

Visvesvaraya Technological University, Belagavi.



PROJECT REPORT
on
**“Traffic Management System Using Computer Vision And Neural
Networks”**

Project Report submitted in partial fulfillment of the requirement for the award of
the degree of
Bachelor of Engineering
in
Electronics and Communication Engineering
For the academic year 2019-20

Submitted by

USN	Name
1CR16EC150	Sanjay Kumar VH
1CR16EC190	Vikas V
1CR17EC411	Karthik S
1CR17EC427	Ramesh D

Under the guidance of

Internal
Prof. Richa Tengshe
Asst. professor
Department of ECE
CMRIT, Bengaluru



Department of Electronics and Communication Engineering
CMR Institute of Technology, Bengaluru – 560 037

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING



CERTIFICATE

This is to Certify that the dissertation work “**Traffic Management System Using Computer Vision And Neural Networks**” carried out by Sanjay Kumar VH, Vikas V, Karthik S, Ramesh D, USN: 1CR16EC150, 1CR16EC190, 1CR17EC411, 1CR17EC427 bonafide students of **CMRIT** in partial fulfillment for the award of **Bachelor of Engineering** in **Electronics and Communication Engineering** of the **Visvesvaraya Technological University, Belagavi**, during the academic year **2019-20**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

Signature of Guide

Signature of HOD

Signature of Principal

Prof. Richa Tengshe,
Asst. Professor,
Dept. of ECE.,
CMRIT, Bengaluru.

Dr. R. Elumalai
Head of the Department,
Dept. of ECE.,
CMRIT, Bengaluru.

Dr. Sanjay Jain
Principal,
CMRIT,
Bengaluru.

External Viva

Name of Examiners

- 1.
- 2

Signature & date

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose consistent guidance and encouragement crowned our efforts with success.

We consider it as our privilege to express the gratitude to all those who guided in the completion of the project.

We express my gratitude to Principal, **Dr. Sanjay Jain**, for having provided me the golden opportunity to undertake this project work in their esteemed organization.

We sincerely thank **Dr. R. Elumalai**, HOD, Department of Electronics and Communication Engineering, CMR Institute of Technology for the immense support given to me.

We express my gratitude to our project guide **Prof. Richa Tengshe**, Asst. Professor, for their support, guidance and suggestions throughout the project work.

Last but not the least, heartfelt thanks to our parents and friends for their support.

Above all, We thank the Lord Almighty for His grace on us to succeed in this endeavor.

ABSTRACT

The importance of traffic signals is increasing owing to the drastic increase in population. The rising number of vehicles and truck plying on road presses the fact that ensuring road safety is of high priority. In this project, we introduce an Intelligent Traffic Management System (ITMS) capable of managing traffic of varying densities, without the need of a traffic warden to physically monitor a particular intersection. As this system is completely automated, there is no requirement of manpower to physically monitor and manage the systems on the road.

This system is designed to retrieve the live traffic feed from a junction and process the same using the TensorFlow Object Detection API over OpenCV to detect the severity of the traffic based on the number of vehicles detected. Upon determining the number of vehicles, the corresponding signal, based on the traffic intensity is given. Additionally, the recorded values which are recorded by the system are recoded in a .csv file, which can help us understand the traffic density at intersections with more clarity, and remove the object-detection component which requires a lot of compute power. The system works on the principle that moere vehicles detected – Green light for longer duration and vice versa.

Thus, this system dynamically adapts to the prevailing traffic conditions and grants the corresponding traffic light sequence for the required duration to maximize the flow of vehicular traffic. The system is designed to ensure smooth traffic flow by decreasing the wait period of vehicles at intersections and automates the process of controlling traffic signal.

Table of Contents

CHAPTER 1	1
INTRODUCTION	ERROR! BOOKMARK NOT DEFINED.
1.1 Problem Statement	2
1.2 Objective	2
1.3 What is CNN	3
1.3.1 Object Detection using CNN	3
1.3.2 Various Components of CNN	4
CHAPTER 2	6
LITERATURE SURVEY	6
2.1 Related Work done in the Field	7
CHAPTER 3	9
METHODOLOGY	9
3.1 System Design	12
CHAPTER 4	13
SYSTEM ANALYSIS	13
4.1 System Requirement Specification	13
CHAPTER 5	14
SYSTEM DESIGN	14
5.1 Use Case Diagram	14
5.2 Sequence Diagram	14
5.3 Data Flow Diagram	15
5.4 Activity Diagram	1Error! Bookmark not defined.
CHAPTER 6	18
SYSTEM IMPLEMENTATION AND CODING	18
CHAPTER 7	2ERROR! BOOKMARK NOT DEFINED.
RESULT AND DISCUSSION	2ERROR! BOOKMARK NOT DEFINED.

CHAPTER 8	31
CHELLENGES	31
CHAPTER 9	32
CONCLUSION	32
CHAPTER 10	33
FUTURE SCOPE	33
REFERENCES	34

List of Figures

Figure 3.1 The sequence of processing the video feed.	1Error! Bookmark not defined.
Figure 3.2 The training loss graph	1Error! Bookmark not defined.
Figure 5.1 The activity diagram of the designed system	14
Figure 5.2 The sequence diagram of the system	15
Figure 5.3.1 Dataflow diagram of the Programmer	16
Figure 5.3.2 Dataflow diagram of the users of system	16
Figure 5.4 Activity Diagram of the System	17
Figure 7.1 Traffic signal cycle in a low traffic condition.	28
Figure 7.2 Traffic signal cycle in a moderate condition.	28
Figure 7.3 Traffic signal cycle in a severe condition.	29
Figure 7.4 Values stored in the .csv file	30

CHAPTER 1

INTRODUCTION

Traffic in urban cities and towns have been a serious problem which deserves high priority and deserves to be resolved at the earliest. Traditional traffic management was done using traffic lights or hand- signals given by the traffic warden, have been mostly ineffective or mismanaged when the warden is away from his kiosk. This problem can be addressed effectively and efficiently by a Smart Traffic Management System, which employs the usage of neural networks and video processing to determine the traffic at any given point of time at a given intersection to determine the traffic density and grant the required signal to ensure that the traffic is cleared, efficiently.

In most of the road transport constructs, live feed video cameras are deployed for video monitoring of the prevailing traffic. Through monitoring, traffic congestional variations are tracked and improvisational measures are taken accordingly. This video monitoring is also useful for surveillance purposes to track accidents. CNN (Convolutional Neural Networks) is a tool which is designed for identification and classification of road traffic which are efficient tools for object-classification. Video traffic monitoring is convenient as the traffic data which is obtained from the feed will be classified by CNN. CNN does the image detection, recognition and classification. CNN is made of ‘neurons’ which have can be trained to have learnable weights and biases through training. The vehicles are extracted from the live video feed background and the object detection is done to identify vehicles in a scene. Detected vehicles are identified real-time.

The system employs the usage of a mini-computer which is capable of communicating through a LAN and a computer with high-performance to process the video and provide the signals to the former to display as the signal for the prevailing traffic situation. The project employs two basic elements for processing the video-feed, namely OpenCV and TensorFlow, which are responsible for processing the live video stream from a traffic intersection and determining the traffic density based on which, a signal-cycle is granted.

Thus, the designed project can solve the ever growing problem of mismanagement of traffic at intersection.

1.1 Problem Statement

Our nation, India is known to have the second largest road networking on the face of earth. It is a herculean task for the nations leaders and government to have good roads with extremely reliable infrastructure such as roads and highways and also our country has spent a lot of money on maintenance of roads which will cause huge mismanagement in the utilization of public funds. It should be understood by us as citizens of this nation that we are the reason for the traffic problem and destroy the traffic rules and blame the government and traffic police for any incidence that happens, and that requires the need to constantly monitor the traffic in our city in our cities of every junction. The problem with the people are seems to not care about the lanes present in the traffic. If the problem is not addressed by the future engineers like us , this problem will become a extremely bigger problem costing huge sum of money and effort to solve.

In our project the Object detection API model takes the video data and the layers within it process it to determine the number of cars in the frame of video. Through the identification and detection of the video, we can solve the ever growing problem of traffic .By this the traffic problem can be reduced and it will improve the performance of the transportation.

1.2 Objectives

The Main Objective of this project is to propose the object detection module (CNN)-based method that is trained over images of cars to help identify and detect them, from which the count can be determined. Using the number of vehicles detected, we can now determine the density of traffic on the roads. A CNN is applied to the image to determine the traffic in an intersection by determining the number of cars in the frame of video. The effectiveness of the proposed method is evaluated by using a scaled down model of cars.

1.3 What is CNN?

Convolutional Neural Network is a tool which is used to analyse and classify images and it is a division of the of deep neural networks. The editions of various multilayer perceptron are categorized under the framework of Convolutional Neural Network. The fully connected networks is simply a collection of neurons which is generally interconnected to one another from one layer to another making it multilayer by design.

Regularization which is a typical way of adding some deterministic additions to the loss function using the concept of weights. Convolutional Neural Network will have strategy of approach towards regularization .The multi-layered pattern of neurons is a advantage of regularization which uses to assemble more complex patterns of simple patters and also smallest patters . Hence, the Convolutional neural networks are lower extreme and valued in the scale of complexity and correctness.

Convolutional neural networks are inspired from the pattern of biological systems where neurons interconnect to process information in cortex of the brain. The Convolutional Neural Network use training images to develop a framework where the network trains itself to learn and identify images. By this technology, there is no need to help the machine perform its tasks without prior knowledge or information.

Convolutional Neural Network has many applications in various domains such as natural language processing, recommendation systems, medical data and image analysis, etc.

1.3.1 Object Detection using CNN

The concept of convolutional neural networks is widely spoken in the field of image processing. The convolution neural network consists of mainly two layers which does the image processing which are the convolution layer and pooling layer. There are other layer also called input layer, output layer and max pooling layer. The CNN takes the video feed to process into image video frames and renders it into matrix which contains of values of each pixels in the obtained video frame. The convolution layer is trained to understand the filters by training itself over the training dataset and applies it through the

frame which appears in the form of testing data and it also extracts the landmarks which is necessary for object detection. The pooling layer based on the filter applied by the convolution layer to reduce the sizes of the images. The pooling layer employs function ReLU , Sigmoid etc. the final output layer is the SoftMax which gives the co-ordinates where the objects have been identified in the processed video feed.

1.3.2 Various components of CNN

Input layer

The input layer is designed to take the frames from the video and represent each individual frame in the form of an RGB matrix. This matrix is the collection on pixel values which will be worked upon

Convolutional layer

Convolutional layer is the layer which computes using the neurons which are connected in the output to the subsequent layer. The dot product between a small area and weights are calculated and they are passed as the input values. In such cases the result will have twelve filters volume. In this way Convolutional layer will transform the image one after another layer to the final score using original pixel value. The convolutional layer develops a mapping where the initial volume on the left using raw image pixel and are mapped one the right it stores class scores. Convolutional layer consists of set of filters in which every filter is spatially small in width and height and also the full depth. These filters learn by itself for the image. Each filter across the height and width of inputs and products between these two height and width are the entries of each input at any position and the filter.

Pooling layer

This layer is placed between the successive convolutional layers of the model. The main functionality of this layer is to reduce size of amount of parameters and computationally in the neural network. Using the mathematical max operation, this layer will independently change depth size and resizes the input obtained from previous later. The left side of pooling layer will perform the task of reducing the size of input and output volume and the right side of pooling layer will have down sampling operation that

will be the output the max pooling. Thus the size of the sample is reduced after being processed by the pooling layer.

Output layer

This is the SoftMax function which will have probability of all the object classes. It also calculates probability of positive label from a random sample of negative labels. In our project this layer gives the density of vehicles on the intersection of the road. It deals with object detection and classification.

CHAPTER 2

LITERATURE SURVEY

As the complexity of the traffic situations increase across the world, researches are trying to find a comprehensive solution to address the ever growing problem of traffic mismanagement and inefficiency. There are many works done in this field to mitigate the ever growing problem of unmanaged traffic, which is leading to a loss of time and gains. The different approaches used to identify and manage the traffic are by using various technology enabled devices such as RFID (Radio frequency based ID), NFC (Near field communication), Pressure sensors and various other devices which can be activated to determine the traffic densities. However, these approaches have their own pitfalls and advantages. The first advantage being that the traffic can be measured accurately and efficiently using the RFID tags, NFC Communication and Pressure sensors. The problem arises at the juncture where the devices are either too costly or the system being too complex to implement. The first thing being that, in case of RFID systems, the RFID sensors need to be strategically located and all the vehicles plying on the road need to be provided with RFID stickers. In a country where there is a huge population, it is nearly impossible to RFID tag the vehicles. Secondly, the system needs to employ a powerful RFID antenna to pick up the signals. This is an additional requirement to the infrastructure. Thus, the technique of object detection is understood to be the best to identify and detect cars in a video frame. This technique can be done through various ways where, we can employ simple computer vision techniques or deep learning to perform the task of object detection. The use of computer vision techniques for object detection has its own drawbacks such as low accuracy and overheads, besides being a legacy technique. The technique which employs deep learning is far more accurate and efficient when compared to other methods. Hence, we have taken the Machine learning approach to solve the problem in hand.

2.1 Related work done in the field

We have studied works where a group of Multilayer perceptron classifiers have been employed to perform the task of object detection with error-correcting outputs is applied, achieving an average overall accuracy of 93.8%. Additionally, in another work an Multilayer Perceptrons classifiers with swarm optimization algorithm is used to classify 6 labels with a peak accuracy of 96.9%. Another system employed the usage of a system called Loop Detection technology where a inductor is employed for the process of identifying and detecting the traffic on the roads, It works on the principle that one or more number of turns of insulated wire roadside to the controller and the controller consists of the electronic unit. When a vehicle passes over the loop or stops, the induction of the wire is moved and changed. Due to change in induction, the frequency varies. These results in the electronic unit to send a signal to the controller; indicating presence of vehicles.

The system employed makes use of a video camera which consists of sensors, one processing unit and communication unit. The traffic is monitored continuously using a CCTV camera. The captured video is then compressed so as to reduce the size taken to transfer. The video analysis extracts scene description from the video feed from the camera. This description is later used to compute traffic data. Another work is an entropy-based minimum description length of features as a pre-processing step to various algorithms such as C4.5, Naïve Bayes, SVM and kNN. Declaring a quality performance of the algorithms, achieving an accuracy of 93.2%- 98%. These are few of the work that are related to our project. For studying the efficiency of vehicle motion detection by RFID, there is a work that inquire about RFID read latency and thus the productiveness of vehicles reader insertion for a vast range of speeds.

The paper [1] proposed by Vikramaditya Dangi, Amol Parab titled Image Processing Based Intelligent Traffic Controller employs the usage of computer vision technologies to identify and detect the traffic using background subtraction and edge detection. In this method, the opensource computer vision library was used to identify cars in the given frame of video, the system was able to identify cars and determine the traffic density. Additionally, the system was empowered to detect essential services

vehicles such as ambulances, fire engines, cavalcades, etc. to help identify vehicles which deserve more priority than other vehicles

The paper titled “Traffic Light Recognition using Image Processing Compared to Learning Processes [2]” discusses about the various techniques which can be used to identify the traffic signals from a distance using Gaussian filters to extract colours from the image obtained. This helps us in recognizing the traffic lights and making further enhancements in providing a real time feedback to the system from the camera module.

Authors, M. Yang, R. Jhang and J. Hou, have published a paper[5] titled “Traffic flow estimation and vehicle-type classification using vision-based spatial-temporal profile analysis to identify and detect the density of traffic plying on the roads to perform an analysis on how can the traffic situation be managed and controlled. Techniques such as foreground and background detection have been used to identify and detect the traffic condition on the road. The density of the traffic plying on the road is determined using the number of objects detected or identified in the frame.

The system which we have developed have used various components from all the above mentioned papers to extract the best of features and ensuring the developed system is empowered to identify detect and manage the traffic at the intersections.

CHAPTER 3

METHODOLOGY

The objective of the algorithm is to use Convolution Neural Networks using TensorFlow is to identify and detect the vehicles in the frame and then, based on the vehicular density, we determine the traffic on the road and grant the traffic signals to clear maximum traffic. Based on these values traffic can be easily managed by changing traffic lights accordingly in a way such that the lanes with higher traffic gives more amount green light and the lanes with the less traffic are gives less amount time of green light. In this way, maximum traffic can be cleared by the system effectively and efficiently.

The data provided to the Neural Network is from the Closed Circuit Television Camera also called as CCTV camera or from a 360 degree camera which can obtain the video feed where camera can tilt toward the visibility of lanes to capture the video of the ongoing traffic. The video is nothing but the sequence of frames. The video feed from the high hosted camera is segmented into individual frames for Convolution Neural Networks to be processed to calculate the traffic of the lanes. To train the neural network model, we need to train the model first, before it is tested, here we have used 400 images of cars in total where 360 are used for training and 40 are used for testing the model. The size of the array is same as the resolution of the image (height x width) times the depth of the image.

The algorithm followed by neural network is $y=w*a+b$, where y is the labels, w is the weights and b is the biases. For each stage, the value of w and b is varied. The result generated produces the output which are in the form of detected objects which have been processed by the model, these detected objects are then observed and counted to get the idea of the prevailing traffic situation

The video feed is processed using OpenCV and the resolution is 640x480x3. The classes for the model are not encoded and require it to be converted back to string when running the model. The output will be the number of cars detected in the frame.

CNN consists of major two alternative layers which is convolution and pooling layers. These both reduces the resolution of images for processing of CNN. The layers of the convolutional neural networks does the image processing. The convolution neural network consists of mainly two layers which does the image processing. These layers are namely the convolution layer and pooling layer. There are other layer also called input layer, output layer and max pooling layer. The input layer receives the obtained image video frames from the video source and converts it into matrix format which is known to consists of those values of each individual pixels of the frame. The computational framework which consists of the convolution layer understands these filters by training applies it through the frame and it also extracts the features which is necessary for object detection (the car in this case) . The pooling layer based on the filter applied by the convolution layer it reduce the sizes of the images. The pooling layer uses certain functions such as ReLU functions, Sigmoid function etc. the final output layer is the SoftMax processed the video feed and identifies the number of cars in the frame and then generate the output which the

In each stage feature extraction is done and it produces the large feature set for original input. These feature play an important role in detecting the objects of the vide frame. These characteristics of data is nothing but the characteristics of the vehicle such as its shape which is recognized in the form of blocks and considering it will check the rating of the traffic of the lanes. The processed frames are then

ReLU (Rectified Linear Unit) is activation function we are applying and it is popular activation function in deep neural networks, shown above, where x is the input to the neurons. The SoftMax function transforms each element of a collection by computing the exponential of each element divided by the sum of the exponentials of all the elements.

SoftMax function

SoftMax function is also called as Normalized exponential function. This function takes vector N real numbers as input and normalizes it to probability distribution consisting of N probabilities. In neural networks it will map non-normalized output to probability

distribution over predicted output classes. It is used as last layer in the neural network to determine the output of the classifier.

All CNNs use 128×128 pixel images, converted to gray scale and normalized, as inputs. Solutions use square convolution masks of different sizes. In the case of the best performing CNN the first convolutional layer consists of 16 filters of the size 9×9 which give 16 feature maps by shifting the center of filters by 4 pixels (stride=4) in each direction and padding 0. There are $30 \times 30 \times 16$ neurons in this layer. Max pooling is applied to the outputs, maximum is calculated in 3×3 windows and stride 2. Max pooling divides the inputs into a set of non-overlapping squares and, for each such square, calculates the maximum value.

The second convolution layer takes as input the pooled output of the first convolutional layer and filters it with 32 filters of the size 5×5 , with stride 1 and padding 1, hence there are 32 feature maps. The layer consists of $12 \times 12 \times 32$ neurons. Max pooling is applied to the outputs, maximum is calculated in 3×3 windows and stride 2.

The last convolutional layer uses 32 filters of the size 3×3 with stride 1 and padding 1. Again there are 32 feature maps. The number of neurons is reduced to $5 \times 5 \times 32$. The same as for the other layers max pooling operation is applied, this gives 32 of 2×2 outputs.

The last two layers are fully connected layers. The fully connected layers are traditional MLPs (hidden layer + logistic regression). The first MLP consists of 256 neurons connected to the outputs of the last pooling layer, the second MLP elaborates the 4 classes of LOS and consists of 4 neurons. Convolutional and fully connected layers use rectified linear units as activation functions, LCN is also used to enhance the performance of the convolutional layers. The total number of neurons in subsequent layers is: 14400—4608—800—256—4, in all there is 20068 neurons in the optimal CNN for classification of LOS.

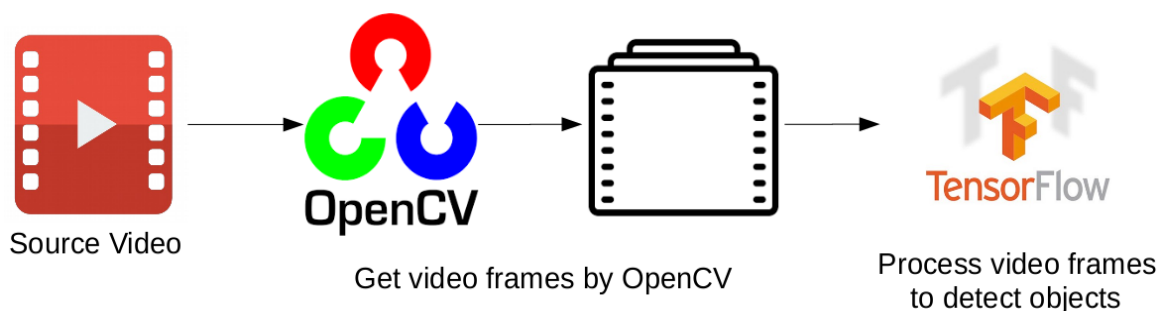


Fig 3.1 The sequence of processing the video feed.

3.1 SYSTEM DESIGN

Logical design

The design flow of the project is to generate the number of cars identified in the video frame of video frames based on the raw comparison with convolutional feature sets. The network is trained with a synthetic dataset to detect small cars made of plastic which will later be the test data on which it carries out predictions. The weights are updated using back propagation using the principle of gradient descent. No signs of vanishing gradient problem was encountered during the training of the mode

Design Goals

The design is to effectively generate feature-sets with above a million weights. The program is designed to Irrespective of the camera resolution, the image resolution is either scaled down or up to 223x223 pixels. And then converted to tensors to be feed to the convolutional model. The output is a matrix of probability distributions feed from the previous ReLU activation layer, alternatively this activation function can be switched to SoftMax based on the type of the data and noise factor.

The Activities of the Design process

The design process first starts with the collection of data a dataset of around 600 images were collected. Once the data is collected it is converted to a suitable uniform format and distributed into training, testing and validation sets with ratios of 0.8 and 0.2 each. Once the data is ready next steps involved in designing a model based on the data. The object detection classifier generates a frozen inference graph to summarize the object detection process and stored as a .tf record

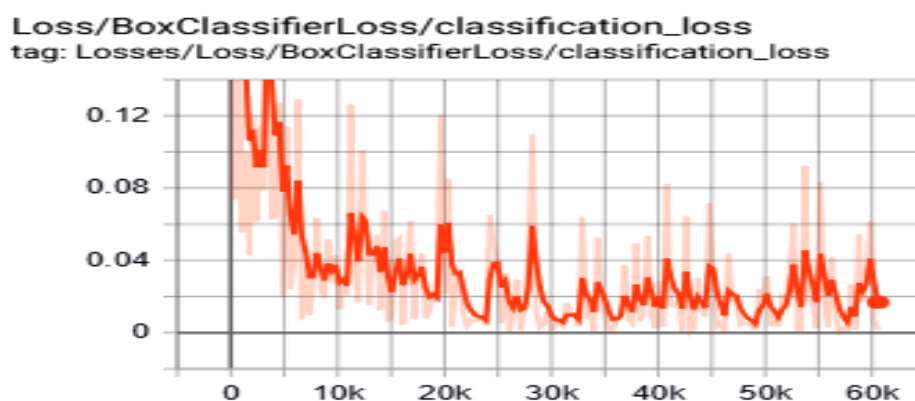


Fig 3.2 The training loss graph

CHAPTER 4

SYSTEM ANALYSIS

4.1 System Requirement Specification

SOFTWARE REQUIREMENTS

- Python 3.6.5
- OpenCV
- Numpy
- TensorFlow – GPU (v1.6 or higher)
- CUDA for maximizing compute capability of the machine.
- Anaconda IDE
- Raspbian for Raspberry Pi.
- GPIO daemon to access GPIO remotely.

HARDWARE REQUIREMENTS

- Raspberry Pi 3 Model B+
- Desktop or laptop with the following or better configurations:
 - RAM – 8GB or higher.
 - Processor – Intel i5 7th Generation or Higher (Base clock rate – 2.5Ghz)
 - Graphics Processor - NVIDIA GTX 1050Ti with 4GB memory or higher.
 - SLI Connectivity (Optional)
- Video Source – Either a Webcam or IPCamera.
- Wireless Router

CHAPTER 5

SYSTEM DESIGN

5.1 Use Case diagram

The use case diagram is a pictorial representation of how the various components of the system and the users interact with each other while performing a specific task. The various steps associated with the execution of the program are specified along with the different roles assumed by the users. The diagram provides a clear image of how the users interact with the system.

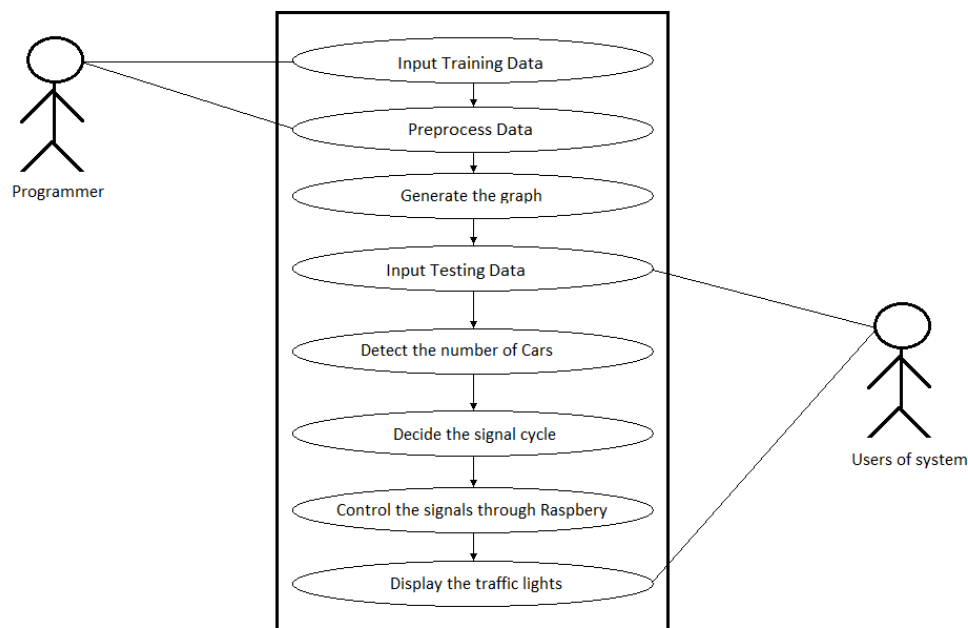


Fig 5.1 The activity diagram of the designed system

5.2 Sequence diagram

The Sequence diagram is a graphical representation of the various interaction which occur between the different components and the users during the runtime of the program in an orderly manner. It also represents the construct of a message sequence chart. The sequence diagram depicts how various objects in a system interact and are arranged in a time sequence. It represents the objects and classes involved in the scenario and the

sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically employed while representing the case realizations in the Logical View of a system, which is under development. Sequence diagrams are also known as event diagrams or event scenarios.

Sequence Diagram of Designed System

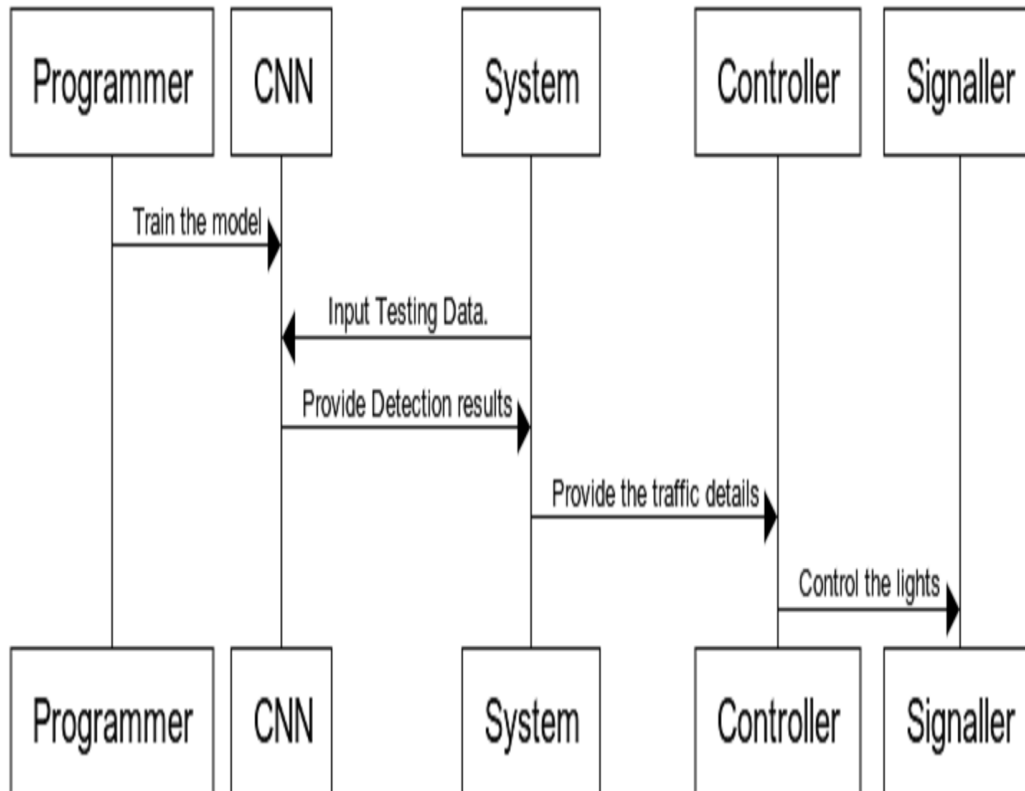


Fig 5.2 The sequence diagram of the system

5.3 Data Flow Diagram

The Data flow diagram, also called as 'DFD' represents how the data "flows" through an information system, modelling the various aspects associated with it. The data flow diagram is often used as the fundamental step to generate an overview of the developed system without going into the details and the specifications of it, which can later be explained. The data flow diagrams is also be used for visualizing how the data is processed by the system.

The data flow diagrams show what kind of data will be input to and output from the system, how the data is processed by the system, and where the data will be stored, either locally or remotely. It does not give us information related to the timing of process or the information about whether processes will run in a parallel or a sequential manner, unlike a flowchart would depict the same

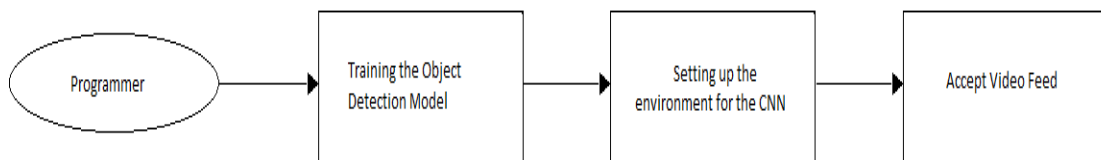


Fig 5.3.1 Dataflow diagram of the Programmer

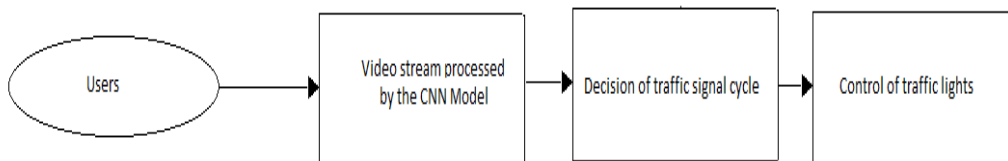


Fig 5.3.2 Dataflow diagram of the users of system

5.4 Activity Diagram

The Activity diagram is a pictorial representation of workflows how the entire system runs, based on the data available to the system, decisions, iterations and flow of data. In Unified Modeling Language (UML), activity diagrams are used to model both, the developmental, procedural and computational processes

Activity diagrams are designed from a limited number of shapes, with arrows directing the flow of control or data. The important shape types are mention as follows:

Arrows run from the start towards the end and represent the order in which activities happen.

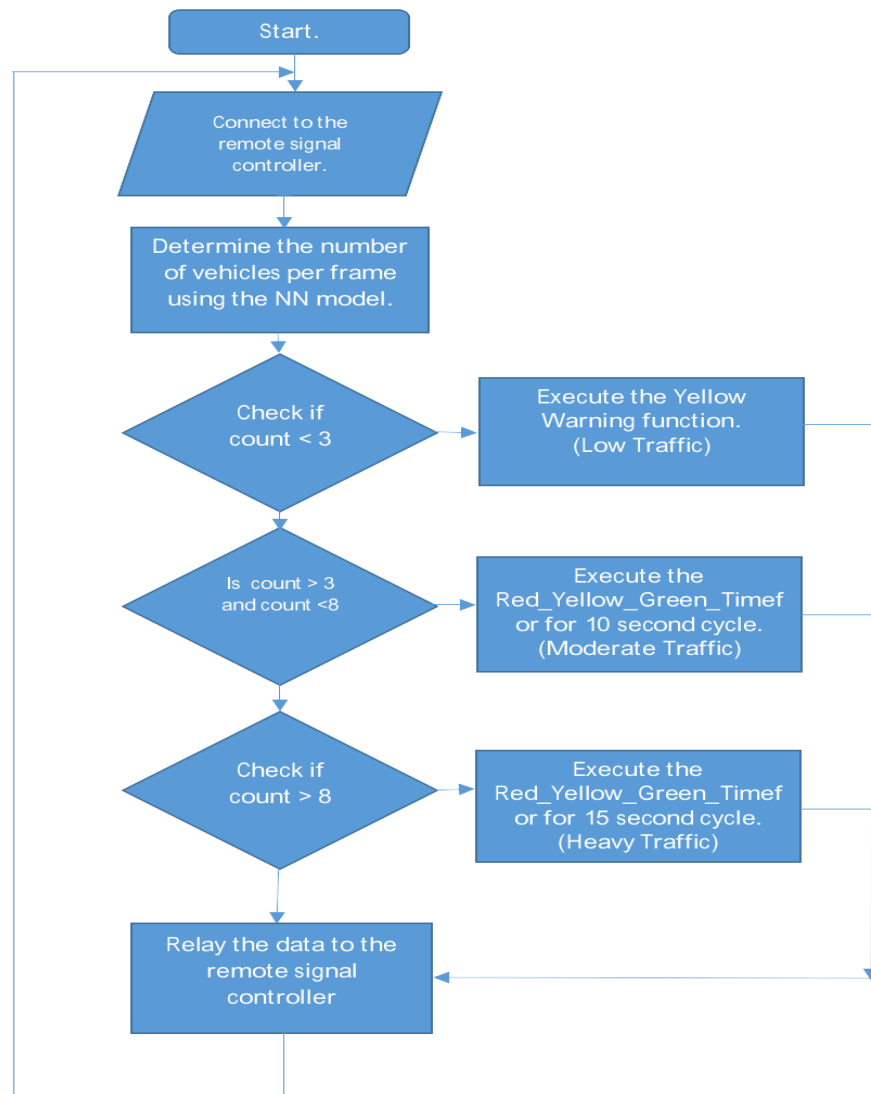


Fig 5.4 Activity Diagram of the System

CHAPTER 6

SYSTEM IMPLEMENTATION AND CODING

6.1 Code and Explanation

```
import os
import cv2
import numpy as np
import tensorflow as tf
import sys
import time
import datetime
from threading import Thread
import csv
from gpiozero import LED
```

The program is first loaded with the required libraries which perform the various functionalities of the program. The `os` module is designed to implement system related operations such as opening and closing of directories, renaming files and folders, listing contents of directory, etc. The `cv2` module provides the base library to perform the computer vision related operations and provides the foundation for the tensor flow CNN based object detection library. The Numpy module is a library for numeric operations related tasks and is a pre requisite for `cv2`, as videos are processed as arrays of pixels. The `sys` library is imported to perform system related operations such as auto-run, scheduling, etc. The `datetime` module is used for the purpose of retrieving the current date and time of the system. The Threading module is used to run multiple scripts parallel in a Python environment. The `csv` module is required for the purpose of storing data locally for data analytics regarding the density of the traffic at various times of the day. The `gpiozero` module of Python is used in communicating with the raspberry pi module in

order to control the signals. It is performed by connecting, both, raspberry pi and the computer to the local area network.

```
global CarsDet
```

```
CarsDet=1
```

The global variable CarsDet is declared to help in retrieving the number of cars detected in the frame and utilizing the same values across the various sections of the program.

```
from utils import label_map_util
```

```
from utils import visualization_utils as vis_util
```

The label maps util plays an important role. The module visualization_utils helps in drawing bounding boxes around the detected objects and the label map utils consists of tools which is used to read and retrieve the values of the label map, which is the list of objects which the system can recognize.

```
red_traffic = LED(17) #assigning red_stop to GPIO 17
```

```
yellow_traffic = LED(4)
```

```
green_traffic = LED(18)
```

```
red_pedestrian = LED(27)
```

```
green_pedestrian = LED(22)
```

The GPIO pins of the raspberry pi are controlled by the server, thus each individual pin of the Raspberry is associated with the pins on the board, which will be controlled from the system.

```
with open('traffic_measurement.csv', 'w', newline = '') as f:
```

```
    writer = csv.writer(f)
```

```
    csv_line = \
```

```
        'Time, Vehicle Count'
```

```
    writer.writerow([csv_line.split(',')])
```

```
    print("Opened .csv successfully")
```

As discussed, a .csv file will store the values of the number of cars detected on the roads during detection. An object to manage the csv file is created using the csv.writer function. This will enable us to perform data analytics and understand the dynamic nature of the traffic on the streets in a simpler manner

```
MODEL_NAME = 'inference_graph'  
CWD_PATH = os.getcwd()  
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')  
PATH_TO_LABELS = os.path.join(CWD_PATH,'training','labelmap.pbtxt')  
# Number of classes the object detector can identify  
NUM_CLASSES = 1
```

The model is named inference graph and it is used in various points. The inference graphs is a trained CNN which is designed for the purpose of identifying and detecting cars in the video frame. The path of the current working directory is generated and stored in an object. The labels is a collection of objects which the CNN is trained on.

```
def CarCounter():  
    global CarsDet  
    cwd= "N.A"  
    label_map = label_map_util.load_labelmap(PATH_TO_LABELS)  
    categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=NUM_CLASSES,  
    category_index = label_map_util.create_category_index(categories)
```

The function CarCounter is implemented to perform the task of object detection and recognition. The label maps are loaded initially in a variable and are then converted into categories to be recongized. The categories generated are indexed, which will be helpful during the runtime of the program.

```
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name="")

sess = tf.Session(graph=detection_graph)
```

The detection graph needs to be created as a tensorflow object before being put to use. The stored graph is now retrieved and loaded into the tensorflow object. The graph which is stored is serialized, and needs to be deserialied before detection. The `tf.Session` function creates an environment for pipelining the data through the graph.

```
# Define input and output tensors (i.e. data) for the object detection classifier

# Input tensor is the image
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

# Output tensors are the detection boxes, scores, and classes
# Each box represents a part of the image where a particular object was detected
detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

# Each score represents level of confidence for each of the objects.
# The score is shown on the result image, together with the class label.
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

# Number of objects detected
num_detections = detection_graph.get_tensor_by_name('num_detections:0')
```

The tensor associated in this step is the image tensor, which contains the live video feed from the camera source, which will be processed by the CNN to identify and detect cars

```
video = cv2.VideoCapture(0 + cv2.CAP_DSHOW)
ret = video.set(3,640)
ret = video.set(4,480)
if not video.isOpened():
    print("Video Source Not Found")
    CarCounter()
print("Initializing Video Source... Please wait... ")
ret=True
while(video.isOpened()):
    if ret==False:
        print("Video Source Disconnected")
        CarCounter()
```

The video feed is obtained using the OpenCV's video capture function and is stored in a numpy object. The dimensionalities of the video is set and is checked if the video source is available. The system generates an error and constantly searches for the video source. If found, the video is fed to the tensorflow CNN to perform the process of detection and recognition.

```
# Acquire frame and expand frame dimensions to have shape: [1, None, None, 3]
# i.e. a single-column array, where each item in the column has the pixel RGB value
ret, frame = video.read()
frame_expanded = np.expand_dims(frame, axis=0)
```

The dimensions of the video are expanded and the corresponding changes are then processed by the system to detect the cars in the given frame of video. The frame is expanded along the 0th axis, using the inbuilt function of the numpy module.

```
# Perform the actual detection by running the model with the image as input
```

```
(boxes, scores, classes, num) = sess.run(  
    [detection_boxes, detection_scores, detection_classes, num_detections],  
    feed_dict={image_tensor: frame_expanded})  
final_score = np.squeeze(scores)  
CarsDet = 0  
for i in range(100):  
    if scores is None or final_score[i] > 0.5:  
        CarsDet = CarsDet + 1
```

The created session is now run to generate the bounding boxes, along with the scores and accuracy of the prediction and the corresponding class of the object is determined. If and only if the accuracy of the predicted object is over 50%, the system detects the objects and increments the count of vehicles that are detected.

```
# Draw the results of the detection (aka 'visulaize the results')
```

```
vis_util.visualize_boxes_and_labels_on_image_array(frame,np.squeeze(boxes),np.squeeze(classes).astype(np.int32),np.sque  
  
if(CarsDet >=3 and CarsDet <=5):  
    cwd = "Medium"  
elif(CarsDet >=6):  
    cwd = "High"  
else:  
    cwd = "Low"  
now = datetime.datetime.now()  
cv2.rectangle(frame, (0,450), (640,480), (255, 255, 255), -1)  
cv2.rectangle(frame, (460,0), (640,20), (255, 255, 255), -1)  
cv2.putText(frame,now.strftime("%H:%M:%S %d-%m"), (460,20), cv2.FONT_HERSHEY_TRIPLEX, 0.66, (0,0,0), 0)  
cv2.putText(frame, "Cars Detected:" +str(CarsDet), (400,frame.shape[0]-12), cv2.FONT_HERSHEY_TRIPLEX, 0.66, (0,0,0), 0)  
cv2.putText(frame, "Traffic:." +cwd, (0,frame.shape[0]-12), cv2.FONT_HERSHEY_TRIPLEX, 0.66, (0,0,0), 0)  
cv2.imshow('Processed Video Feed', frame)
```

The `vis_util_boxes_and_labels_on_image_array()` function is called to perform the visualization operations on the processed video feed, where the bounding boxes, class of the object detected, accuracy are printed over the image array and is displayed in the feed. The computing of the traffic signal which is best suited for the moment is decided using simple if statements, where if less than 5 and greater than 3, the medium traffic signal cycle is activated. If not, the heavy traffic signal cycle is activated, and when there are less than 3 vehicles, the warning signals is activated. This dynamic options available ensures that the traffic is well managed based on the number of vehicles detected.

```
if cv2.waitKey(1) == ord('q'):  
    break  
video.release()  
cv2.destroyAllWindows()
```

If the program needs to be terminated, the 'y' key on the keyboard needs to be pressed to ensure that the video-feed source is destroyed leading to the termination of the program.

```
def Write_to_csv():  
    global CarsDet  
    time.sleep(2)  
    now = datetime.datetime.now()  
    Curr_Time = now.strftime("%H:%M:%S %d-%m")  
    with open('traffic_measurement.csv', 'a', newline='') as f:  
        writer = csv.writer(f)  
        csv_line = (Curr_Time, CarsDet)  
        writer.writerow([csv_line])  
    print("Data Saved Successfully")  
    time.sleep(600)  
    Write_to_csv()
```

As discussed earlier, we have implemented a Python module to write the traffic measurements into a .csv file which can be used for future data analytics and research to identify the underlying patterns in the traffic data. These patterns can be easily understood

when visualized thus, the number of cars identified are written into a .csv file every 600 seconds (10 Minutes). The traffic details is recorded along with the time at which the observation are made.

```
def Signaller():
    global CarsDet
    print("Value Received :"+str(CarsDet))
    if(CarsDet <= 3):
        print("Low Traffic - Yellow Warning")
        Yellow_Warning()
    elif(CarsDet > 3 and CarsDet < 8):
        print("Moderate Traffic - 10 second cycle")
        Red_Yellow_Green_Timer(10,10)
    else:
        print("Severe Traffic - 15 second cycle")
        Red_Yellow_Green_Timer(15,15)
    time.sleep(1)
    Signaller()
```

The signaller module of the program is designed to control the remote computer, raspberry pi module which is connected through a network port and IP through a LAN. This module crafts a network packet and sends it over to the raspberry pi which has a daemon running in the background to accept the data through the LAN and process the same to control the GPIO pins on the Pi, in order to manage the traffic. These functions in turn manage the traffic signaller. Once the number of cars have been detected, the call to signaller function is given to manage the signaller.

```
def Red_Yellow_Green_Timer(a,b):
    yellow_traffic.on()
    print("Yellow Signal")
    time.sleep(2)
    yellow_traffic.off()
    print("Red Signal")
    red_traffic.on()
    Pedestrian_Crossing_Green()
    time.sleep(a)
```



```
green_pedestrian.off()
Pedestrian_Crossing_Red()
time.sleep(2)
red_traffic.off()
print("Green Signal")
green_traffic.on()
time.sleep(b-2)
green_traffic.off()
```

The above defined function is designed to manage the task of controlling the various traffic lights in a sequential manner. This function is designed to control the lights in a sequential manner with both traffic lights and pedestrian lights. The function receives the period of time for which the lights needs to be on or off from the signaller function, which in turn received the values from the detector function.

```
def Yellow_Warning():
    for r in range(0,5):
        print("Yellow Signal")
        red_pedestrian.off()
        yellow_traffic.on()
        time.sleep(1)
        yellow_traffic.off()
        time.sleep(1)
        yellow_traffic.on()
        time.sleep(1)
        yellow_traffic.off()
        time.sleep(1)
```

The yellow_warning() lights are used to control the warning lights for a period of 5 seconds after which the signal cycle changes from one to another, or remains the same based on the existing traffic condition.

```
def Pedestrian_Crossing_Green():
    red_pedestrian.off()
    green_pedestrian.on()
    print("Pedestrian Green")
```

The pedestrian_crossing_green() is used to manage the traffic signal which is meant for crossing the road for pedestrians.

```
def Pedestrian_Crossing_Red():  
    red_pedestrian.on()  
    print("Pedestrian Red")
```

The pedestrian_crossing_red() will manage the red traffic signal, by controlling the GPIO to the red pin whenever function is called.

```
thread1 = Thread(target = CarCounter,args=[])  
thread1.start()  
thread2 = Thread(target = Write_to_csv,args=[])  
thread2.start()  
thread3 = Thread(target = Signaller,args=[])  
thread3.start()
```

The program is split into 3 sections where, the first component of the program is the detector function which deals with object detection and counting. The write to csv module is used to enter the values into the .csv file to perform data analysis and visualtions. The signaller module is defined to manage and run the raspberry pi script and control the lights remotely through LAN.

CHAPTER 7

RESULTS AND DISCUSSION



Fig 7.1 Traffic signal cycle in a low traffic condition.

This represents how the system was correctly able to identify cars in a scene where there are less cars in the video frame and the corresponding “Yellow Warning” signal cycle was initiated where the yellow traffic lights blinked. This cycle automatically ends after 5 seconds and based on the number of cars identified in the next cycle, the corresponding signal is given.



Fig 7.2 Traffic signal cycle in a moderate condition.

This output generated presents how the system was correctly able to identify cars in a scene where there was a moderate traffic situation where the number of cars in the video frame was found to be 4 and the corresponding “Moderate Traffic” signal cycle was initiated where the red light glowed. This cycle automatically ends after 23 seconds, where 10 seconds is given for both, red and green signal along with 3 seconds delay for yellow signal.

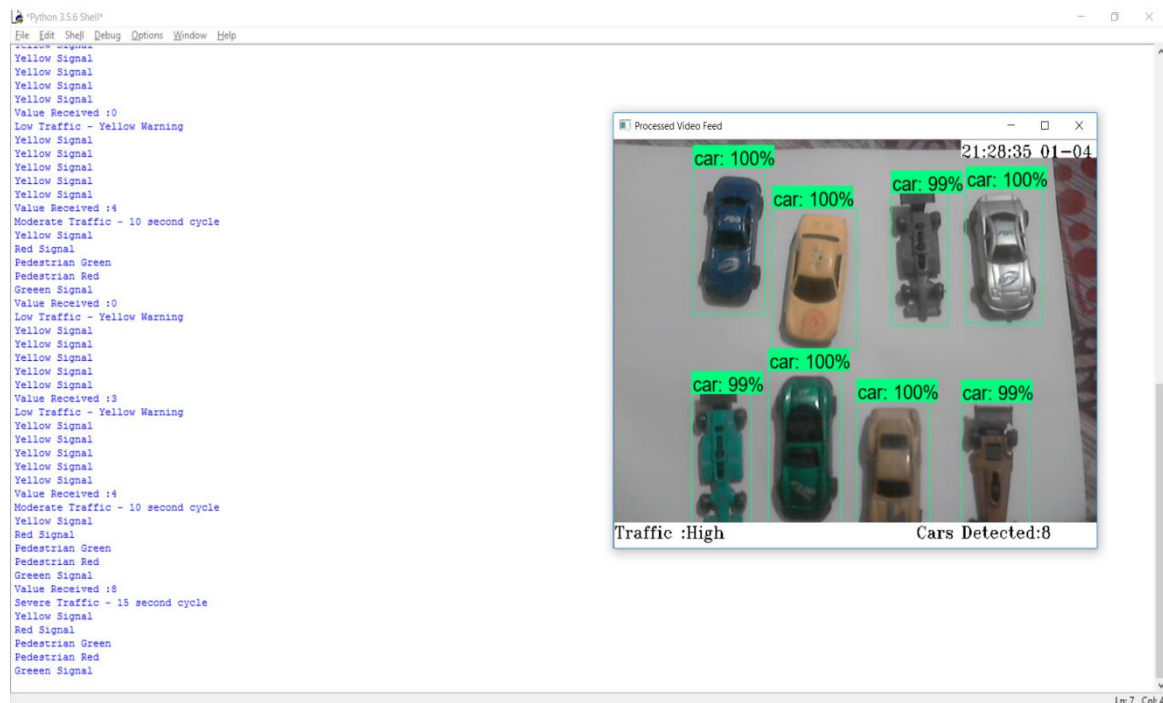
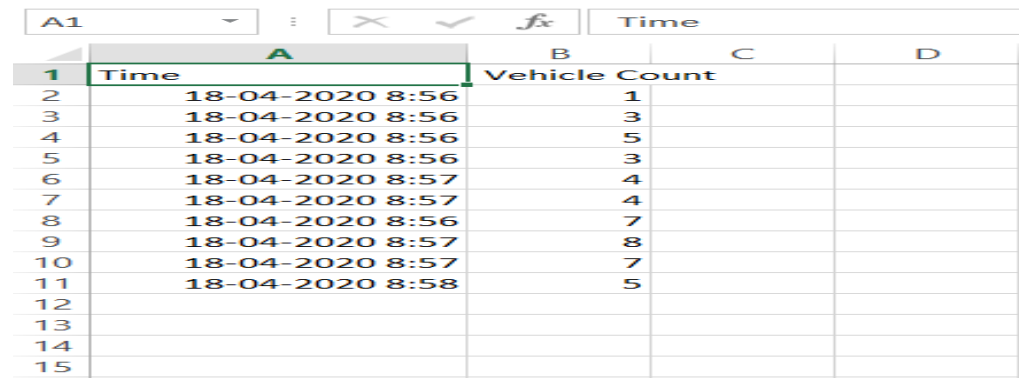


Fig 7.3 Traffic signal cycle in a severe condition.

This output generated presents how the system was correctly able to identify cars in a scene where there was heavy traffic with over 9 cars detected in the video frame and the corresponding “Severe traffic” signal cycle was initiated where the red light glowed, This cycle automatically ends after 33 seconds, where 15 seconds is given for both, red and green signal along with 3 seconds delay for yellow signal.

This way, the traffic can be efficiently and effectively managed by the system during any time of the day by the system, ensuring that there is no way in which a particular intersection of the traffic is left unattended.



The image shows a screenshot of an Excel spreadsheet. The spreadsheet has four columns labeled A, B, C, and D. Column A is titled 'Time' and contains a list of timestamps from 18-04-2020 8:56 to 18-04-2020 8:58. Column B is titled 'Vehicle Count' and contains corresponding numerical values ranging from 1 to 8. Columns C and D are currently empty. The spreadsheet interface includes a formula bar at the top with the text 'Time' and a grid of cells below.

	A	B	C	D
1	Time	Vehicle Count		
2	18-04-2020 8:56	1		
3	18-04-2020 8:56	3		
4	18-04-2020 8:56	5		
5	18-04-2020 8:56	3		
6	18-04-2020 8:57	4		
7	18-04-2020 8:57	4		
8	18-04-2020 8:56	7		
9	18-04-2020 8:57	8		
10	18-04-2020 8:57	7		
11	18-04-2020 8:58	5		
12				
13				
14				
15				

Fig 7.4 Values stored in the .csv file

This number of cars detected is also recorded at fixed intervals to understand and make require arrangements to manage the traffic during peak hours of the day. Additionally, it can also be used for data analytics and forecasting to help understand the traffic situation better.

CHAPTER 8

CHALLENGES

The challenges faced during the development of the project is was mainly due to the non availability of the training dataset. The program required ample training data in the for of toy cars to develop the training and the testing datasets. These images were then needed to be cleansed for erroneous pictures and then the model was trained to detect the cars.

The dataset which is available now needs to be labeled with the objects which are needed to be identified individually in every picture before the commencement of the training. This process took a considerable amount of time to mention the co ordinates of the object and label the images accordingly.

The model took around 8 hours for training itself over images of cars and this was the single most longest step required in the entire project. The model was then validated for its accuracy using the test dataset and the system was known to have an accuracy of 99.54% as the loss was reasonably low

The development of a network for establishing the connection between the computer and the remote computer (raspberry pi) was initially a challenge as the IP addresses were needed to be identified in the network, followed by which the ports to retrieve data at both ends were needed to be established.

CHAPTER 9

CONCLUSION

As the traffic situation round the world worsens day-by-day, this system aims at ending the ever-growing problem efficiently by using modern-day technologies which will not only provide a potential solution, but also reduce the human intervention required to manage the system. The system will make use of IoT, along with concepts from Machine Learning and Computer Vision. Further, this system can be designed to monitor traffic violations and automate the process of catching those guilty of committing traffic rule violations.

The developed system will be a boon in countries where the traffic situation is known to be unmanageable and complex, this system proves to be not only effective, but also efficient in terms of adaptability and complexity. The system uses cost effective hardware components which can be purchased in mass and a single powerful GPU to perform computations. The hardware tends to pay up for itself in the long run, bringing down the cost to run the system. It effectively brings down the manpower to manage the traffic.

CHAPTER 10

FUTURE SCOPE

The program is designed to manage and update the traffic scenario without the intervention of humans, furthermore the system can be run independent without the video processing, as we can use techniques such as traffic forecasting and automate the junctions.

The delays in traffic can also be updated to a 3rd party application to understand and divert the traffic if and when required to reduce the burdens on the main roads which link and connect to the major parts of the city where the traffic is maximum.

The processed feed can provided to the control room to ensure that there are no hustles in the traffic which is plying on the road, therefore remote administration can also be done in this regard.

A number plate detection technique can also be included along this project to ensure that the registration number of those vehicles which violate the traffic rules are automatically penalized by the competent authorities.

REFERENCES

1. Vikramaditya Dangi, Amol Parab, "Image Processing Based Intelligent Traffic Controller", Undergraduate Academic Research Journal (UARJ), ISSN: 2278 – 1129, Volume-1, Issue-1, 2012.
2. Raoul de Charette and Fawzi Nashashibi, "Traffic light recognition using Image processing Compared to Learning Processes".
3. Mriganka Panjwani, Nikhil Tyagi, Ms. D. Shalini, Prof. K Venkata Lakshmi Narayana, "Smart Traffic Control Using Image Processing".
4. F. Mehboob, M. Abbas, R. Almotaeryi, R. Jiang, S. Al-Maadeed and A. Bouridane, "Traffic Flow Estimation from Road Surveillance," *2015 IEEE International Symposium on Multimedia (ISM)*, Miami, FL, 2015, pp. 605-608
5. M. Yang, R. Jhang and J. Hou, "Traffic flow estimation and vehicle-type classification using vision-based spatial-temporal profile analysis," in *IET Computer Vision*, vol. 7, no. 5, pp. 394-404, October 2013.
6. Ankit Chaudhary, Reinhard Klette, J. L. Raheja and Xia Jin, "Introduction to the special issue on computer vision in road safety and intelligent traffic", *EURASIP Journal on Image and Video Processing 2017*:16
7. F. Espinosa, C. Gordillo, R. Jiménez and O. Avilés, "Dynamic traffic light controller using machine vision and optimization algorithms," *2012 Workshop on Engineering Applications, Bogota*, 2012, pp. 1-6.