

TRAFFIC RULES VIOLATION DETECTION USING DEEP LEARNING

A Major Project Work

Submitted in partial fulfilment of the requirements for the award of the
degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

K. BHAVYA	17H61A0423
P. POOJITHA	17H61A0441
P. SAI JYOTHI	17H61A0443
U. SANDEEP	17H61A0454

Under the guidance of

DR. M. SANTHOSH

Associate Professor

Department of ECE



Department of Electronics and Communication Engineering

ANURAG GROUP OF INSTITUTIONS

AUTONOMOUS

SCHOOL OF ENGINEERING

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

Venkatapur(V), Ghatkesar(M), Medchal - Malkajgiri(D)

2020-2021

ANURAG GROUP OF INSTITUTIONS

AUTONOMOUS

SCHOOL OF ENGINEERING

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad

Venkatapur(V),Ghatkesar(M), Medchal-Malkajgiri (D)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

ENGINEERING



CERTIFICATE

This is to certify that the project report entitled **Traffic rules violation detection using deep learning** being submitted by

K. Bhavya	17H61A0423
P. Poojitha	17H61A0441
P. Sai Jyothi	17H61A0443
U. Sandeep	17H61A0454

in partial fulfillment for the award of the Degree of Bachelor of Technology in Electronics & Communication Engineering to the Anurag Group of institutions, Hyderabad is a record of bonafide work carried out under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Dr. M. Santhosh
Associate Professor

Dr. S. Sathees Kumaran
Head of the Department
DEPT OF ECE

External Examiner

DECLARATION

We hereby declare that the result embodied in this project report entitled “**Traffic rules violation detection using deep learning**” is carried out by us during the academic year 2020-2021 for the partial fulfilment of the award of **Bachelor of Technology in Electronics and Communication Engineering**, from Anurag group of Institutions. We have not submitted this project report to any other Universities/Institute for the award of any degree.

K. Bhavya	17H61A0423
P. Poojitha	17H61A0441
P. Sai Jyothi	17H61A0443
U. Sandeep	17H61A0454

ACKNOWLEDGEMENT

This project is an acknowledgement to the inspiration, drive and technical assistance contributed by many individuals. This project would have never seen light of this day without the help and guidance we have received. We would like to express our gratitude to all the people behind the screen who helped us to transform an idea into a real application.

It's our privilege and pleasure to express our profound sense of gratitude to **Dr. M. Santhosh**, Associate Professor, Department of ECE for her guidance throughout this dissertation work.

We express our sincere gratitude to **Dr. S. Sathees Kumaran**, Head of Department of Electronics and Communication Engineering for his precious suggestions for the successful completion of this project. He is also a great source of inspiration to our work.

We would like to express our deep sense of gratitude to **Dr. K. S. Rao** , Director of Anurag group of Institutions for his tremendous support, encouragement and inspiration.

Lastly, we thank almighty, our parents, friends for their constant encouragement without which this assignment would not be possible. We would like to thank all the other staff members, teaching and non-teaching, who have extended their timely help and eased our work.

K. BHAVYA	17H61A0423
P. POOJITHA	17H61A0441
P.SAI JYOTHI	17H61A0443
U. SANDEEP	17H61A0454

ABSTRACT

In this covid era the duties of Traffic Police are numerous and it would be very helpful if we can make their work a little simpler on a daily basis. Rather than manually taking picture of each traffic rule violator and then fine each of them manually this project is a first step towards automating the entire process and reducing the manual efforts in the task. In this project we propose an end to end framework for capturing images, detection of violations and notifying violators, and also letting them know the amount of fine to be paid . The two violations that are being detected in this project are a) Helmet violation for two wheelers, b) Crosswalk violation for two, three and four-wheeler vehicles. In the proposed approach, we first detect vehicles using object detection which is performed using YOLO, and then accordingly each vehicle is checked against appropriate violations viz. not wearing a helmet, violation of crosswalks. Helmet violation is detected using a CNN (Convolutional neural network) based classifier. Crosswalk violation is also detected in the same way but after segmenting the images such that there is a part of the crosswalk visible in each of image. After violations are detected, vehicle numbers are obtained of respective violators using OCR, and violators are notified. Thus an end to end autonomous system will help enforcing strong regulation of traffic rules.

TABLE OF CONTENTS

Title	Page no
ABSTRACT	v
LIST OF FIGURES	ix
LIST OF TABLES	xi
CHAPTER 1: INTRODUCTION	1
1.1 Object Detection Using Yolo	2
1.2 CNN Based Classifier	3
1.3 Optical Character Recognition	4
1.4 Notifying Using Smtip	5
CHAPTER 2: LITERATURE SURVEY	6
2.1 Introduction	6
2.2 Existing System	6
2.3 Disadvantages of Existing System	6
2.4 Proposed System	7
CHAPTER 3: SOFTWARE AND HARDWARE REQUIREMENTS	8
3.1 Software Requirements	8
3.1.1 Operating System – WINDOWS 10	8
3.1.2 Kaggle	8
3.1.3 Labellmg	8
3.1.4 Anaconda 3-2021.05	8
3.1.4.1 Jupyter Notebook v6.3.0	9
3.1.4.2 Programming Language – Python 3.8	9
3.1.4.3 OpenCV -python	10
3.1.5 Yolo	10
3.1.5.1 YOLOv3	10
3.1.6 Tensor flow	11

3.1.7 OCR python	11
3.1.7.1 Tesseract v0.3.7	12
3.1.7.2 NumPy v1.20.1	12
3.1.7.3 PyWhatKit	12
3.1.7.4 secure -smtplib	13
3.1.8 Teachable Machine	13
3.2 Hardware Requirements	15
3.2.1 Laptop	15
3.2.1.1 RAM – 8GB	15
3.2.1.2 Processor – Intel Core i5	15
3.2.2 CCTV cameras	16
3.2.2.1 Traffic cameras	16
3.2.2.2 Pan Tilt Zoom cameras(PTZ)	16
CHAPTER-4: PROPOSED METHOD	17
4.1 Training the Dataset	18
4.2 Vehicle Detection	19
4.2.1 YOLO	19
4.2.1.1 YOLO V3	20
4.2.2 Working of YOLO	22
4.3 Helmet and Crosswalk Violations	24
4.3.1 Convolution Neural Networks	24
4.3.2 Helmet Classifier	26
4.3.3 Crosswalk Classifier	29

4.4 License Plate Recognition	30
4.4.1 Edge Detection	32
4.4.1.1 Concept of Canny Edge Detection	32
4.4.1.2 Contours	34
4.4.2 Ocr Pytesseract	35
CHAPTER-5: RESULTS AND DISCUSSION	36
5.1 Performance of The Model	36
5.1.1 Accuracy	36
5.1.2 Precision	36
5.2 Output Results	39
5.3 Violation Notification	43
5.4 Cost	45
5.5 Usefulness to Society and Environmental Safety	46
5.5.1 Environmental Safety	46
CHAPTER-6: CONCLUSION AND FUTURE SCOPE	47
6.1 Conclusion	47
6.2 Future Scope	47
CHAPTER-7: REFERENCES	48

LIST OF FIGURES

S.NO	NAME OF THE FIGURE	PageNo
1.1	Traditional way of regulating traffic violations.	2
3.1	Classes in Teachable Machine	14
3.2	Training model in Teachable Machine	14
1.3	Exporting the model.	16
4.1	This process has been divided into the following stages	18
4.2	Object detection and classification	20
4.3	Darknet-53	22
4.4	CNN Architecture	26
4.5	Architecture of CNN for Helmet Classification	27
4.6	Helmet classification using CNN. Input image is classified as “wearing helmet” or “not wearing helmet”	28
4.7	Crosswalk detection using CNN	30
4.8	Shows the detection of the number plate of the vehicle, then the characters are read using OCR	31
4.9	Edge Detection	33
4.10	Graph showing Hysteresis Threshold	34
4.11	Block Diagram of OCR	35
5.1	Output image showing the bounding boxes for case-1	40
5.2	Console output for case-1	40
5.3	Output image showing the bounding boxes for case-2	41

5.4	Console output for case-2.	42
5.5	Output image showing the bounding boxes for case-3	42
5.6	Console output for case-3	43
5.7	E-mail notification for helmet violation	44
5.8	E-mail notification for crosswalk violation	44
5.9	Whatsapp notification for violation	45

LIST OF TABLES

S. NO	NAME OF THE TABLE	PageNo
5.1	Results of Vehicle Detection.	37
5.2	Results of Helmet Classification	38
5.3	Results of Crosswalk Violation	38
5.4	Results of Number Plate Detection	39

CHAPTER 1

INTRODUCTION

Between 2001 and 2017, only due to road accidents, India has experienced about 20.42 lakh fatalities and 82.30 lakh people had been injured. According to the data given by the ministry, DIU has found that two-wheelers are hit the most in 2017 road accidents. Two-wheeler accidents make up 33 percent of the total deaths in road accidents. 4 two-wheeler users lose their life every hour just because of not wearing a helmet.

Two-wheeler is a highly common mode of transport, but due to less protection, there is a high risk involved. It is highly advisable for bike-riders to use a helmet to reduce the danger involved. Observing the importance of the helmet, governments have made riding a bike without a helmet a punishable offense and have instructed to prosecute the violators. However, current video surveillance technologies are passive and require substantial human assistance. Typically, these systems are inefficient due to the presence of human intervention where efficiency tends to decline over a long period of time due to human fatigue. Also, manual intervention for capturing helmet violation is inefficient because out of many two-wheeler riders violating only a single vehicle rider can be caught at a time.

One of the most neglected road signs in our city's roads is the 'stop' line. Not only do motorists brazenly cross this line, but the Corporation, too, has done little to keep them from fading. As is the case with most traffic violations today, commuters seldom realise that such a rule exists. And those who are aware, still cross the line at traffic signals. Not stopping at a pedestrian crossing will cost you a fine up to Rs. 100 for the first offence and up to Rs. 300 for subsequent offences. Yet, this seems to have been conveniently forgotten. With commuters hardly paying heed to signboards placed at eye level, it is no surprise that signs on the road are ignored. Four wheelers and two-wheeler riders violate crosswalk at signals. Manual intervention for detecting crosswalk violations turns out to be inefficient and would lead to traffic jams due to the interruption of the flow of traffic. Detecting crosswalk violations through CCTV cameras would be highly beneficial as the flow of traffic wouldn't be interrupted and multiple violations can be detected simultaneously.



Fig 1.1 Traditional way of regulating traffic violations.

In this project, we propose an end to end system for detection of traffic rules violations namely not wearing helmet and cutting the crosswalk using advanced computer vision techniques like object detection, image classification, and image segmentation on CCTV footages. Vehicles are first detected using Object Detection Algorithm (YOLO), then the detected vehicles are checked against violations. Helmet violation is captured with the help of a CNN based Image Classifier. Crosswalk violation by a vehicle is determined using Image Segmentation employing Mask R-CNN architecture. Then if any violation(s) are identified, the vehicle number of the violator is obtained using OCR and a notification is sent to the violator. The proposed methodology focuses on an end to end autonomous system right from capturing the image frames to penalizing the violator.

1.1 Object Detection Using Yolo

Object detection is a supervised machine learning problem, which means you must train your models on labeled examples. Each image in the training dataset must be accompanied with a file that includes the boundaries and classes of the objects it contains. There are several open-source tools that create object detection annotations.

Object detection is a computer vision task that involves both localizing one or more objects within an image and classifying each object in the image.

It is a challenging computer vision task that requires both successful object localization in order to locate and draw a bounding box around each object in an image, and object classification to predict the correct class of object that was localized.

The “*You Only Look Once*,” or YOLO, family of models are a series of end-to-end deep learning models designed for fast object detection, developed by Joseph Redmon, et al. and first described in the 2015 paper titled “You Only Look Once: Unified, Real-Time Object Detection.”

The approach involves a single deep convolutional neural network (originally a version of GoogLeNet, later updated and called DarkNet based on VGG) that splits the input into a grid of cells and each cell directly predicts a bounding box and object classification. The result is a large number of candidate bounding boxes that are consolidated into a final prediction by a post-processing step.

There are three main variations of the approach, at the time of writing; they are YOLOv1, YOLOv2, and YOLOv3. The first version proposed the general architecture, whereas the second version refined the design and made use of predefined anchor boxes to improve bounding box proposal, and version three further refined the model architecture and training process.

1.2 CNN Based Classifier

CNNs have broken the mold and ascended the throne to become the state-of-the-art computer vision technique. Among the different types of neural networks (others include recurrent neural networks (RNN), long short term memory (LSTM), artificial neural networks (ANN), etc.), CNNs are easily the most popular.

These convolutional neural network models are ubiquitous in the image data space. They work phenomenally well on computer vision tasks like image classification, object detection, image recognition, etc.

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

1.3 Optical Character Recognition

Optical character recognition or optical character reader (OCR) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example: from a television broadcast).

Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision

1. **Number Plate Detection.** This problem can be tackled using the Object Detection approach where we need to train our model using the car/other vehicle images with number plates.

2. **Extracting text from the detected Number Plate.** This problem can be solved using OCR(Optical Character Recognition) which can be helpful in extracting alphanumeric characters from cropped Number Plate images.

1.4 Notifying Using Smtplib

The violators after being detected are notified using the SMTP library to send a mail about the challan details. The Simple Mail Transfer Protocol (SMTP) is an internet standard communication protocol for electronic mail transmission. Mail servers and other message transfer agents use SMTP to send and receive mail messages. User-level email clients typically use SMTP only for sending messages to a mail server for relaying, and typically submit outgoing email to the mail server on port 587 or 465 per RFC 8314. For retrieving messages, IMAP and POP3 are standard, but proprietary servers also often implement proprietary protocols, e.g., Exchange ActiveSync.

The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon. For details of SMTP and ESMTP operation, consult RFC 821 (Simple Mail Transfer Protocol) and RFC 1869 (SMTP Service Extensions).

```
class smtplib.SMTP(host="", port=0, local_hostname=None, [timeout, ]source_address=
None)
```


CHAPTER-2

LITERATURE SURVEY

To make the project successful, we have gone through many real time projects but future scope in this project where there can be drastic implementation of new technology in the detection of traffic rules violations to improve the accuracy and speed of detecting violations as per the increase in the demand which have guided us to build our project.

2.1 Introduction

“TRAFFIC RULES VIOLATION DETECTION USING DEEP LEARNING” implements the emerging advanced technology. This project is implemented using machine learning by training the model and then giving the images as input to the model. For the given input the images are classified into different categories such as two-wheeler, with helmet, without helmet and violating the cross walk.

2.2 Existing System

There are many such projects which are used to identify the traffic rules violations through the given input image. But there are many projects, which are also producing a high rate of false-negative results, so the existing projects failed to come under implementation. So, we proposed a project called “TRAFFIC RULES VIOLATION DETECTION USING DEEP LEARNING”.

2.3 Disadvantages of Existing System

Here in the present model, the accuracy of the model is less and there may be some false output which may lead to punish the innocent.

2.4 Proposed System

These problems can be solved using present technologies. A deep learning model is proposed for classification and detection of traffic rules violations. This model is trained with multiple images, which are not in a regular form. The crucial finding shows us that most of the detected violations are false. Therefore, more accurate methods for the detection are needed.

Several researches have been done in this domain to detect the traffic rules violations for different types of violations.

- J. Chiverton has proposed “Helmet presence classification with motorcycle detection and tracking,” Intelligent Transport Systems (IET), which enhanced the importance of wearing helmet for a motorcycle rider.
- Gomathi, et al, developed “Automatic Detection of Motorcycle without helmet using IOT”, which automatically detects the vehicles without helmet and warns the violators.
- Samir Ibadov, et al, modeled the “Algorithm for detecting violations of traffic rules based on computer vision approaches” which detected different violations.
- Amey Narkhede, et al, came up with detecting the number plate along with the violations named as “Automatic Traffic Rule Violation Detection and Number Plate Recognition”.
- Kunal, Dahiya; Dinesh, Singh; and C. Krishna, Mohan developed “Automatic Detection of Bike-riders without Helmet using Surveillance Videos in Real-time” which detects the violations in real time from the CCTV’s fixed on roads.
- Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, have done the traffic rules violations detection using “You Only Look Once: Unified, Real-Time Object Detection”.
- Pooya Sagharichi Ha, Mojtaba Shakeri, used “License Plate Automatic Recognition based on Edge Detection” to detect the license plate number of the violated vehicles.

CHAPTER – 3

SOFTWARE AND HARDWARE REQUIREMENTS

3.1 Software Requirements

3.1.1 Operating System – WINDOWS 10

Windows 10 is a Microsoft operating system for personal computers, tablets, embedded devices and internet of things devices. We did this project on a laptop with the WINDOWS 10 operating system.

3.1.2 Kaggle

Kaggle is the world's largest data science community with powerful tools and resources to help you achieve your data science goals. Kaggle offers a no-setup, customizable, Jupyter Notebooks environment. Access free GPUs and a huge repository of community published data & code. We have downloaded few data sets of images violating traffic rules and the other images were collected by us, each of us has gone on roads and took few images violating the traffic rules to train the model.

3.1.3 LabelImg

LabelImg is a free, open-source tool for graphically labelling images. It's written in Python and uses QT for its graphical interface. It's an easy, free way to label a few hundred images to try out your next project.

LabelImg does require a little bit of technical awareness (like using the command line) to get going. LabelImg supports labelling in VOC XML or YOLO text file format.

3.1.4 Anaconda 3-2021.05

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was

founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as **Anaconda Distribution** or **Anaconda Individual Edition**, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

Package versions in Anaconda are managed by the package management system *conda*. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called **Miniconda**, which includes only *conda*, Python, the packages they depend on, and a small number of other packages.

3.1.4.1 Jupyter Notebook v6.3.0

Jupyter is a free, open-source, interactive web tool known as a computational notebook, which researchers can use to combine software code, computational output, explanatory text and multimedia resources in a single document. Computational notebooks have been around for decades, but Jupyter in particular has exploded in popularity over the past couple of years. This rapid uptake has been aided by an enthusiastic community of user-developers and a redesigned architecture that allows the notebook to speak dozens of programming languages — a fact reflected in its name, which was inspired, according to co-founder Fernando Pérez, by the programming languages Julia (Ju), Python (Py) and R.

3.1.4.2 Programming Language – Python 3.8

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

3.1.4.3 OpenCV -python

OpenCV (*Open-Source Computer Vision Library*) 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 (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations.

3.1.5 Yolo

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals. Object detection is a phenomenon in computer vision that involves the detection of various objects in digital images or videos. Some of the objects detected include people, cars, chairs, stones, buildings, and animals.

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals. Object detection consists of various approaches such as fast R-CNN, Retina-Net, and Single-Shot MultiBox Detector (SSD). Although these approaches have solved the challenges of data limitation and modeling in object detection, they are not able to detect objects in a single algorithm run. YOLO algorithm has gained popularity because of its superior performance over the aforementioned object detection techniques.

3.1.5.1 YOLOv3

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. Versions 1-3 of YOLO were created by Joseph Redmon and Ali Farhadi.

The first version of YOLO was created in 2016, and version 3, which is discussed extensively in this article, was made two years later in 2018. YOLO is implemented using the Keras or OpenCV deep learning libraries.

Object classification systems are used by Artificial Intelligence (AI) programs to perceive specific objects in a class as subjects of interest. The systems sort objects in images into groups where objects with similar characteristics are placed together, while others are neglected unless programmed to do otherwise.

3.1.6 Tensor flow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.

3.1.7 OCR python

Optical character recognition or optical character reader (OCR) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example: from a television broadcast).

Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive

computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

OCR consists of few libraries that are:

3.1.7.1 Tesseract v0.3.7

Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License. Originally developed by Hewlett-Packard as proprietary software in the 1980s, it was released as open source in 2005 and development has been sponsored by Google since 2006.

The initial versions of Tesseract could only recognize English-language text. Tesseract v2 added six additional Western languages (French, Italian, German, Spanish, Brazilian Portuguese, Dutch). Version 3 extended language support significantly to include ideographic (Chinese & Japanese) and right-to-left (e.g., Arabic, Hebrew) languages, as well as many more scripts

3.1.7.2 NumPy v1.20.1

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. NumPy stands for Numerical Python.

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

3.1.7.3 PyWhatKit

PyWhatKit is a Python library with various helpful features. It is an easy-to-use library which does not requires you to do some additional setup.

Python offers numerous inbuilt libraries to ease our work. Among them pywhatkit is a Python library for sending WhatsApp messages at a certain time, it has several other features too.

Following are some features of pywhatkit module:

- 1.Send WhatsApp messages.
- 2.Play a YouTube video.
- 3.Perform a Google Search.
- 4.Get information on particular topic.

3.1.7.4 secure -smtplib

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers. Python provides **smtplib** module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

An SMTP object has an instance method called **sendmail**, which is typically used to do the work of mailing a message. It takes three parameters –

- The *sender* – A string with the address of the sender.
- The *receivers* – A list of strings, one for each recipient.
- The *message* – A message as a string formatted as specified in the various RFCs.

3.1.8 Teachable Machine

Teachable Machine is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone. You train a computer to recognize your images, sounds, and poses without writing any machine learning code. Then, use your model in your own projects, sites, apps, and more.

Teachable Machine uses TensorFlow.js, a library for machine learning in Javascript, to train and run the models you make in your web browser.

Teachable Machine is an experiment from **Google** to bring a no-code and low-code approach to training AI models. Anyone with a modern browser and webcam can quickly train a model with no prior knowledge or experience with AI.

1. Gather

Gather and group your examples into classes, or categories, that you want the computer to learn.

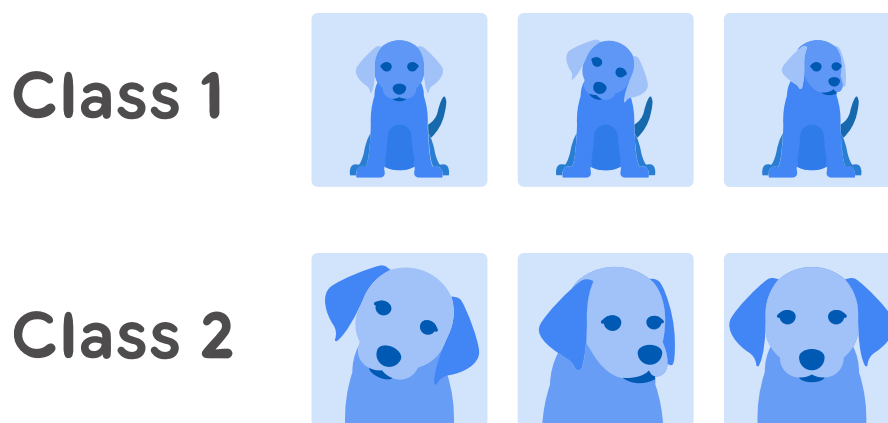


Figure 3.1 Classes in Teachable Machine

2. Train

Train your model, then instantly test it out to see whether it can correctly classify new examples.



Figure 3.2 Training model in Teachable Machine

3.Export

Export your model for your projects: sites, apps, and more.



Figure 2.3 Exporting the model.

3.2 Hardware Requirements

3.2.1 Laptop

We have used a DELL Laptop to perform our Project. For Machine learning, we require nothing expensive or latest as long as we have lots of RAM and a fast CPU while for deep learning, a GPU is required in addition to RAM and CPU.

3.2.1.1 RAM – 8GB

We have used a laptop with 8GB RAM. Generally, the larger the RAM the higher the amount of data it can handle, hence faster processing. With larger RAM we can use your machine to perform other tasks as the model trains. Although a minimum of 8GB RAM can do the job, **16GB RAM** and above is recommended for most deep learning tasks.

3.2.1.2 Processor – Intel Core i5

We have used a laptop that has an Intel Core i5 processor for this project. It is developed and manufactured by Intel, the Core i5 is a dual-core computer processor, available for use in both desktop and laptop computers. It is one of three types of processors in the "i" series (also called the Intel Core family of processors).

3.2.2 CCTV cameras

CCTV cameras play an important part in **road** network management. They are installed at sensitive locations on the network to support traffic management, where congestion and traffic queues are frequent and at other locations where there is an increased risk of accidents, traffic incidents and to monitor traffic rules violations.

3.2.2.1 Traffic cameras

Traffic cameras are an innovative and extremely functional use of video surveillance technology. You've seen their footage during traffic reports on the TV news. They're atop traffic signals and placed along busy roads, and at busy intersections of the highway. Whether they're recording traffic patterns for future study and observation or monitoring traffic and issuing tickets for moving violations, traffic cameras are an explosively popular form of video surveillance.

3.2.2.2 Pan Tilt Zoom cameras(PTZ)

Pan, Tilt Zoom (PTZ) cameras are commonly used for:

- elevated motorways in built-up areas
- exposed motorway bridges or roads at high elevations
- urban motorways with tidal flow and / or reduced lane width
- locations where congestion frequently occurs and queues exceed 1km

Either fixed or PTZ cameras can be used:

- in road tunnels and on the approaches to those tunnels
- at the termination of a motorway, at busy interchanges and places where there are lane reductions.

CHAPTER-4

PROPOSED METHOD

In this section, we will explain the proposed end to end system. This system consists of four major components. The system architecture is shown in figure 3.1. In the proposed system we first give an image frame from CCTV footage as input to the system then perform vehicle detection using Object Detection. Object Detection is carried out using YOLO (you look only once) for detecting vehicles in the image frame. After detecting vehicles, individual vehicles are cropped out using the coordinates obtained from bounding boxes given by Object Detection Algorithms. Now an individual vehicle is checked against different violations. Violations included in this proposed system are Helmet Violation (Two wheeler rider not wearing a helmet) and Crosswalk Violation (Violating Zebra Crossing). Two-wheeler vehicles will be checked against Helmet and Crosswalk Violation and Four Wheeler vehicles will be checked against Crosswalk Violation. Helmet violation and Crosswalk violation is detected using CNN (Convolutional Neural Network) based classifier which works well on visual data. Once violation(s) are detected for a vehicle then the number plate of the corresponding vehicle is detected using Object Detection. Again YOLO is used for detection of the number plate of a vehicle. OCR (Optical Character Recognition) is used to obtain license number from the number plate. Vehicle users are notified with associated violations and violations are inserted into the database. The database can be used to obtain statistical analysis on traffic rules violations that previously occurred. Now we will look at each individual module in detail.

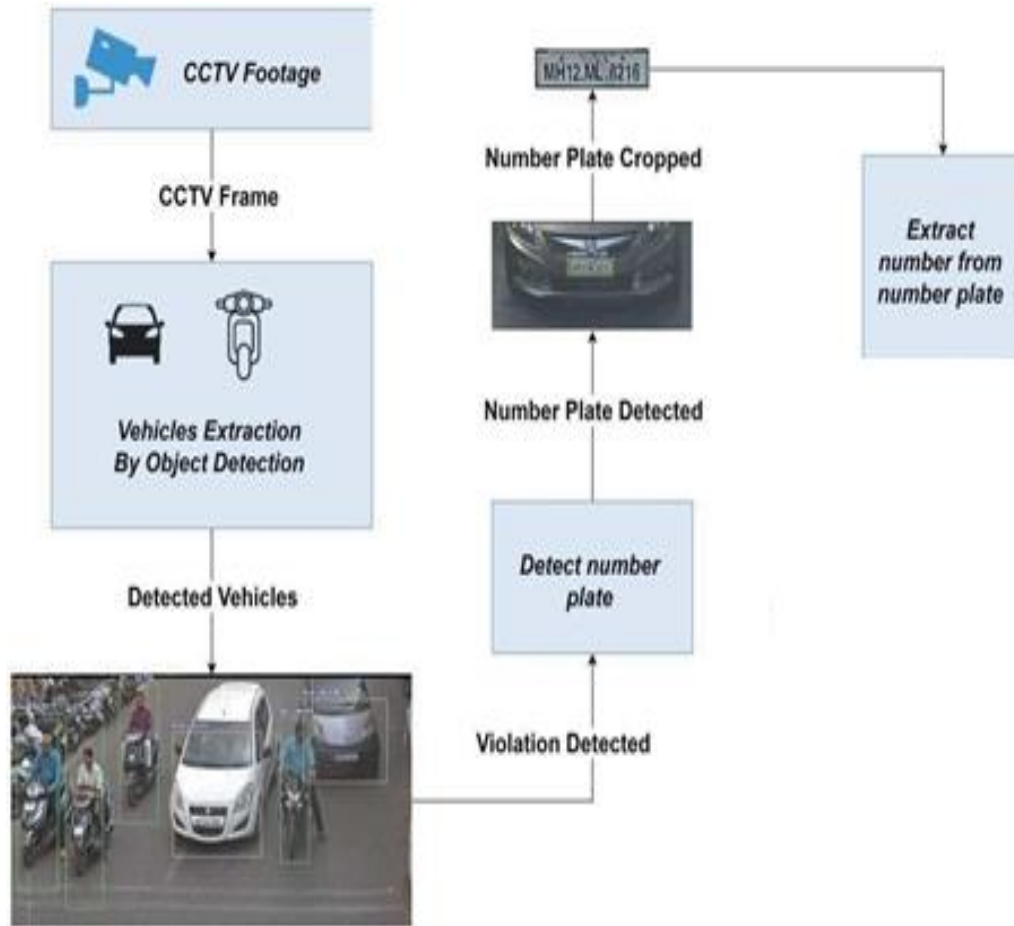


Fig 4.1 This process has been divided into the following stages:

4.1 Training the Dataset:

We have collected data from various sources such as Kaggle and Google. We also collected few images by clicking pictures at the zebra crossing when the traffic was at a halt near the junction signal.

After gathering all the images, we have divided them into different classes where in each class consists of approximately 300 images for both helmet and crosswalk violations. To train the data we have decided to implement transfer learning to obtain maximum accuracy. In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.

To implement Transfer Learning we have used Google's Teachable Machine which is a web-based resource for training and developing ML models for image classification, sound classification, and pose classification for full-body poses, we managed to train our model accordingly. The resource allows you to either export your model for use in your app or publish it online where Google hosts the model for free and provides a URL, we have decided to export these models into our project as .h5 files. Once you have the weight file in .h5 format, the next step is to develop a script that will take an image and use the model to predict the class the image belongs to.

```
import tensorflow.keras  
  
from PIL import Image, ImageOps  
  
import numpy as np
```

These are the few libraries that we need to install to implement Teachable Machine in our project.

4.2 Vehicle Detection

After obtaining the image, the frame is passed as an input to the object detection module for vehicle detection. We are using YOLO (You Only Look Once) for this purpose. YOLO is an object detection algorithm. YOLO predicts the coordinates of bounding boxes directly using fully connected layers on top of the convolutional feature extractor. Predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn. YOLO is better than other object detection algorithms like R-CNN, fast R-CNN, and faster R-CNN as they use pipelines to perform the detection of objects which incorporates multiple steps. Hence these algorithms are slow to run and hard to optimize as each individual component should be trained separately.

4.2.1 YOLO

You only look once, or YOLO, is one of the faster object detection algorithms out there. Though it is no longer the most accurate object detection algorithm, it is a very good

choice when you need real-time detection, without loss of too much accuracy. YOLO algorithm divides any given input image into $S \times S$ grid system. Each grid on the input image is responsible for detection on object. Now the grid cell predicts the number of boundary boxes for an object.

For every boundary box has five elements (x, y, w, h, confidence score). X and y are the coordinates of the object in the input image, w and h are the width and height of the object respectively. Confidence score is the probability that box contains an object and how accurate is the boundary box.

YOLO algorithm:

It is based on regression where object detection and localization and classification the object for the input image will take place in a single go. This type of algorithms is commonly used real-time object detection.

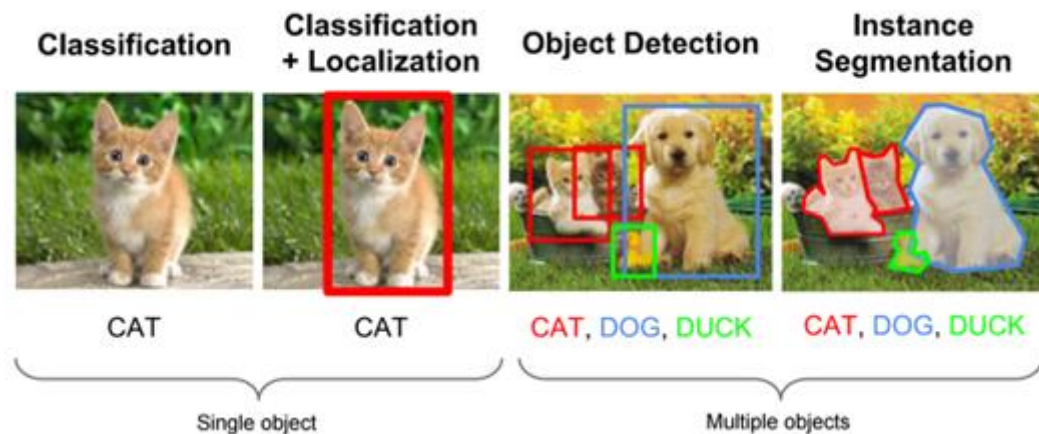


Fig 4.2 Object detection and classification

4.2.1.1 YOLO V3

The previous version has been improved for an incremental improvement which is now called YOLO v3. As many object detection algorithms are been there for a while now the competition is all about how accurate and quickly objects are detected. YOLO v3 has all we need for object detection in real-time with accurately and classifying the objects. The authors named this as an incremental improvement.

Here we will have look what are the so called Incremental improvements in YOLO v3

Bounding Box Predictions: In YOLO v3 gives the score for the objects for each bounding boxes. It uses logistic regression to predict the objectiveness score.

Class Predictions: In YOLO v3 it uses logistic classifiers for every class instead of softmax which has been used in the previous YOLO v2. By doing so in YOLO v3 we can have multi-label classification. With softmax layer if the network is trained for both a person and man, it gives the probability between person and man let's say 0.4 and 0.47. With the independent classifier gives the probability for each class of objects. For example if the network is trained for person and a man it would give the probability of 0.85 to person and 0.8 for the man and label the object in the picture as both man and person.

Feature Pyramid Networks (FPN): YOLO v3 makes predictions similar to the FPN where 3 predictions are made for every location the input image and features are extracted from each prediction. By doing so YOLO v3 has the better ability at different scales. As explained from the paper by each prediction is composed with boundary box, objectness and 80 class scores. Doing upsampling from previous layers allows getting meaning full semantic information and finer-grained information from earlier feature map. Now, adding few more convolutional layers to process improves the output .

Darknet-53: the predecessor YOLO v2 used Darknet-19 as feature extractor and YOLO v3 uses the Darknet-53 network for feature extractor which has 53 convolutional layers. It is much deeper than the YOL v2 and also had shortcut connections. [6]. Darknet-53 composes of the mainly with 3x3 and 1x1 filters with shortcut connections.

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

Fig 4.3 Darknet-53

Object detection reduces the human efforts in many fields. Object detection in real-time and accurately is one of the major criteria in the world where self-driving cars are becoming a reality. There is a lot of scope for the improvements in the object detection algorithms such as YOLO v3, faster R-CNN, SSD and many. Slightest improvements in these algorithms can change entire perception in real world.

4.2.2 Working of YOLO

Here we are loading the YoloV3 weights and configuration file with the help of **dnn** module of OpenCV. The **coco.names** file contains the names of the different objects that our model has been trained to identify. We store them in a list called **classes**. Now to run a forward pass using the **cv2.dnn** module, we need to pass in the names of layers for which the output is to be computed. **net.getUnconnectedOutLayers()** returns the indices of the output layers of the network.

Coco names is an image folder that comes with a list of upto 80 datasets. Using this list we detect the vehicles whether it is a bike or a car.

To correctly predict the objects with deep neural networks, we need to preprocess our data and `cv2.dnn` module provides us with two functions for this purpose: `blobFromImage` and `blobFromImages`. These functions perform scaling, mean subtraction and channel swap which is optional. We will use `blobFromImage` in a function called `detect_objects()` that accepts image/frame from video or webcam stream, model and output layers as parameters.

We have used the scalefactor of 0.00392 which can also be written as $1/255$. Hence, we are scaling the image pixels to the range of 0 to 1. There is no need for mean subtraction and that's why we set it to `[0, 0, 0]` value.

The `forward()` function of `cv2.dnn` module returns a nested list containing information about all the detected objects which includes the x and y coordinates of the centre of the object detected, height and width of the bounding box, confidence and scores for all the classes of objects listed in `coco.names`. The class with the highest score is considered to be the predicted class.

In `get_box_dimensions()` function, a list called *scores* is created which stores the confidence corresponding to each object. We then identify the index of class with highest confidence/score using `np.argmax()`. We can get the name of the class corresponding to the index from the *classes* list we created in `load_yolo()`.

We have selected all the predicted bounding boxes with the confidence of more than 70 %. You may play around with this value.

Now that we have the vertices of the predicted bounding box and `class_id` (index of predicted object class), we need to draw the bounding box and add object label to it. We will do that with the help of `draw_labels()` function.

Although we removed the low confidence bounding boxes, there is a possibility that we will still have duplicate detections around an object. You may observe that some objects have been detected multiple times and we have more than one bounding box for it. To fix this situation we'll need to apply Non-Maximum Suppression (NMS), also called Non-Maxima Suppression.

We pass in confidence threshold value and NMS threshold value as parameters to select one bounding box. From the range of 0 to 1, we should select an intermediate value like 0.4 or 0.5 to make sure that we detect the overlapping objects but do not end up getting multiple bounding boxes for the same object.

After the objects are detected and classified we send all the motorcycle image frames to the helmet classifier and all motorcycle and car image frames to the crosswalk classifier.

4.3 Helmet and Crosswalk Violations:

After vehicles are detected from images, Two-wheelers are considered for the Helmet Classification task and all the vehicles are considered for Crosswalk Classification . Individual vehicles are cropped from the cctv frame with the help of corresponding bounding box, if the vehicle belongs to two wheeler class they are further cropped where we consider the upper half of the individual vehicle (upper 50% of vehicle frame). Likewise if the vehicle belongs to crosswalk class we make sure that a part of the lane is added in each bounding box and that is how the classifier identifies the lane violation.

4.3.1 Convolution Neural Networks:

Convolutional neural network (CNN) is network architecture for deep learning. CNN are deep artificial neural networks that are used primarily to classify images cluster them by similarity and perform object recognition within scenes. A CNN is comprised of one or more convolutional layers and then followed by one or more fully connected layers as in a standard multilayer neural network .It learns directly from images. A CNN can be trained to do image analysis tasks including classification, object detection, segmentation and image processing. CNN Architecture. CNN is made of several types of layer, like Convolutional Layer, Non-Linearity Layer, Rectification Layer, Rectified Linear Units (ReLU), Pooling Layer, Fully Connected Layer, Dropout Layer.

- **Convolutional Layer:** The main task of the convolutional layer is to detect local conjunctions of features from the previous layer and mapping their appearance to a feature map.
- **Non-Linearity Layer:** A non-linearity layer in a convolutional neural network consists of an activation function that takes the feature map generated by the convolutional layer and creates the activation map as its output.
- **Rectification Layer:** A rectification layer in a convolutional neural network performs element-wise absolute value operation on the input volume.
- **Rectified Linear Units (ReLU):** The rectified linear units (ReLUs) are a special implementation that combines non-linearity and rectification layers in convolutional neural networks.
- **Pooling Layer:** The pooling or down sampling layer is responsible for reducing the spatial size of the activation maps.
- **Fully Connected Layer:** The fully connected layers in a convolutional network are practically a multilayer perceptron (generally a two or three layer MLP) that aims to map the $m_1(l-1) \times m_2(l-1) \times m_3(l-1)$ activation volume from the combination of previous different layers into a class probability distribution.
- **Dropout Layer:** Dropout is a technique used to improve over-fit on neural networks, you should use Dropout along with other techniques like L2 Regularization.
- **Softmax:** Softmax function calculates the probabilities distribution of the event over 'n' different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs. In building neural networks softmax functions used in different layer level.

CNN

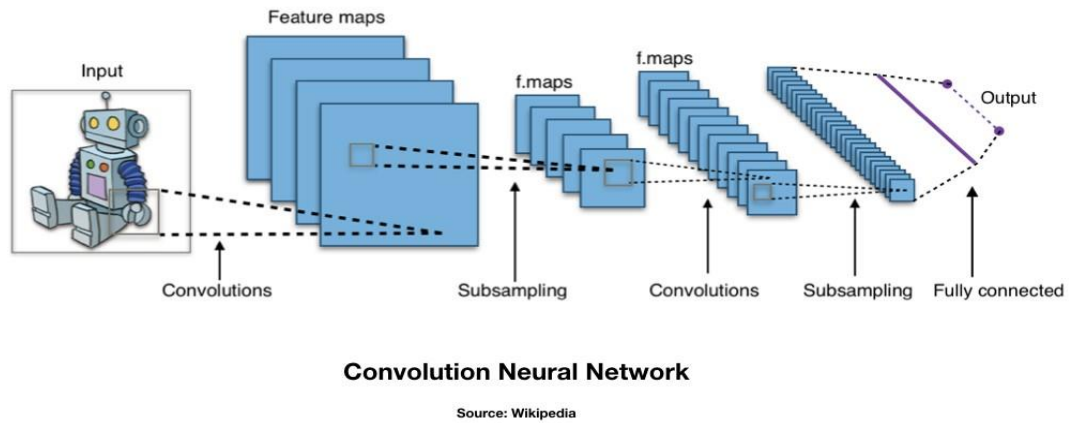


Fig 4.4 CNN Architecture

4.3.2 Helmet Classifier

Now the upper half is considered as statistically, heads (helmets) are located in the upper half of the image. By considering the upper half of the image computational complexity of the system could be reduced. This upper half is given as input to the classifier, which outputs whether the image belongs to helmet or non-helmet class. Fig depicts the flowchart for helmet classification.

The classifier used for helmet classification employs a CNN architecture as shown in fig. It is a 12 layered architecture that includes various layers such as 5 Convolutional layers using ReLu as an activation unit, 4 pooling layers, and a single dense layer using softmax for classification into two classes. CNN is preferred over other methods as it is better at extracting visual features from image data. Fig. represents the feature maps that are generated from output of convolutional layer.

These feature maps illustrate that the CNN learns common hidden features and structures among helmets and heads in the training set while training, thus being able to distinguish between helmet and non-helmet class.

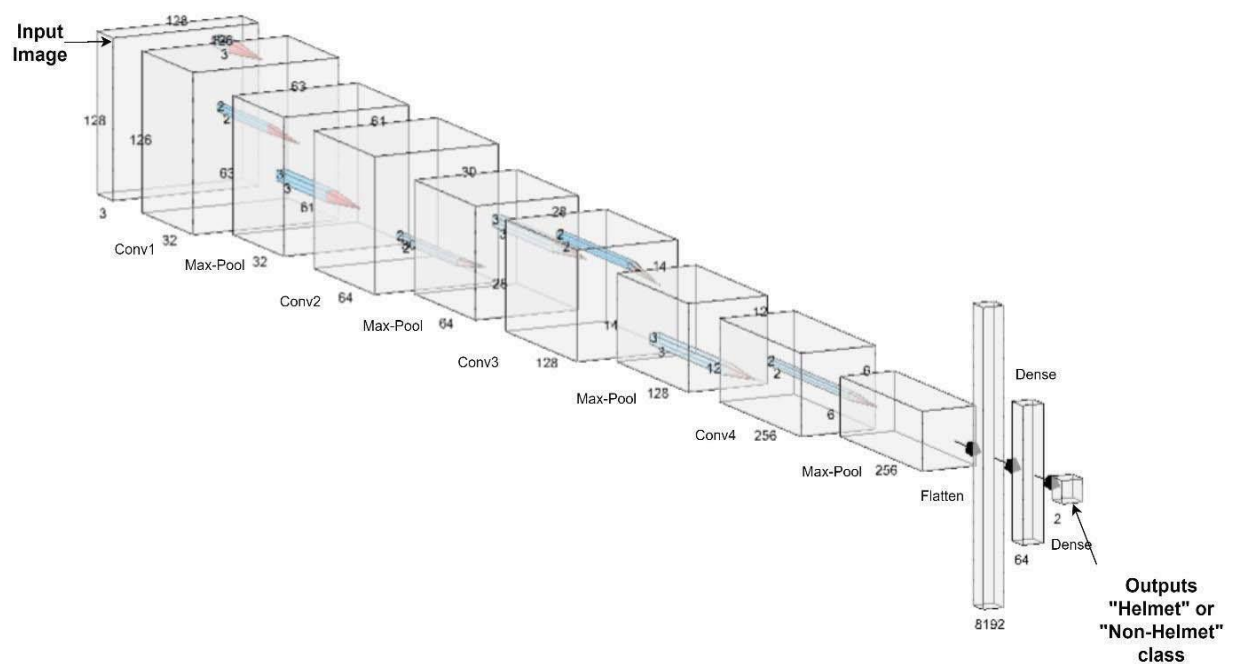


Fig4.5 Architecture of CNN for Helmet Classification

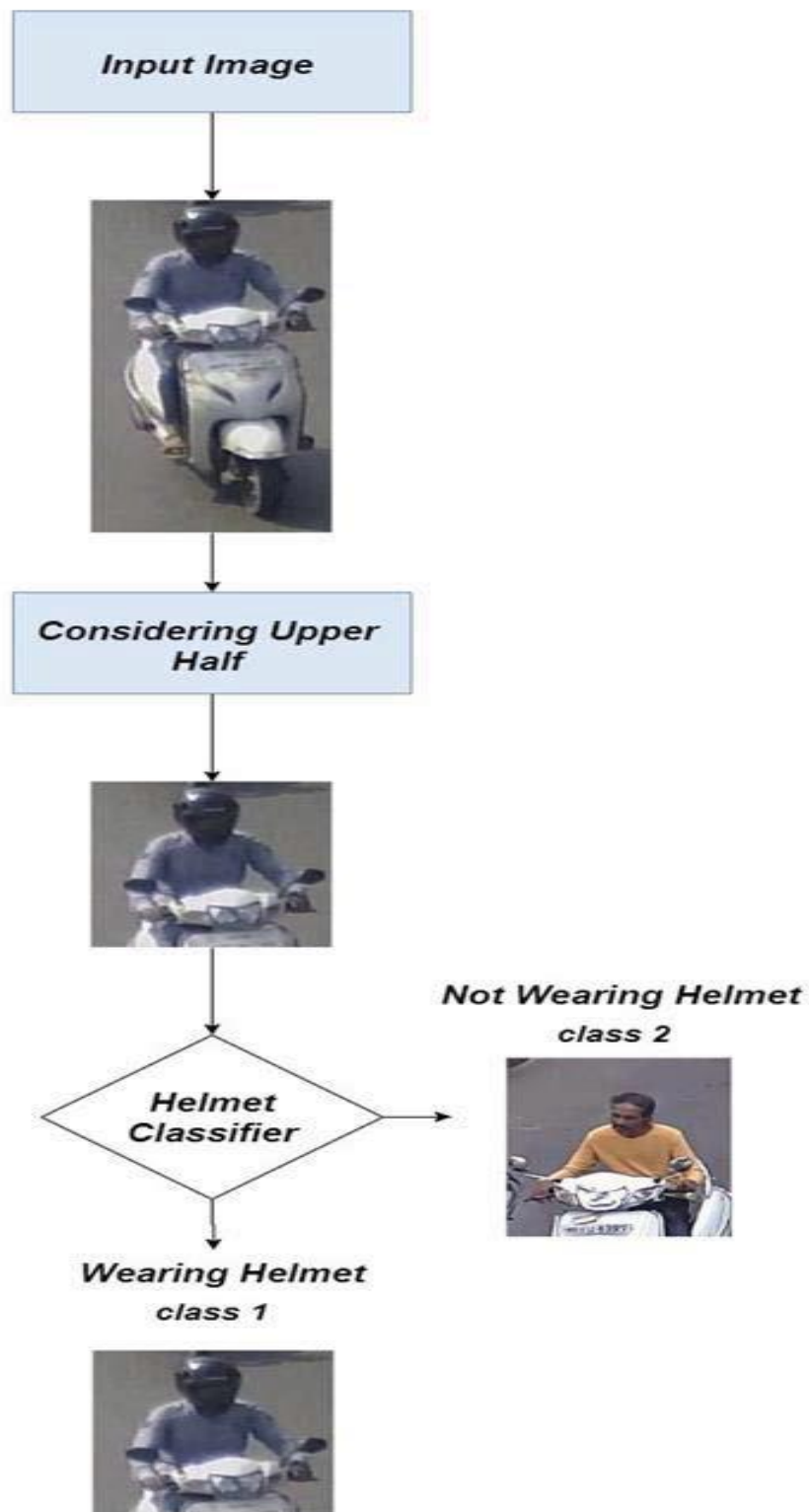


Fig 4.6 Helmet classification using CNN. Input image is classified as “wearing helmet” or “not wearing helmet”.

4.3.3 Crosswalk Classifier

Crosswalk violation is detected using instance segmentation. . Segmentation gives us a pixelwise mask for each object. Instance segmentation model was trained for two classes namely vehicle and crosswalk. Training CNN requires significant amount of time, hence we made use of pretrained mask CNN weights .After segmentation is performed we obtain a mask for each vehicle and zebra in the image along with their bounding boxes.

Crosswalk violation occurs if a vehicle is present on a crosswalk (Zebra crossing). First, we obtain coordinates of the bottom of the tyre with the help of the bounding box of the respective vehicle. Now we check if the obtained coordinates overlap with the crosswalk, this can be achieved by comparing coordinates obtained with those of the bounding box of the crosswalk. We assume top-left corner of image frame as origin (0,0), Considering $(Zx1, Zy1)$ and $(Zx2, Zy2)$ as coordinates of bounding box of crosswalk of top-left corner and bottom-right corner respectively, $(Vx1, Vy1)$ and $(Vx2, Vy2)$ as coordinates of bounding box of a particular vehicle of top-left corner and bottom-right corner respectively. We compare $Zy1$ with $Vy2$, if $Vy2 > Zy1$ where $Vy2$ will be y coordinate of bottom of tyre, then a violation has said to be occurred. Crosswalk violation is carried out for each vehicle by checking if the coordinate of the bottom of tyre lies within the crosswalk. It can be observed from fig that a four-wheeler and 2 two-wheelers (marked as green and purple) have violated crosswalk and their coordinates of the bottom of tyre lie over the crosswalk.

Crosswalk violation occurs if a vehicle is present on a crosswalk (Zebra crossing). First, we obtain coordinates of the bottom of the tyre with the help of the bounding box of the respective vehicle. Now we check if the obtained coordinates overlap with the crosswalk, this can be achieved by comparing coordinates obtained with those of the bounding box of the crosswalk. We assume top-left corner of image frame as origin (0,0), Considering $(Zx1, Zy1)$ and $(Zx2, Zy2)$ as coordinates of bounding box of crosswalk of top-left corner and bottom-right corner respectively, $(Vx1, Vy1)$ and $(Vx2, Vy2)$ as coordinates of bounding box of a particular vehicle of top-left corner and

bottom-right corner respectively. We compare $Zy1$ with $Vy2$, if $Vy2 > Zy1$ where $Vy2$ will be y coordinate of bottom of tyre, then a violation has said to be occurred. Crosswalk violation is carried out for each vehicle by checking if the coordinate of the bottom of tyre lies within the crosswalk. It can be observed from fig 3.8 that a four-wheeler (masked yellow) and 2 two-wheelers (marked as green and purple) have violated crosswalk and their coordinates of the bottom of tyre lie over the crosswalk.



Fig 4.7 Crosswalk detection using CNN

4.4 License Plate Recognition

After detecting a violation of traffic rules, the task to be carried out is to get the vehicle number from the vehicle that violated traffic rules. First, we localize the number plate of the vehicle using Object Detection. Localization of number plates is carried out using YOLO again. Localized license plates of the vehicles that violated any traffic rule are cropped out. The vehicle number is extracted by performing OCR (Optical Character Recognition) on the cropped number plate. In OCR character segmentation is carried out followed by template matching. Characters are segmented individually and then compared with characters in the database using template matching. Once the vehicle number is obtained the violation is stored in the database and the corresponding vehicle user is notified about the same through whatsapp message or email.



Fig 4.8 Shows the detection of the number plate of the vehicle, then the characters are read using OCR

4.4.1 Edge Detection

Edge detection is a technique of image processing used to identify points in a digital image with discontinuities, simply to say, sharp changes in the image brightness. These points where the image brightness varies sharply are called the edges (or boundaries) of the image. It is one of the basic steps in image processing, pattern recognition in images and computer vision. When we process very high-resolution digital images, convolution techniques come to our rescue.

4.4.1.1 Concept of Canny Edge Detection

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in

1. It is a multi-stage algorithm and we will go through each stages.
2. Noise Reduction

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter. We have already seen this in previous chapters.

3. Finding Intensity Gradient of the Image

Smoothed image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel as follows:

$$\text{Edge_Gradient}(G) = \sqrt{G_x^2 + G_y^2} \quad \text{Angle}(\theta) = \tan^{-1}(G_y/G_x)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

4. Non-maximum Suppression

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Check the image below:

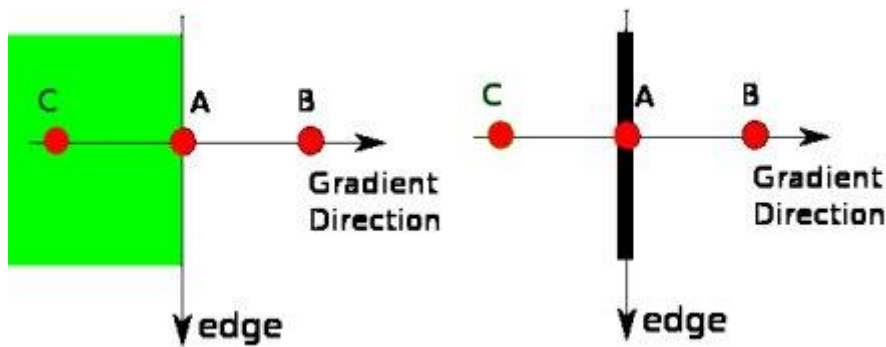


Fig 4.9 Edge Detection

Image

Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero).

In short, the result you get is a binary image with "thin edges".

5. Hysteresis Thresholding

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:

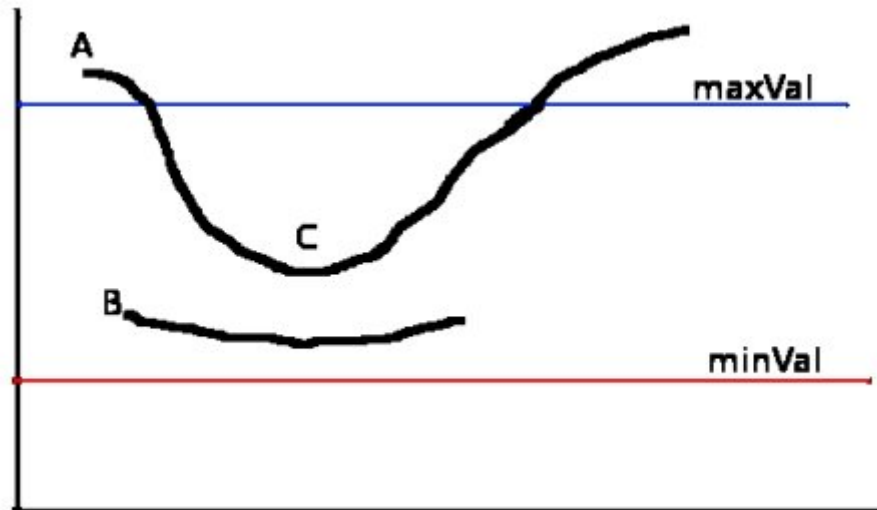


Fig 4.10 Graph showing Hysteresis Threshold

Image

The edge A is above the maxVal, so considered as "sure-edge". Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select minVal and maxVal accordingly to get the correct result.

This stage also removes small pixels noises on the assumption that edges are long lines.

So what we finally get is strong edges in the image.

4.4.1.2 Contours

Contours are defined as the line joining all the points along the boundary of an image that are having the same intensity. Contours come handy in shape analysis, finding the size of the object of interest, and object detection.

OpenCV has findContour() function that helps in extracting the contours from the image. It works best on binary images, so we should first apply thresholding techniques, Sobel edges, etc.

There are three essential arguments in cv2.findContours() function. First one is source image, second is contour retrieval mode, third is contour approximation method and it outputs the image, contours, and hierarchy. 'contours' is a Python list of all the contours

in the image. Each individual contour is a Numpy array of (x, y) coordinates of boundary points of the object.

4.4.2 Ocr Pytesseract

Tesseract — is an optical character recognition engine with open-source code, this is the most popular and qualitative OCR-library.

OCR uses artificial intelligence for text search and its recognition on images.

Tesseract is finding templates in pixels, letters, words and sentences. It uses two-step approach that calls adaptive recognition. It requires one data stage for character recognition, then the second stage to fulfil any letters, it wasn't insured in, by letters that can match the word or sentence context.

In general, the training step of Tesseract is :

1. Merge training data to .tiff file using jTessBoxEditor
2. Create a training label, by creating a .box files containing predictions of the Tesseract from .tiff file and fix each inaccurate predictions
3. Train the tesseract

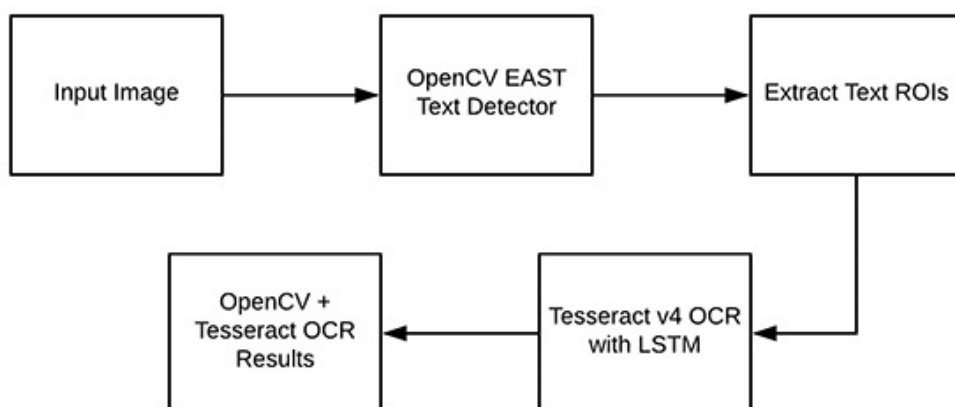


Fig 4.11 Block Diagram of OCR

CHAPTER-5

RESULTS AND DISCUSSION

In this section, we present experimental results of our modules viz. vehicle detection, helmet classifier and Image Segmentation. The experiments are carried on Ubuntu 18.04 machine consisting of i5 10th gen processor, 8 GB of ram and 256 GB SSD. Python 3.65 was used for evaluation of results along with libraries such as opencv-python for image processing, tensorflow as a Deep Learning framework (for training and prediction), numpy for mathematical operations and matplotlib for visualization.

5.1 Performance of The Model

AP (Average precision) is a popular metric in measuring the accuracy of object detectors like Faster R-CNN, SSD, etc. Average precision computes the average precision value for recall value over 0 to 1. It sounds complicated but actually pretty simple as we illustrate it with an example. But before that, we will do a quick recap on precision, recall, and IoU first.

5.1.1 Accuracy

Accuracy has two definitions:

More commonly, it is a description of systematic errors, a measure of statistical bias; low accuracy causes a difference between a result and a "true" value. ISO calls this trueness.

Alternatively, ISO defines accuracy as describing a combination of both types of observational error above (random and systematic), so high accuracy requires both high precision and high trueness.

$$\text{Accuracy} = \frac{\{TP+TN\}}{\{TP+TN+FP+FN\}}$$

where TP = True positive; FP = False positive; TN = True negative; FN = False negative

5.1.2 Precision

To evaluate object detection models like R-CNN and YOLO, the mean average precision (mAP) is used. The mAP compares the ground-truth bounding box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections.

Here are the few precision models:

- From Prediction Score to Class Label
- Precision-Recall Curve
- Average Precision (AP)
- Intersection over Union (IoU)
- Mean Average Precision (mAP) for Object Detection

The following are the calculated values for Accuracy and Precision in our model:

1. Vehicle Detection:

Vehicle detection is performed using YOLO algorithm, which was trained on 420 images consisting of both the classes, two and four. This was evaluated against 105 test images considering metric as average precision per class and also calculating the mean average precision. The results for which are as follows:

Table i

Results Of Vehicle Detection

Class	Average Precision
Four	0.9406
Two	0.9594

Table 5.1 Results of Vehicle Detection.

2. Helmet Classification:

Helmet Classifier performs image classification using CNN consisting of five convolutional layers with ReLu activation units, 4 max-pooling layers, and one fully connected dense layer with final softmax unit for classification into two classes. The proposed model architecture is evaluated using metrics such as precision, recall, accuracy, and F1-score on a test-set consisting of 215 images where 98 images belonged to helmet class and 117 belonged to non-helmet class.

Table ii

Results Of Helmet Classification

Metric	Value
Precision	0.8580
Recall	0.8925
Accuracy	87%
F1 score	0.8852

Table 5.2 Results of Helmet Classification

3. Crosswalk Violation:

Crosswalk Violation employs use of cnn network . This network is trained on 150 images with 2 classes namely vehicles, and crosswalk(zebra) using the concept of transfer learning. This network was then evaluated against 50 images each consisting of all of the 3 classes. Metric used for evaluation is mean Average Precision(mAP) is computed on a testset consisting of 50 images.

Table iii

Results Of Crosswalk Violation

Metric	Value
mean Average Precision (mAP)	0.96

Table 5.3 Results of Crosswalk Violation.

4. Number Plate Detection:

Number plate detection is performed using YOLO again which was trained on 450 images . This was evaluated against 100 test images considering metric as average precision. The results for which are as follows:

Table iv

Results Of Number Plate Detection

Metric	Value
mean Average Precision (mAP)	0.80

Table 5.4 Results of Number Plate Detection

5.2 Output Results

In this section the we will see the various results that we get when we give an image of traffic that has different violations. We tried to produce the output image with bounding boxes and also with the respective label written on the top indicating which violation it is. As we used a line `img.show()` in the code we get the output image for each input image given.

These are the images to show all the violations that we have covered in the project:

Case-1:

The conclusions we can draw by looking at the image are:

- 1.Car is on the crosswalk.
- 2.Bike is on the crosswalk.
- 3.The man on the bike has his helmet on.

Output:

The output we got after giving this image is as follows:

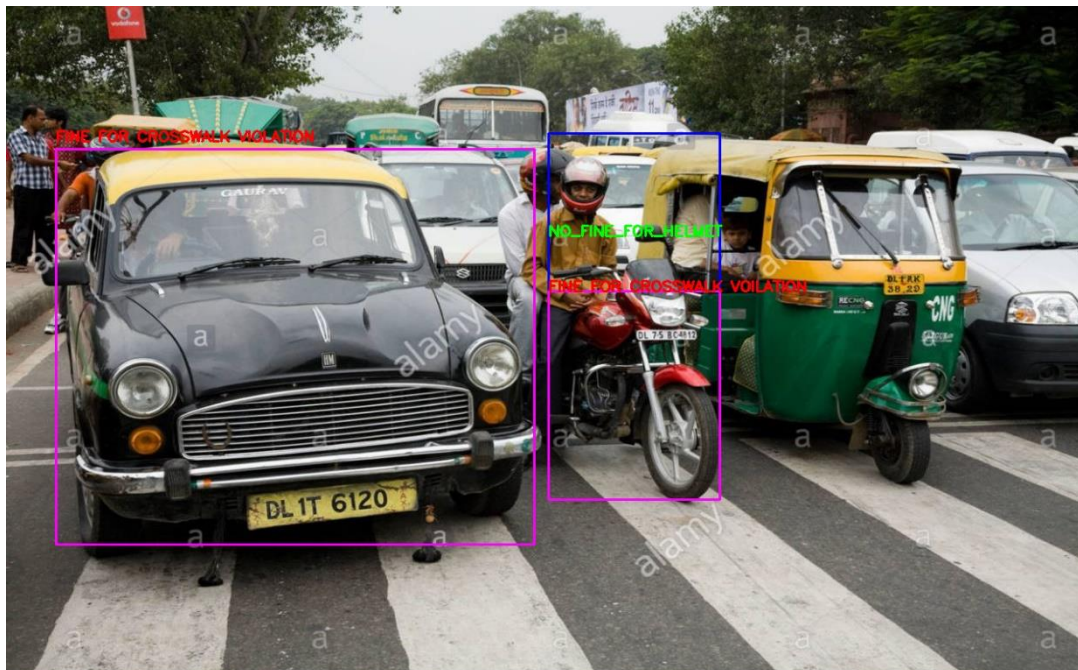


Fig 5.1 Output image showing the bounding boxes for case-1.

Console Output for the same is as follows:

```
jupyter final_code Last Checkpoint: 06/27/2021 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [ ]:
path = 'C:/Users/Bhavya/Desktop/final_traffic_detection'
img_name = 'fine2.jpeg' # name of the image
img = cv.imread(os.path.join(path,img_name)) # reading the image using the opencv
blob = cv.dnn.blobFromImage(img, 1/255, (wHt, wHt), [0, 0, 0], 1, crop=False) # creating the blob of the image to give blob
net.setInput(blob) # giving the blob of image to network to detect objects in the image
layersNames = net.getLayerNames() # we are using Layername to names Layer name in network
outputNames = [(layersNames[i][0] - 1)] for i in net.getUnconnectedOutLayers() # getting the output names from the Layers
outputs = net.forward(outputNames)
findObjects(outputs,img) # calling the method findobjects to detect the objects in the image and giving image as input to the
cv.imshow('img',img) # showing the output image
cv.waitKey(0)
cv.destroyAllWindows()
car is on the lane, mail is sent
This is number plate : None
person has his helmet on, no violation
bike is on the lane: mail is sent
This is the number plate None
In [ ]:
In [ ]:
```

Fig 5.2 Console output for case-1.

Case-2:

The conclusions we can draw by looking at the image are:

- 1.Car is on the crosswalk.
- 2.The man on the bike has his helmet on.

Output:

The output we got after giving this image is as follows:

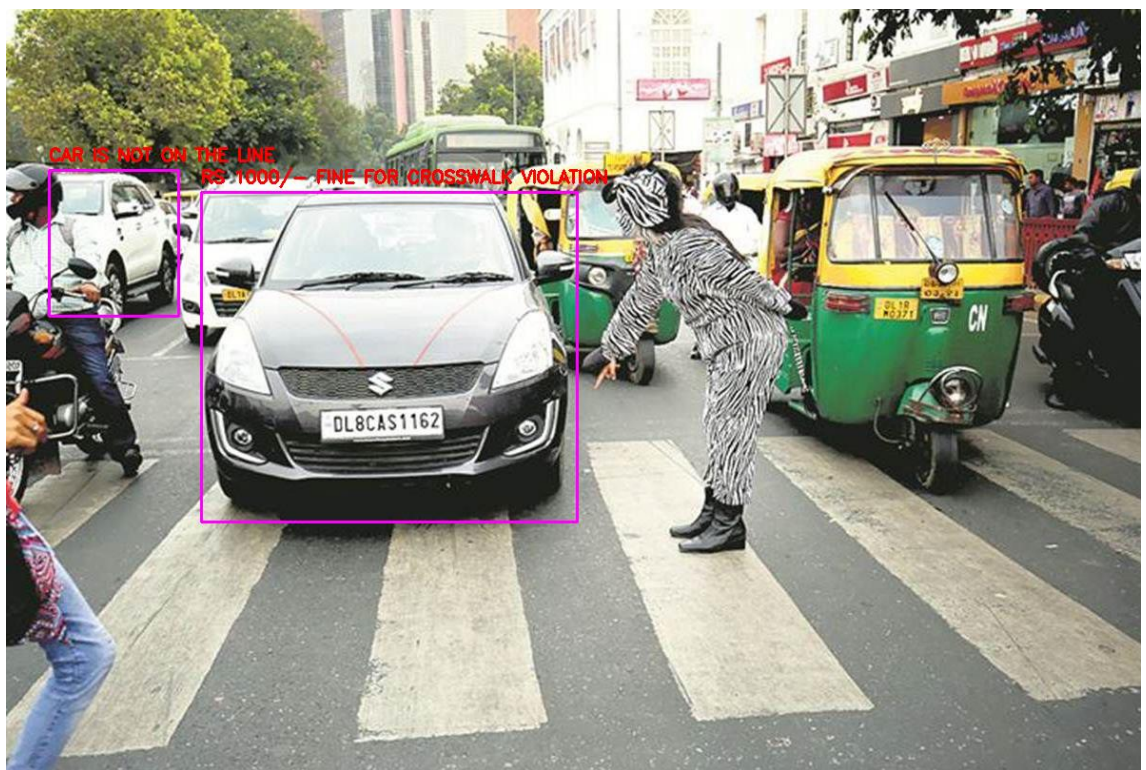


Fig 5.3 Output image showing the bounding boxes for case-2.

Console Output for the same is as follows:

```
jupyter final_code Last Checkpoint: 06/27/2021 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [*]:
1 path = 'C:/Users/pooji/final_traffic_detection/final_traffic_detection'
2 img_name = 'fine1.jpeg' # name of the image
3 img = cv.imread(os.path.join(path,img_name)) # reading the image using the opencv
4 blob = cv.dnn.blobFromImage(img, 1/255, (wHt, wHt), [0, 0, 0], 1, crop=False) # creating the blob of the image to give blob
5 net.setInput(blob) # giving the blob of image to network to detect objects in the image
6 layersNames = net.getLayerNames() # we are using layernames to names layer name in network
7 outputNames = [layersNames[i][0] - 1] for i in net.getUnconnectedOutLayers() # getting the output names from the layers
8 outputs = net.forward(outputNames)
9
10 findObjects(outputs,img) # calling the method findobjects to detect the objects in the image and giving image as input to th
11
12 cv.imshow('img',img) # showing the output image
13
14 cv.waitKey(0)
15 cv.destroyAllWindows()
16
car is on the lane mail is sent
This is number plate : LECAS 1162
car is not on the lane
```

Fig 5.4 Console output for case-2.

Case-3:

The conclusions we can draw by looking at the image are:

- 1.No vehicle is on the crosswalk.
- 2.Man on the bike does not have his helmet on.

Output:

The output we got after giving this image is as follows:

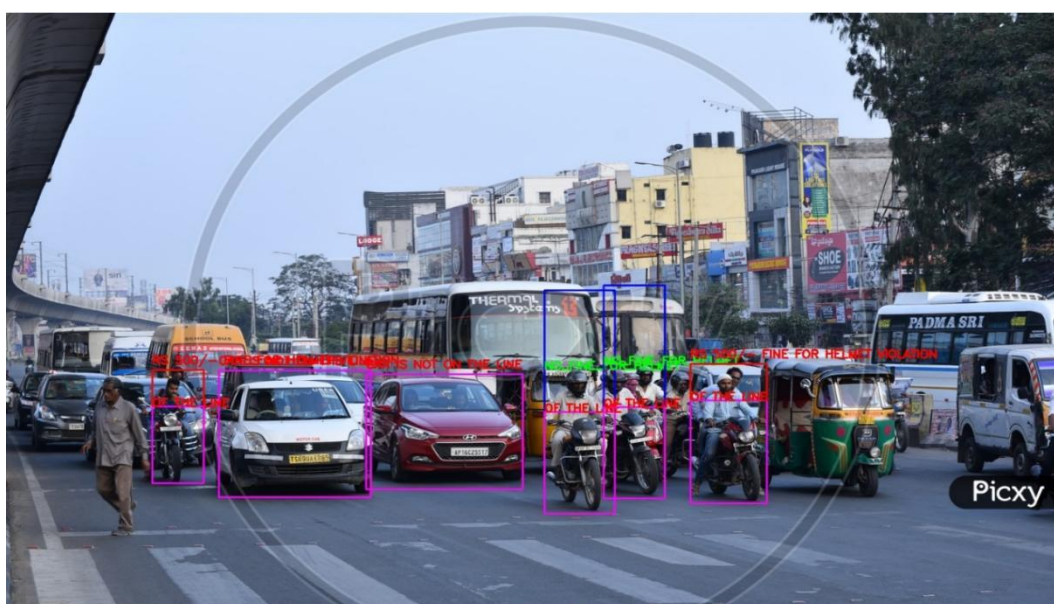
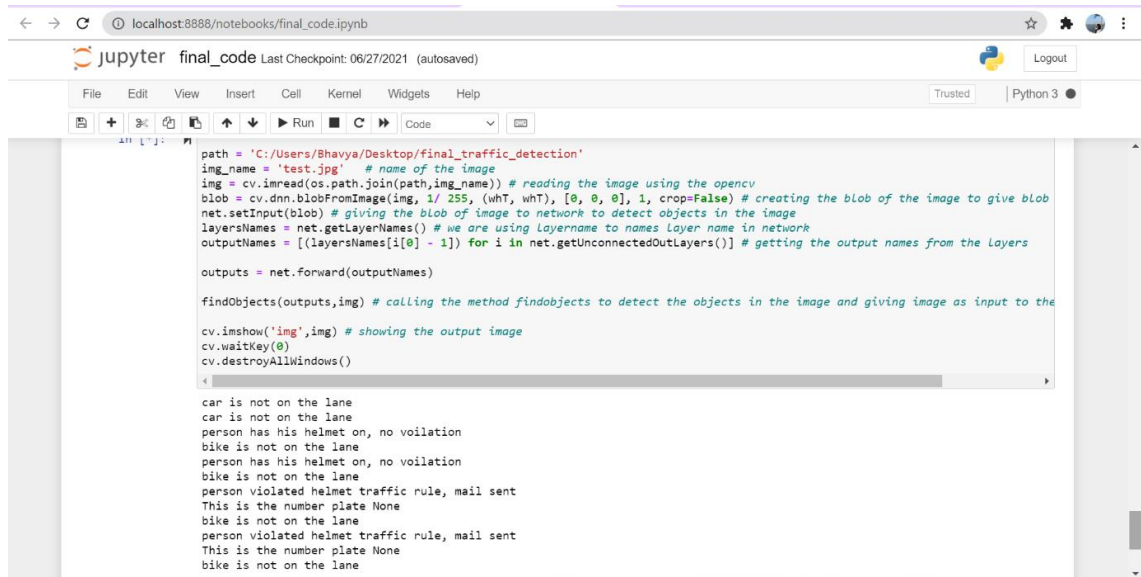


Fig 5.5 Output image showing the bounding boxes for case-3.

Console Output for the same is as follows:



```
path = 'C:/Users/Bhavysa/Desktop/final_traffic_detection'
img_name = 'test.jpg' # name of the image
img = cv.imread(os.path.join(path,img_name)) # reading the image using the opencv
blob = cv.dnn.blobFromImage(img, 1/255, (wHt, wHt), [0, 0, 0], 1, crop=False) # creating the blob of the image to give blob
net.setInput(blob) # giving the blob of image to network to detect objects in the image
layersNames = net.getLayerNames() # we are using Layername to names Layer name in network
outputNames = [(layersNames[i][0] - 1)] for i in net.getUnconnectedOutLayers() # getting the output names from the layers

outputs = net.forward(outputNames)

findObjects(outputs,img) # calling the method findobjects to detect the objects in the image and giving image as input to the
cv.imshow('img',img) # showing the output image
cv.waitKey(0)
cv.destroyAllWindows()

car is not on the lane
car is not on the lane
person has his helmet on, no violation
bike is not on the lane
person has his helmet on, no violation
bike is not on the lane
person violated helmet traffic rule, mail sent
This is the number plate None
bike is not on the lane
person violated helmet traffic rule, mail sent
This is the number plate None
bike is not on the lane
```

Fig 5.6 Console output for case-3.

5.3 Violation Notification

Once we detect the violations we then send the notification to the violator through email as well as Whatsapp. The notification of violation through email is sent using the library `secure-smtp` and the whatsapp notification is sent using `Py-whatkit`.

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers. Python provides **`smtplib`** module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon

PyWhatKit is a Python library with various helpful features. It is an easy to use library which does not requires you to do some additional setup.

We did this by mentioning the emails we want to send the mail to in our code, we also mentioned the whatsapp number to send the messages to. The whatsapp messages are sent only when the whatsapp web is given access in that particular Web Browser.

The following are the screenshots of the mails that have been sent:

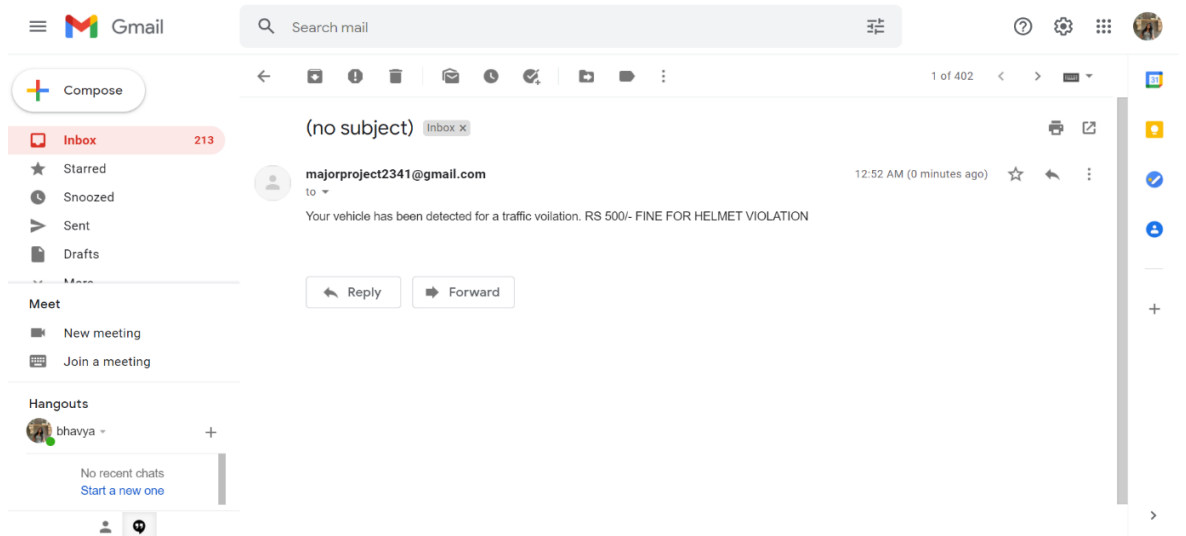


Fig 5.7 E-mail notification for helmet violation.

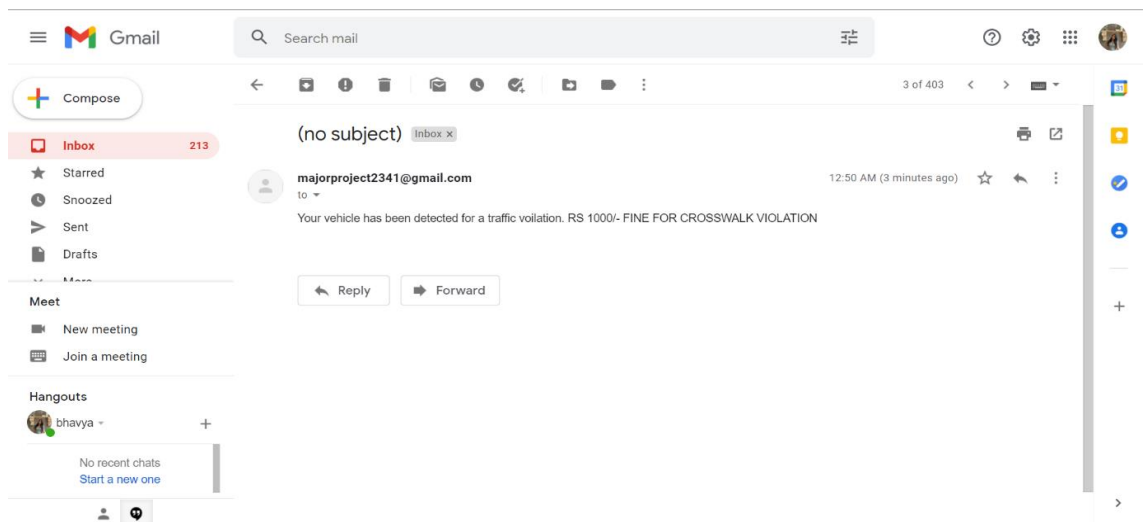


Fig 5.8 E-mail notification for crosswalk violation.

The following is the screenshot of the Whatsapp messages that have been sent:

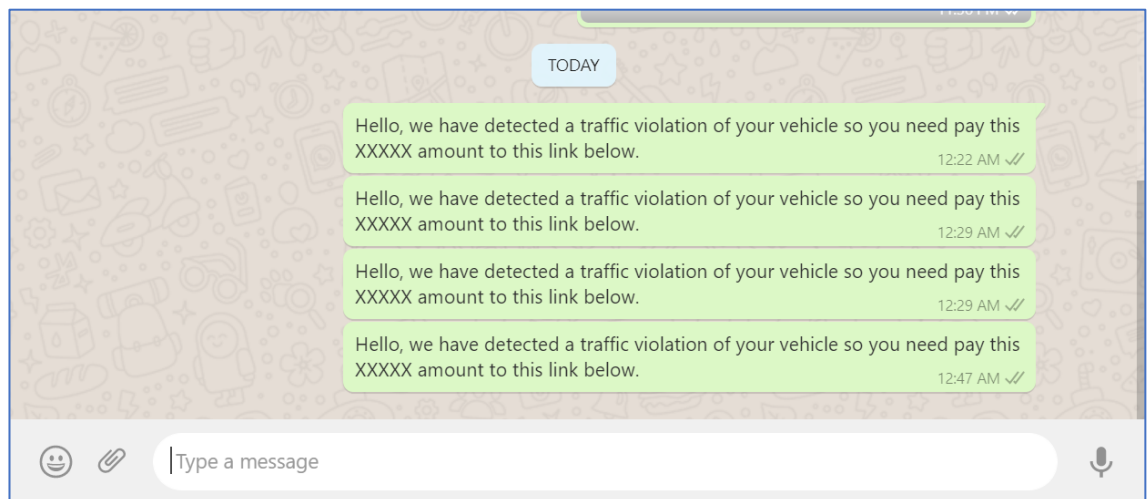


Fig 5.9 Whatsapp notification for violation.

5.4 Cost

The government faces a big challenge for the regulators to be put in place for an effective system to enforce motorist to wear helmet because even a metropolitan cities only have around 2000-3000 full time traffic personals are deployed for law and order maintenance. It is very difficult to increase the traffic personals as it will increase the cost to the government.

Also the problem is with general public mindset, who is watching us ? Most of the Accidents are happening at early & late hours of the day, so most of the traffic personals & signal systems are shut down by that time. We need the help of AI to resolve the challenge in all the Dimensions (human interference, cost, accuracy , data retrieval & punish the violators).

That is why the proposed system is highly efficient because though the initial installation and set-up costs might be high later on, the costs will be significantly low as

the government does not have to appoint new Traffic police. And if the system works accurately the entire traffic monitoring can be automated in the near future. The proposed system thus shows great promise and it is something the government must invest in as it is highly advisable to be up to date with the modern technology.

5.5 Usefulness to Society and Environmental Safety:

In India, over 78% of vehicles on the road are two-wheelers and they account for about 29 % of road accidents, a statistic that has risen steadily over the years. According to a road accident report, at least 98 two-wheeler riders died daily in 2017 because of not wearing helmets.

For Public safety services in India, Traffic control and enforcement of various traffic regulations still pose a major challenge because of manual detection procedures. In a place of millions, many tend to violate the traffic laws as the chance of getting caught is far less due to manual systems.

An accurate, state of the art and robust system would ensure that the people are fined every single time they violate rules and that would help our society reduce the no.of accidents on the roads. Maximizing the number of daily processed violations will ensure a much more safer driving environment for everyone.

5.5.1 Environmental Safety:

Environmental safety is something that we should always think about before starting to deploy a new system. In this proposed system there would not be any harm to the environment as all the hardware that is required is pretty basic and very environment friendly.

Wireless communication between sites and central servers ensures installing a city-wide system without stressing the environment with serious construction work. We can reduce the number of CCTV cameras by covering three lanes with a single high quality camera.

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The proposed approach employs the use of concepts such as YOLO, CNN and OCR for automatic detection of traffic rules violations. It achieves the desired goal with precision and ease, but requires high computational power as it makes use of concepts like image segmentation and object detection. The advantage of proposed system is it has the ability to catch more number of violations as compared to the human intervened system. In addition, the proposed methodology provides an end to end autonomous system, when implemented would prove an upper hand in detecting violations. We have then notified the violators about the same through E-mail and Whatsapp message. Thus strict regulations of traffic rule violations can be implemented resulting in better road safety and bring awareness among vehicle users.

6.2 Future Scope

Future work for the proposed system can have a broad perspective. The scope can be extended to detect other violations of traffic rules that occur. This can include whether a four-wheeler driver is wearing a seat belt or not, detection of more than two persons riding a two-wheeler, detection of vehicles traveling against allowed direction (vehicles driving the wrong side), etc.

We have executed it for existing images and the same can be done for images that are extracted from CCTV footage. Another issue in India is not all license plates have standardized format thus detection of fancy number plates could be a big help in standardizing number plates thus strengthening traffic rules discipline. The standardization would help in better OCR number plate detection in the future.

CHAPTER-7

REFERENCES

- 1.J. Chiverton, “Helmet presence classification with motorcycle detection and tracking,” *Intelligent Transport Systems (IET)*, vol. 6, no. 3, pp. 259–269, September 2012
2. Gomathi, et al, “Automatic Detection of Motorcycle without helmet using IOT”, *South Asian Journal of Engineering and Technology*
3. Samir Ibadov, et al, “Algorithm for detecting violations of traffic rules based on computer vision approaches”, *MATEC Web of Conferences* 132, 05005 (2017)
- 4.Aaron Christian P. Uy, et al, “Automated Traffic Violation Apprehension System Using Genetic Algorithm and Artificial Neural Network”, *2016 IEEE Region 10 Conference*
- 5.Amey Narkhede, et al, “Automatic Traffic Rule Violation Detection and Number Plate Recognition”, *IJSTE - International Journal of Science Technology & Engineering | Volume 3 | Issue 09 | March 2017*
- 6.Ana Riza, et al, “Localization of License Plates Using Optimized Edge and Contour Detection Technique ”, *2017 IEEE Region 10 Conference (TENCON)*, Malaysia, November 5-8, 2017
- 7.Prem Kumar Bhaskar, et al, “Image Processing Based Vehicle Detection and Tracking Method “, *Universiti Teknologi PETRONAS Seri Iskandar, 31750, Perak, Malaysia*
- 8.Kunal, Dahiya ; Dinesh, Singh ; and C. Krishna, Mohan “Automatic Detection of Bike-riders without Helmet using Surveillance Videos in Real-time”. *2016 International Joint Conference on Neural Networks (IJCNN)*
- 9.Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”
- 10.Kaiming He, Georgia Gkioxari, Piotr Doll’ar, Ross Girshick, “Mask R-CNN”, *Facebook AI Research (FAIR)* ,24 JAN 2018
- 11.<http://pjreddie.com/media/files/yolo.weights>

12.https://github.com/matterport/Mask_RCNN/releases/download/v2.0/mask_rcnn_coco.h5

13.Pooya Sagharichi Ha, Mojtaba Shakeri, “License Plate Automatic Recognition based on Edge Detection”

14.Abdullah Asım, et al, “A Vehicle Detection Approach using Deep Learning Methodologies”