**TRAFFIC MONITORING ANALYSIS USING PYTHON**

**A MAJOR PROJECT REPORT**

***Submitted by***

**ROKIBUL ISLAM RONI**

***In partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Under the guidance of**

**xxxxxx**



**ALAKH PRAKASH GOYAL SHIMLA UNIVERSITY**

**SHIMLA 171009**

**xxxxxxx**

**MAY 2020**

**BONAFIDE CERTIFICATE**

Certified that this project report “**TRAFFIC MONITORING ANALYSIS USING PYTHON**” is the bonafide work of “**ROKIBUL ISLAM RONI**” who carried out the project work under my supervision.

**……………………. …………………….**

MR. ANUJ GUPTA xxxxxxxxxxxxxxxx

**CLASS MENTOR SUPERVISOR**

Assistant Professor

Department of computer science

ALAKH PRAKASH GOYAL

SHIMLA UNIVERSITY

**……………………………………….**

Dr. ARUN KUMAR CHOUDHARY

**HEAD OF DEPARTMENT**

**ABSTRACT**

Due to increase in population, number of vehicles on the road has increased. Also, there is a rush on the road specially at traffic while crossing signals. At traffic signals there is always high chance of accident due to breaking traffic rules by drivers and pedestrians. Therefore, to ensure safety at road it is important to manage it well and continuously monitor it. In case if any mishap happen, immediate action should be taken to save precious lives. However, managing and monitoring traffic is very difficult and it require human effort. Generally, official used CCTV camera to see live frames but it requires human effort to monitor it. It became difficult to manage if one need to count how many cars crossed a particular street in a day? What is the frequency of people crossing signals? Answers of these questions help to decide whether this area is accidental prone or not.

Now-a-days traffic signals and sign board are placed to manage traffic and avoid accidents. However, sometimes driver can’t see the sign board such as ‘speed breaker’ and drive car without slowing it down which results accident on the road. Sometimes, driver cannot see a person who is walking beside road or crossing signals causes accidents and sometimes driver can’t take instant decision to avoid accidents.

One solution of these problems may be if a system should detect persons walking beside road/crossing signals, vehicles passing through street, traffic signs such as speed breaker, speed limit, U-turn prohibited than and help driver by converting detected objects into audio signals so that even if driver cannot see objects system would remind them. This concept can also be used in automatic driving applications. This system can also be used to monitor traffic on the road.

Here, I developed a real time traffic monitoring application is developed using python. It also showing alert to the drivers who are using the app.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **CHAPTER NAME** | **PAGE NO** |
|  | TITLE PAGE |  |
|  | BONAFIDE CERTIFICATE |  |
|  | ABSTRACT |  |
|  | LIST OF CONTENT |  |
|  | LIST OF TABLES |  |
|  | LIST OF FIGURES |  |
|  | ABBREVIATION |  |
| **CHAPTER 1** | **INTRODUCTION** | 08 |
| 1.1 | OBJECTIVES | 08 |
| 1.2 | LITERATURE SURVEY | 09 |
| 1.3 | METHOLOGY | 10 |
| **CHAPTER 2** | **FEASIBILITY REPORT** | 11 |
| 2.1 | ECONOMIC FEASIBILITY | 11 |
| 2.2 | OPERATIONAL FEASIBILITY | 12 |
| 2.3 | TECHNICAL FEASIBILITY | 12 |
| **CHAPTER 3** | **FACILITIES REQUIRED** | 13 |
| 3.1 | HARDWARE REQUIREMENTS | 13 |
| 3.2 | SOFTWARE REQUIREMENTS | 13 |
| **CHAPTER 4** | **SOFTWARE REQUIREMENT SPECIFICATION** | 14 |
| 4.1 | REQUIREMENT SPECIFICATION | 14 |
| 4.2 | FUNCTIONAL REQUIREMENT | 14 |
| 4.3 | USER INTERFACE DESIGN | 18 |
| 4.4 | COMPUTER INITIATED INTERFACE | 18 |
| **CHAPTER 5** | **MODULE AND EXPRESSION** | 19 |
| 5.1 | OPENCV | 19 |
| 5.2 | OPENCV VIDEO PROCESSING | 19 |
| 5.3 | READING VIDEO - SCIKIT | 21 |
| 5.4 | NUMPY | 22 |
| 5.5 | NUMPY INSTALLATION | 22 |
| **CHAPTER 6** | **VEHICLE IDENTIFICATION** | 23 |
| 6.1 | BACKGROUND | 23 |
| 6.2 | SCOPE | 23 |
| **CHAPTER 7** | **SPEED DETECTION** | 24 |
| 7.1 | BACKGROUND | 24 |
| 7.2 | GOALS | 24 |
| 7.3 | SCOPE | 24 |
| **CHAPTER 8** | **IMPLEMENTATION** | 25 |
| 8.1 | PROBLEM DECOMPOSITION | 25 |
| 8.2 | DESIGN AND ALGORITHM | 26 |
| 8.3 | SUBSEQUENT | 30 |
| 8.4 | DESIGN JUSTIFICATION, TESTING & RESULT | 33 |
| **CHAPTER 9** | **SOURCE CODE STRUCTURE** | 35 |
| 9.1 | PYTHON CODE STRUCTURE | 35 |
| 9.2 | IMPORTED LIBRARIES | 35 |
| **CHAPTER 10** | **TESTING** | 36 |
| **CHAPTER 11** | **IMPLEMENTATIONS** | 38 |
| **CHAPTER 12** | **FUTURE IMPROVEMENT** | 39 |
| **CHAPTER 13** | **CONCLUSION** | 40 |
|  | **APPRENDIX – SOURCE CODE** | 41 |
|  | **REFERENCE** | 50 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **TITLE** | **PAGE NO** |
| 1 | TRAINING & TESTING DATA | 27 |
| 2 | NEURAL NETWORK ARCH | 28 |

**TABLE OF FIGURE**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | **PAGE NO** |
| 1 | DESIGN & ALGORITHMS FIG | 26 |
| 2 | THE BOUNDING BOXES | 29 |
| 3 | SUBSEQUENT | 30 |
| 4 | CALCULATED VALUES | 32 |
| 5 | THE DETECTED CARS AND THEIR SPEEDS | 33 |
| 6 | PREDICTION | 34 |

**ABBREVIATION**

|  |  |
| --- | --- |
| FACS | FACIAL ACTION CODING SYSTEM |
| FER | FACIAL EXPRESSION RECOGNITION |
| LDA | BONAFIDE CERTIFICATE |
| GPU | GRAPHICS PROCESSING UNIT |
| CNN | CONVOLUTION NEURAL NETWORK |
| LDA | LINEAR DISCRIMINANT ANALYSIS |
| PCA | PRINCIPLE COMPONENT ANALYSIS |
| RELU | RECTIFIED LINEAR UNIT |
| SIANN | SPACE INVARIANT ARTIFICIAL NEURAL NETWORK |

**CHAPTER 1**

**INTRODUCTION**

* 1. **OBJECTIVES:**

Traffic Monitoring System is an advanced Machine Learning Technology. It monitors vehicles speed, display the data and alert the driver at the real time by an android app. It is based on Machine Learning. The key processing steps involved in the Traffic Monitoring Systems which extracting the features of images in video frames and classifying them using trained component classifiers to determine the class of the object.

**It has three major functionalities:**

1. Vehicle Detection

2. Vehicle Speed Detection

3. Get Notification by Mobile App

**Vehicle Detection**

Vehicles/vehicle boundaries (primarily Cars) were identified automatically using Haar Cascade Classifier specifically trained to detect cars. This involved training and optimizing the model to be specific to vehicle classification.

**Speed Detection**

This part mainly focused on detecting the speed of the vehicle in real time by keeping track of the relative pixel positions in every single frame and identifying the changes in positions of the cars detected by using pre-trained vehicle classifier and converting them into real object parameters. To generalize the direction in which the vehicle is progressing subjecting to change in pixel values, Manhattan distance is being calculated between the centroids of the vehicles in subsequent frames and resulting pixel value is converted from image plane to object plane.

**1.2 LITERATURE SURVEY**

Traffic Monitoring System is a project for analyzing and predicting the Vehicles and Vehicle Speed Detection. The platform I used here is python-openCV.

In order to find out the speed of any object the basic parameters that will be needed are distance and time. In this project, making all calculations in the image plane and then later projecting them in the object plane. In this case, the distance a particular vehicle has travelled in the image plane would be the relative differences in pixels at various vehicle positions in subsequent frames. Time would be the rate at which the frames were being processed, which I assumed to be 30 frames per second for the purposes of this project.

From the above description,

**Time = Number of pixels progressed by vehicle/frames per second and Distance**

**= sqrt (mid1\*mid1+mid2\*mid2)**

If the difference is less than 10 pixels, then the car is assumed to be stationary and is neglected when updating values. Once the speed of the vehicle in the image plane is calculated, then I convert the pixel values obtained into km/sec which is representative of the object plane, using the distance of the object from the camera in the given formula exact distance to

**object (mm) = focal length (mm) \* real height of the object (mm) \* image height (pixels)/object height (pixels) \* sensor height (mm).**

However, the trained classifier is unable to detect all the cars in every single frame and thus the obtained speed values are not always accurate and subject to deviate from original value depending on the following assumptions on the parameters. The height of the object in object plane is assumed to be 4meters in general approximating to median values for all cars which is not the same in every case.

**1.3 METHODOLOGY**

1. Captured images from video frames.

2. Acquiring data.

3. Extracting the features of images in video frames.

4. Process Data.

5. Classifying the extracted images data.

6. Use trained component classifiers.

7. Univariate Analysis.

8. Bi-variate Analysis.

9. Missing values treatment.

10. Outlier treatment.

11. Determine the class of the object.

12. Locate vehicles in a video sequence.

13. Identify vehicles in a video sequence.

14. Model building and prediction.

15. Validate and Execute.

16. Display the results of data.

17. Deploy.

**CHAPTER 2**

**FEASIBILITY REPORT**

Feasibility is an important phase in the software development process it enables the developers to have an assessment of the product being developed It refers to the feasibility study of the product in terms of outcomes of the product, operational required for implement ting it. Feasibility study should be performed on the basic of various criteria and parameters. The various feasibility studies are:-

* Economic Feasibility
* Operational Feasibility
* Technical Feasibility

# ECONOMIC FEASIBILITY

It refers to the benefits or outcomes we are deriving from the product as compared to the total cost we are spending for developing the benefits are more or less the same as the older system then it is not feasible to develop the product. The product is economical feasible.

* + - Reduces the time to understand people
    - Reduces the work load
    - User can understand the people’s reaction
    - Beneficial for business

# OPERATIONAL FEASIBILITY

It refers to the feasibility of the product to be operational. Some products may work very well at the design and implementation but many fail in the real time environment. It introduces the study of human reactions required and their technical expertise. This product is operationally feasible as it is designed specifically for business / understand the emotions for better communication.

# TECHNICAL FEASIBILITY

The system is self-explanting and does not need any entire sophisticated training. A system has been built by concentrating on the graphical user interface concepts, the application can also be handled very easily with a novice uses. The overall time that a user needs to get trained is less than 15 minutes. The system has been added with features of menu device and button interaction methods, which makes him the master as he starts working through the environment. As the software that were used as developing this application are very economical and are readily available is the market the only time that is lost by the customer is just installation time.

**CHAPTER 3**

**FACILITIES REQUIRED**

**3.1 HARDWARE REQUIREMENTS**

1. DISK: 100 GB

2. RAM: 4 GB (Min)

3. PROCESSOR: Core-i7 3.40 GHz (Commercial)

**3.2 SOFTWARE REQUIREMENTS**

1. cmake: 3.12.0

2. dlib: 19.16.0

3. numpy: 1.15.3

4. opencv-python: 3.4.3.18

**CHAPTER 4**

**SOFTWARE REQUIREMENT SPECIFICATION**

**4.1 REQUIREMENT SPECIFICATION**

System requirements are expressed in a software requirement document. The Software requirement specification (SRS) is the official statement of what is required of the system developers. This requirement document includes the requirements definition and the requirement specification. The software requirement document is not a design document. It should set out what the system should do without specifying how it should be done. The requirement set out in this document is complete and consistent.

**DEVELOPERS RESPONSIBILITIES OVERVIEW**

• Developing the system, which meets the SRS and solving all the requirements of the systems

• Demonstrating the system and installing the system at client’s location after the acceptance testing is successful.

• Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.

• Conducting any user training that might be needed for using the system.

• Maintain the system for a period of one year after installation.

**4.2 FUNCTIONAL REQUIREMENTS**

**OUTPUT DESIGN**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

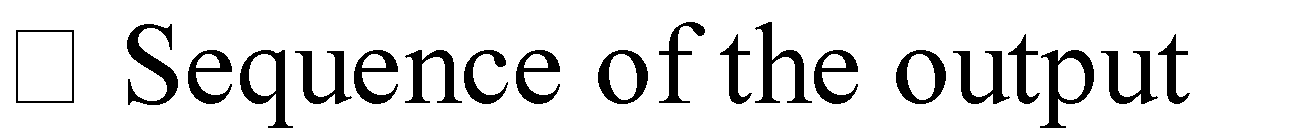
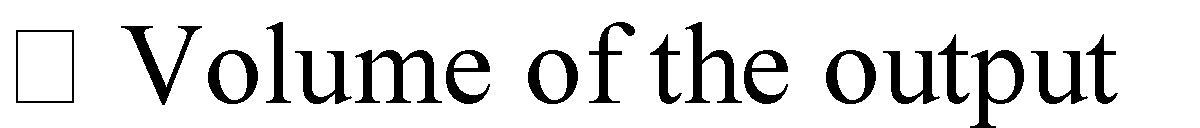
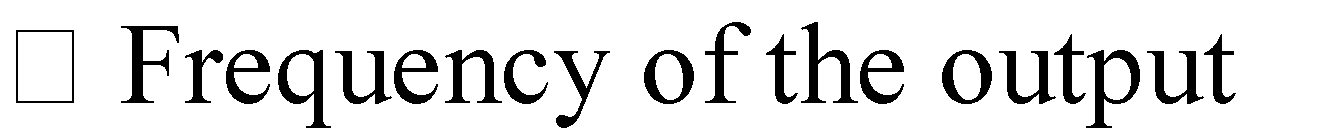
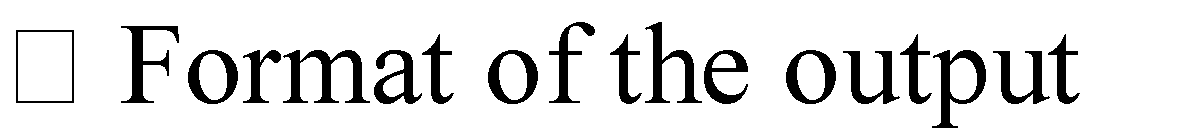
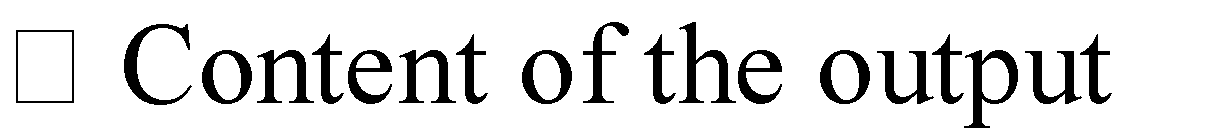
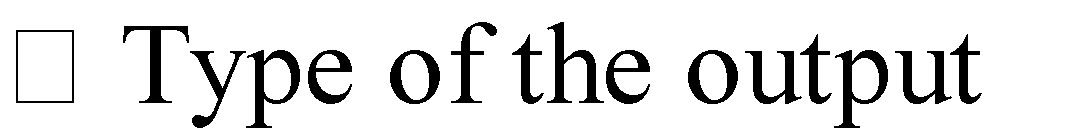
• External Outputs, whose destination is outside the organization.

• Internal Outputs whose destination is within the organization and they are the user’s main interface with the computer.

• Operational outputs whose use is purely within the computer department.

**OUTPUT DEFINATION**

The outputs should be defined in terms of the following points:

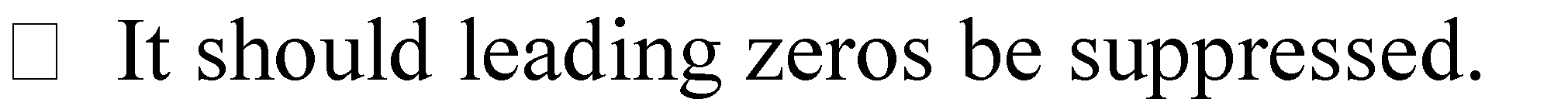
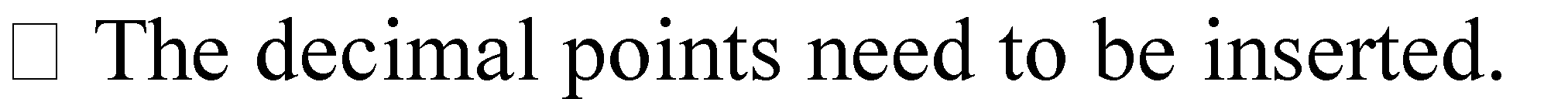




tput

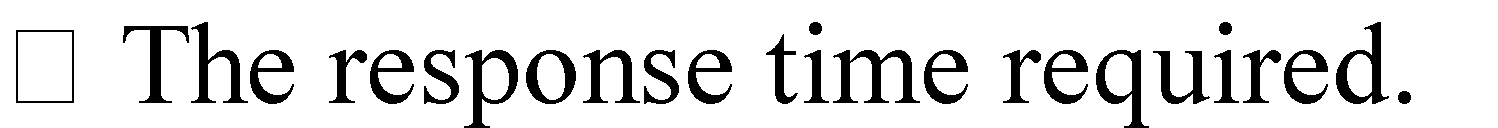
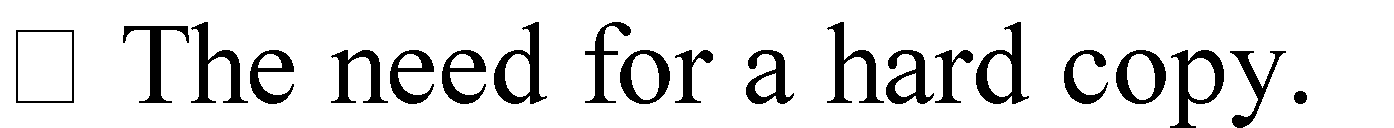
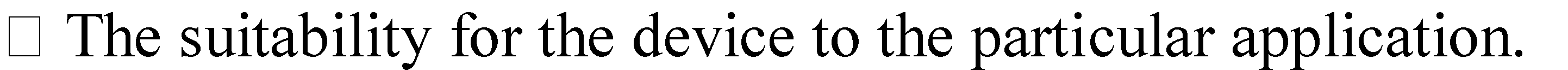
It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

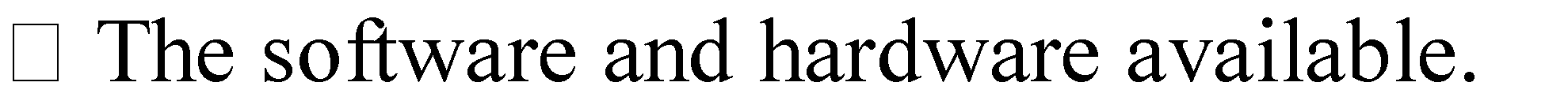
For Example



**OUTPUTS MEDIA**

In the next stage it is to be decided that which medium is the most appropriate for the output. The main considerations when deciding about the output media are:





Keeping in view the above description the project is to have outputs mainly coming under the category of internal outputs. The main outputs desired according to the requirements specification are: The outputs were needed to be generated as a hot copy and as well as quires to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

**INPUT DESIGN**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To archive the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT MEDIA**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to

• Type of input

• Flexibility of format

• Speed

• Accuracy

• Verification methods

• Easy of correction

• Storage and handling requirements

• Easy to use

• Portability

• Probabilistic

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As input data is to be the directly keyed in by the user, the keyboard can be considered to be most suitable input device.

**ERROR AVIDANCE**

At this stage care is to be taken to ensure that input data remains accurate from the stage at which it is recorded unto the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**ERROR SECTION**

Even though every effort is making to avoid the occurrence of errors, still a small proportion of errors are always likely to occur these types of errors can be discovered by using validations to check the input data.

**DATA VALIDATION**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been including where necessary. The system is designed to be a user friendly one. In other words, the system has been designed to communicate effectively with the user. The system has been designed with pop-up menus.

**4.3 USER INTERFACE DESIGN**

It is essential to consult the system users and discuss their needs while designing the user interface.

**USER INTERFACE SYSTEM CAN BE BROADLY CLASSIFIED AS**

User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.

**Computer initiated interfaces**

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**USER-INITIATED INTERFACES**

**User initiated interfaces fall into approximate classes**

Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer. Forms oriented interfaces the user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**4.4 COMPUTER-INITIATED INTERFACES**

The following computer-initiated interfaces were used:

* The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
* Questions-answer type dialog system where the computer asks question takes action based on the basis of the users reply.
* Right from the start the system is going to be menu driven, the opening menu

**CHAPTER 5**

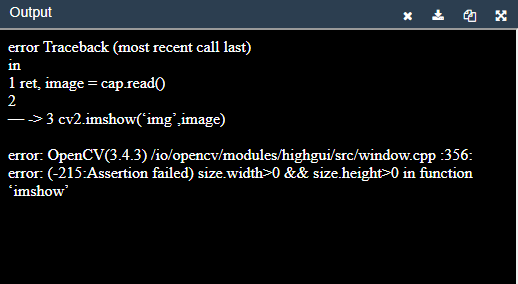
**MODULES AND EXPRESSION**

**5.1 OpenCV**

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.

OpenCV is an open source library which provides us with the tools to perform almost any kind of image and video processing. OpenCV is written in C++ and its primary interface is in C++. The OpenCV code is hard to develop and maintain, due to its methods naming, error logging, and sometimes weird code structures.

For example, when the last frame passes out, openCV simply crashes, with such error to console:



**5.2 OpenCV VIDEO PROCESSING**

First of all, we need to create a *cv2.VideoCapture object*, *cv2.VideoCapture is a class* for video capturing from video files, image sequences, or cameras.

**>>> import cv2**

#source might be provided as video filename or integer number for camera capture

**>>> cap = cv2.VideoCapture(source)**

Then, in order to play video, we create a non-trivial cycle:

**>>> while True:**

**… ret, frame = cap.read()**

It’s interesting that it’s an unbounded cycle.

**>>> while True:**

**… ret, frame = cap.read()**

**… cv2.imshow(‘window name’, frame)**

The cv2.imshow method displays an image in the specified window. We specify a window name as a first argument, and the frame we would like to display as a second.

And now, we need to somehow break an infinite cycle:

**>>> while True:**

**… ret, frame = cap.read()**

**… cv2.imshow(‘window name’, frame)**

**… if cv2.waitKey(1) & 0xFF == ord(‘q’):**

**… break**

**… cap.release()**

**… cv2.destroyAllWindows()**

cv2.waitKey() is a required building block for OpenCV video processing.

**waitKey** is a method which displays the frame for specified milliseconds.

The **‘0xFF == ord(‘q’)’** inside the ‘if’ statement is a special syntax to provide the ‘while’ loop break, by a keyboard key pressing event.

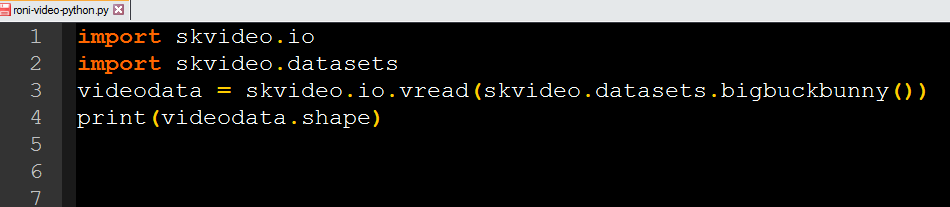
**cap.release()** and **cv2.destroyAllWindows()** are the methods to close video files or the capturing device, and destroy the window.

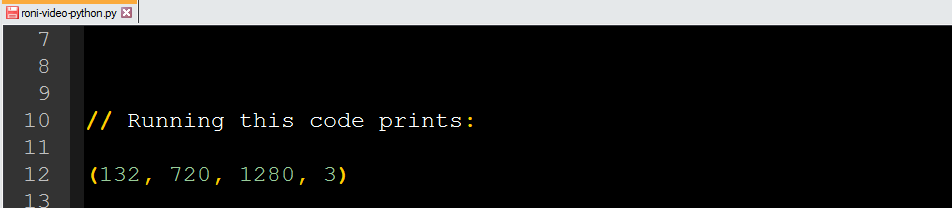
**5.3 VIDEO READING VIDEO – SCIKIT**

Scikit is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

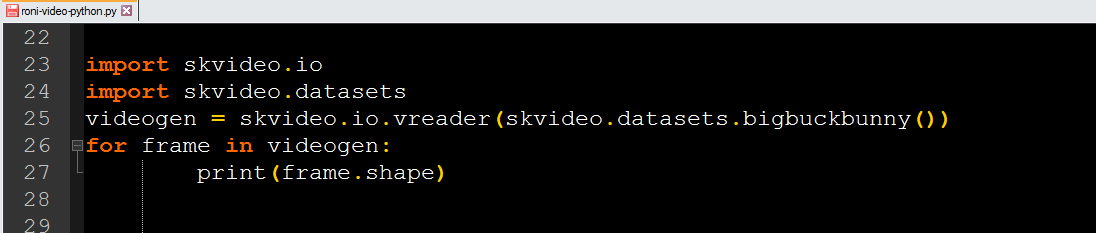
**skvideo.io** is a module created for using a *FFmpeg/LibAV* backend to read and write videos. Depending on the available backend, the appropriate probing tool (*ffprobe*, **avprobe**, or even *mediainfo*) will be used to parse metadata from videos.

Use *skvideo.io.vread* to load any video (here *bigbuckbunny*) into memory as a single *ndarray*. Note that this function assumes you have enough memory to do so, and should only be used for small videos.

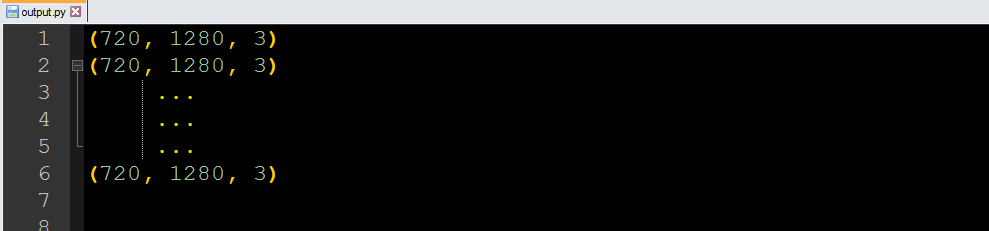




Use *skvideo.io.vreader* to load any video (here *bigbuckbunny*) frame-by-frame. This is the function to be used for larger files, and is actually faster than loading a video as 1 *ndarray*. However, it requires handling each frame as it is loaded. An example snippet:



And the output will be…



**5.4 numpy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

**5.5 numpy Installation**:

*pip install numpy*

**CHAPTER 6**

**VEHICLE IDENTIFICATION**

**6.1 BACKGROUND**

Vehicle detection and identification systems have become increasingly common in the computer vision community over the last few years.[1-5] The key processing steps involved in these systems are extracting the features of images in video frames and classifying them using trained component classifiers to determine the class of the object.

**6.2 SCOPE**

It’s identifying vehicles in real time video environment, by training a neural network to spot vehicles and subsequently place a bounding box over any vehicles detected in the frame. We have added in two new features for our project, one is calculating the speed of the vehicle using the relative pixel co-ordinates of the vehicles detected in subsequent frames divided by the focal length and the other is detecting the lane the vehicle is travelling on by analyzing the white lane markers on the road, relative to the vehicles position in the video.

**CHAPTER 7**

**SPEED DETECTION**

**7.1 BACKGROUND**

Speed detection has traditionally been achieved via directly measuring the distance from the camera to the vehicle in order to calibrate the speed of the vehicle across frame.[6] For speed cameras specifically, they use Doppler radar technology[15] will measures the changes of microwaves reflected from vehicles in order to obtain the speed.

Many approaches have been applied to lane detection, which can be classified as either feature-based or model-based[7], feature-based[8] methods detect lanes by low-level features like lane-mark edges[9-11]. The feature-based methods are highly dependent on clear lane marks, and suffer from weak lane-marks, noise and occlusions. Model-based methods represent lanes as a kind of curve model which can be determined by a few critical geometric parameters[12-14]

**7.2 GOALS**

My goal has been defined as follows:

• Speed Detection: Detecting the speed of the identified vehicles (km/hr) in a video sequence.

**7.3 SCOPE**

In the project centre’s on detecting speeds of the identified vehicles in real time video environment, by using trained vehicle classifier to spot vehicles and subsequently place a bounding box over any vehicles detected in the frame and then calculating the speed of the vehicle using the relative pixel co-ordinates of the vehicles detected in subsequent frames divided by the focal length relative to the vehicles position in the video.

**CHAPTER 8**

**IMPLEMENTATION**

**8.1 PROBLEM DECOMPOSITION**

The project was decomposed into the following sub-tasks:

**A. Vehicle Detection**

In this section, Vehicles/vehicle boundaries (primarily Cars) were identified automatically using Haar Cascade Classifier specifically trained to detect cars. This involved training and optimizing the model to be specific to vehicle classification.

**B. Speed Detection**

This part mainly focused on detecting the speed of the vehicle in real time by keeping track of the relative pixel positions in every single frame and identifying the changes in positions of the cars detected by using pre-trained vehicle classifier and converting them into real object parameters. To generalize the direction in which the vehicle is progressing subjecting to change in pixel values, Manhattan distance is being calculated between the centroids of the vehicles in subsequent frames and resulting pixel value is converted from image plane to object plane.

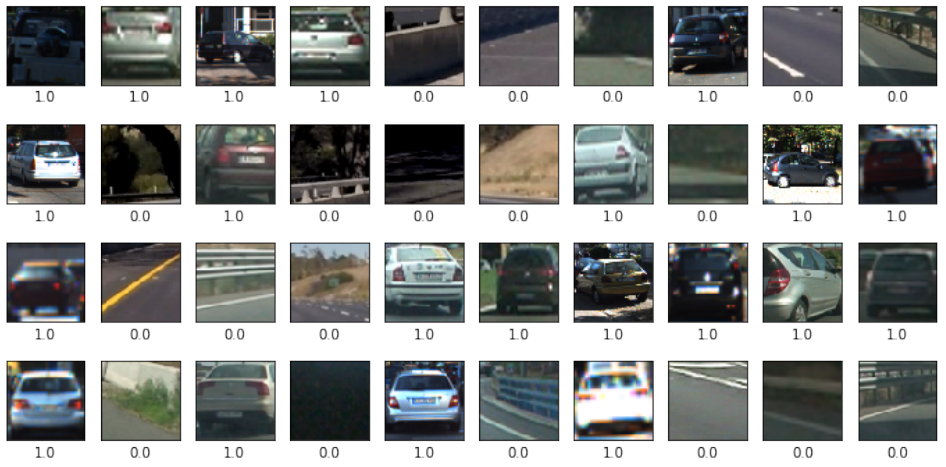
**C. Notification**

Get alert by mobile app, which I developed by Android. It just a simple app for get traffic alert over the app. It’s a radio station for that specific road. It also shows the current vehicle speed at the real time by google map and gps. Its doesnot required internet connection for the speed detection. The app download the whole graphical map inside the mobile which the user install the app at the 1st time. It also shows the current location of the vehicle.

**8.2 Design & Algorithms**

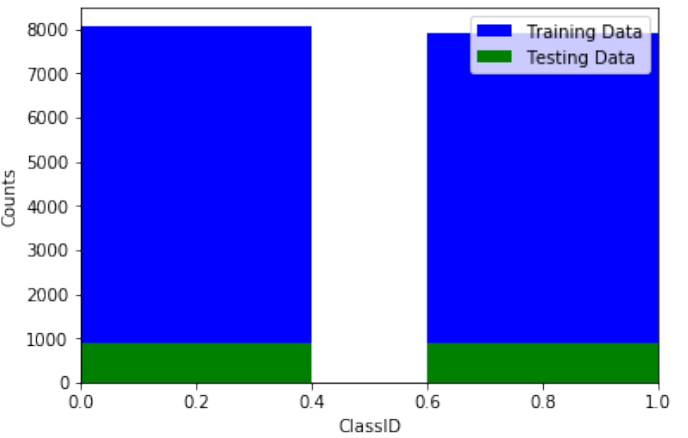
Vehicles are detected using a deep learning approach with a fully convolutions network, details of which are explained below.

The data-set is explored here. It is comprised of images taken from the GTI vehicle image database[16], the KITTI vision benchmark suite[18], and the examples were extracted from the test video itself. The data-set is labelled with two classes, cars and non-cars. Cars have a label of 1.0, whereas non-cars have a label of 0.0, as can be seen from the following figure:



A total of 17760 samples were acquired from the above mentioned data-sets, each image is colored and has a resolution of 64x64 pixels. The data-set was then further split into the training set (90%, 15984 samples) and validation set (10%, 1776 samples). From the following distribution, it can be seen that the data-set is nicely balanced, which is important for training the neural network. This balance ensures that the neural network is not biased towards either of the classes.

**The distribution looks as follows:**



The Neural Network’s architecture is explained here. This is essentially a binary classification problem, where the neural network has to decide whether the given image is a vehicle or not. The model has 10 layers excluding the input and output layers.

**The architecture of the model is as follows:**

• Input layer, where the 64x64 pixels were fed with all the 3 color channels

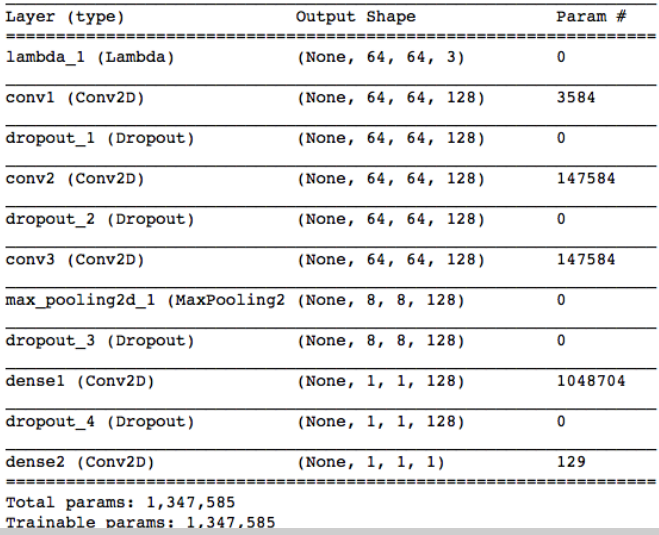
• Convolutional Layer 1 & dropout, a 2-dimensional conv layer with a 128 filter, 50% dropout and ReLU activation

• Convolutional layer 2 & dropout, a 2D conv layer with a 128 filter, 3x3 each, 50% dropout and ReLU activation.

• Convolutional layer 3, dropout & maxpool, a 2D conv layer with 8x8 max-pooling, 50% dropout and ReLU activation.

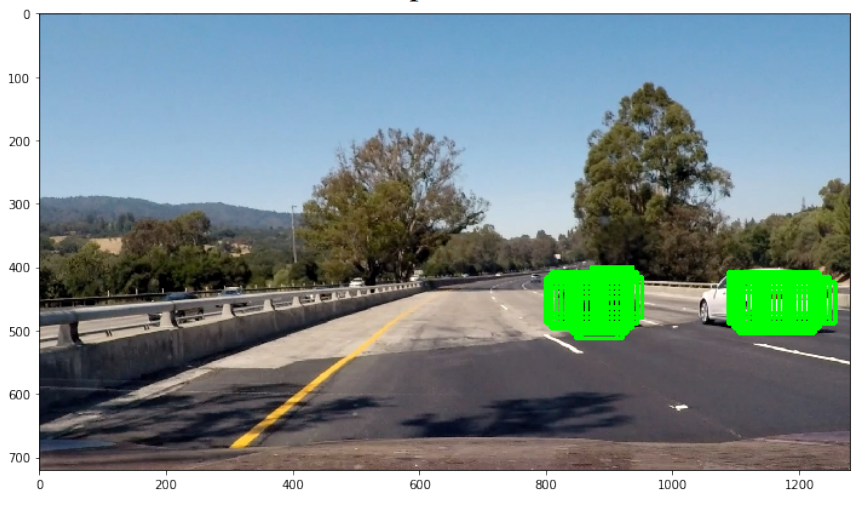
• Dense convolutional layer, 128 dense neuron layer with a ReLU activation

• Dense neuron layer, with hyperbolic tangent (tanh) activation

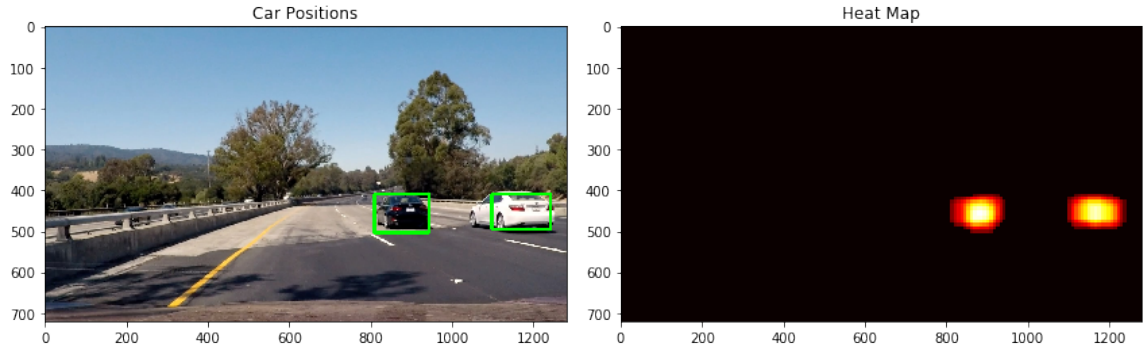
**The full Neural network architecture is summarized below:**

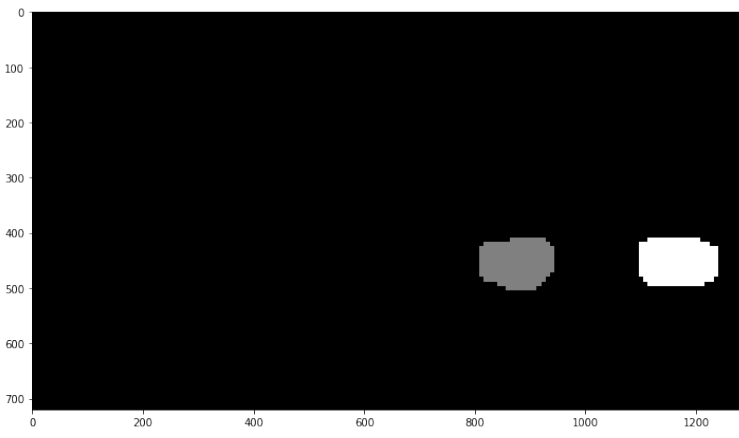
Each frame from the video was fed into the Neural Network, which has been used to detect vehicles everywhere in the frame. This network scales up to be compatible with whatever the input size is, as there is no fully-connected layer at the end, but just a conv layer with max pooling and dropout. Since there is no single-neuron output, the neural network outputs an image like a virtual heat-map.

**The bounding boxes were then drawn on the hot positions as can be seen below:**



A heatmap was created and a small threshold was added to it to avoid false positives.

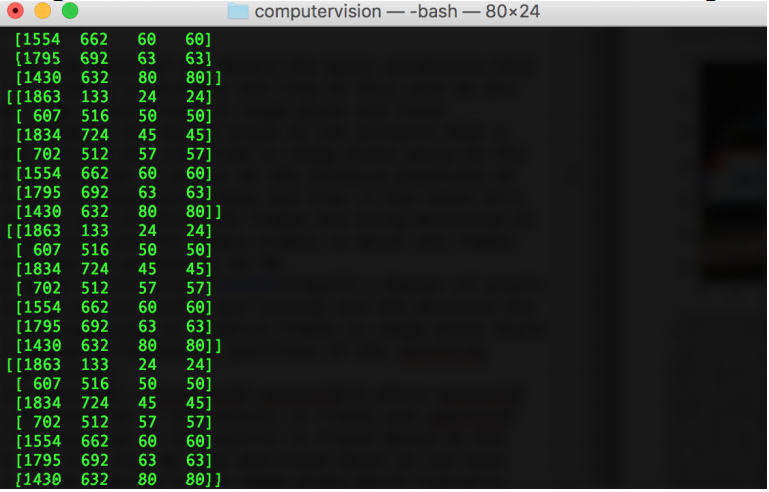




The network parameters (weights) were frozen and stored for future use, since the neural network has to be trained only once. These weights were then merged with Google TensorFlow’s pre-trained Object detection models[1] and were later used in predictions.

**8.3 SUBSEQUENT**

Once the vehicle is detected using the above trained Classifier, the subsequent part that needed to be solved was the speed of the vehicle. In order to achieve this, the video was read using Scikit video package and all video frames with vehicle being detected and bounding boxes drawn around with the help of the Cascade Classifier, were output to a folder. Subsequently, the video frames were iterated over one by one and their coordinates or absolute pixel values of the positions of the detected vehicles were obtained, using the HOG Classifier in order to track the ROI of the vehicle/s present in every frame. The ROI or the coordinates of all the cars that being tracked and calculated are shown in the below figure.



Once the regions of interest of all the vehicles in the frames were obtained, the midpoints of the bounding boxes surrounding the vehicles were obtained, which is held as the “position” of the vehicle which will be key for future frames. This process was repeated for all vehicles to create a list of midpoint values which were updated every frame.

In order to find out the speed of any object the basic parameters that will be needed are distance and time. In this project, we instead are making all calculations in the image plane and then later projecting them in the object plane. In this case, the distance a particular vehicle has travelled in the image plane would be the relative differences in pixels at various vehicle positions in subsequent frames. Time would be the rate at which the frames were being processed, which we assumed to be 30 frames per second for the purposes of this project.

From the above description, Time = Number of pixels progressed by vehicle/frames per second and

**Distance = sqrt(mid1\*mid1+mid2\*mid2)**

where mid1 and mid2 are the positions of the vehicle in frame 1 and 2 respectively.

If the difference is less than 10 pixels, then the car is assumed to be stationary and is neglected when updating values.

Once the speed of the vehicle in the image plane is calculated, we then convert the pixel values obtained into km/sec which is representative of the object plane, using the distance of the object from the camera in the given formula exact distance to ---

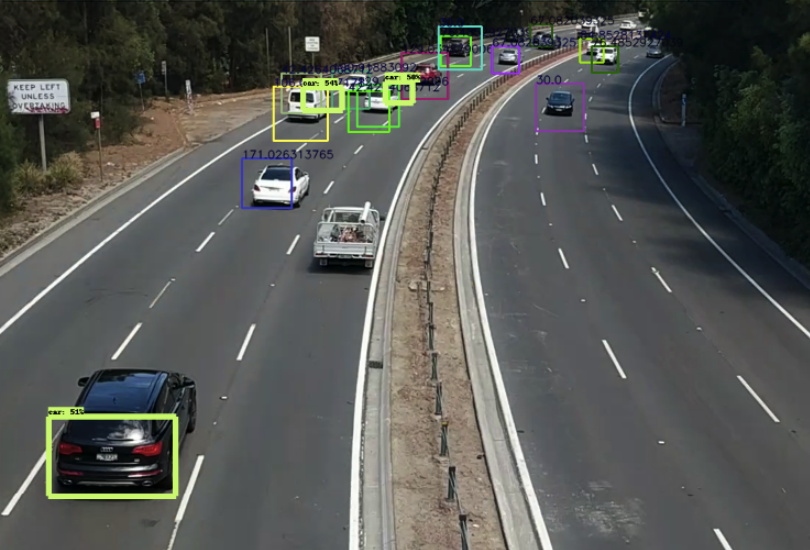
**object (mm) = focal length (mm) \* real height of the object (mm) \* image height (pixels)/object height (pixels) \* sensor height (mm)**

The calculated values are displayed in every frame is shown in below figure.



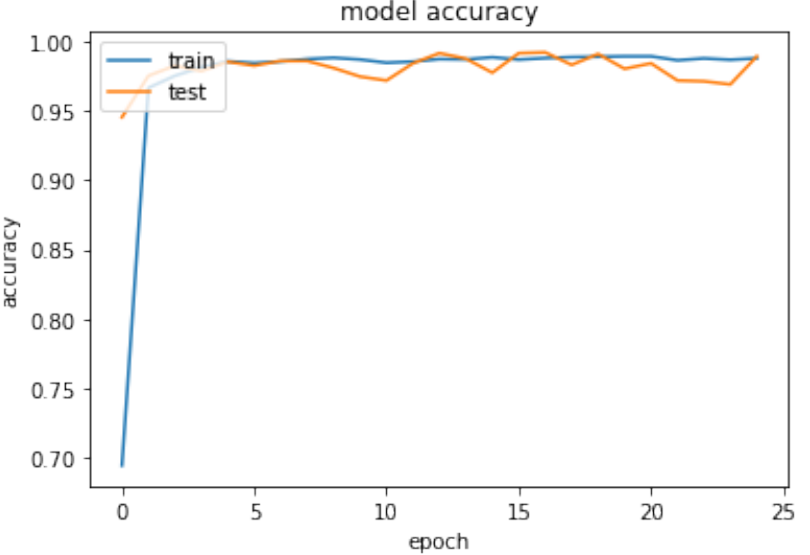
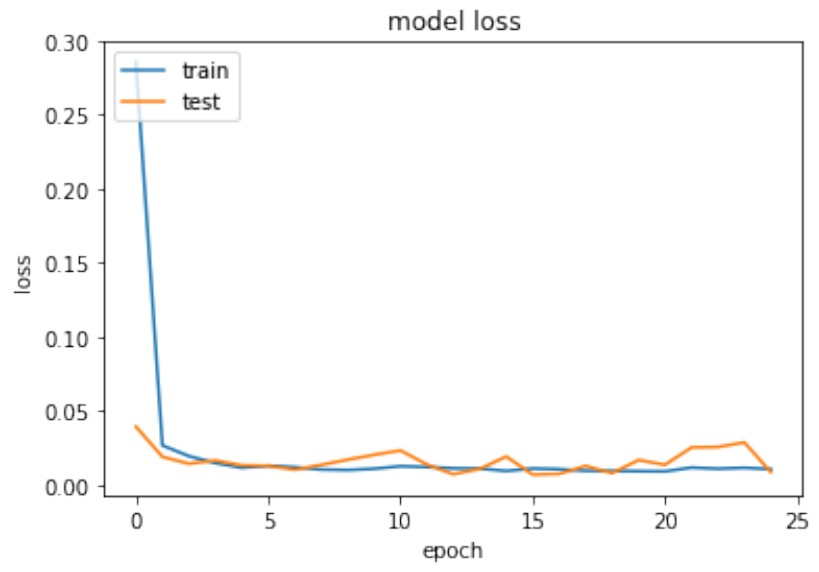
However, the trained classifier is unable to detect all the cars in every single frame and thus the obtained speed values are not always accurate and subject to deviate from original value depending on the following assumptions on the parameters. The height of the object in object plane is assumed to be 4meters in general approximating to median values for all cars which is not the same in every case. The distance between the camera and the road is approximated which is not accurate due to practical constraints and hence approximated.

**The detected cars and their speeds are best demonstrated in the below figure:**

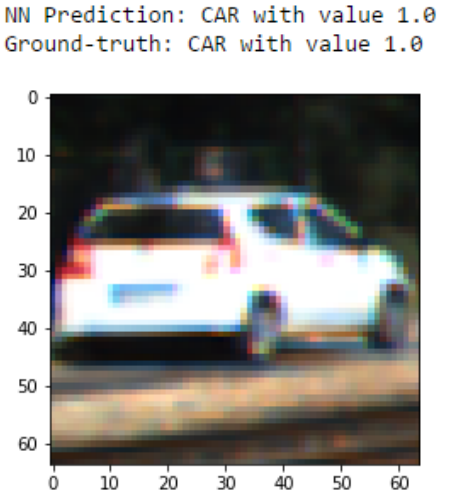


**8.4 DESIGN JUSTIFICATIONS, TESTING & RESULTS**

The Neural Network architecture was finalized because of it’s overall accuracy for the task, after training and testing various other models. The current model achieves an accuracy of approximately 99% after 20 epochs of training. The loss and accuracy over time can be seen from the figure below:



**A prediction made by the model on a random sample test image can be seen:**



**CHAPTER 9**

**SOURCE CODE STRUCTURE**

**9.1 Python Codes**

speed\_check.py

**9.2 IMPORTED LIBRARY**

import cv2

import numpy as np

import dlib

import time

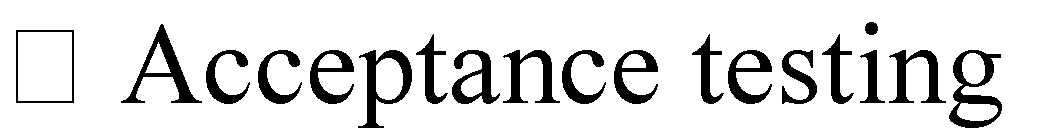
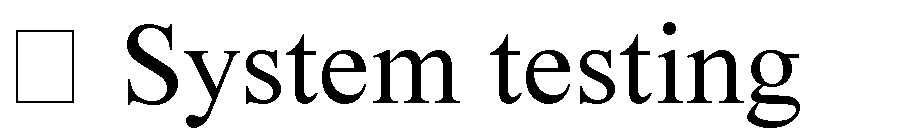
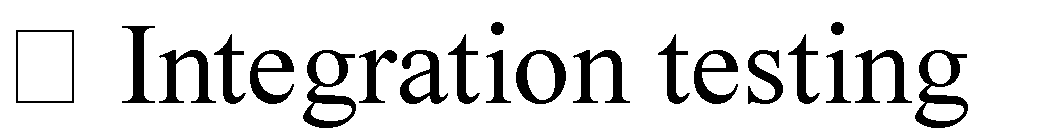
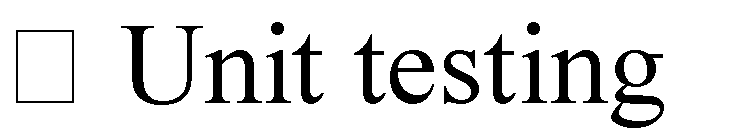
import threading

import math

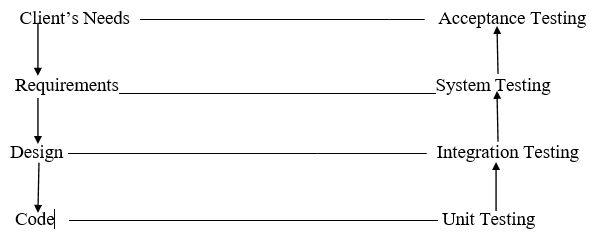
**CHAPTER 10**

**TESTING**

Testing is the process of executing the program with the intention of finding out errors. During testing, the program to be tested is executed with a set of test cases and the output of the programs for the test case is evaluated to determine if the program is performing as it is expected to be. The success of testing in revealing errors in program depends critically on the test cases. In software system the use of testing is not limited to the testing phase. The results of testing are used later on during maintenance also. During testing a test suite can be used to see that modification doesn’t have any undesirable effect the basic levels of testing are:



These different levels of testing attempt to detect different types of faults. The relation of faults introduced in different phases and the different levels of testing are as shown in bellow:



1. **Unit Testing:** The level of testing is called unit testing. In this, different modules are tested against the specifications produced during design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goal is set to test the internal logic of the modules.
2. **Integration Testing:** The next level of testing is often called the integration testing. In this, many tested modules are combined into subsystems, which are then tested. The goal here is to see if the modules can be integrated properly, the emphasis being on testing interface between modules. This testing activity can be considered as testing design, and hence the emphasis on testing modules interactions.
3. **System Testing:** During system testing, the system is used experimentally to ensure that the software doesn’t fail, i.e. it will run according to its specifications and in the way users expect, special test data input for processing, and the results examined. A limited number of users may be allowed to use the system can see whether they try to use it in unforeseen ways.
4. **Acceptance Testing:** It is sometimes performed with realistic data of the client to demonstrate that the software is working properly. Testing here focuses on the external behavior of the system. The internal logic of the program is not emphasized.
5. **Validation Check:** During testing section validations checks are made. Appropriate actions are taken after testing.

**CHAPTER 11**

**IMPLEMENTATIONS**

Implementation is the stage of the project when the theoretical design turned into a working system. At this stage the main workload, the up heal and the major impact on the existing practices shift to user department. If the implementation stage is not carefully planned and controlled, it can cause chaos. Thus, it can be considered to be the most crucial stage in achieving a new successful system and in giving the users confidence that the users confidence that the new system will work and be effective. The implementation view of software requirements presents the real worlds manifestation of processing functions and information structures. In some cases, a physical representation is developed as the first step in software design. However, most computer-based systems are specified in a manner that dictates accommodation of certain implementation details. Implementation involves careful planning, investigation of current system and constraints on implementation, design of methods to achieve the changeover, training of staff in the changeover procedures and evaluation of changeover methods. The first task is the implementation planning.

i.e. deciding the methods and time scale to be adopted. Once the planning has been completed, the major effort in the computer department is to ensure that the programs in the system are working properly. At the same time the user department must concentrate on training user staff. What the staffs have been trained, a full system test can be carried out, involving both the computer and clerical procedures. The main step of implementation includes:

>> Installing software in machine.

>> Installing the requirements in the machine.

>> Training the operational staff.

**CHAPTER 12**

**FUTURE IMPROVEMENT**

# Making more interesting the system.

# User can get more accurate result.

# Will be adding more features as like real time alert to the driver.

# Data will be stored for future research.

# Can be added more features like re-check vehicle, where any vehicle can be identify anywhere.

**CHAPTER 13**

**CONCLUSION**

In conclusion, although there were few issues with false positives and the inability to detect cars, the prototype generally performed well during the lab demonstration. This project being one of the most focused topics in real time vehicle identification systems can be extended to detect collisions. One obvious extension would be to design a program to detect if there any accidents depicted in real time video and notify emergency services if so. Another possible extension is to perhaps solely focus on lane detection, and adding features to detect abrupt lane departures.

**APPENDIX**

**Annexures**

**Car Detect**

import cv2

cascade\_src = 'cars.xml'

video\_src = 'video.avi'

cap = cv2.VideoCapture(video\_src)

car\_cascade = cv2.CascadeClassifier(cascade\_src)

while True:

ret, img = cap.read()

if (type(img) == type(None)):

break

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cars = car\_cascade.detectMultiScale(gray, 1.1, 2)

for (x,y,w,h) in cars:

cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)

cv2.imshow('video', img)

if cv2.waitKey(33) == 27:

break

cv2.destroyAllWindows()

**Bus Detect**

import cv2

cascade\_src = 'Bus\_front.xml'

video\_src = 'bus1.mp4'

cap = cv2.VideoCapture(video\_src)

car\_cascade = cv2.CascadeClassifier(cascade\_src)

while True:

ret, img = cap.read()

if (type(img) == type(None)):

break

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cars = car\_cascade.detectMultiScale(gray, 1.16, 1)

for (x,y,w,h) in cars:

cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)

cv2.imshow('video', img)

if cv2.waitKey(33) == 27:

break

cv2.destroyAllWindows()

**Bike Detect**

import cv2

cascade\_src = 'two\_wheeler.xml'

video\_src = 'two\_wheeler2.mp4'

cap = cv2.VideoCapture(video\_src)

car\_cascade = cv2.CascadeClassifier(cascade\_src)

while True:

ret, img = cap.read()

if (type(img) == type(None)):

break

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cars = car\_cascade.detectMultiScale(gray,1.01, 1)

for (x,y,w,h) in cars:

cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,215),2)

cv2.imshow('video', img)

if cv2.waitKey(33) == 27:

break

cv2.destroyAllWindows()

**SOUCE CODE**

import cv2

import dlib

import time

import threading

import math

carCascade = cv2.CascadeClassifier('myhaar.xml')

video = cv2.VideoCapture('cars.mp4')

WIDTH = 1280

HEIGHT = 720

def estimateSpeed(location1, location2):

d\_pixels = math.sqrt(math.pow(location2[0] - location1[0], 2) + math.pow(location2[1] - location1[1], 2))

ppm = 8.8

d\_meters = d\_pixels / ppm

fps = 18

speed = d\_meters \* fps \* 3.6

return speed

def trackMultipleObjects():

rectangleColor = (0, 255, 0)

frameCounter = 0

currentCarID = 0

fps = 0

carTracker = {}

carNumbers = {}

carLocation1 = {}

carLocation2 = {}

speed = [None] \* 1000

out = cv2.VideoWriter('output.avi',cv2.VideoWriter\_fourcc('M','J','P','G'), 10, (WIDTH,HEIGHT))

while True:

start\_time = time.time()

rc, image = video.read()

if type(image) == type(None):

break

image = cv2.resize(image, (WIDTH, HEIGHT))

resultImage = image.copy()

frameCounter = frameCounter + 1

carIDtoDelete = []

for carID in carTracker.keys():

trackingQuality = carTracker[carID].update(image)

if trackingQuality < 7:

carIDtoDelete.append(carID)

for carID in carIDtoDelete:

print ('Removing carID ' + str(carID) + ' from list of trackers.')

print ('Removing carID ' + str(carID) + ' previous location.')

print ('Removing carID ' + str(carID) + ' current location.')

carTracker.pop(carID, None)

carLocation1.pop(carID, None)

carLocation2.pop(carID, None)

if not (frameCounter % 10):

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

cars = carCascade.detectMultiScale(gray, 1.1, 13, 18, (24, 24))

for (\_x, \_y, \_w, \_h) in cars:

x = int(\_x)

y = int(\_y)

w = int(\_w)

h = int(\_h)

x\_bar = x + 0.5 \* w

y\_bar = y + 0.5 \* h

matchCarID = None

for carID in carTracker.keys():

trackedPosition = carTracker[carID].get\_position()

t\_x = int(trackedPosition.left())

t\_y = int(trackedPosition.top())

t\_w = int(trackedPosition.width())

t\_h = int(trackedPosition.height())

t\_x\_bar = t\_x + 0.5 \* t\_w

t\_y\_bar = t\_y + 0.5 \* t\_h

if ((t\_x <= x\_bar <= (t\_x + t\_w)) and (t\_y <= y\_bar <= (t\_y + t\_h)) and (x <= t\_x\_bar <= (x + w)) and (y <= t\_y\_bar <= (y + h))):

matchCarID = carID

if matchCarID is None:

print ('Creating new tracker ' + str(currentCarID))

tracker = dlib.correlation\_tracker()

tracker.start\_track(image, dlib.rectangle(x, y, x + w, y + h))

carTracker[currentCarID] = tracker

carLocation1[currentCarID] = [x, y, w, h]

currentCarID = currentCarID + 1

for carID in carTracker.keys():

trackedPosition = carTracker[carID].get\_position()

t\_x = int(trackedPosition.left())

t\_y = int(trackedPosition.top())

t\_w = int(trackedPosition.width())

t\_h = int(trackedPosition.height())

cv2.rectangle(resultImage, (t\_x, t\_y), (t\_x + t\_w, t\_y + t\_h), rectangleColor, 4)

carLocation2[carID] = [t\_x, t\_y, t\_w, t\_h]

end\_time = time.time()

if not (end\_time == start\_time):

fps = 1.0/(end\_time - start\_time)

for i in carLocation1.keys():

if frameCounter % 1 == 0:

[x1, y1, w1, h1] = carLocation1[i]

[x2, y2, w2, h2] = carLocation2[i]

carLocation1[i] = [x2, y2, w2, h2]

if [x1, y1, w1, h1] != [x2, y2, w2, h2]:

if (speed[i] == None or speed[i] == 0) and y1 >= 275 and y1 <= 285:

speed[i] = estimateSpeed([x1, y1, w1, h1], [x2, y2, w2, h2])

if speed[i] != None and y1 >= 180:

cv2.putText(resultImage, str(int(speed[i])) + " km/hr", (int(x1 + w1/2), int(y1-5)),cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (255, 255, 255), 2)

cv2.imshow('result', resultImage)

if cv2.waitKey(33) == 27:

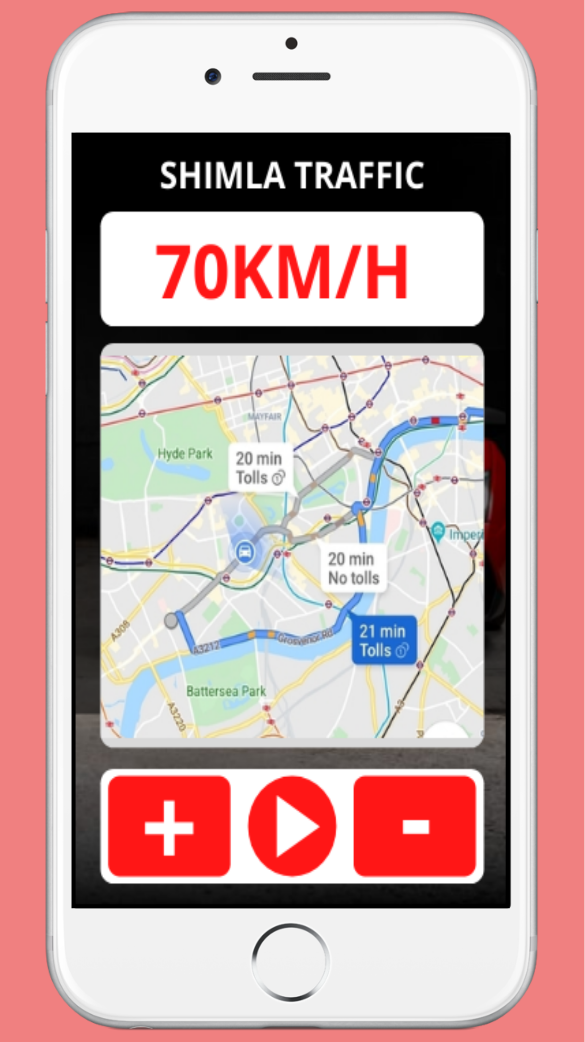
break

cv2.destroyAllWindows()

if \_\_name\_\_ == '\_\_main\_\_':

trackMultipleObjects()

**ANDROID ALERT APP OUTPUT**



**REFERENCES**

[1] Bernd Heisele, Ivaylo Riskov, and Christian R. Morgenstern, Components for Object Detection and Identification.

[2] Bileschi, S., Wolf, L.: A unified system for object detection, texture recognition, and context analysis based on the standard model feature set. In: British Machine Vision Conference (BMVC) (2005)

[3] Heisele, B., Serre, T., Pontil, M., Vetter, T., Poggio, T.: Categorization by learning and combining object parts. In: Neural Information Processing Systems (NIPS), Vancouver (2001)

[4] Poggio, T., Edelman, S.: A network that learns to recognize 3-D objects. Nature 343, 163266 (1990)

[5] Weber, M., Welling, W., Perona, P.: Towards automatic dscovery of object categories. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (June 2000)

[6] Yong-Kul Ki, Doo-Kwon Baik, ”Model for Accurate Speed Measurement Using Double-Loop Detectors”, The IEEE transactions on Vehicular Technology, vol. 55, no. 4, pp. 1094-1101, July 2006.

[7] J.C. McCall and M.M. Trivedi, Video-based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation, IEEE Transactions on Intelligent Transportation Systems, vol.7, pp.20-37, 2006.

[8] Y.Wang, E. K.Teoh and D. Shen, Lane Detection and Tracking Using B-snake, Image and Vision Computing, vol. 22, pp. 269-280, 2004.

[9] A. Broggi and S. Berte, Vision-based Road Detection in Automotive Systems: a Real-time Expectation-driven Approach, Journal of Artificial Intelligence Research, vol.3, pp. 325-348, 1995.

[10] M. Bertozzi and A. Broggi, GOLD: A Parallel Realtime Stereo Vision System for Generic Obstacle and Lane Detection, IEEE Transactions of Image Processing, pp. 62-81, 1998.

[11] S.G. Jeong, C.S. Kim, K.S. Yoon, J.N. Lee, J.I. Bae, and M.H. Lee, Realtime Lane Detection for Autonomous Navigation,IEEE Proc. Intelligent Transportation Systems, pp. 508513, 2001.

[12] Y.Wang, E.K. Teoh and D. Shen, Improved Lane Detection and Tracking Using B-snake, Image and Vision Computing, vol. 20, pp. 259-272, 2005.

[13] C. R. Jung and C. R. Kelber, A Lane Departure Warning System Using Lateral Offset with Uncalibrated Camera, Proc. IEEE Conf. on Intelligent Transportation Systems, pp.102-107, 2005.

[14] D.J. Kang, J. W. Choi and I.S. Kweon, Finding and Tracking Road Lanes Using Line-snakes, Proceedings of Conference on Intelligent Vehicle, pp. 189-194, 1996.

[15] S. M. Paing, S. S. Y. Mon, H. M. Tun, ”Design And Analysis Of Doppler Radar-Based Vehicle Speed Detection”, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 5, ISSUE 06, JUNE 2016.

[16] J. Arrspide, L. Salgado, M. Nieto, Video analysis based vehicle detection and tracking using an MCMC sampling framework, EURASIP Journal on Advances in Signal Processing, vol. 2012, Article ID 2012:2, Jan. 2012.

[17] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, ”Vision meets Robotics: The KITTI Dataset”, International Journal of Robotics Research, 2013.

[18] Google, ”Tensorflow pre-trained Object Detection Models”, https://github.com/tensorflow/models/tree/master/research/slim, 2017.

[19] Y. LeCunn, L. Bottou, Y. Bengio, P. Haffner, ”Gradient-Based Learning Applied to Document Recognition”, IEEE, 1998.

[20] A. Krizhevsky, I. Sutskever, G. E. Hinton.