 Scope

This project finds its scope wherever there is road, a nearby school and hospital. It is an innovating state of the art software integrated hardware prototype powered by machine learning and image processing techniques, providing a smart administrative tool to reduce the traffic congestion and prioritizing ambulances over normal traffic to save the sacred life inside it as its not only about saving a single life but the whole humanity.

1.7 Deliverables

1.7.1 Hawk eye

It serves as a hawk eye to observe and decide the best traffic flow time by using a combination of image processing and machine learning techniques with the help of camera and a pre-fed data set in real time

1.7.2 Object of interest:

It can detect the object of interest by using the same combination of image processing and machine learning techniques. By detecting the object of interest, we mean detecting the emergency vehicles like ambulances and Fire Brigades vehicles and the pedestrians who are about to cross the roads.

1.7.3 Special privileges:

It provides the special privileges to the ambulances in order to save “The Sacred Life” that is inside it because it's not only about saving a single life but saving a family, and humanity so whenever it will detect an ambulance it will prioritize it over normal traffic.

1.8 Structure of Thesis

Chapter 2 contains the literature review and the background and analysis study this thesis is based upon.

Chapter 3 contains the design and development of the project.

Chapter 4 introduces detailed evaluation and analysis of the code.

Chapter 5 contains the conclusion of the project.

Chapter 6 highlights the future work needed to be done for the commercialization of this project.

Chapter 2: Literature Review

A new product is launched by modifying and enhancing the features of previously launched similar products. Literature review is an important step for development of an idea to a new product. Likewise, for the development of a product, and for its replacement, related to traffic system, a detailed study regarding all similar projects is compulsory. Our research is divided into the following points.

- Industrial Background
- Existing solutions and their drawbacks
- Research Papers

2.1 Industrial background

In today's era, one of the major issues faced across the world is traffic congestion. Traffic congestion has led to many further problems, as discussed in Problem Statement, which increases need for a smart system. Ultimately, results in a big marketplace for industrial development.

Initially, Pakistan Industries were barren. Then, these started exporting under liberal policies resulting in increase in industrial growth due to the rapid expansion of domestic demand and encouragement for export. Despite of declination of growth, Pakistan managed to make progress and growth in the new century. And now industries are inclining towards smart industries, automation, based on new technologies (Internet of Things (IOT), Machine Learning (ML) techniques, Artificial Intelligence (AI)). Hence, smart traffic system provides good market growth and impacts economy directly as it is a fully automated system.

2.2 Existing solutions and their drawbacks

Different solutions are previously being provided for the traffic congestion problem, but every product has some pros and cons. Following are some solutions which are already being prepared and being implemented.

- Microcontroller with fixed control
- Systems using edge detection techniques [5]
- Systems with fuzzy based controller and Morphological edge detection [6][7][8][9]
- IoT Based Smart Traffic Signal Monitoring [10][11][12][13]
- Smart Traffic Control System using YOLO [14][15][16]
- Automatic Traffic Density Calculation using SSD [17]

2.2.1 Microcontroller with fixed control / Manual Controlling

Traffic systems that are conventionally used are usually with fixed control or is controlled manually. For fixed control, is a microcontroller with a timer which opens each side of the road for a time slot that does not change irrespective of the density of vehicles on the road. It must allot time to a road which is even empty because there is no such check. To cater this issue, manual controlling is used. A person is controlling whole traffic, so, will not open a road if it is empty. But this requires manual power. Era like today where everything is being automated, such systems become outdated.

2.2.2 Systems using edge detection techniques

To automate a traffic signal, different detection techniques can be used to detect vehicles and measure their densities to control traffic. Different edge detection techniques are hence being provided for this purpose including Robert, Sober, Prewitt and Canny.[5]

Each of the techniques has its own advantages along with the disadvantages. However, many other techniques are being provided now which has replaced detection using these filters because of their less effectiveness. Moreover, while using video systems where object is extracted by consecutive pixels, number of static or unattended vehicles may be considered as no vehicles.[5]

2.2.3 Systems with fuzzy based controller and Morphological edge detection

It is based on the measurement of the traffic density by correlating the live traffic image with a reference image, of an empty road. Image matching is performed by the edge detection techniques. The percentage of matching is calculated which controls the traffic lights on the basis of comparing reference and input image. The greater is the difference of both images, greater is the traffic density. Obtained traffic density is then used to control traffic lights. Object detection can be improved using image segmentation and noise removal operations. But efficiency is not good using this technique if our interest is only vehicles. [6] It may count persons or other unnecessary stuff as vehicles, incrementing the vehicle counter.[7][8][9]

2.2.4 IoT Based Smart Traffic Signal Monitoring

These systems extract traffic densities from camera's output using Image processing techniques and sensors data. An algorithm predicts traffic density to control traffic congestion. RFIDs are also used to provide special privileges. The problem in this solution is use of sensors to predict traffic which may not work for long distance. The limitation of this technique is the accuracy which depends on sensors output. [10][11][12][13]

2.2.5 Smart Traffic Control System using YOLO

This system is based on a deep convolutional neutral network called You Only Look Once (YOLO). This algorithm detects the vehicles (any object of interest) and then density of each lane is counted. A real time traffic light control algorithm is used for this system. The purpose of this system is only limited to reduce waiting time for vehicles. The drawback of this technique is the bad performance in localization of boxes because bounding boxes depend totally on the dataset images. [14][15][16]

2.2.6 Automatic Traffic Density Calculation using SSD

In [17] Computer vision technology is used in the object detection process using Single Shoot multi box Detector (SSD) algorithm which detects the number of passing vehicles. SSD is capable of handling different shapes and sizes of the object. It also caters the different view angles. This technology is complex to implement, but high in efficiency (about 82.90%)[17].

2.3 Components Used:

After a month of research which included understanding concepts used by several individuals (students and professionals) in IEEE research papers, online research, and consulting with our professors, we decided to use raspberry pi 4 as hardware.

2.3.1 Raspberry pi

2.3.1.1 Specifications

This project uses Raspberry-PI model 4b, with 2GB RAM, 1.5 GHz 64-bit quad processor, two 2.0 and two 3.0 USB ports, built-in WIFI and Bluetooth, power and ethernet ports, and 4K display through HDMI port. [4]

2.3.1.2 Operating System

The Raspberry Pi just like computers needs operating system to work, the project uses Raspberry Pi OS (Raspbian). Raspbian is a Debian based operating system specially design for Raspberry Pi. One of the key points about Raspbian OS is that it provides user friendly interface that looks like many Windows and Mac OS. Raspbian provides a traditional CMD prompt as well as a User interface GUI.

2.3.1.3 General Configuration and Framework

2.3.1.3.1 Temperature meter:

Raspberry Pi is a temperature sensitive device. Its performance is directly affected from the temperature change. So, a constant check on the device temperature is utmost necessary, specially working with algorithms that requires large processing power. This project works with an on-screen temperature meter that shows temperature stats constantly of the device.

2.3.1.3.2 Cooling fans and Heat Sinks:

Cooling fans and heat sinks are additional accessories that can be attached with RPI device to reduce the internal temperature of the device.

2.3.1.3.3 Protective casing:

Due to its small design, RPI is a sensitive device. So, an external protective casing provides the needed agility to RPI to work in harsh environments.

2.3.1.4 Power Supply

Raspberry Pi needs a constant supply of voltage, 5V to be exact. There are two ways to supply power:

2.3.1.4.1 Direct connection:

Raspberry Pi can be attached with a USB-C connector at the designated port. Through this port a constant power supply can be send to the device.

2.3.1.4.2 POE (power over ethernet):

Raspberry Pi can be connected to a network using it's ethernet port. We can also provide power using the same port by using a special attribute of ethernet connections called power over ethernet. This method reduces the wire count by providing power and internet through one cable.

2.3.1.5 Monitoring Options

Raspbian OS provides CMD and user interface which can be observed on screen. Raspberry Pi can be connected to a monitor through a wire or can be accessed using IP.

2.3.1.5.1 VNC-local/global IP

An application called VNC uses local/global IP assigned to the Raspberry Pi to create an SSH connection, through which the device can be accessed wirelessly from any other device connected to internet.

2.3.1.6 File Sharing

Raspberry Pi can be used with the same SSH connection to transfer files to another device through IP. An application called WINSCP initiates a connection between two devices using their IPs, which can then share files on the network.

2.3.1.7 Camera and Configuration

Raspberry Pi can connect to live camera feed. There are two possible ways: a USB camera that can be attached to the USB slots present on RPI or, the RPI camera that has a dedicated camera connection port. This project uses a USB camera which can be easily configured.

2.3.2 Machine Learning (ML) Algorithms

This project works with two different ways of object detection method that are based on ML that are You Only Look Once (YOLO) and Single Shot Detector (SSD) Mobile Net.

The project mainly focuses on efficient ambulance detection along with a fast vehicle detection thus, we have used a combination of two object detection algorithms that can serve accordingly, to fulfill the requirements: SSD mobile-net (from here on referred as SSD) detects regular traffic fast and YOLO detects the ambulances efficiently. These algorithm works on the same basic principle of single shot detection using convolutional neural networks (CNN). SSD requires less processing power to detect objects, but yields low accuracy as compared to YOLO. Meanwhile, YOLO algorithm requires large processing power to detect objects, but provides high accuracy to detect objects.

2.3.2.1 You Only Look Once (YOLO)

YOLO algorithm detects and recognizes various objects in an image. It also shows the class probabilities of the detected images and through regression problem. This algorithm utilizes CNN

for real time object detection and requires a single forward propagation through a neural network. This means that the object detection prediction is done while the algorithm runs only once. YOLO comprises of various variants including tiny YOLO and YOLOv3.

The YOLO algorithm is important because it is fast and works in real time. Another important factor is its high efficiency with minimal background errors. [14] YOLO has excellent learning capabilities and is easy to use as well.

2.3.2.1.1 Working of YOLO

The YOLO algorithm works in the following three steps.

2.3.2.1.1.1 Residual Blocks

The image is divided in equal SxS grids as shown below:

Figure 2: Residual Blocks (x and y label shows the pixel numbers) [18]

The object will be detected separately in each grid cell.

2.3.2.1.1.2 Bounding Box Regression

A bounding box is made that defines the object in an image. It has a width (bw) and a height (bh). An object is considered to belong to a class. For example, car, motorcycle, person, ambulance etc.

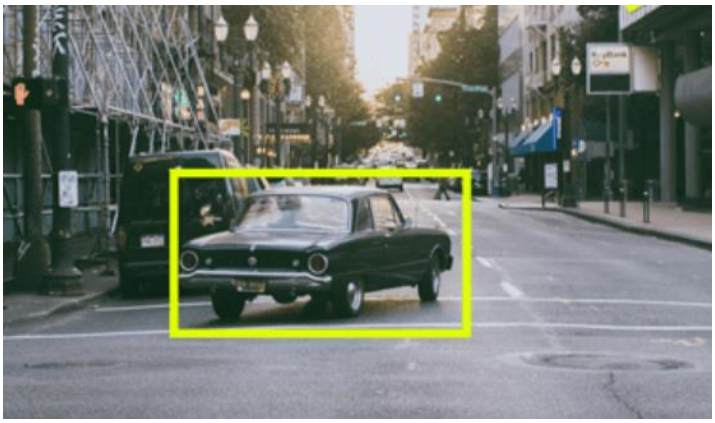


Figure 3: Bounding Box Regression [18]

The bounding box also depicts the probability of the object appearing inside it.

2.3.2.1.1.3 Intersection over Union (IOU)

This phenomenon describes how boxes overlap and provides an output box that fits the object completely. There are multiple grids, and each is responsible for prediction of object and has confidence scores. If equal to 1, then the bounding box is the same as real box. In this way, the bounding boxes that are not equal to real boxes are eliminated.

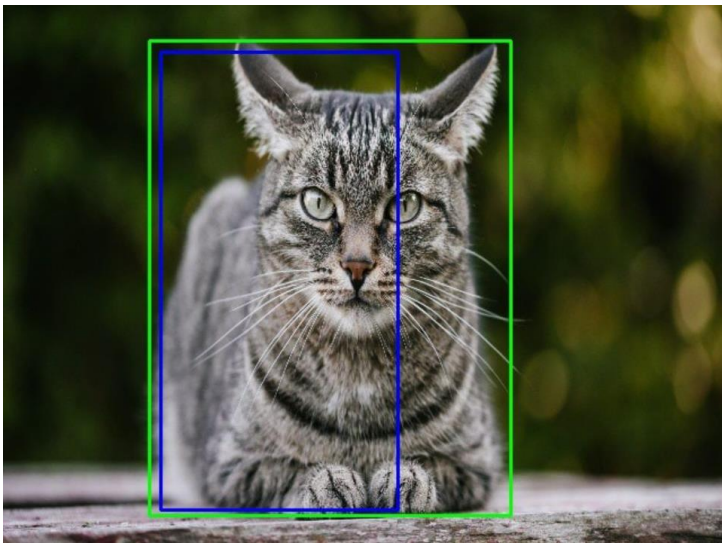


Figure 4: Intersection over Union

2.3.2.2 Single Shot Detector (SSD) Mobile Net

Main objective of SSD Mobile net [17][19] is to make neural networks lighter and portable for mobile and embed vision applications, this objective is primarily achieved by using depth-wise separable convolutional and introducing two global hyper parameters that efficiently tradeoff between latency and accuracy. A Light object detection model with an acceptable accuracy. Detects patterns rather than complete objects.

Architecture of SSD:

Regular convolution v. Depth wise separable convolution:

Applies single filter to every individual channel, point wise convolution uses 1x1 convolution to combine the outputs of depth wise convolution.

Standard convolutions computational cost:

$\boldsymbol{D_K . D_K . M . N . D_F . D_F}$ **--- Equation no.1**

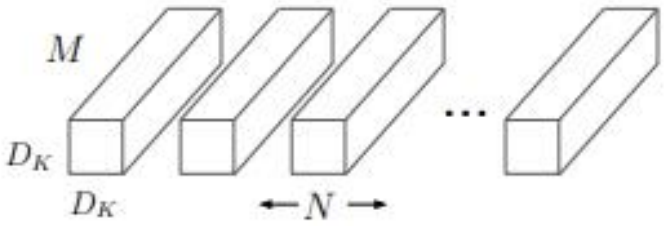
Depth wise separable convolution cost:

$\boldsymbol{D_K . D_K . M . D_F . D_F + M . N . D_F . D_F}$ **--- Equation no.2**

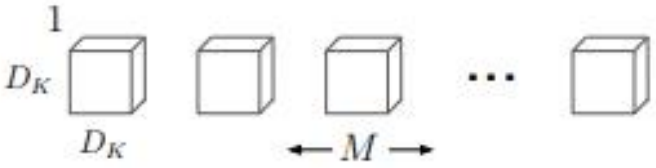
Ratio b/w standard and depth wise convolution:

The ratio shows a huge decrease in computational resources required. This implements that mobile-net uses 8 to 9 times less computation as compared to standard convolutions baring only a small reduction in accuracy.

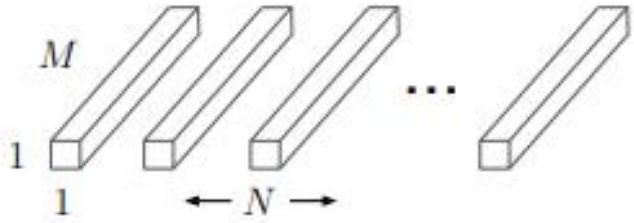
$$\frac{D_K.D_K.M.D_F.D_F+M.N.D_F.D_F}{D_K.D_K.M.N.D_F.D_F} = \frac{1}{N} + \frac{1}{D^2{}_K}$$ **--- Equation no.3**



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1 × 1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

Figure 5:Architecture of SSD [20]

Chapter 3: Interfacing and Detection

3.1 Ambulance Detection

Every year, around 150,000 people die in traffic accidents, which rounds up to 400 fatalities a day. [21] Out of these, 2 in 10 patients die before even reaching hospitals due to their ambulance stuck in traffic. More than 50% of patients that suffer from heart attack reach hospitals late. [21]The treatment that is initially provided to patients as first aid is considered very critical. It is vital that these patients reach hospitals on time and seek medical assistance but unfortunately, the average time in which an ambulance reaches a hospital is going up each year. [21]

Detection of ambulances is an essential part of our project. When it is detected, traffic signal for that road turns green, allowing it to pass without wasting any time being stuck in traffic. The process compromises of three parts. [22]

3.1.1 Preparing Dataset

Dataset refers to a compilation of instances that share a mutual attribute. Dataset is used to sculpt a machine learning algorithm going forward. The more data is provided, the better is the efficiency. In this model we used about 1300+ images of Pakistani ambulances which includes different organization's ambulances like Edhi Foundation, Al-Khidmat, Aman Foundation, Chippa, CMH, Red Cresent.

Figure 6-a : Proposed Dataset

To create bounding boxes, the software **LabelImg version Windows_v1.8.0** is used, in YOLO format.

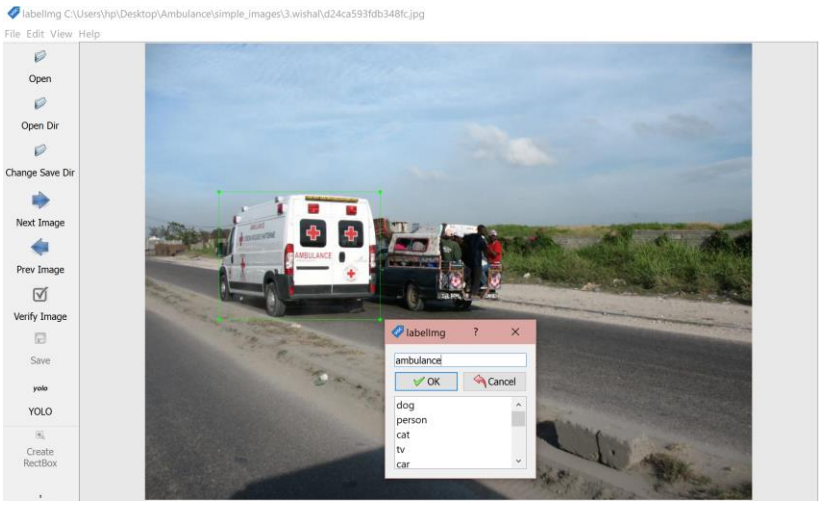


Figure 6-b: Bounding box in LabelImg

The output is an image in JPG format with a text file that describes the position of ambulance on image. It has the following attributes:

i) No. of classes

ii) Centre x of the object

iii) Centre y of the object

iv) Width of bounding box

v) Height of bounding box

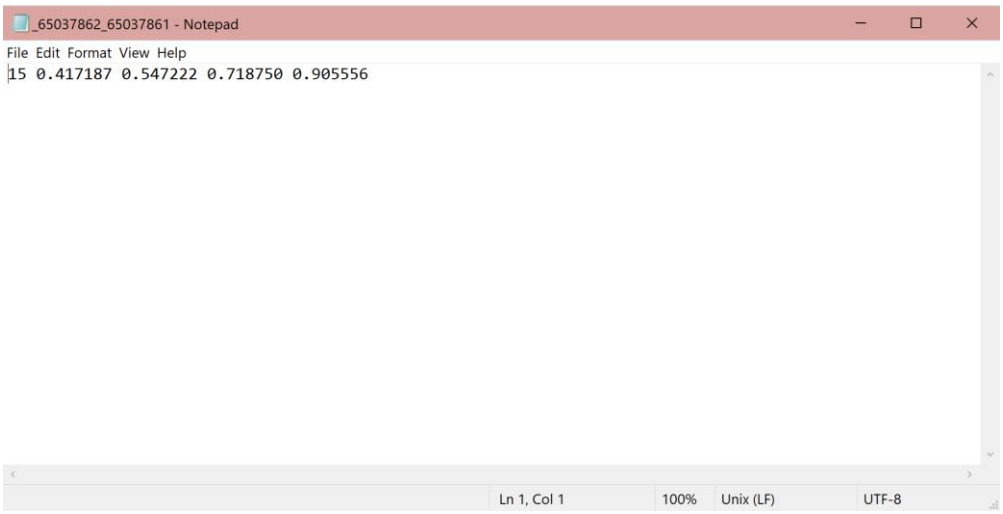


Figure 6-c: Text Document Attributes

3.1.2 Training

The next step is to train the Machine Learning model to detect the object. We used Google Colaboratory to gain free access to GPUs, A python editor and Ubuntu that is preinstalled. Google Colab offers several free GPUs that are very expensive otherwise including the following:

- HPJOG95A NVIDIA Tesla K80 Dual GPU Module.
- NVIDIA Tesla T4 Processore GPU computing 16GB – GDDR6 SDRAM
- NVIDIA Tesla P100 Processore GPU computing 16GB – HBM2
- NVIDIA Tesla V100 Processore GPU computing 16GB – HBM2

The access is limited only to 12 hours. Hence, we will be connecting it with google drive to save the weights file directly. The training process consists of the following steps:

3.1.2.1 Clone the darknet:

Darknet is an open-source neural network framework that is speedy and easy to install. Since Darknet supports GPU computation so we clone the darknet.

3.1.2.2 Compile Darknet using NVIDIA GPU:

It is important to ensure that GPU and open-cv are enabled at this step.

3.1.2.3 Configure Darknet network for training YOLO V3

3.1.2.4 Extract Images:

This step is where we extract the zip file that contains dataset. Dataset comprises of jpg image files along with their text files.

3.1.2.5 Start the training:

The training process continued for about 6+ hours. The number of iterations were 1000. The more time the model is left to train, the better is the efficiency and the number of iterations is improved.

Figure 7: Training of YOLO model using Darknet

3.1.3 Testing

Till this point, the weight files are ready to use. It is better to download the latest weight files since it has the highest number of iterations. The configuration file was downloaded and used with the weights file and python code. The results are shown in Chapter 4.

3.2 Vehicle Detection

3.2.1 SSD mobile-net:

This part of the project works with the SSD mobile net algorithm for the vehicular detection. The major aim to use SSD is to create a fast-processing object detection program.

3.2.2 Dataset:

For the training of the SSB mobile-net algorithm Common Objects In Context (COCO) data set is used. The COCO dataset is a huge pre-annotated set of daily life objects. For the specific nature of the project the COCO data set is altered accordingly from 80 different objects, 5 common traffic vehicles(cars, buses, trucks, motorcycle and, ambulances).

3.2.3 Weights and configuration files:

The training of COCO data set using SSD mobile-net algorithm in a tensor flow environment yields two important files: weight files (.pb) and configuration files (.pbtxt). These two files are of utmost importance in detection of objects, a brief definition of these files is as below

3.2.3.1 Weight files:

The weight files store a model in an efficient way. In tensor flow environment the weight files use .pb (protocol buffer) format, storing models in binary form (computer readable).

3.2.3.2 Configuration files:

The configuration files are stored as .pbtxt (protocol buffer) format, which stores model and numerical readings in a human readable form. This allows user to read an alter files.

3.2.4 Code functional parts:

3.2.4.1 Weights and Configuration files:

As discussed above the weights and configuration files are of pivotal importance in our vehicle detection code.

```
configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weightsPath = 'frozen_inference_graph.pb'


net = cv2.dnn_DetectionModel(weightsPath, configPath)
net.setInputSize(320, 320)
net.setInputScale(1.0/ 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
```

Figure 8-a: Weights and Configuration files STCS code

3.2.4.2 Bounding boxes

This function detects objects using weights files and configuration files. Also, it draws bounding boxes around the objects and states objects needed to be detected.

```
def getObjects(img, thres, nms,  draw=True, objects=[]):
    classIds, confs, bbox = net.detect(img, confThreshold=thres, nmsThreshold=nms)
    if len(objects) == 0: objects = classNames
    objectInfo = []
    if len(classIds) != 0:
        for classId, confidence, box in zip(classIds.flatten(), confs.flatten(),bbox):
            className = classNames[classId - 1]
            if className in objects:
                objectInfo.append([box, className])
                if(draw):
                    cv2.rectangle(img, box, color=(8, 255, 8), thickness=2)
                    cv2.putText(img, className.upper(), (box[0]+10, box[1]+38),
                                cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
                    cv2.putText(img, str(round(confidence*100, 2)), (box[0]+200, box[1]+38),
                                cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
    return img, objectInfo
```

Figure 8-b: Bounding box STCS Code

3.3 Output shown on GUI

GUI is a “graphical user interface” that is used for making an application in any programming language that can be more friendly for the user to interact with. So basically, a graphical user interface is an application that consists of a lot of options like windows, buttons, segments and, a lot of other widgets that can interact with the main application in a very user-friendly way.

Normally if you visualize a statement in a graphical form, it will be easier for you to understand what that statement is all about. For example, if someone says you how much 2 + 2 is equal to then you will have to think that 2 + 2 is equal to 4 but if someone shows you 2 balls and then two balls more and then he asks you how much the balls you have seen in total so it will be too much easy for you to say that in total there were four balls[23].

Or if someone wants you to create a program that is going to expand a given body based on input that is going to be fed by the user so it will be difficult for you to understand in which direction it's going to be expanded but if this statement is given to you in a graphical user interface medium then

it will be too much easy for you to understand that in which direction your body is going to be enhanced or expanded because all that is visualized graphically on the window.

3.3.1 Ways to create GUI

In Python, there are multiple ways to create a graphical user interface that are mentioned below

3.3.1.1 Kivy [24]

Kiyi is an OpenGL ES 2 accelerated framework that is used for the creation of new and attractive user interfaces in multiple platforms. You can use this methodology in multiple platforms like MacOSX, Linux, Raspberry Pi, Android-iOS, and windows and it offers around 20 widgets with an open-source availability.

3.3.1.2 PyQT [25]

It is one of the most favored cross-platform that Python bindings implementing the Qt library for the Qt that was owned by the Nokia, applications development framework. Nowadays Qt is available for Linux/Unix, Windows, Mac OS X, Sharp Zaurus, and windows. It combines both the best of Python and Qt and then it is up to the programmer to decide whether he has to use Qt Designer to create visual dialogues or just simply create a program by simple programming.

3.3.1.3 Tkinter [25]

This is the most common method that is used for creating a graphical user interface this is because of its simplicity and compatibility with almost all the platforms on which Python can be used. It is commonly bundled with the Python using its Python standard GUI framework and TK. It is also open source and is available under the Python license. As discussed above this methodology can be used in MacOSX, Linux, Raspberry Pi, Android iOS and windows, etc.

3.3.1.4 WxPython [25]

It was implemented as a Python extension module, and it is a wrapper for cross-platform GUI library WX widgets. It is also an open-source wrapper and supports Windows, Mac OS, and Unix to create native applications.

3.3.1.5 PyGUI [25]

It is a graphical application cross-platform framework that is offered by Unix, Windows, and Macintosh. If you compare it to the other available GUI frameworks, then it will be far the simplest and the lightest to all the available options as the API that is used in it is purely synchronized with Python so GUI inserts comparatively a very little code between the Python application and the platform used by the GUI. So, the display of the application normally displays the very nature of the platform.

3.3.1.6 PySide [25]

PySide is a free and cross-platform GUI toolkit **Qt** that was initiated and sponsored by the pioneer Nokia. **Qt** is a user interface framework and a cross-platform application. Pyside is nowadays offered by Linux /X11, Mac OSX, windows, Maemo and in the close near future, it will also be going to support Android. It is also a provider to work with XML documents, databases, multimedia, GUI, and networks.

The main feature due to which it is preferred is its API compatibility with PyQT4. So, if someone wants to migrate to Pyside then the process is going to be hassle-free of the whole lot.

So, from the above-mentioned features of the options that are available for creating a graphical user interface one can see that every platform has its pros and cons. So, the usability of these platforms depends on the needs and requirements of the user.

In our final year project, the platform that we used for creating our graphical user interface is Tkinter and the sole reason behind this selection is its compatibility with Windows and Raspberry Pi without any much difference in coding and its working fashion that is asynchronous methodology.

3.3.2 Working of Tkinter:

This is the most common method that is used for creating a graphical user interface this is because of its simplicity and compatibility with almost all the platforms on which Python can be used. It is commonly bundled with the Python using its Python standard GUI framework and TK. It is also open source and is available under the Python license. As discussed above this methodology can be used in Mac OSX, Linux, Raspberry Pi, Android iOS and windows, etc [23].

To create a Tkinter application one needs to follow the following steps.

- Download the Tkinter library.
- Importing the Tkinter module.
- Creating the container (main window).
- Add widgets to the main window that are useful for your application.
- Applied suitable conditions to the widgets.

3.3.2.1 Importing the Tkinter module:

```
from tkinter import *
```

3.3.2.2 Creating the container (main window):

```
window = Tk()
window.title("Traffic Signal")
```

By using command window.title(“title”) you can give any name to the graphical user interface window.

3.3.2.3 Add widgets to the main window that are useful for your application:

For adding a button in window

m=button(parent window, option=value, ….)

where m is the name of button and in options we can have different choices like active background, active foreground, and command etc.

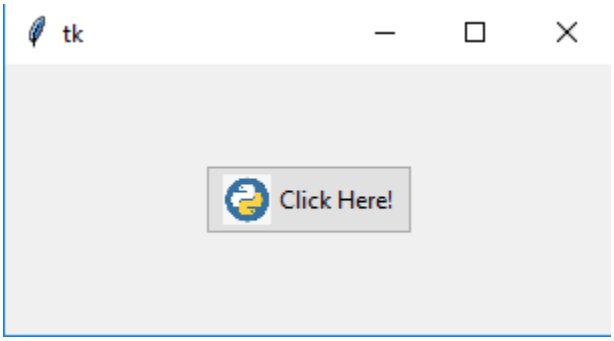


Figure 9: Tkinter library

Canvas command is used for adding shapes, graphics and graphs etc

3.3.2.4 Creating Shapes

3.3.2.4.1 Creating Rectangle:

Canvas.create_rectangle(x0,y0,x1,y1,fill=”color”)

Where x0 and y0 are the upper left coordinates of the rectangle from where it is starting and x1 and y1 are they lower right coordinates of the rectangle at where it is ending while for adding color you need to write the name of color that is specified and available in Python in the place of color in the above command [26].

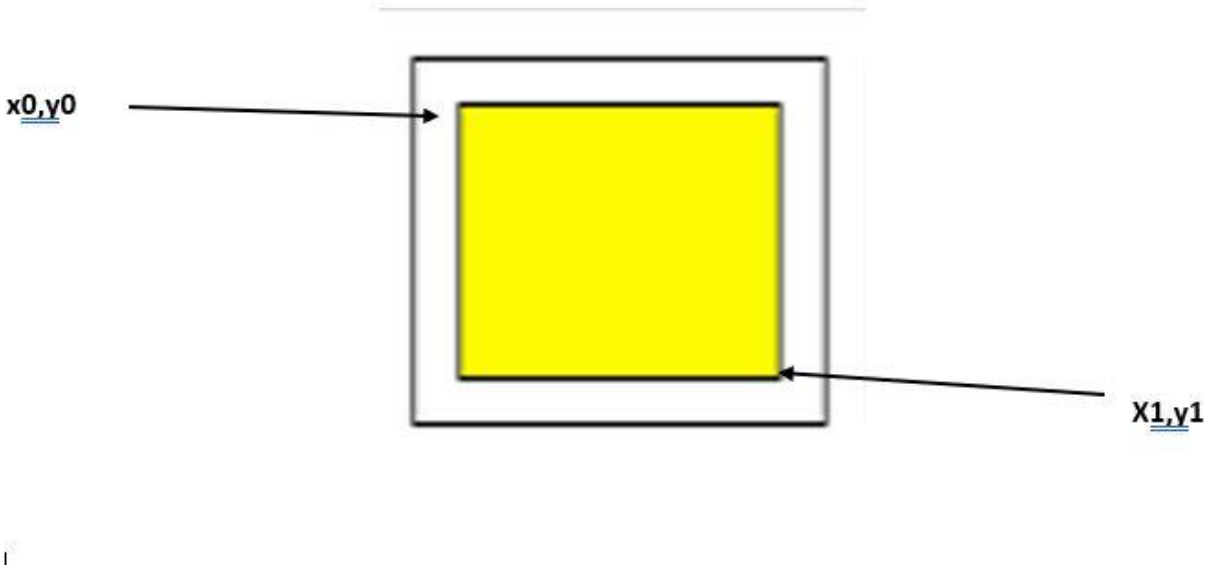


Figure 10-a: Creating Rectangle in GUI

3.3.2.4.2 Creating line:

Canvas.create_line(x0,y0,x1,y1,fill='color',arrow='start/last', width=num)

Where x0 and y0 are the upper left coordinates of the line from where it is starting and x1 and y1 are they lower right coordinates of the line at where it is ending while for adding color you need to write the name of color that is specified and available in Python in the place of color in the above command and you can place arrow head on start, last or on both sides of the line and you can assign any width just by writing any number in the place of num in above command [26].

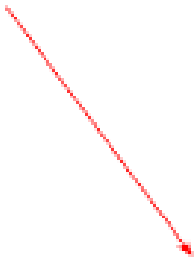


Figure 10-b: Creating line in GUI

3.3.2.4.3 Creating Circle:

Canvas.create_oval(x0,y0,x1,y1,fill="color")

Where x0 and y0 are the upper left coordinates of the circle from where it is starting and x1 and y1 are they lower right coordinates of the circle at where it is ending while for adding color you need to write the name of color that is specified and available in Python in the place of color in the above command [26].

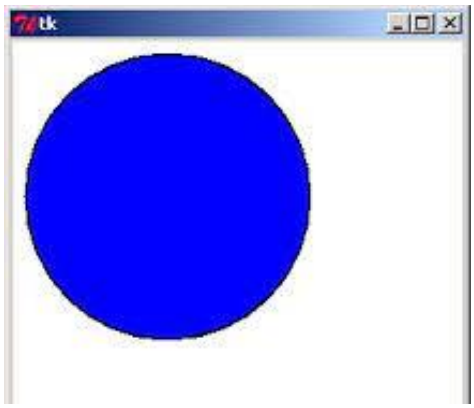


Figure 10-c: Creating circle in GUI

3.4 Decision making:

We designed our final year project such that it can be implemented on the chowks where there is a junction of four roads. In Pakistan normally each main road has three lanes which means that three cars can accommodate side by side in one row of the road so all of the cars in the same row need the same time to get out of that row. We designed our traffic signal in such a way that if there are 1 to 6 cars on a road then they will require 5 seconds to cross the road and three more seconds are additionally given to them for the safer end so in total they will get 8 seconds.

Similarly, if there are 7 to 12 cars on a road then they will require about 10 seconds to clear that road and similarly 3 more seconds will be given to them additionally to cross the road for the safer

side so in total they will get 13 seconds. And if there are 13 to 18 cars on the road then they will require 15 seconds to cross the choke and similarly, three more seconds are allotted to them for the safer end so in total they will get 18 seconds. The methodology that we have used in our final year project is like we are using pre-recorded data set on two of the roads and for the 3rd and 4th one, we are using live feed by mounting a camera on that road. The camera is programmed in such a way that it detects the vehicles on the road, counts them, and returns the value of the vehicles on the road and it has also the capability of differentiating an ambulance from normal vehicles by using the combination of image processing techniques and machine learning algorithms [23]

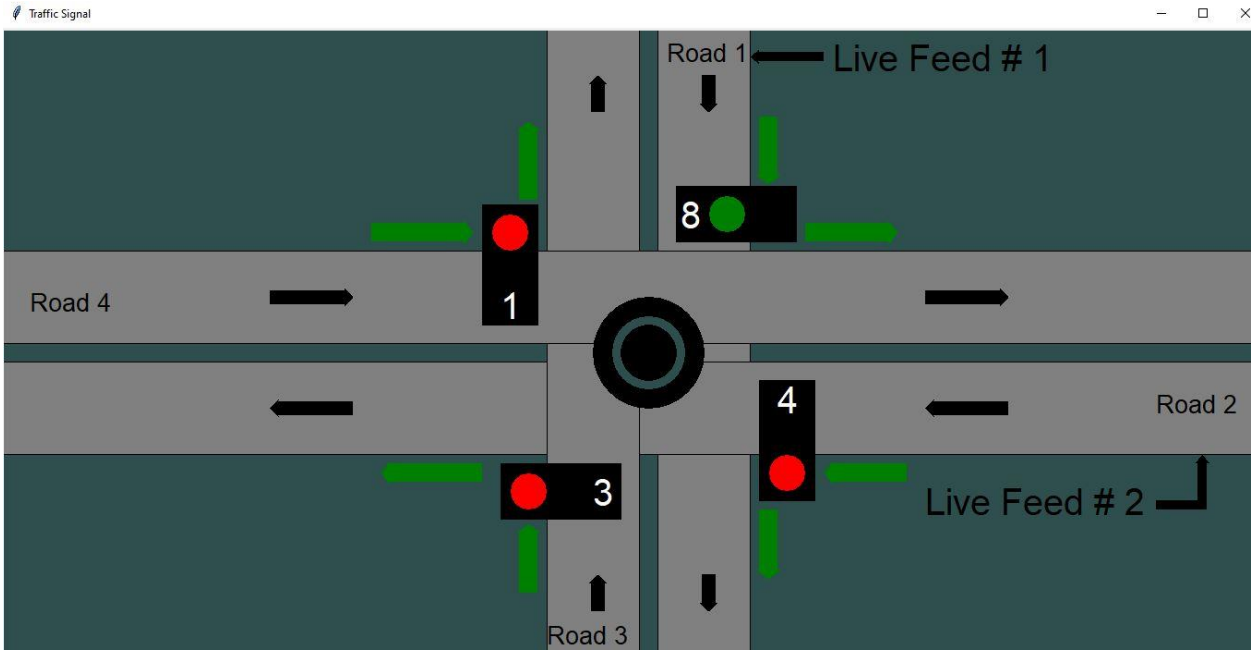


Figure 11: Decision making

It stores two pre-recorded data values and two live values in an array, then it sorts them in ascending order, and then it gives that sorted array to our program that operates our traffic signal according to the densities of the traffic on the roads.

For example, if roads 1 and 2 are connected to the live feed which means that the camera is mounted on road 1 and 2 which is giving the density of the traffic on road 1 and 2while for roads 3, and 4 it is taking density from pre-recorded data set.

First, that road will be open which has the maximum density then after opening the signal of that road, the density of that road will be replaced by zero which means that all the vehicles on that road have successfully crossed the choke and then the maximum of the three remaining roads will be open and the same fashion is carried out until all the roads get the value 0 which means that according to the starting time of our that loop all the cars have cleared the chowk.

And then the same loop is repeated, and the same procedure is carried out further [26].

In case an ambulance approaches the road, the signal of that road will be open regardless of the density of that road. By density, we mean the number of vehicles on that road.

The signal of that road will be open until the ambulance has crossed that road so that the ambulance doesn't get stuck in somewhere between due to traffic signal as it's not just about saving a life but it's about saving a family, saving the whole humanity.

3.4.1 Working of GUI:

As we know that in Pakistan, always open to left condition is implemented which means that traffic signals are not implemented on the vehicles that are intended to go left.

A traffic signal is only implemented for those who want to go straight or to take right or to make a U-turn. In the graphical user interface that we have created for our final year project, green arrows are facing towards the left from all the roads which are indicating that the vehicles that intend to take left turn can take a turn without bothering for the traffic signal on that road because that traffic signal is not applicable on them [23].

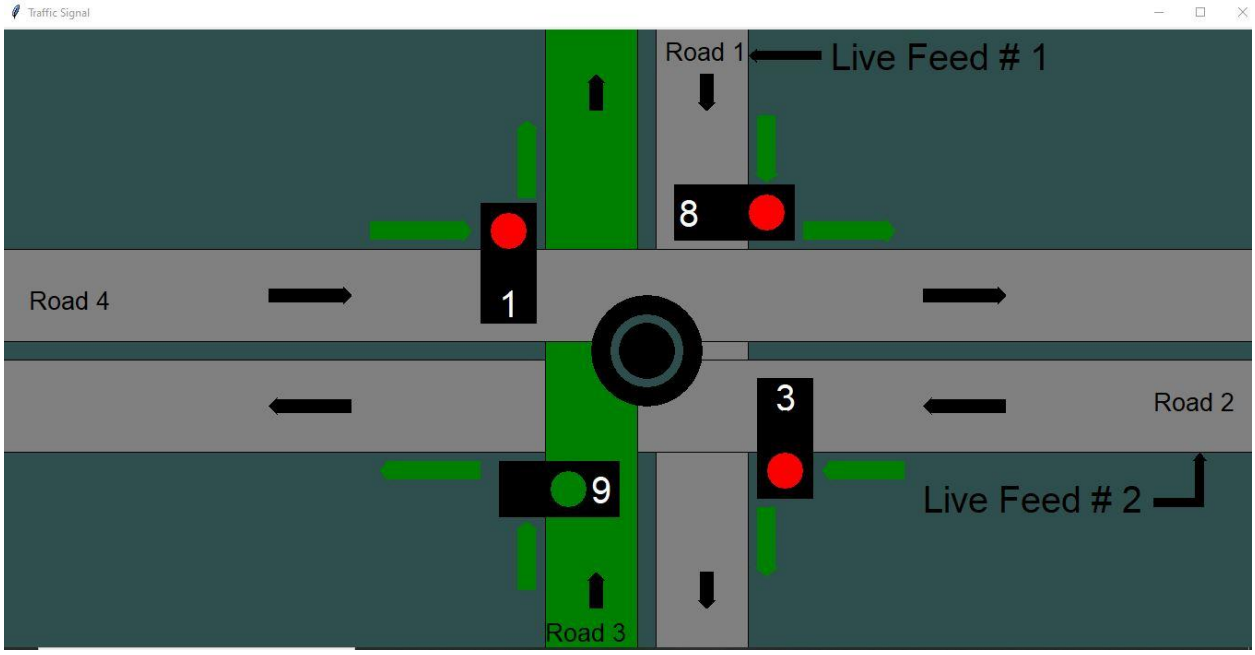


Figure 12: Working of GUI

For the opening of the traffic signal of the particular road, we have highlighted that road in green color which means that the traffic signal of that particular road is open either because of its maximum density or because of an ambulance on that road. So, the road which is green in GUI is indicating that the traffic signal of that particular road is open so the vehicles on that road can take left, can go straight, can go right after the roundabout, or can make a U-turn also.

Moreover, you can see that there are two lights on each road from which one of them is on at a particular instant either green or red. Red light indicates that the traffic signal of the road is close which means that vehicles on that road can only take left turn and they're not allowed to take U-turn or making a straight move while a green light indicates that the traffic signal of that particular road is open and they are allowed to take a left turn, right turn, straight move or U-turn

Besides the green light, there is a number written on each traffic signal which is representing the number of vehicles on that road at that instant [26].

3.5 Proposed Block Diagram

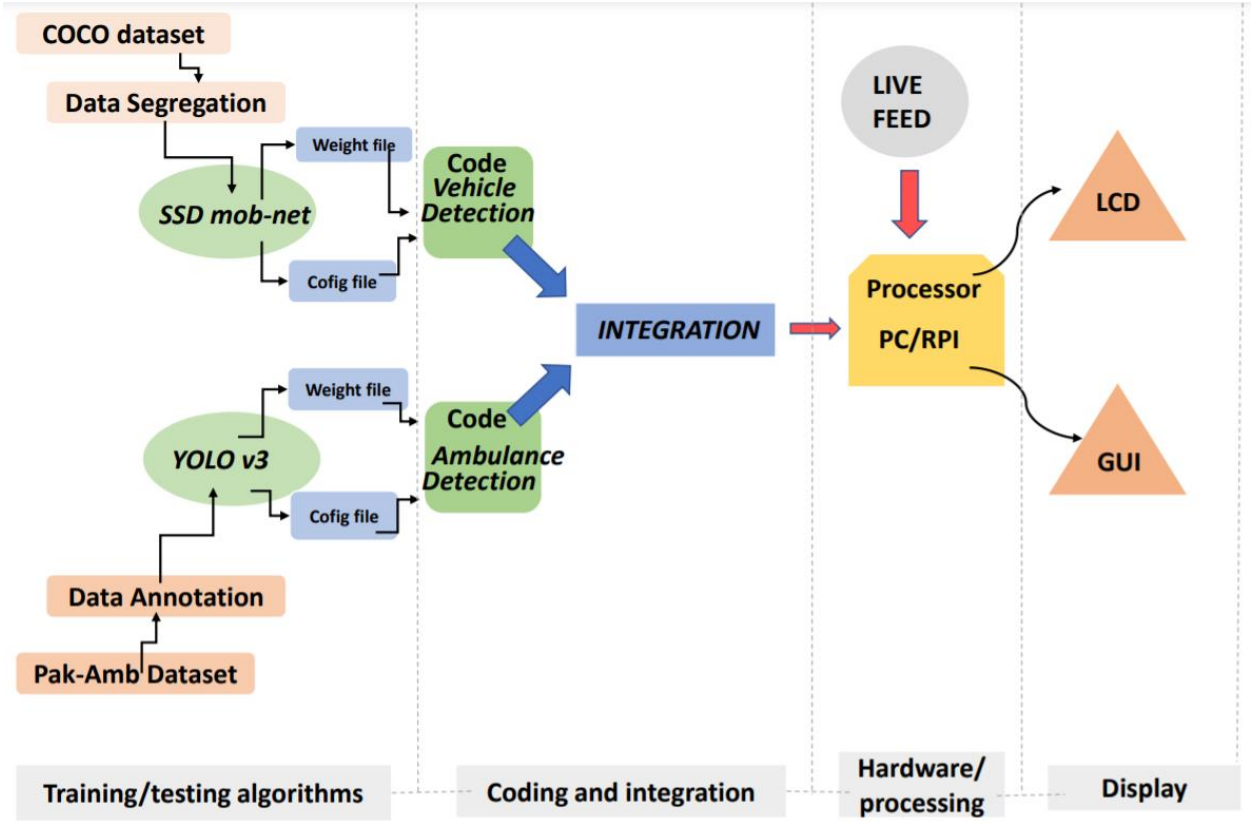


Figure 13: Proposed Block Diagram

Chapter 4: Code Analysis and Evaluation

Our code is divided into following main chunks:

4.1 Ambulance Detection

As discussed earlier in Section 3.1, YOLO algorithm is used to detect an ambulance.

Enlisted below are some parts to detect an ambulance.

4.1.1 Initialization Code:

Program fetches a video or a live feed and stores it. Weights and configuration files are then loaded to the YOLO algorithm, from which layers name are extracted for further processing.

Figure 14-a shows code for initialization.

```
import cv2
import math
import numpy as np


cap = cv2.VideoCapture('use_me.webm')
# Load Yolo
net = cv2.dnn.readNet("yolov3_training_lastt.weights", "yolov3_testing.cfg")
classes = ["ambulance"]
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))
```

Figure 14-a: Initialization code for Ambulance Detection

4.1.2 Display Code

For each ambulance detected, following code draws a rectangle / bounding box around it and puts a text according to its class ID. Then video/ live feed is displayed, text and rectangle included. (Section 3.3.1)

Figure 14-b shows a part of the code for display.

```
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        color = colors[class_ids[i]]
        cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
        cv2.putText(img, label, (x, y + 30), font, 3, color, 2)
cv2.imshow("Image", img)
key = cv2.waitKey(1)
```

Figure 14-b: Display code for Ambulance Detection

4.1.3 Output

Figure 14-c shows output of ambulance detection code.

Figure 14-c: Output for Ambulance Detection

4.2 Vehicle Detection

As discussed earlier in Section 3.2, SSD algorithm is used to detect vehicles. Enlisted below are some parts to detect vehicles.

4.2.1 Initialization Code

Program fetches dataset named ‘coco names’ and stores it. Weights and configuration files are then loaded to the SDD algorithm, then some important parameters are set. Get Objects is a function which does the detection part and returns a frame with density of vehicles in it.

Figure 15-a shows a part of the code for initialization.

```
#vehicle initialization
import cv2
classNames = []
classFile = 'coco.names'
with open(classFile, 'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')


configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weightsPath = 'frozen_inference_graph.pb'


net = cv2.dnn_DetectionModel(weightsPath, configPath)
net.setInputSize(320, 320)
net.setInputScale(1.0/ 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)

def getObjects(img, thres, nms, draw=True, objects=[]):...
```

Figure 15-a: Initialization code for Vehicle Detection

4.2.2 Display Code

This code fetches a video or live feed and stores it. Calls getobject function, as discussed above. As this function returns a density hence it is printed here per frame. For each vehicle detected, following code draws a rectangle / bounding box around it and puts a text according to its class ID. Then video/ live feed is displayed, text and rectangle included. Moreover, it also displays densities of each lane on the bottom left corner per frame.

Figure 15-b shows a part of the code for display.

```
if __name__ == "__main__":

    cap = cv2.VideoCapture('use_me.webm')

    cap.set(3, 648)
    cap.set(4, 488)
    while x < 50 and True:
        # while True:
        x = x + 1
        success, img = cap.read()
        result, objectInfo, d1 = getObjects(img, 0.45, 0.5, True, objects=['car'])
        print(d1)
```

```
        cv2.imshow("Output", img)
        cv2.waitKey(1)
        k = k + 1
    cv2.destroyAllWindows()
    cv2.waitKey(1)
    # sys.exit('ruk ja yaaar')

```

Figure 15-b: Display code for Vehicle Detection

4.2.3 Output

Figure 15-c shows output of ambulance detection code.

Figure 15-d displays density of vehicles per frame.

Figure 15-c: Output for Vehicle Detection

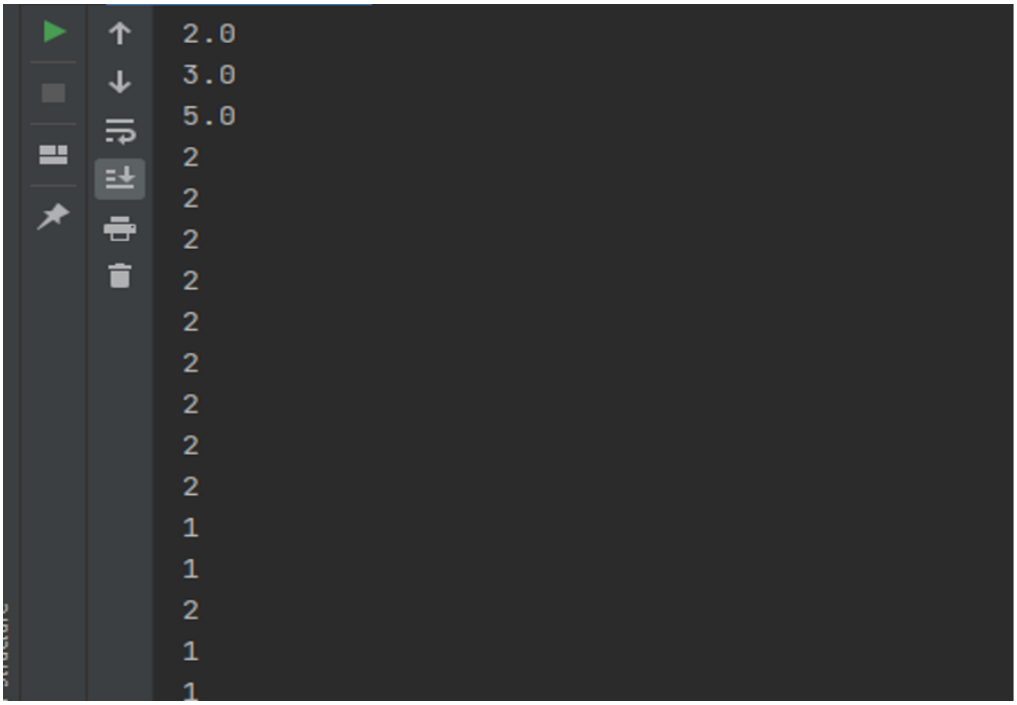


Figure 15-d: Density of vehicles per frame

4.3 Excel sheets

Excel sheets are used to store the densities at all instants. Moreover, prerecorded data is also fetched from excel sheets.

4.3.1 Creating Sheets for storing data

A new excel sheet is created which is used to store the densities of vehicles and the ambulance per frame. **Figure 16-a** shows the code for this program.

Figure 16-b shows the excel sheet having densities stored per frame.

```
#excel create and stores value continuously
import pandas as pd
my_dict1={...}
dfv = pd.DataFrame(data=my_dict1)
dfv.to_excel('new1.xlsx', index =False, sheet_name='vehicles')
dfv = pd.DataFrame(data=my_dict1)

import pandas as pd
# over writing protection
df1 = pd.read_excel('new1.xlsx')
SeriesA = df1['Vehicle']
```

Figure 16-a: Code for creating excel sheets

	A	B	C	D	E	F
1	vehicles					
2	1					
3	1					
4	0					
5	1					
6	1					
7	1					
8	1					
9	1					
10	1					
11	2					
12	2					
13	2					
14	3					
15	2					
16	1					
17	3					
18	2					

Figure 16-b: Densities per frame

4.3.2 Fetching pre-recorded data

As discussed in Section 3.4, two lanes are connected via live feed and other two lanes are connected to the excel sheet. The code and excel sheet are displayed in **Figure 17-a** and **Figure 17-b**.

```
#excel for pre recorded data
import xlrd
import time
from xlwt import Workbook
wb = Workbook()
sheet1 = wb.add_sheet('Sheet 2')
book = xlrd.open_workbook("fyp.xls")
sheet = book.sheet_by_index(0)
```

Figure 17-a: Code for fetching pre-recorded data

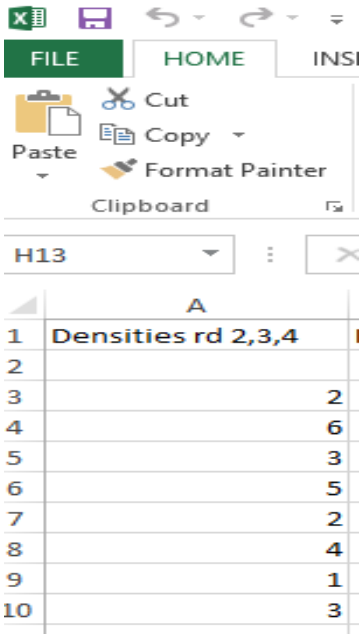


Figure 17-b: Excel Sheet

4.4 Decision Making

As discussed earlier in Section 3.4.1, decision has to be done on the basis of densities of all 4 lanes. Its code and output are as follows:

4.4.1 Code

It has two functions; time is for allocated time when density is between the ranges. Fyp is the function where decision has to be made with respect to density of each lane hence taking 4 inputs: density of each lane.

Referred to Section 3.4)

Figure 18-a shows the code for decision:

```
#time slots for densities
def timee(tt):...

#decision
def fypp(d1,d2,d3,d4) :...
```

Figure 18-a: Decision making code

4.4.2 Output

Figure 18-b shows the output of decision which can be also connected to the traffic signal.

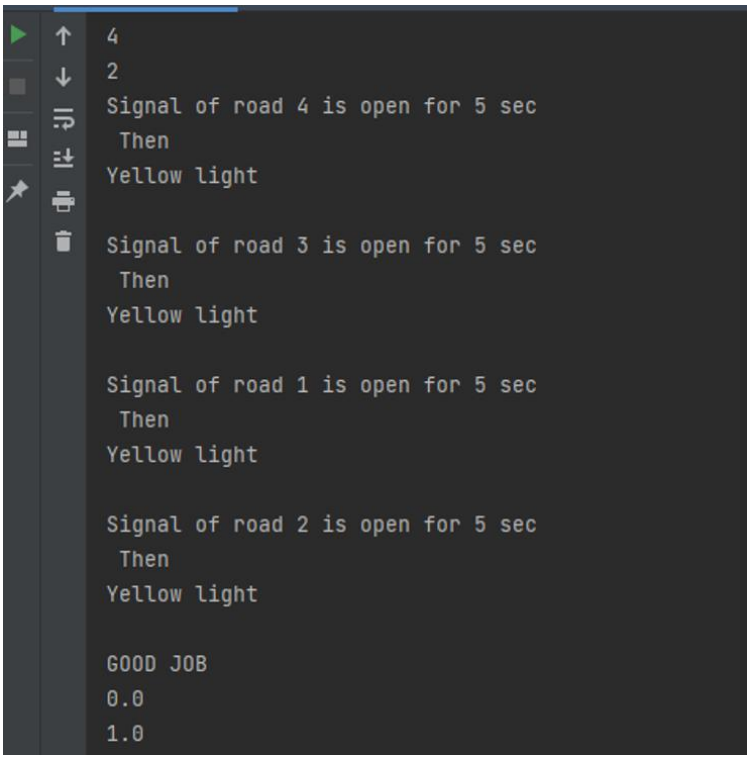


Figure 18-b: Output of Decision-making code

4.5 GUI

4.5.1 Code

This is the code for GUI which displays the output for controlling traffic signals.

```
def start():
    controlLogic()
    window.after(1000, start)
window = Tk()
window.title("Traffic Signal")
frame = Frame(window)
frame.pack()
color = StringVar()
canvas = Canvas(window, width=1400, height=700, bg="dark slate grey")
canvas.pack()
roads=[...]
direction = [...]
always_left_direction = [...]
rectangles =[...]
names=[...]
density=[...]
chowk = canvas.create_oval(640, 290, 760, 410, fill="black")
chowk_1 = canvas.create_oval(660, 310, 740, 390, fill="dark slate grey")
chowk_3 = canvas.create_oval(670, 320, 730, 380, fill="black")
reds = [...]
greens = [...]
start()
window.mainloop()
```

Figure 19-a: GUI Code

4.5.2 Output

Figure 19-b shows the output.

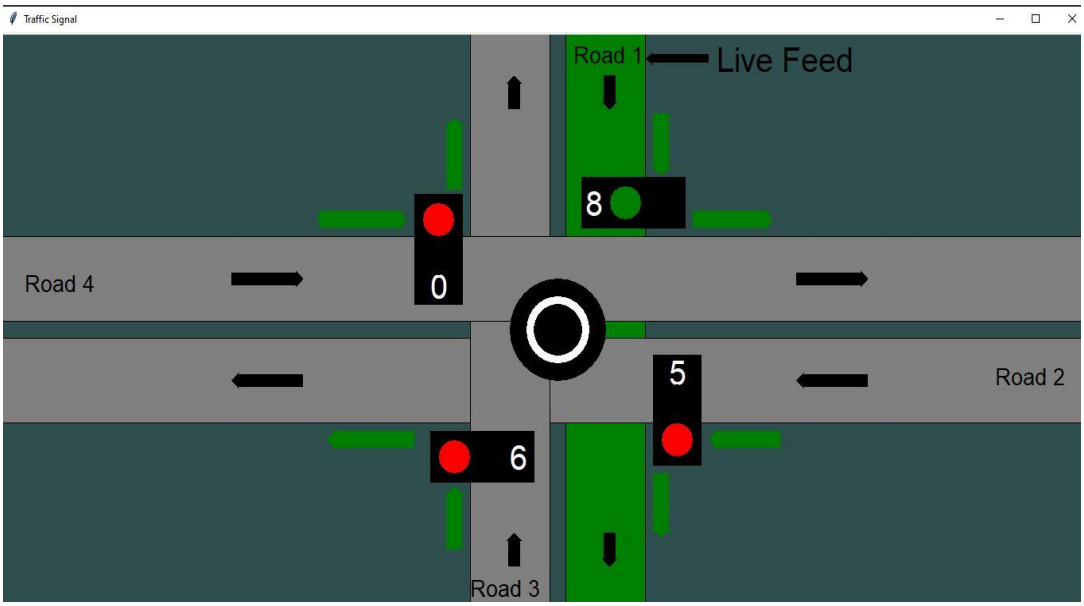


Figure 19-b: GUI Output

Chapter 5: Conclusion

In this thesis, we discussed a traffic control system that can handle traffic problems smartly and more efficiently than the typically used sensor-based or fixed time slot systems by using real-time processing. Our proposed system has an advantage over other traditional systems due to the latest algorithms used for the detection of objects of interest. Techniques used in our proposed system; Image Processing techniques to process the video, frame by frame, and Machine Learning included algorithms such as YOLO and SSD, which were used to detect the objects of interest; are also briefly explained included their working and importance. The purpose of increasing productivity and overcoming problems in existing solutions is being achieved by using the modern techniques. Additionally, the objectives; smooth traffic and providing special privileges, are attained. Hence saves time as well as the prestigious lives of patients.

Simulated results are shown for two live feeds using OpenCV, which is open-source software, easy to install, and can be used in real-time. The same algorithm can be used to have live feeds for all four lanes. Only the interfacing of the cameras would be required.

Our proposed system, STCS, is cost-effective as it is purely made for the core purpose of serving Pakistan, any system that could be beneficial to its citizens. Else, similar solutions provided by other countries are very costly. Moreover, STCS provides an ease to adoption which can be adopted by beneficiaries, mass deployment can also be done and most importantly no product training is required.

Chapter 6: Future Work

Future milestones that need to be achieved to commercialize this project are the following.

6.1 Access to real life traffic signal:

The main objective of this project is to produce a product that is portable, easily attached with a conventional traffic signal. Physical access to a traffic signal is vital. During the preparation of this product, we tried to access many civil-military authorities for the allocation of a traffic signal, but due to covid restrictions and security reasons we were unable to gain physical access of a traffic signal. We overcame this problem by testing our project on GUI (Graphical User Interface) that portrays a random traffic routine around a traffic signal. For real time processing, in future we are looking forward to acquire physical access of a traffic signal.

6.2 Camera type:

The camera that we are using for the project prototype is a USB camera, also known as rolling shutter camera. This camera works fine as for the prototype model, but it is prone to cause blurriness while capturing a high-speed object. For practical implementation of our project, we have to amalgamate a global shutter camera, which can capture high speed objects without causing blurriness in the imagery.

6.3 Vast Imagery of traffic from a certain angle of inclination:

The angle at which the camera captures images is very important for efficient performance of STCS. This is based on a ML algorithm, which learns to detect objects with respect to input images. If the camera is to be placed at a certain angle on the traffic signals, then for

detection purpose, the ML algorithm must be trained to detect objects from the input images at specific angle. So, we need to collect dataset captured at that specific angle of inclination.

6.4 Finding Investors:

We are going to convert this project into a market compatible product. The first step is thorough research and development to find innovation ways to reduce cost of the product and increase overall durability and efficiency. Secondly, to broadcast the product into open market we need investors and stake holders. We are planning to find such investors for future market ventures of our product. Special thing about this project is that it solves a civic problem: traffic congestion. This problem directly affiliates governmental institution for Research and Development and can be contacted for installation of this project.

References and Work Cited

1. Highways.dot.gov. 2021. Federal Highway Administration. [online] Available at: <https://highways.dot.gov/>

2. Cocodataset.org. 2021. *COCO - Common Objects in Context*. [online] Available at: <https://cocodataset.org/>

3. Engineering Education (EngEd) Program | Section. 2021. *Introduction to YOLO Algorithm for Object Detection*. [online] Available at: <https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/>

4. The MagPi magazine. 2021. *Raspberry Pi 4 specs and benchmarks — The MagPi magazine*. [online] Available at: <https://magpi.raspberrypi.org/articles/raspberry-pi-4-specs-benchmarks>

5. Walad, K.P. and Shetty, J., (2014). Traffic light control System using image processing. *International Journal of Innovative Research in Computer and Communication Engineering*, vol.*2*, no.5, p.289.

6. Gaikwad, O.R., Vishwasrao, A., Pujari, P.K. and Talathi, T., (2014). Image Processing Based Traffic Light Control. International Journal of Science, Engineering and Technology Research (IJSETR), vol.3, no.4, p.1010-1014.

7. Abbas, N., Tayyab, M. and Qadri, M.T., (2013). Real time traffic density count using image processing. *International Journal of Computer Applications*, vol.*83*, no.9, p.16-19.

8. Vidhya, K. and Banu, A.B., (2014). Density based traffic signal system. *International Journal of Innovative Research in Science, Engineering and Technology*, vol.*3*, no.3, p.2218-2222.

9. Parthasarathi, V., Surya, M., Akshay, B., Siva, K.M. and Vasudevan, S.K., (2015). Smart control of traffic signal system using image processing. Indian journal of Science and Technology, vol.8, no.16, p.1.

10. Ravish, R., Shenoy, D.P. and Rangaswamy, S., Sensor-Based Traffic Control System. In *Proceedings of the Global AI Congress (2019)* (p. 207-221). Springer, Singapore.

11. Chattaraj, A., Bansal, S. and Chandra, A., (2009). An intelligent traffic control system using RFID. *IEEE potentials*, vol.*28*, no.3, p.40-43.

12. Mohammadi, S., Rajabi, A. and Tavassoli, M., (2012). Controlling of traffic lights using RFID technology and neural network. In *Advanced Materials Research*, vol. 433, p. 740-745. Trans Tech Publications Ltd.

13. Rajesh, G., Raajini, X.M. and Dang, H. eds., (2021). Industry 4.0 Interoperability, Analytics, Security, and Case Studies. CRC Press, p. 96-262.

14. Masurekar, O., Jadhav, O., Kulkarni, P., & Patil, S. (2020). Real Time Object Detection Using YOLOv3. International Research Journal of Engineering and Technology (IRJET), p. 3764-3768.

15. Lu, J., Ma, C., Li, L., Xing, X., Zhang, Y., Wang, Z. and Xu, J., (2018). A vehicle detection method for aerial image based on YOLO. *Journal of Computer and Communications*, vol.*6*, no.11, p.98-107.

16. Shindel, P., Yadav, S., Rudrake, S. and Kumbhar, P., (2019). Smart traffic control system using YOLO. *Int. Res. J. Eng. Technol.(IRJET)*, vol.*6*, p.966-970.

17. Biswas, D., Su, H., Wang, C., Stevanovic, A. and Wang, W., (2019). An automatic traffic density estimation using Single Shot Detection (SSD) and MobileNet-SSD. *Physics and Chemistry of the Earth, Parts A/B/C*, vol.*110*, p.176-184.

18. Lu, S., Wang, B., Wang, H., Chen, L., Linjian, M. and Zhang, X., 2021. *A real-time object detection algorithm for video*. [online]. Available at: < https://www.mdpi.com/2079-9292/10/1/14/htm>

19. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications, p.1.

20. Industrial, D., 2021. *Understanding SSD Architecture| Delkin Devices*. [online] Delkin Industrial. Available at: <https://www.delkin.com/blog/understanding-ssd-architecture/>

21. Oakridge. 2021. *Importance of giving way to ambulance | Best Schools in Chandigarh - Oakridge.in*. [online] Available at: <https://www.oakridge.in/blog/mohali/school-students-stress-the-importance-of-giving-way-to-ambulance-when-on-roads/>

22. Canu, S., 2021. *Train YOLO to detect a custom object (online with free GPU) - Pysource*. [online] Pysource. Available at: <https://pysource.com/2020/04/02/train-yolo-to-detect-a-custom-object-online-with-free-gpu/>

23. Beniz, D. and Espindola, A., (2017), September. Using Tkinter of python to create graphical user interface (GUI) for scripts in LNLS. In *Proc., 11st Int. Workshop on Personal*

Computers and Particle Accelerator Controls, PCaPAC2016. Geneva: JACoW. https://doi.org/10.18429/JACoW-PCaPAC2016-WEPOPRPO25, p 1-6.

24. Kivy.org. 2021. *Kivy: Cross-platform Python Framework for NUI*. [online] Available at: <https://kivy.org/>

25. Realpython.com. 2021. *PySimpleGUI: The Simple Way to Create a GUI With Python – Real Python*. [online] Available at: <https://realpython.com/pysimplegui-python/>

26. Lin, J. and Zhou, A., (2018). PyDraw: a GUI drawing generator based on Tkinter and its design concept. arXiv preprint arXiv:1808.09094, p 1-15.