**ii**

**ABSTRCAT:**

Road accidents are one of the major causes of human deaths. Among the different types of road accidents, motorcycle accidents are common and cause severe injuries. The helmet is the motorcyclist’s main protection.in current situation, we come across various problems in traffic regulations in India which can be solved with different ideas. Riding motorcycle without wearing helmet is a traffic violation which has resulted in increase in number of accidents and deaths in India. As a solution to this, it is highly desirable for bike-riders to use helmet. Observing the usefulness of helmet, Governments have made it punishable offense to ride a bike without helmet and have adopted manual strategies to catch the violators which has limitations of speed. Using video surveillance of the street, the proposed approach detects if the bike rider is wearing a helmet automatically without manual help. If a bike rider is detected not wearing a helmet, the number plate of the vehicle read and noted.

E-challan will also be generated with offender details. A database will be generated with records to identify every offender accurately. The system implements pure machine learning in order to identify every type of helmet that it comes across with minimum computation cost.

**iii**

**TABLE OF CONTENTS**

**ACKNOWLEDGEMENT i**

**ABSTRACT ii**

**TABLE OF CONTENTS iii**

**LIST OF FIGURES iv**

**CHAPTER 1: PREAMBLE**

* 1. Introduction
  2. Project overview

**CHAPTER 2: LITRETURE SURVEY**

2.1 Literature Survey

2.2 Problem Definition

**CHAPTER 3: OVERVIEW OF OBJECT DETECTION**

3.1 Concept

3.2 Existing Methods

3.2.1 SSD

3.2.2 R-CNN, Fast R-CNN & Faster R-CNN

3.2.3 RESNET

3.2.4 YOLOv5

**CHAPTER 4: SYSTEM DESIGN**

4.1 Procedure

4.2 Flowchart of proposed System

**CHAPTER 5: SYSTEM REQUIREMENTS**

5.1 Software’s and Libraries Used

5.2 TensorFlow

5.3 OpenCV

5.4 Selenium Web Driver

**CHAPTER 6: SYSTEM IMPLEMENTATION**

6.1 Model for Motorcycle Detection

6.2 Model for Helmet Detection

6.2.1 Training Yolov5 Helmet Model

6.3 Number plate Recognition Web API

6.4 Automated E-challan Generation

**CHAPTER 7: SYSTEM TESTING**

7.1 Account setup on Gdrive

7.2 Detection of Motorcycle in Frames

7.3 Detection of Helmet

7.4 Recognition of Number plate

7.5 E-Challan Generation

**CONCLUSION**

**REFERENCES**

**iv**

**LIST OF FIGURES**

**Fig No Title**

1. Retina Net Detector Architecture
2. Yolov5 Architecture
3. Bounding Box Prediction
4. Algorithms Comparisons
5. Flowchart of Workflow
6. Flowchart for Helmet Model
7. Helmet Dataset
8. Helmet ROI Cropping
9. Helmet Yolov5 Prediction
10. Number plate Recognition
11. RTO database Extraction
12. E-challan Generation

**CHAPTER 1:**

**1.1 Introduction**

All over the world around 1.35 million lives are lost each year, 50 million people are getting injured due to road accidents, according to a report titled “The Global status Revised Manuscript Received on December 05, 2019 report on road safety 2018” released by world health organization. It is very hard to imagine that this burden is unevenly borne by motorcyclists, cyclists, and pedestrians. This report noted that a comprehensive action plan must be set up in order to save lives.

Two-wheeler is a very popular mode of transportation in almost every country. However, there is a high risk involved because of less protection. When a two-wheeler meets with an accident, due of sudden deceleration, the rider is thrown away from the vehicle. If head strikes any object, motion of the head becomes zero, but with its own mass brain continues to be in motion until the object hits inner part of the skull. Sometimes this type of head injury may be fatal in nature. In such times helmet acts as life saviour. Helmet reduces the chances of skull getting decelerated, hence sets the motion of the head to almost zero. Cushion inside the helmet absorbs the impact of collision and as time passes head comes to a halt. It also spreads the impact to a larger area, thus safeguarding the head from severe injuries. More importantly it acts as a mechanical barrier between head and object to which the rider came into contact. Injuries can be minimized if a good quality full helmet is used. Traffic rules are there to bring a sense of discipline, so that the risk of deaths and injuries can be minimized significantly. However strict adherence to these laws is absent. Hence efficient and feasible techniques must be created to overcome these problems. To reduce the involved risk, it is highly desirable for bike-riders to use helmet. Worrying fact is that India ranks in top as far as road crash deaths are considered. Rapid urbanization, avoiding helmets, seat belts and other safety measures while driving are some of the reasons behind this trend according to analysis done by experts. In 2015 India signed Brasilia Declaration on Road Safety, where India committed to reduce road crash deaths to 50 percent by 2020.

Observing the usefulness of helmet, Governments have made it a punishable offense to ride a bike without helmet and have adopted manual strategies to catch the violators. However, the existing video surveillance-based methods are passive and require significant human assistance. In general, such systems are infeasible due to involvement of humans, whose efficiency decreases over long duration. Automation of this process is highly desirable for reliable and robust monitoring of these violations as well as it also significantly reduces the amount of human resources needed.

Recent research has successfully done this work based on CNN, R-CNN, LBP, HoG, HaaR features, etc. But these works are limited with respect to efficiency, accuracy or the speed with which object detection and classification is done.

**1.2 Project Overview:**

In this Project Work, a Non-Helmet Rider detection system is built which attempts to satisfy the automation of detecting the traffic violation of not wearing helmet and extracting the vehicles’ license plate number. The main principle involved is Object Detection using Deep Learning at three levels. The objects detected are person, motorcycle at first level using YOLOv5, helmet at second level using YOLOv5, License plate at the last level using Web API. Then the license plate registration number is extracted using Web Automation. Hence a database will be available for analysis for the police authority.

**CHAPTER 2**

**2.1 Literature Survey**

In various fields, there is a necessity to detect the target object and track them effectively while handling occlusions and other included complexities. Many researchers (Almeida and Guting 2004, Hsiao-Ping Tsai 2011, Nicolas Papadakis and Aure lie Bugeau 2010) attempted for various approaches in object tracking. The nature of the techniques largely depends on the application domain. Some of the research works which made the evolution to proposed work in the field of object tracking are depicted as follows

Until very recently, most of the methods used for object detection and object classification used methods such as Haar, HOG, local binary patterns (LBP), the scale invariant feature transform (SIFT), or speeded up robust features (SURF) for feature extraction and then support vector machines (SVM), random forests, or AdaBoost for the classifier. Silva et al. [1] use methods such as histograms of oriented gradient (HOG), LBP, and the wavelet transform (WT) for feature extraction for classifying motorcyclists with helmets and without helmets. They use multiple combinations of the base features such as HOG+LBP+WT, obtaining seven possible feature sets. In [5], K. Dahiya et al. came up with helmet detection from surveillance videos where they used an SVM classifier for classifying between motorcyclist and non-motorcyclist and another SVM classifier for classifying between helmet and without helmet. For both classifiers, three widely used features - HOG, SIFT and LBP - were implemented and the performance of each was compared with that of other two features. They concluded that HOG descriptor helped in achieving the best performance.

In [6], C. Vishnu et al. proposed an approach using Convolutional Neural Networks (CNNs) for classification. In recent years, CNNs performing both automatic feature extraction and classification have outperformed previously dominant methods in many problems. Advances in graphical processing units (GPUs), along with the availability of more training data for neural networks to learn, have recently enabled unprecedented accuracy in the fields of machine vision, natural language processing, and speech recognition. Nowadays, all state-of-the-art methods for object classification, object detection, character classification, and object segmentation are based on CNNs. See for example the methods used in the ImageNet large scale visual recognition challenge [2].

Li and Shen [3] use a deep convolutional neural network and long-short term memory (LSTM) for the license plate recognition and character extraction process. They use two methods for segmentation and recognition. [4] have shown the use of CNNs for text detection and recognition provides significant improvement over existing methods.

The YOLOv5 algorithm is capable of accurate object detection (traffic participants) with near real-time performance (~ 25 fps on HD images) in the variety of the driving conditions (bright and overcast sky, snow on the streets, and driving during the night).

YOLO v5 algorithm consists of fully CNN and an algorithm for post-processing outputs from neural network. CNNs are special architecture of neural networks suitable for processing grid-like data topology. The distinctive feature of CNNs which bears importance in object detection is parameter sharing. Unlike feedforward neural networks, where each weight parameter is used once, in CNN architecture each member of the kernel is used at every position of the input, which means learning one set of parameters for every location instead a separate set of parameters.

**2.2 Problem Definition:**

Existing system monitors the traffic violations primarily through CCTV recordings, where the traffic police must investigate the frame where the traffic violation is happening, zoom into the license plate in case rider is not wearing helmet. But this requires lot of manpower and time as the traffic violations frequently and the number of people using motorcycles is increasing day-by-day. What if there is a system, which would automatically look for traffic violation of not wearing helmet while riding motorcycle and if so, would automatically extract the vehicles’ license plate number.

**CHAPTER 3: OVERVIEW OF OBJECT DETECTION**

**3.1 Concept**

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance. It is widely used in computer vision task such as face detection, face recognition, video object co-segmentation. It is also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, tracking a person in a video.

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a distance from a point (i.e. the centre) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin Colour and Distance Between Eyes Can Be Found.

**3.2 Existing Methods**

Methods for object detection generally fall into either machine learning-based approaches or deep learning-based approaches. For Machine Learning approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as support vector machine (SVM) to do the classification. On the other hand, deep learning techniques that can do end to-end object detection without specifically defining features and are typically based on convolutional neural networks (CNN).

**Deep Learning approach:**

**3.2.1 Single Shot Multibox Detection The paper about SSD:**

Single Shot MultiBox Detector (by C. Szegedy et al.) was released at the end of November 2016 and reached new records in terms of performance and precision for object detection tasks, scoring over 74% mAP (mean Average Precision) at 59 frames per second on standard datasets such as PascalVOC and COCO. To better understand SSD, let’s start by explaining where the name of this architecture comes from:

**• Single Shot:** this means that the tasks of object localization and classification are done in a single forward pass of the network

**• MultiBox:** this is the name of a technique for bounding box regression developed by Szegedy et al. (we will briefly cover it shortly)

**• Detector:** The network is an object detector that also classifies those detected objects

The SSD object detection composes of 2 parts:

• Extract feature maps, and

• Apply convolution filters to detect objects. Modified from SSD:

Single Shot MultiBox Detector SSD uses VGG16 to extract feature maps. Then it detects objects using the Conv4\_3 layer. For illustration, we draw the Conv4\_3 to be 8 × 8 spatially (it should be 38 × 38). For each cell (also called location), it makes 4 object predictions. Each prediction composes of a boundary box and 21 scores for each class (one extra class for no object), and we pick the highest score as the class for the bounded object. Conv4\_3 makes a total of 38 × 38 × 4 predictions: four predictions per cell 10 regardless of the depth of the feature maps. As expected, many predictions contain no object. SSD reserves a class “0” to indicate it has no objects. Each prediction includes a boundary box and 21 scores for 21 classes (one class for no object).

**Convolutional predictors for object detection**:

SSD does not use a delegated region proposal network. Instead, it resolves to a very simple method. It computes both the location and class scores using small convolution filters. After extracting the feature maps, SSD applies 3 × 3 convolution filters for each cell to make predictions. (These filters compute the results just like the regular CNN filters.) Each filter outputs 25 channels: 21 scores for each class plus one boundary box. Apply a 3x3 convolution filter to make a prediction for the location and the class. 11 For example, in Conv4\_3, we apply four 3 × 3 filters to map 512 input channels to 25 output channels.

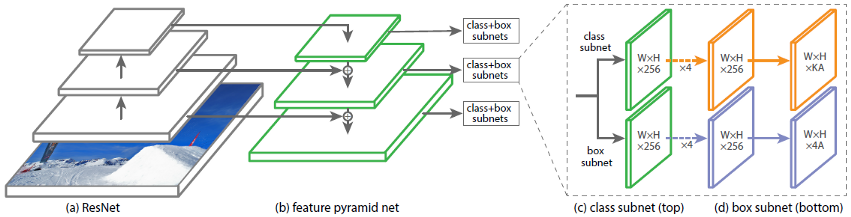
**3.2.2 Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)**:

In this, algorithms try to draw a bounding box around the object of interest to locate it within the image. Also, you might not necessarily draw just one bounding box in an object detection case, there could be many bounding boxes representing different objects of interest within the image and you would not know how many beforehand. The major reason why you cannot proceed with this problem by building a standard convolutional network followed by a fully connected layer is that, the length of the output layer is variable — not constant, this is because the number of occurrences of the objects of interest is not fixed. A naive approach to solve this problem would be to take different regions of interest from the image and use a CNN to classify the presence of the object within that region. The problem with this approach is that the objects of interest might have different spatial locations within the image and different aspect ratios. Hence, you would have to select a huge number of regions and this could computationally blow up. Therefore, algorithms like R-CNN, YOLO etc have been developed to find these occurrences and find them fast.

**3.2.3 Resnet:**

To train the network model in a more effective manner, we herein adopt the same strategy as that used for DSSD (the performance of the residual network is better than that of the VGG network). The goal is to improve accuracy. However, the first implemented for the modification was the replacement of the VGG network which is used in the original SSD with ResNet. We will also add a series of convolution feature layers at the end of the underlying network. These feature layers will gradually be reduced in size that allowed prediction of the detection results on multiple scales. When the input size is given as 300 and 320, although the ResNet–101 layer is deeper than the VGG–16 layer, it is experimentally known that it replaces the SSD’s underlying convolution network with a residual network, and it does not improve its accuracy but rather decreases it.

RetinaNet, a one-stage detector,**by using focal loss, lower loss is contributed by “easy” negative samples so that the loss is focusing on “hard” samples,** which improves the prediction accuracy.With **[ResNet](https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8" \t "_blank)+**[**FPN**](https://towardsdatascience.com/review-fpn-feature-pyramid-network-object-detection-262fc7482610)**as backbone**for feature extraction**, plus two task-specific subnetworks for classification and bounding box regression**, forming the**RetinaNet**, which achieves state-of-the-artperformance,**outperforms**[**Faster R-CNN**](https://towardsdatascience.com/review-faster-r-cnn-object-detection-f5685cb30202), the well-known two-stage detectors**.**



**Fig 1: Retina Net Detector Architecture**

[**ResNet**](https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8)is used for deep feature extraction.

**3.2.4 Yolov5**

YOLO v5 algorithm consists of fully CNN and an algorithm for post-processing outputs from neural network. CNNs are special architecture of neural networks suitable for processing grid-like data topology. The distinctive feature of CNNs which bears importance in object detection is parameter sharing. Unlike feedforward neural networks, where each weight parameter is used once, in CNN architecture each member of the kernel is used at every position of the input, which means learning one set of parameters for every location instead a separate set of parameters. This feature plays important role in capturing whole scene on the road. On Fig is presented the overview of YOLO v5 algorithm.

A screenshot of a social media post

Description automatically generated

Fig2: Yolov5 Architecture

This algorithm starts with extraction single image from video stream, in a next step extracted image is resized and that represent input to Yolo network. YOLO v3 neural network consist of 106 layers. Besides using convolutional layers, its architecture also contains residual layers, up sampling layers, and skip (shortcut) connections.

CNN takes an image as an input and returns tensor (see Fig 3) which represents:

A screenshot of a cell phone

Description automatically generated

Fig. 3. Bounding box prediction.

* Coordinates and positions of predicted bounding boxes which should contain objects,
* A probability that each bounding box contains object,
* Probabilities that each object inside its bounding box belongs to a specific class.

The detection is done on the three separate layers. Object detection done at 3 different scales addresses the issue of older YOLO neural network architectures, the detection of the small objects. Output tensors from those detection layers have the same widths and heights as their inputs, but depth is defined as: the number of bounding box properties such as width (bw), height (bh), x and y position of the box (bx, by) inside the image, 1 is the probability that box contains the detectable object (pc) and class probabilities for each of the classes (c1, c2, …, c5).

That sum is multiplied by 3, because each of the cells inside the grid can predict 3 bounding boxes. As the output from the network, we get 10 647 bounding box predictions.

This network has an ability to simultaneously detect multiple objects on the single input image. Features are learned during the network training process when the network analyses the whole input image and does the predictions. In that way, the network has knowledge about the whole scenery and objects environment, which helps the network to perform better and achieve higher precision results comparing to the methods which use the sliding window approach. The concept of breaking down the images to grid cells is unique in YOLO, as compared to other object detection solutions. Predictions whose pc is lower than 0.5 are ignored and that way, most of the false predictions are filtered out. Remaining bounding boxes are usually prediction of the same object inside the image. They are filtered out using the non max suppression algorithm.

A screenshot of a social media post

Description automatically generated

Fig 4: algorithm comparison

**CHAPTER 4: SYSTEM DESIGN**

**4.1 Procedure:**

1. Install TensorFlow-GPU and all required libraries
2. Set up Object Detection directory structure and Google collab Environment
3. Gather and label pictures
4. Generate training data
5. Create label map and configure training
6. Train object detector
7. Test it out.
8. Extract Licence plate number
9. Generate E-challan

|  |  |
| --- | --- |
|  |  |

**4.2 Flowchart of Proposed System**



Fig 5: flowchart of workflow

**CHAPTER 5: SYSTEM REQUIREMENTS**

**5.1 Major Software’s and Libraries Used:**

**5.1.1 Google Colab**

Google Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

As a programmer, we can perform the following using Google Colab:

* Write and execute code in Python
* Document your code that supports mathematical equations
* Import/Save notebooks from/to Google Drive
* Integrate PyTorch, TensorFlow, Keras, OpenCV
* Free Cloud service with free GPU

**Python:** The programming style of Python is simple, clear and it also contains powerful different kinds of classes. Moreover, Python can easily combine other programming languages, such as C or C++. As a successful programming language, it has its own advantages:

* Simple and easy to learn
* Open source
* Scalability

**5.2 OpenCV**

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. The library is cross-platform and free for use under the open-source BSD license. OpenCV supports the deep leaning framework TensorFlow, Torch/PyTorch and caffe.

**NumPy:** In Python, there is data type called array. To implement the data type of array with python, NumPy is the essential library for analysing and calculating data. They are all open source libraries. NumPy is mainly used 22 for the matrix calculation

**Pandas , Matplotlib**: **pandas**is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool,built on top of the [Python](https://www.python.org/) programming language. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

**Pillow:** Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) additional [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)) that adds support for opening, [manipulating](https://en.wikipedia.org/wiki/Image_editing), and saving many different [image file formats](https://en.wikipedia.org/wiki/Image_file_formats).

The **Python Imaging Library** adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

**5.3 TensorFlow**

TensorFlow, an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays. These arrays are referred to as "tensors".

Why TensorFlow for Object Detection:

* It allows Deep Learning.
* Known as the second-generation machine learning system, it performs numerical computations through data flow graphs.
* It is open source and free.
* It is reliable (and without major bugs). 26
* It is backed by Google and a good community.
* It is a skill recognized by many employers.
* It is easy to implement.
* With capability of running on CPUs and GPUs, it can be deployed in broad range of products of Google such as Speech Recognition, Google Photos, Gmail and even Search.

**ImageAI**: is a python library built to empower developers, researchers and students to build applications and systems with self-contained Deep Learning and Computer Vision capabilities using simple and few lines of code.

**Keras:** Keras is an open source neural network library written in Python. It can run on top of TensorFlow, Microsoft Cognitive Toolkit, or Theano Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.  Keras this can be done via the **keras. preprocessing. image. ImageDataGenerator**class.

This class allows you to: configure random transformations and normalization operations to be done on your image data during training instantiate generators of augmented image batches (and their labels) via .flow(data, labels)  These generators can then be used with the Keras model method that accepts datainputs, fit\_generator, evaluate\_generator and predict\_generator.

**5.4 Selenium Web driver:**

Selenium is an umbrella project for a range of tools and libraries that enable and support the automation of web browsers. Selenium supports automation of all the major browsers in the market using *WebDriver*. WebDriver is an API and protocol that defines a language-neutral interface for controlling the behavior of web browsers.

Web scraping is a technique which could help us transform HTML unstructured data into structed data in spreadsheet.

Refers to both the language bindings and the implementations of the individual browser controlling code. This is commonly referred to as just WebDriver. Selenium WebDriver is a [W3C Recommendation](https://www.w3.org/TR/webdriver1/).

* WebDriver is designed as a simple and more concise programming interface.
* WebDriver is a compact object-oriented API.
* It drives the browser effectively.

**CHAPTER 6: SYSTEM IMPLEMENTATION**

**Methodology**

One important element of deep learning and machine learning at large is dataset. A good dataset will contribute to a model with good precision and recall. In the realm of object detection in images or motion pictures.

* **For Motorcycle detection**: we used trained model with COCO Dataset with accuracy of 99%.
* **For Helmet Detection:** We created our own Yolov5 Model with our own dataset with 1000+ images of helmet and non-helmet riders.
* **For License plate:** We used API for extraction and web automation for getting details for E-Challan Generation.

**6.1 Model for Motorcycle Detection:**

**Coco Dataset for Motorcycle Detection:** COCO is a large-scale object detection, segmentation, and captioning dataset. This version contains images, bounding boxes " and labels for the 2017 version. Coco defines 80 classes.

COCO stands for Common Objects in Context. As hinted by the name, images in COCO dataset are taken from everyday scenes thus attaching “context” to the objects captured in the scenes. COCO was an initiative to collect natural images, the images that reflect everyday scene and provides contextual information. COCO dataset provides the labelling and segmentation of the objects in the images. A machine learning practitioner can take advantage of the labelled and segmented images to create a better performing object detection model.

**Framework: Darknet** is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation.

Download theYolov5 pre-trained model file that will be used for object detection using following link https://pjreddie.com/media/files/yolov5-.h5

If everything is working properly, the object detector will initialize for about 10 seconds and then display a window showing any objects it’s detected in the image!

**6.2 Model for Helmet Detection:**

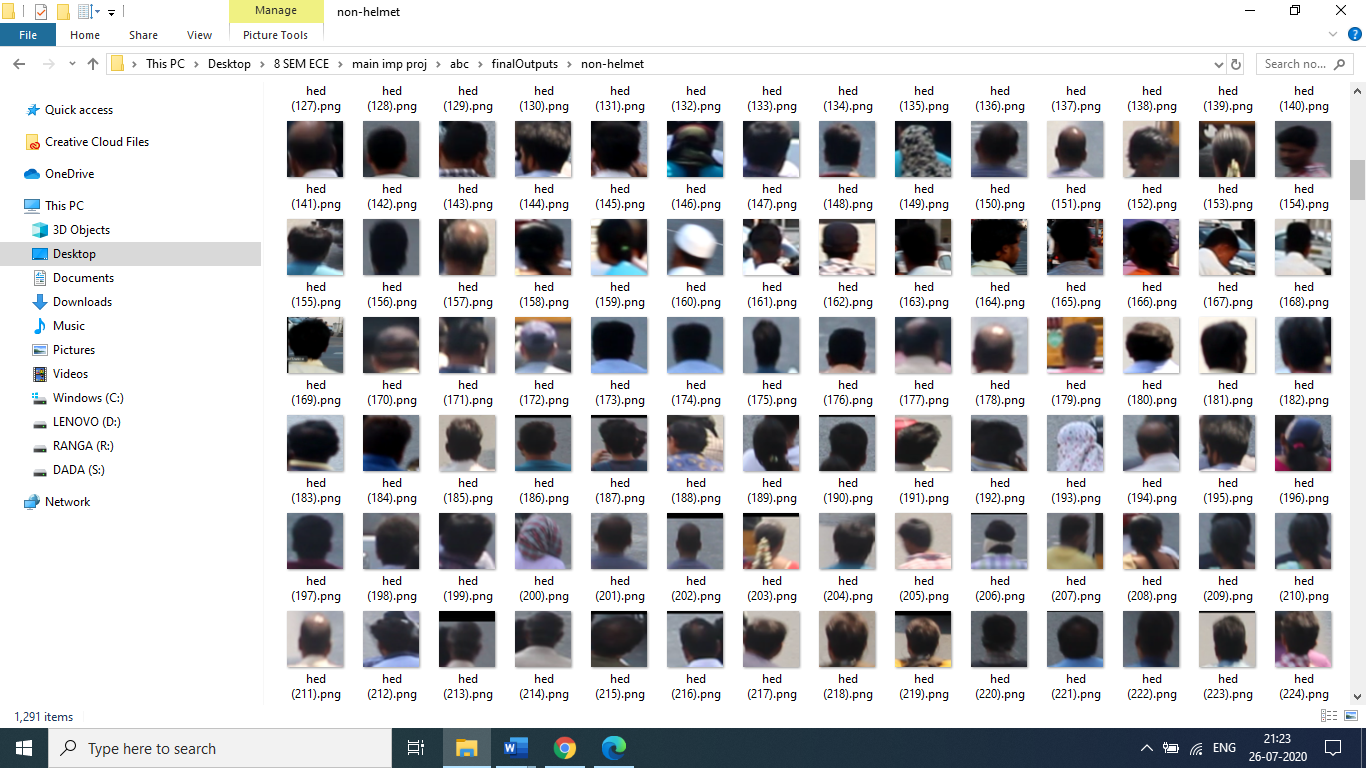
**Fig 6: FLOWCHART FOR HELEMET MODEL**



**6.2.1 Procedure for training a YOLOV5 HELMET model:**

Gathering images (Creating data set): To detect a bike rider with helmet or without helmet. We need bunch of images of bike-riders with helmet, bike-rider without helmet and bike license plate. In this project, we used 1000+ images.

A screenshot of a computer

Description automatically generated 

HELMET IMAGES NON-HELMET IMAGES

Fig 7: Helmet dataset

**Label Images:**

Label the all images with the help of **LableImg tool**. In this project, **Helmet class** was created with the help ofLableImg tool. Create .xml file corresponding to each image with the above following categories of classes

Now that our dataset labels are in the required format, we need to create a train-test split. I chose to create a test set containing 10% of the images in the dataset. Configuring YOLO with your dataset. Now that we have created our train and test sets, we need to make some changes to train the YOLO model on the dataset.

**Training:**

Now that our dataset is ready to use, we can begin training. Before we start, compile the darknet repository with the make command. To compile with specific options, such as GPU, CUDNN and OPENCV. This will create a darknet executable.

Trained weights for model. You can set other parameters (learning rate, momentum, weight decay etc by editing the corresponding lines). Finally, model is ready to use.

**6.3 Number plate Detection:**

**Platerecognizer**: is an open source Automatic License Plate Recognition library written in C++ with bindings in C#, Java, Node.js, and Python. The library analyses images and video streams to identify license plates. The output is the text representation of any license plate characters.

**Plate Recognizer Snapshot API!** You can use our API to access our API endpoints, which can read license plates from images. Plate Recognizer provides accurate, fast, developer-friendly Automatic License Plate Recognition (ALPR) software that works in all environments.

The software can be used in many ways:

1. Recognize license plates from camera streams. The results are [browsable, searchable, and can trigger alerts](http://doc.openalpr.com/watchman.html#web-server). The data repository can be in the cloud or stored entirely within your on-site network.
2. Recognize license plates from camera streams and [send the results to your own application](http://doc.openalpr.com/watchman.html#on-premises).
3. Integrate license plate recognition into your application [directly in-code.](http://doc.openalpr.com/api.html#language-bindings)

**6.4 Automated E-Challan Generation:**

Further this Detected LP Numbers are injected to **RTO Database** for extracting the further details of the Violator. Now, we do **Automation of web Browser,** to get the details of offender from **RTO Database.**

**Selenium** is an umbrella project for a range of tools and libraries that enable and support the automation of web browsers. Selenium supports automation of all the major browsers in the market using ***WebDriver*.**

WebDriver is an API and protocol that defines a language-neutral interface for controlling the behavior of web browsers. **Web scraping** is a technique which could help us transform HTML unstructured data into structed data in spreadsheet.

With **Pillow Image Library,** with extracted details saved in excel sheet an automatic E-challan is generated with details Including Date and time & further it can be sent through message, mail or post

**CHAPTER 7: WORKFLOW & SYSTEM TESTING**

**7.1 Account Setup Gdrive**

In **Google collab** Editor,

First, upload all required data into google drive and now sync Gdrive with collab editor.

A screenshot of a computer screen

Description automatically generated

Create **PROJECT** Folder:

* Motorcycle\_Detection\_Model File
* Helmet Detection Model File
* Final Outputs (Folder)

Subfolders

* Full Frame Image
* Bikes Image
* Rider Image
* Challan (Folder)
* Chrome WebDriver.exe

**Importing Required Libraries**

A screenshot of a computer screen

Description automatically generated

* In this section we explain different processing steps. initial phase, frames are collected at regular intervals from video file and passed into detection model for processing.
* All these techniques are subjected to predefined conditions and constraints, especially the license plate number extraction part. Since, this work takes video as its input, the speed of execution is crucial. We have used above said methodologies to build a holistic system for both helmet detection and license plate number extraction.

**7.2 Detection of Motorcycle**

The frame chosen is given as input to YOLOv5 Motorcycle detection model, where the classes to be detected are “**Motorcycle**‟. At the output, image with required class detection along with confidence of detection through bounding box and probability value is obtained as shown in the Fig. 1 (a) and Fig. 1(b)

**A person riding a bicycle on the side of a road

Description automatically generatedA group of people walking down the street

Description automatically generated**

The details of these extracted images which is stored in a dictionary which can be later used for further processing.

**Output for each object:** [{**'name':** 'motorcycle', **'percentage\_probability**':89.4,

**'box\_points':** [104, 84, 265, 400]}

Fig. 7.2.1(b) Frame-2

Case 2

Front View

Rear View

**Fig**: Frame with ‘**motorcycle’** classes detected

Fig. 7.2.1(a) Frame-1

Case 1

Fig. 1(b) Frame 2

Case 2

Front View

With the help of functions given by **Image AI** library, only the detected objects are extracted as shown below, and stored as separate images and named with class name and image number in order.

We crop these detected frames in 3 formats:

1. Full Image with motorbike and rider
2. Bike Image
3. Rider Image

For example, it will be saved as Full-1, Full-2, etc. || Bike-1, Bike-2…etc. || Rider-1, Rider-2…etc.

A person riding on the back of a motorcycle

Description automatically generatedA picture containing outdoor, road, motorcycle, street

Description automatically generatedA close up of a person

Description automatically generatedA picture containing sitting, snow, covered, riding

Description automatically generatedA picture containing motorcycle, sitting, small, holding

Description automatically generatedA picture containing outdoor, person, people, standing

Description automatically generatedA person riding on the back of a motorcycle

Description automatically generatedA picture containing outdoor, road, motorcycle, street

Description automatically generated

**Rider-1.jpg Bike-1.jpg Full-1.jpg Rider-2.jpg Bike-2.jpg Full-2.jpg**

Fig 8 : Helmet ROI cropping

**7.3 Detection of Helmet:**

Once the **Motorcycle** class is obtained, the Rider images is given as input to **Helmet detection model**. While testing the helmet detection model, some false detections were observed. So, the person image was cropped to get only top one-fourth portion of image, as shown in Fig. 2 (Rider.jpg). This ensures that false detection cases are eliminated as well as avoid cases leading to wrong results when the rider is holding helmet in hand while riding or keeping it on motorcycle while riding instead of wearing.

Now two cases Arise:

**Case 1:** When the motorcycle rider is wearing helmet

**Case 2:** When the motorcycle rider is not wearing helmet

After applying cropped image to helmet detection model, output is as shown.

The bounding box around helmet along with the detection probability is displayed as shown in (Rider-2.jpg) .

As the rider wearing helmet in Case 2, no further processing is necessary. Since in Case 1, rider is not wearing helmet, no bounding box is created.

**HELMET DETECTION MODEL**

Rider-1.jpg Rider-2.jpg

A screenshot of a social media post

Description automatically generatedA screenshot of a social media post

Description automatically generated

CASE -1 CASE-2

Fig 9: Helmet yolov5 prediction

After applying cropped image to helmet detection model, output is as shown. The bounding box around helmet along with the detection probability is displayed as shown in (Rider-2.jpg). As the rider wearing helmet in Case 2, no further processing is necessary. Since in Case 1, rider is not wearing helmet, no bounding box is created.

**7.4 Detection of Number Plate:**

If the helmet is found, there is no need for this step. However, if the helmet is not found, then the motorcycle image is given as input to license plate detection phase. Where the image is passed to this library to detect the License Plate Number and Return it and Store it for Further use.

**Platerecognizer** is an open source *Automatic License Plate Recognition* library. The library analyzes images and video streams to identify license plates. The output is the text representation of any license plate characters. Further this Detected LP Numbers are injected to **RTO Database** for extracting the further details of the Violator.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Fig 10: Number plate recognition

Now, we do **Automation of web Browser,** to get the details of offender from **RTO Database.**

Selenium supports automation of all the major browsers in the market using ***WebDriver*.** WebDriver is an API and protocol that defines a language-neutral interface for controlling the behavior of web browsers.

**Web scraping** is a technique which could help us transform HTML unstructured data into structed data in spreadsheet.

A screenshot of a cell phone

Description automatically generatedA screenshot of a computer

Description automatically generated

Automated Browsing

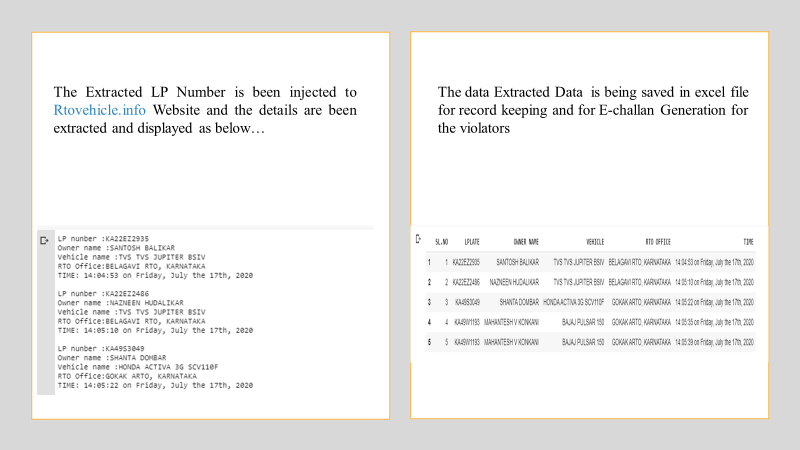


Fig 11: RTO details extraction

**7.5 E-Challan Generation:**

With **Pillow Image Library,** with extracted details saved in excel sheet an automatic E-challan is generated with details Including Date and time & further it can be sent through message, mail or post

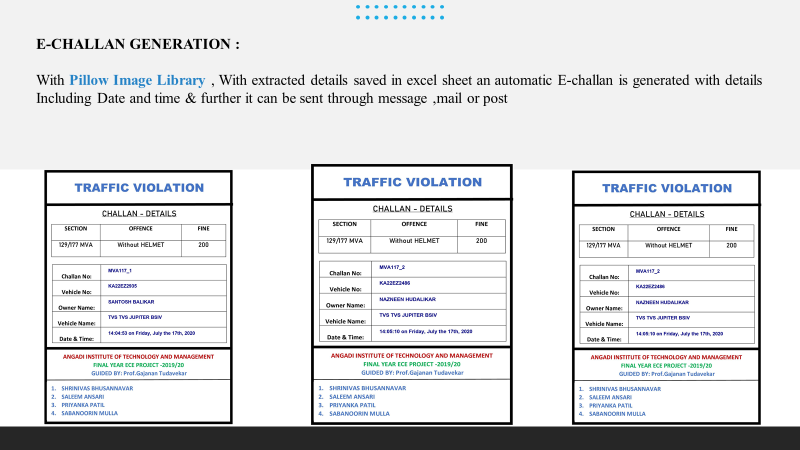


Fig 12: E-challan generation

**CONCLUSION:**

A Non-Helmet Rider Detection system is developed where a video file is taken as input. If the motorcycle rider in the video footage is not wearing helmet while riding the motorcycle, then the license plate number of that motorcycle is extracted and displayed. Object detection principle with YOLO architecture is used for motorcycle, person, helmet, and license plate detection. Web api is used for license plate number extraction if rider is not wearing helmet. Not only the characters are extracted, but also the frame from which it is also extracted so that it can be used for other purposes. All the objectives of the project are achieved satisfactorily.

**REFERENCES:**

[1] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov 1998

[2] O. Russakovsky et al., “ImageNet large scale visual recognition challenge”, International Journal of Computer Vision (IJCV), Dec 2015, 115(3), 211–252.

[3] H. Li and C. Shen, “Reading car license plates using deep convolutional neural networks and LSTMs”, arXiv preprint arXiv:1601.05610, 2016

[4] M. Jaderberg, K. Simonyan, A. Vedaldi and A. Zisserman, “Reading text in the wild with convolutional neural networks”, International Journal of Computer Vision (IJCV), 2016, 116(1), 1–20

[5]. K. Dahiya, D. Singh and C. K. Mohan, "Automatic detection of bike-riders without helmet using surveillance videos in real-time," 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, 2016, pp. 3046-3051.

[6]. C. Vishnu, D. Singh, C. K. Mohan, and S. Babu, "Detection of motorcyclists without helmet in videos using convolutional neural network," 2017 International Joint Conference on Neural Networks (IJCNN)

[7] Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.