A REPORT

on

TRAFFIC SURVEILLANCE SYSTEM

Submitted

by

Tushar Rawat
Vaibhav Banga
Rahul Kadam
Prashant Kumar
Mrinal Shahi
Akash Sarkar
Kumar Shubham
Vikash Rathod

Under the Team Lead of

Vivek Chaudhary

to

Packt Publishing Private Limited

Mumbai, Maharastra

1. INTRODUCTION

1.1 Traffic Surveillance System

In a traffic management system, the surveillance component is the process in which data is collected in the field. This data is used to supply information about conditions in the field to other system components. Surveillance provides the information needed to perform the following functions:

- Measure traffic and environmental conditions.
- Make control decisions.
- Monitor system performance.

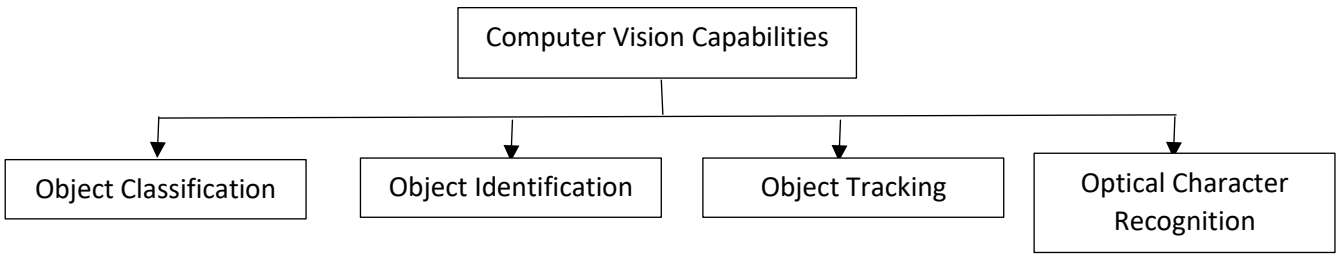
Surveillance is intended to provide support for other elements in the system (e.g., incident detection, information dissemination, ramp metering, etc.), not to drive the decisions about what system elements should be included. In other words, the goals and objectives of a surveillance system should be defined first, and then the system should be designed to meet these goals and objectives. A common mistake to be avoided is to first install a surveillance system, then ask the question "What can this system do for me?". It is essential to determine what system elements are to be supported before selecting and designing a surveillance system.

1.2 Role of Computer Vision

Computer vision is a field of computer science that focuses on enabling computers to identify and understand objects and people in images and videos. Like other types of AI, computer vision seeks to perform and automate tasks that replicate human capabilities. In this case, computer vision seeks to replicate both the way humans see, and the way humans make sense of what they see. Computer vision applications use input from sensing devices, artificial intelligence, machine learning, and deep learning to replicate the way the human vision system works. Computer vision applications run on algorithms that are trained on massive amounts of visual data or images in the cloud. They recognize patterns in this visual data and use those patterns to determine the content of other images.

Deep Learning & Computer Vision

Modern computer vision applications are shifting away from statistical methods for analyzing images and increasingly relying on what is known as deep learning. With deep learning, a computer vision application runs on a type of algorithm called a neural network, which allows it deliver even more accurate analyses of images. In addition, deep learning allows a computer vision program to retain the information from each image it analyzes—so it gets more and more accurate the more it is used.



There are potentially n number of classes in which a given image can be classified. Manually checking and classifying images could be a tedious task especially when they are massive in number (say 10,000) and therefore it will be very useful if we could automate this entire process using computer vision.

Structure of an Image Classification Task

- **Image Preprocessing**- The aim of this process is to improve the image data(features) by suppressing unwanted distortions and enhancement of some important image features so that our Computer Vision models can benefit from this improved data to work on.
- **Detection of an object**- Detection refers to the localization of an object which means the segmentation of the image and identifying the position of the object of interest.
- **Feature extraction and Training**- This is a crucial step wherein statistical or deep learning methods are used to identify the most interesting patterns of the image, features that might be unique to a particular class and that will, later on, help the model to differentiate between different classes. This process where the model learns the features from the dataset is called model training.
- **Classification of the object**- This step categorizes detected objects into predefined classes by using a suitable classification technique that compares the image patterns with the target patterns.

Steps for image Pre-Processing:

a) Read image
b) Resize image
c) Data Augmentation
d) Gray scaling of image
e) Reflection
f) Gaussian Blurring
g) Histogram Equalization
h) Rotation
i) Translation

As to follow traffic safety rules it's very difficult to track people in metropolitan cities over the highways to reduce the number of accidents at the same time. At this stage government is looking for some advance solution without any human intervention which is the need of an efficient traffic surveillance system to make people aware for the same to reduce the accident over the National Highways.

2. OBJECTIVES

We are going to build all models except our custom OCR as it is the unique one to make us stand out here in the market & we worked so hard on this OCR but the rest model that we have integrated in real time we are going to build them together & show the solution at the same time.

Considering this need, the government of India has decided to build a custom object detection

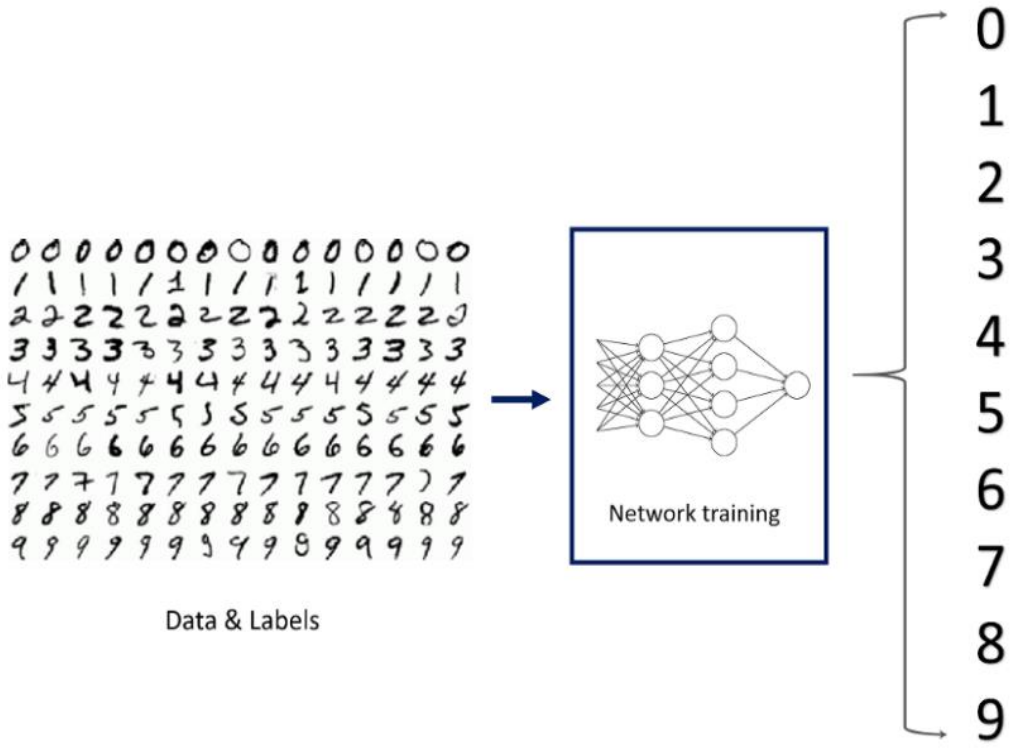
system with respect to traffic surveillance like vehicle classification, helmet detection, triple riding detection & number plate detection. So, the complete project aims at developing a model to classify the type of vehicles, helmet detection, number plate detection and triple riding detection.

3. METHODOLOGY

3.1 Available Models

a) ***MNIST:*** The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset. It is a dataset of 70,000 small square 28×28-pixel grayscale images of handwritten single digits between 0 and 9.

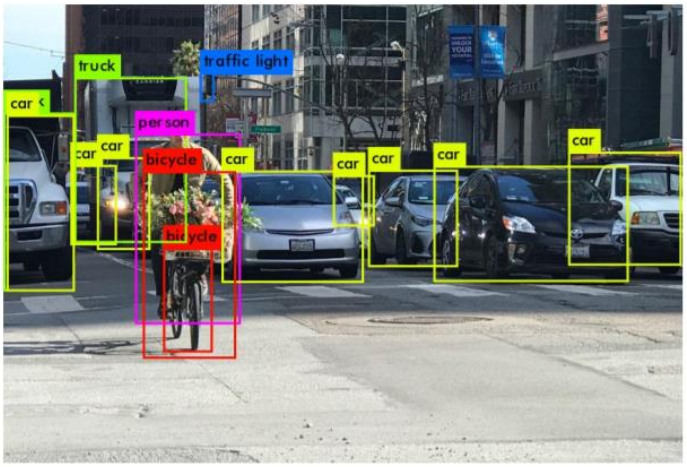
The MNIST database, an extension of the NIST database, is a low-complexity data collection of handwritten digits used to train and test various supervised machine learning algorithms. The database contains 70,000 28x28 black and white images representing the digits zero through nine. The data is split into two subsets, with 60,000 images belonging to the training set and 10,000 images belonging to the testing set. The separation of images ensures that given what an adequately trained model has learned previously, it can accurately classify relevant images not previously examined.

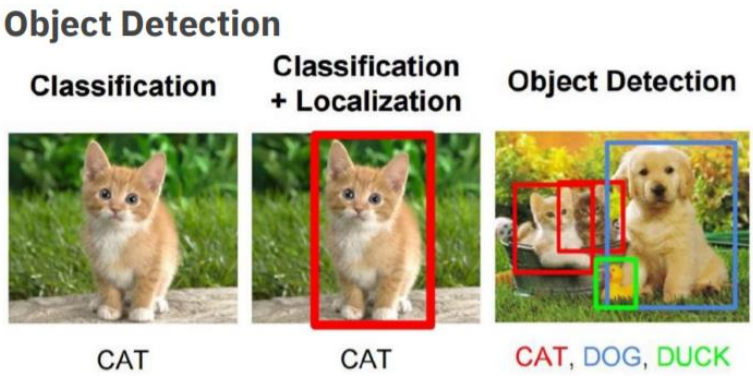


MNIST Dataset and Number Classification by Katakoda

Convolutional Layer → Pooling Layer → A set of Fully Connected Layers → Using MNIST Dataset → Reshaping and Normalizing the Images → Building the CNN → Compiling and Fitting the Model → Evaluating the Model

b) ***YOLO:*** YOLO stands for You Only Look Once, it an algorithm based on regression, instead of selecting the interesting part of an Image, it predicts classes and bounding boxes for the whole image in one run of the Algorithm. To understand the YOLO algorithm, first we need to understand what is actually being predicted YOLO divides up the image into a grid of 13 by 13 cells: Each of these cells is responsible for predicting 5 bounding boxes. It applies a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. YOLO works faster (45 frames per second) than other object detection algorithms. YOLO is used for object recognition for speed and real-time use. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions Whereas Faster R-CNN detect small objects well since it has nine anchors in a single grid, however it fails to do real-time detection with its two-step architecture.





From the image above, the objects have been recognized (two objects in the image), they have been localized (bounding boxes around the two images) and they have been classified (dog and cat with 98.0% and 88.0% respectively). This is the working of object detection. Object detection has several uses, some of which are:

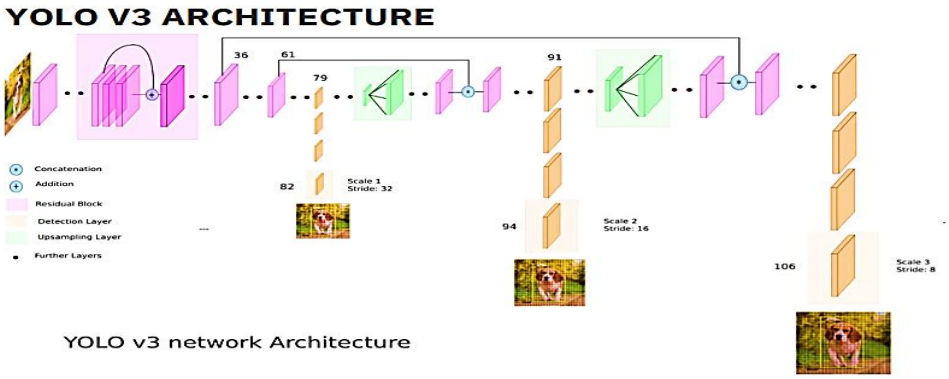
- For tracking objects, example, tracking a ball during a football match or tennis game, also for traffic checking
- For automated CCTV surveillance, to help increase security and protect people and property
- For detecting people, mostly for the police and security services, to use to detect any unscrupulous activities by individuals

- For vehicle detection, mostly for self-driving cars, to detect vehicles and respond accordingly

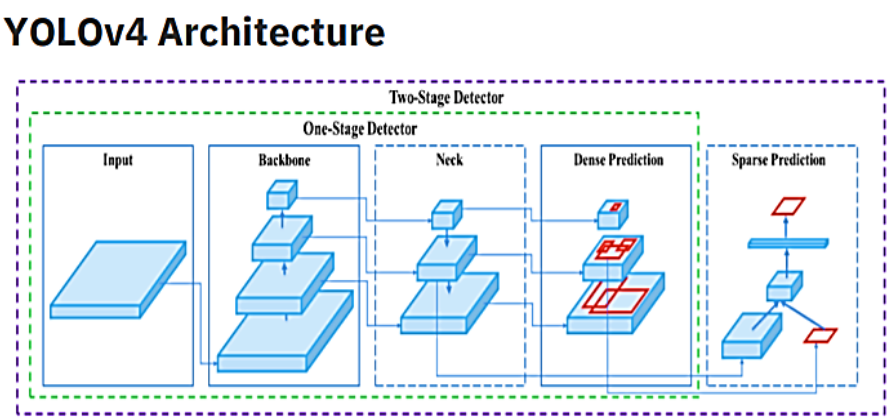
  Over the past few years, deep learning architectures are achieving extraordinary and state-of-the-art performances for object detection. There are many families of object detection created over the years, but the most popular ones are the YOLO and RCNN object detection families. This article would focus on the YOLO architecture.

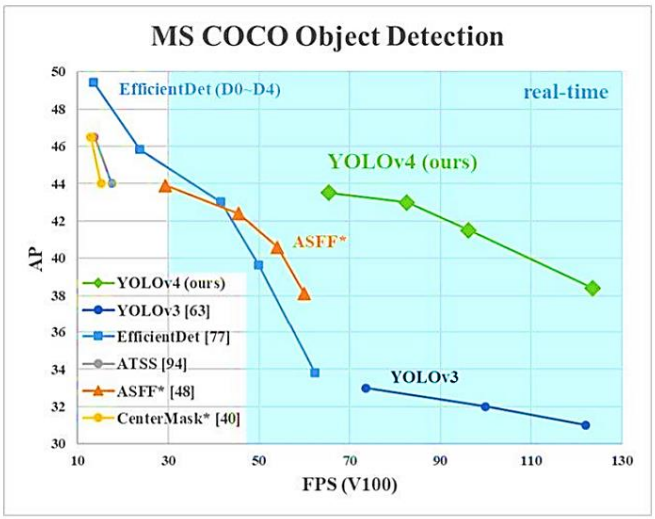
  In this study, we are using the following models:

  a) Yolov3 tiny
  b) Yolo v3
  c) Yolo v4 tiny
  d) Yolo v4

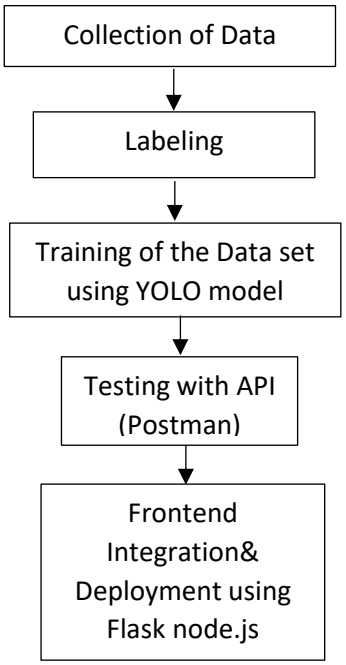


YOLO v3 network Architecture

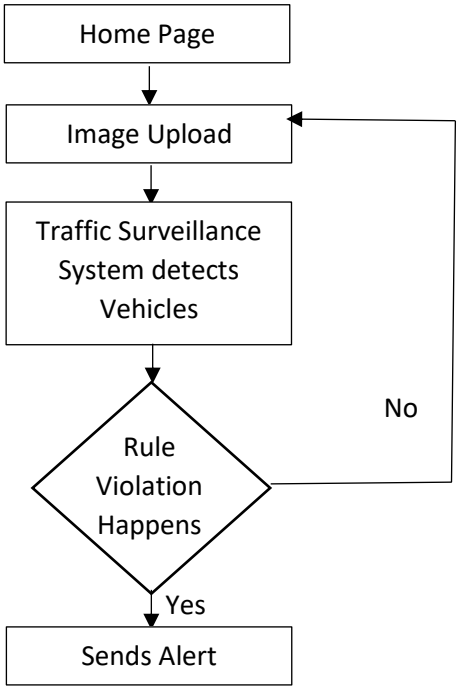




3.2 Flowchart depicting the complete methodology of project



3.2.1 Used Cases



3.2.2 Data Collection

Real time data were collected from various road sections of our locality. The locations were selected in such a way that we can get a heterogenous traffic volume and varying traffic flow so that we can observe various categories of vehicles, large number of vehicles so that chances of getting such data set where it can be observed that traffic rules are violated like absence of number plate, helmet and triple riding cases.

3.2.3 Coding for the Model

```
plate_wgh_pth = resource_path(r'checkpoint\NUm_plate\yolov4_training_final.weights')
plate_cfg_pth = resource_path(r'checkpoint\NUm_plate\yolov4_training.cfg')
plate_label_pth = resource_path(r'checkpoint/NUm_plate/objplate.names')

# Vehicle Detection Checkpoint
vd_wgh_pth = resource_path(r'checkpoint/vehicleDetection/yolov4_vehicle_detection-288.weights')
vd_cfg_pth = resource_path(r'checkpoint/vehicleDetection/yolov4_vehicle_detection-288.cfg')
vd_label_pth =resource_path(r'checkpoint/vehicleDetection/yolov4_vehicle_detection-288.names')

# Helemet - Tripple Ridding Checkpoint
violation_wgh_pth = resource_path(r'checkpoint\helmat\yolov4-training-288-final_final.weights')
violation_cfg_pth = resource_path(r'checkpoint\helmat\yolov4-training-288.cfg')
violation_label_pth = resource_path(r'checkpoint\helmat\helmet-yolov4-288.names')

# Loading Plate Detection Model
platedetection = yolo.LoadYOLO(input_h=256, input_w=256, \
    labelPath=plate_label_pth, cfgPath=plate_cfg_pth, weightPath=plate_wgh_pth)
# Loading OCR Model
# Loading Vehicle Detection Model
vehicledetection = yolo.LoadYOLO(input_h=288, input_w=288, \
    labelPath=vd_label_pth, cfgPath=vd_cfg_pth, weightPath=vd_wgh_pth)
# Loading Helemet - Tripple Riddingn Model
helemt = yolo.LoadYOLO(input_h=288, input_w=288, \
    labelPath=violation_label_pth, cfgPath=violation_cfg_pth, weightPath=violation_wgh_pth)
```

```
##########################################################################################################################
@app.route('/',methods=['GET'])
def welcommessage():
    return 'Hello Welcome to the API Building'
@app.route('/image/',methods=['POST'])
def getimage():
    data=[]

    userid=''
    timsestamp=''
    withoutVehicle = False
    final_saved_data_in_firebase=[]
    outputdata=[]
    filestr=request.files['files']
    npimg=np.fromfile(filestr,np.uint8)
    frame=cv2.imdecode(npimg,cv2.IMREAD_COLOR)
    unique_id = uuid.uuid1()
    var2 = str(unique_id)+str(time.time())+".jpg"
    def vehicle_detection(frame):
        outputvehicle = vehicledetection.detect(frame)
        vehicle_detail = []
        if type(outputvehicle)!=str:
            vehicle_detail = outputvehicle
            return vehicle_detail
```

```
const [uploadedFile, setUploadedFile] = useState({});
const [message, setMessage] = useState('');
const [uploadPercentage, setUploadPercentage] = useState(0);
const [process,setprocess]=useState(0)
const [output_data,setoutput_data]=useState([]);
const [res,setres]=useState([]);
const onChange = e => {
  setFile(e.target.files[0]);
  setFilename(e.target.files[0].name);
};
const onSubmit = async e => {
  e.preventDefault();
  const formData = new FormData();
  formData.append('files', file);
  try {
     const res = await axios.post('http://localhost:5000/image/', formData, {
      headers: {
        'Content-Type': 'multipart/form-data'
      },
      onUploadProgress: progressEvent => {
        setUploadPercentage(
          parseInt(
            Math.round((progressEvent.loaded * 100) / progressEvent.total)
          )
        );
```

4. OBSERVATIONS

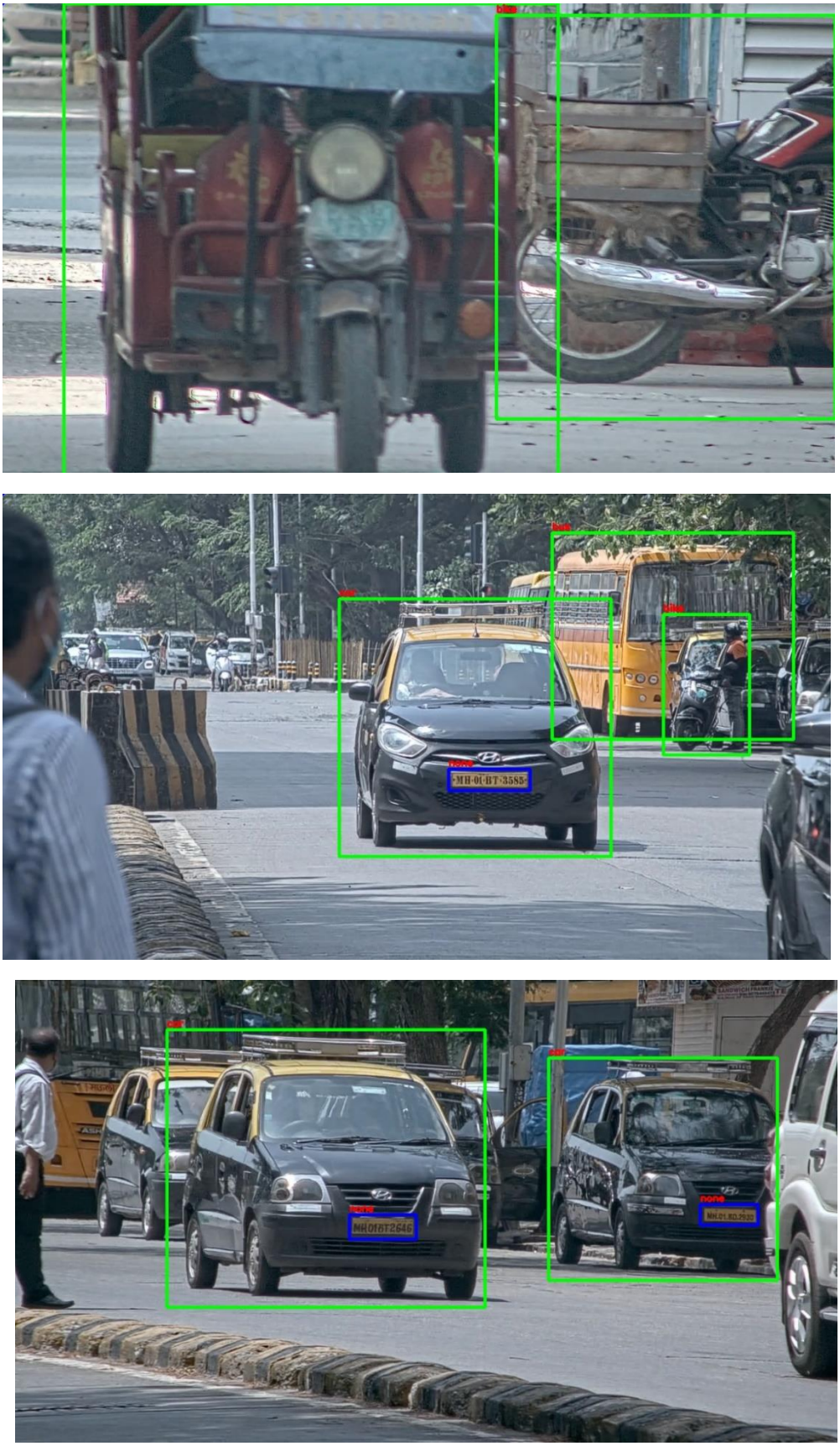


Fig: Various Classes of vehicles detected along with the Number Plate

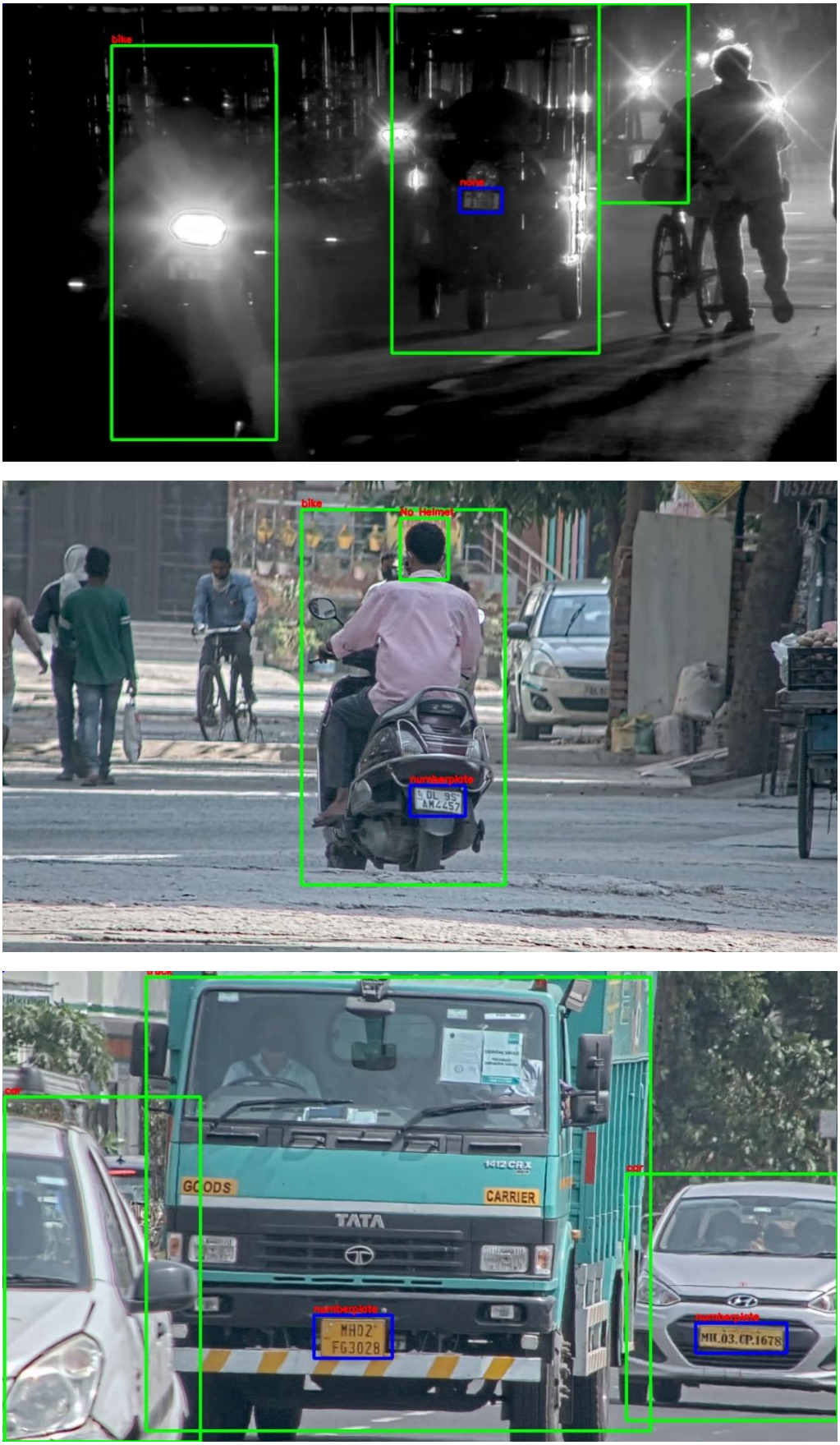


Fig: Various Classes of vehicles detected along with the Helmet detection

5. CONCLUSION

The developed model has the ability to detect the number plates, to detect the helmet, triple riding and the type of vehicle. These test cases, thus, can be used to for the real time monitoring of the traffic Relevant authorities can monitor road conditions with the live surveillance cameras. So, they can make real-time traffic reports, helping people choose the right time and road when going out. So, following will be the benefits of using this model:

- The presence of a traffic camera often influences people's driving behavior in a positive way. For fear of being caught or ticketed, drivers usually slow down and abide by traffic rules.
- Aside from influencing behaviors, these traffic cameras can catch those who engage in traffic violations. Ticketing these drivers helps to deter drivers from making the same mistakes again.
- Rather than employing officers at all intersections, the traffic cameras can monitor activity. This allows officers to tend to more pressing duties.

6. FUTURE SCOPE

- Development of an AI based Traffic Light System based on the traffic volume in each lane.
- Models to detect wrong lane driving, Signal jumping, Seatbelt detection, over speeding etc. to work as an eye for the traffic police department.
- Development of a model to predict future vehicle load based on traffic heterogeneity of an area for further expansion of the road facilities.