CHRIST
(DEEMED TO BE UNIVERSITY)
BENGALURU · INDIA

# Feasible Real Time Helmet Detection Using Raspberry PI - SLES

By,

| | |
|---|---|
| Lynford Valentino D’souza | (2047221) |
| Yash Himmat Kataria | (2047235) |
| Sumi Thomas | (2047259) |

Under the guidance of

Dr Nizar Banu K

Feasible Real Time Helmet Detection Using
Raspberry PI - SLES project report submitted in partial fulfilment of
the requirements of 3 semester MCA, CHRIST
(Deemed to be University)
October - 2021

CHRIST
(DEEMED TO BE UNIVERSITY)
BENGALURU · INDIA

# CERTIFICATE

*This is to certify that the report titled* ***Feasible Real Time Helmet Detection Using Raspberry PI - SLES*** *is a bona fide record of work done by* ***Lynford Valentino D'souza (2047221)*** *of CHRIST (Deemed to be University), Bangalore, in partial fulfilment of the requirements of 3 Semester MCA during the year 2021.*

**Head of the Department** **Project Guide**

Valued-by:

1.

Name : Lynford Valentino Dsouza
Register Number : 2047221
Examination Centre : CHRIST (Deemed to be University)

2.

Date of Exam:

# ACKNOWLEDGMENT

First of all, we thank the lord almighty for his immense grace and blessings showered on us at every stage of this work. We are greatly indebted to our Head of the Department of Computer Science, Prof. Joy Paulose, CHRIST (Deemed to be University) for providing the opportunity to take up this project as part of our curriculum.

We are greatly indebted to our MCA Course Coordinator, Prof. Tulasi B, for providing the opportunity to take up this project and for helping with her valuable suggestions.

We are deeply indebted to our project guide Dr. Nizar Banu PK, for her assistance and valuable suggestions as a guide. She made this project a reality.

We express our sincere thanks to Dr Smitha Vinod and Roseline Mary R, lecturers of the Department of Computer Science, CHRIST (Deemed to be University), for their valuable suggestions during the course of this project. Their critical suggestions helped us to improve the project work.

Acknowledging the efforts of everyone, their chivalrous help in the course of the project preparation and their willingness to corroborate with the work, their magnanimity through lucid technical details lead to the successful completion of our project.

We would like to express our sincere thanks to all our friends, colleagues, parents and all those who have directly or indirectly assisted during this work.

# ABSTRACT

With the ever-increasing population of today's world, the obvious outcome was an increase in the number of vehicles on road. Due to this increase in traffic and the increasing Fuel prices, Two-Wheelers have become the most popular mode of transport. Two-Wheelers have the significant disadvantage of safety, and therefore not surprisingly, the most common road accidents involve motorcycles and maximum cases result in fatal injuries.

To combat this issue, helmets were introduced which have been shown to be extremely effective in saving lives of riders. Witnessing the practicality of helmet, several state governments have made it a punishable offense to ride the bike without a helmet. The technique used to enforce the law includes Police Officers spending a significant proportion of their time spotting riders without helmets, stopping them and issuing them with traffic violation tickets. Besides the obvious time wastage, the incapability of catching each and every violator, and lack of police force, citizens are ignorant of the situation and a significant portion of the population avoid using the helmet. Even though in recent times the government has taken several steps such as increasing the fine amount, having more officers on road and so on, its efforts were futile and its effects only lasted for a short period.

This project is aimed at getting rid of all such disadvantages by automating the entire process and therefore freeing up the time of police officers, reducing traffic issues and ensuring the safety of citizens. The system will an object detection model for classifying riders not wearing helmets. The model once trained is converted to a TensorFlow Lite format to improve overall efficiency and effective cost. The proposed system will automatically detect bike riders not wearing helmets and transfer it to a website where it can be viewed by the law enforcement officers. It uses a Raspberry Pi 4 along with a web camera and processing is done on board and in real time.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1: INTRODUCTION

## 1.1 PROJECT DESCRIPTION

India has the worst road safety standards in the world, a fact repeatedly outlined in World Health Organization reports and backed up by the government's own reports. The NCRB data shows that as many as 43,540 people were killed in accidents involving Two-Wheelers in a single year. More Indians die each and every year in road accidents than the total casualties suffered by India's armed forces in all the wars fought since independence.

Helmets were therefore introduced which have been shown to be extremely effective in saving lives of riders. Seeing the obvious use of the helmet, Governments have made it punishable offense to ride a bike without helmet.

To enforce the Helmet wearing Law, Police Officers spend a significant proportion of their time spotting riders without helmets, stopping them and issuing them with traffic violation tickets. This system obviously has several drawbacks. Besides the obvious time wastage, the incapability of catching each and every violator, and lack of police force, citizens are ignorant of the situation and a significant portion of the population avoid using the helmet. Besides that, catching hold of violators includes the cops usually causing a commotion by coming in between the road to stop the violator. The bikes in turn speed up and narrowly avoid accidents. This in turn causes traffic jams. In recent times the government has taken several steps to implement road such as increasing the fine amount, having more officers on road, having awareness programs and so on. But its efforts were futile and its effects only last for a short period.

The proposed system at its core is basically an automated surveillance system along with a companion android application. The automated surveillance system is made up of a camera and a Raspberry PI 4, which has the necessary computing power to perform object detection and spot Two-Wheeler riders not wearing Helmets and send their images to an online cloud database. The application provides Police officers the ability to verify the violators collected by the object detection system and send them challans using their email or phone number.

## 1.2 EXISTING SYSTEM

Most existent methods suffer from several problems such as occlusion of objects and varying illumination conditions. They tried to address it by using SVM [14], [15], [16] for classification between bikers and non-bikers and bikers wearing helmets and not, which made localization of occluded objects easier. But for any of that to work efficiently, we need to have good quality features from the bikers to classify accurately which is difficult using HOG [12] or LBP [13] or SIFT [11] on images with less pixels. Also, most of the Deep Learning-based approaches use CNNs but ultimately don't work well real time detection and are computationally expensive.

Limitations of Existing System:

- The proposed systems which do not use specialized hardware usually have low accuracy or work only under certain conditions.
- The systems which use modern machine learning techniques are power hungry and this creates a bottleneck on the feasibility of deploying such systems in large scale.
- Systems which use special hardware have better accuracy, performance and versatility. These systems are not implemented due to increased cost of customized hardware which is not economically possible considering the large-scale requirement.
- Most of the suggested methods also don't work in real time.

## 1.3 OBJECTIVES

The aim of this project is to develop a surveillance system which is efficient and has very low power usage overall. Its needs a companion application with Easy-to-use interface for Law enforcements officers. It should be a truly feasible system with small budget and energy requirements. It should effectively make law enforcement officers free for better traffic management and flow. It should be easy to maintain and be able to work on solar power thus making it eco-friendly. It should be extensible so that new features can be added eventually.

## 1.4 PURPOSE, SCOPE AND APPLICABILITY

### 1.4.1 PURPOSE

The project is built to automate the process of catching violators of traffic laws and enable police officers to verify the violators and send challans to them.

- The system is developed for Traffic law enforcement.
- The system's companion app allows Police officers to register and login into the system thus making them a user.
- The admin has the major authority over all the modules.
- The admin mainly maintains the Police and violators details and can generate or view reports.
- The user can verify the violators.
- The user can send challans to the violators via email or phone.
- The system can be used to monitor the overall violators for a given area.
- Reports related the challans sent and violators verified can be generated when required.

### 1.4.2 SCOPE

The scope of the project is within the police organization. The limitation here is that the application can only be used by police officers with a valid phone number and email. The project contains 2 parts: a) the surveillance system which is a camera along with the necessary software codded into a Raspberry PI 4. b) The companion application – SLES which is used by the Police to verify violators. Only Police officers who are verified by the app can verify and assign challans to the violators.

### 1.4.3 APPLICABILITY

The application is applicable to any State or National police of any given country. It is mainly used by Police officers who are the primary users of the application. They will use the application to verify the violators and assign challans. They can also view statistics of the violators for a given location and the challans sent per day for a given location.

## 1.5 OVERVIEW OF THE REPORT

The report explains in detail the system requirements in which the problem statement, hardware requirements, software requirements and constraints are briefed followed by the system design in Chapter 3 containing architecture, model and database designs. In Chapter 4 the implementation details are included. The various test cases are given in Chapter 5 and finally the conclusion of the project is given along with references.

# CHAPTER 2: SYSTEM ANALYSIS AND REQUIREMENTS

## 2.1. PROBLEM DEFINITION

India has the worst road safety standards in the world, a fact repeatedly outlined in World Health Organization reports and backed up by the government's own reports. The NCRB data shows that as many as 43,540 people were killed in accidents involving Two-Wheelers in a single year. More Indians die each and every year in road accidents than the total casualties suffered by India's armed forces in all the wars fought since independence.
In case of fatality, State Highways and other roads accounted for 26.9 percent and 37.1 percent, respectively. The current system has too many problems. Trauma to the brain can occur as a result of an impact, which can cause a concussion or open skull fracture, or a jarring motion, such as a quick turn or sudden stop. Even seemingly mild head injuries, where you don't lose consciousness, can cause permanent behavioral and cognitive problems, such as memory loss, inability to concentrate, sleep disorders and, in some cases, permanent disability or death
Helmets, reduce risk of serious brain injury resulting into death, impact energy is absorbed by the helmet, thus saving head. In 91 per cent cases of two-wheeler accidents, a non-helmet rider was hurt, i.e. they either died or were grievously injured or faced minor injuries. For every hundred two-wheeler accidents in Tamil Nadu, 126 people were hurt. Tamil Nadu is followed by Maharashtra, Rajasthan, West Bengal, Himachal Pradesh, Jharkhand and Karnataka where at least one two-wheeler rider who didn't wear a helmet faced at least one injury. Governments, punishable offense, ride a bike without helmet.Police Officers, significant proportion of time, spotting riders without helmets, stopping them, traffic violation tickets, incapable of catching everyone, disruption of the flow of traffic. Besides that, catching hold of violators includes the cops usually causing a commotion by coming In between the road to stop the violator. The bikes in turn speed up and narrowly avoid accidents. This in turn causes traffic jams which is very ironic all things considered. In recent times the government has taken several steps to implement road such as increasing the fine amount, having more officers on road, having awareness programs and so on. But its efforts were futile and its effects only last for a short period.

## 2.2. REQUIREMENTS SPECIFICATION

The users require a stable internet connection. The administrator must feed the details of the violators along with images for the application to monitor the violators accurately. The police officers logging into the system must verify the image of the violators in order to send the challan to the respective violator.

### 2.3. BLOCK DIAGRAM

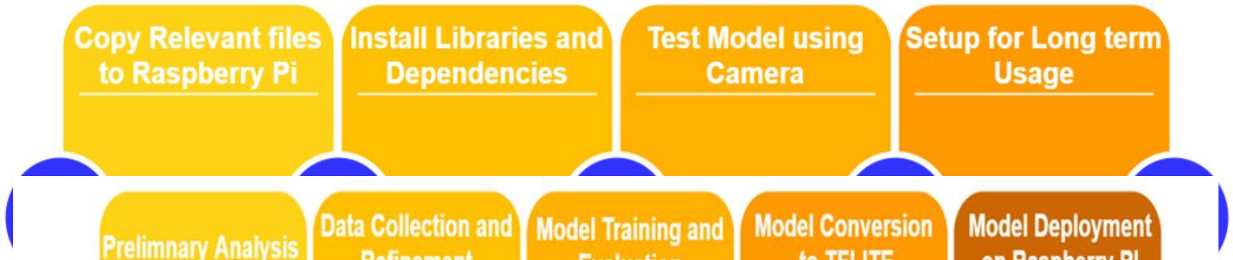

Copy Relevant files to Raspberry Pi
Install Libraries and Dependencies
Test Model using Camera
Setup for Long term Usage

**Fig 2.1: Object Detection Process**

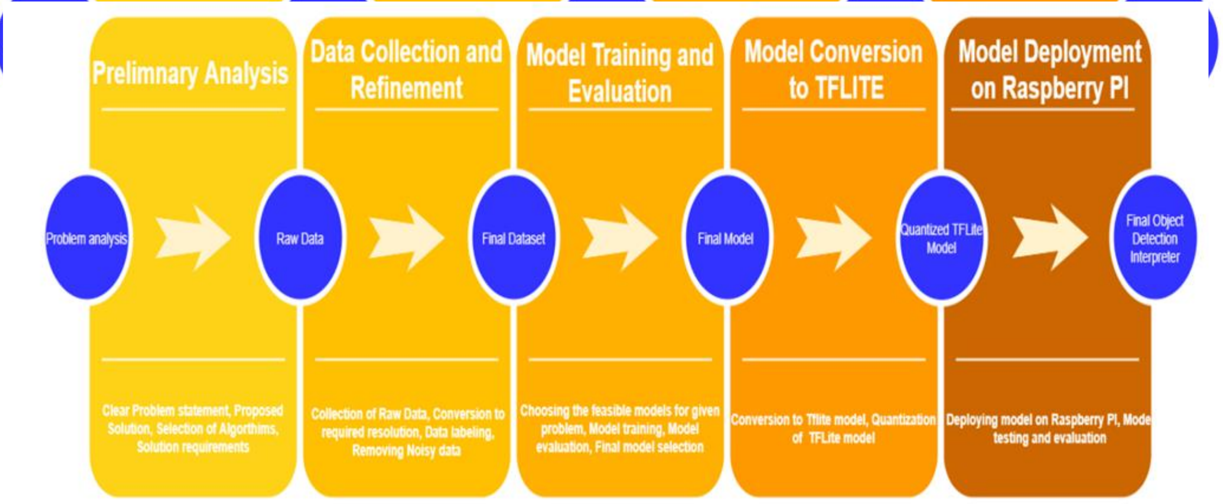

Prelimnary Analysis
Data Collection and Refinement
Model Training and Evaluation
Model Conversion to TFLITE
Model Deployment on Raspberry PI
Problem analysis
Raw Data
Final Dataset
Final Model
Quantized TFLite Model
Final Object Detection Interpreter
Clear Problem statement, Proposed Solution, Selection of Algorthims, Solution requirements
Collection of Raw Data, Conversion to required resolution, Data labeling, Removing Noisy data
Choosing the feasible models for given problem, Model training, Model evaluation, Final model selection
Conversion to Tflite model, Quantization of TFLite model
Deploying model on Raspberry PI, Model testing and evaluation

**Fig 2.2: Image Capture Process**

### 2.4. SYSTEM REQUIREMENTS

#### 2.4.1. USER CHARACTERISTICS

| User | Characteristics |
|---|---|
| User Interface | Police officers can login into the mobile app through their credentials.<br><br>They can verify the violators based on specific locations and accordingly the challan would be sent to that specific violator. |

#### 2.4.2. SOFTWARE AND HARDWARE REQUIREMENTS

##### 2.4.2.1. Software Requirements

| | |
|---|---|
| Operating System | Windows 10, Raspberry Pi OS, Android |
| Software Selection | Flutter, Firebase/Amazon web services, Visual Studio, Jupyter, Anaconda |
| Programming Languages | HTML, CSS, JAVASCRIPT, JAVA, XML, DART, PYTHON |
| Web Browser | Google chrome |

### 2.4.2.2. Hardware Requirements

| Sr No. | Requirement Name | Description |
|---|---|---|
| 1 | Processor | Quad-Core 64-bit Broadcom 2711, Cortex A72 Processor (Raspberry Pi), Cortex A-53 cores and above (Mobile app), Intel i7 core. |
| 2 | ROM | 2 - 50 GB |
| **3** | RAM | 1 - 16 GB |
| **4** | Processor Speed | 1.3 GHz and above |
| **5** | System Type | 64-bit Operating System. |

|  |  |  |
|---|---|---|
| **6** | Internet Connection | 512kb/s |

### 2.4.3. NON-FUNCTIONAL REQUIREMENTS

- **Usability:** The interface should use terms and concepts which are drawn from experience of the people who will make most of the system.
- **Efficiency**: The system should provide easy and fast access without consuming more cost.
- **Reliability:** Users should never run into a situation where the system is not able to handle any unexpected circumstances. The system behaviour must be well defined.
- **Security**: Users information must be as confidential as possible.
- **Maintainability:** There will be no maintenance required for the software. The database is provided by the end-user and therefore is maintained by the user itself.
- **Availability:** The system is available where it is installed and is running.

### 2.4.4. CONSTRAINTS

The set of constraints for the system to work accordingly are,

#### 2.4.4.1 ASSUMPTIONS

● The code should be free from compilation errors/ syntax errors.

● The systems interface must be simple to understand

● The camera should be placed on electric poles to capture image of the violator

#### 2.4.4.2 DEPENDENCIES

● All necessary hardware and software requirements are available

● End users must have basic knowledge about mobile apps

● Users should have a stable internet connection

### 2.5.1 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and the boundaries of the system as a whole.

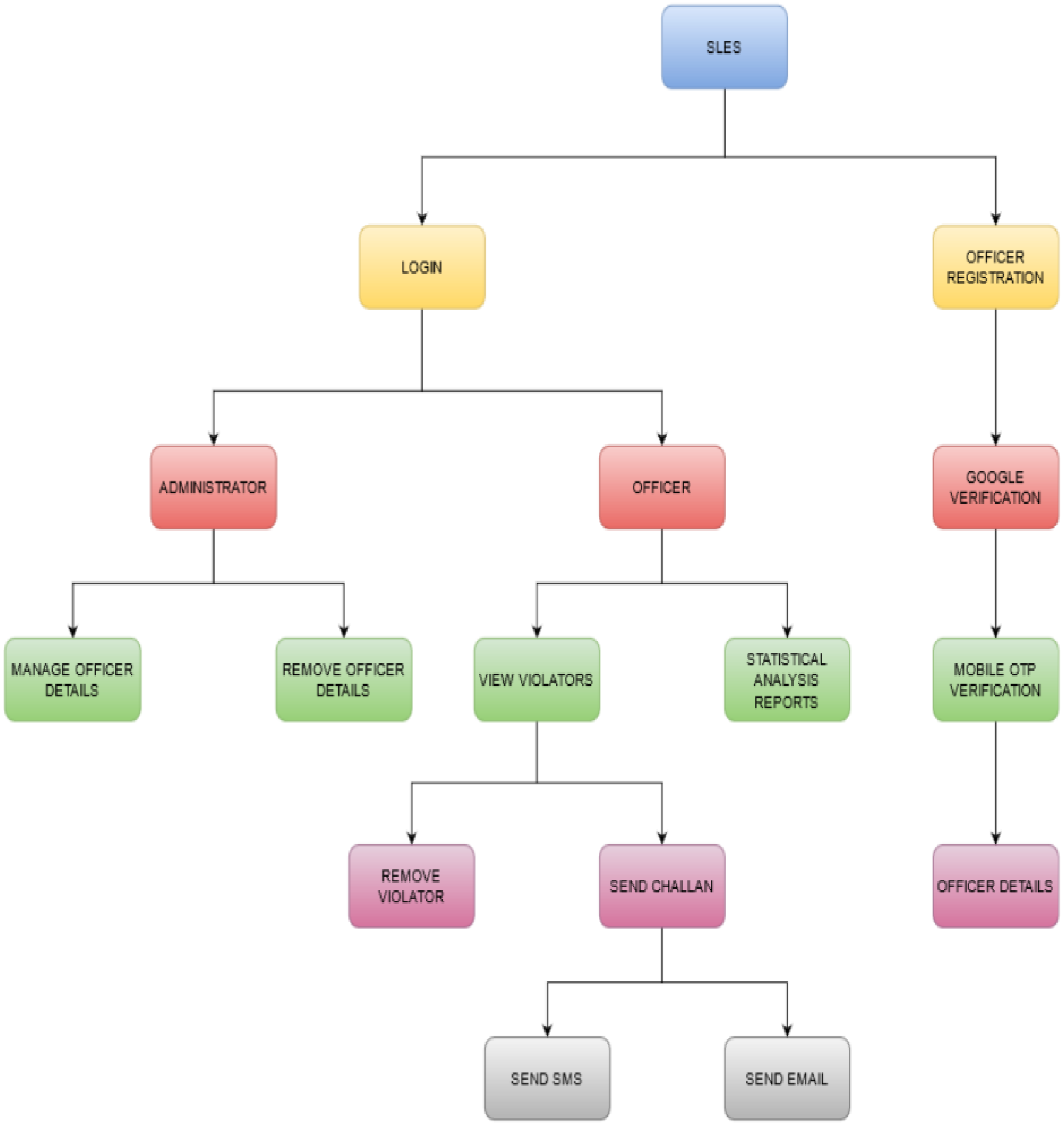


Fig 2.3 : Main Application Flow Diagram

This chapter discusses the required features, functionalities and operations in detail. The description includes rules, process diagrams, pseudocode and other relevant documentation.

## 3.1. SYSTEM ARCHITECTURE

Flutter is an open source framework to create high quality, high performance mobile applications across mobile operating systems - Android and iOS. It provides a simple, powerful, efficient and easy to understand SDK to write mobile application in Google's own language, *Dart*. This tutorial walks through the basics of Flutter framework, installation of Flutter SDK, setting up Android Studio to develop Flutter based application, architecture of Flutter framework and developing all type of mobile applications using Flutter framework.

Firebase is a backend platform for building Web, Android and IOS applications. It offers real time database, different APIs, multiple authentication types and hosting platform. This is an introductory tutorial, which covers the basics of the Firebase platform and explains how to deal with its various components and sub-components.

## 3.2 MODULE DESIGN

1. Registration:

   The public users will undergo the registration procedure by specifying the identity proof and obtain a username and password, which is used for authentication.

2. Login:

   This module contains the login. The admin can access his admin dashboard and perform all functionalities and also control user groups based on their access to the system. The user and can use the login module to access all the functionalities.

3. Object detection in Raspberry PI:

   Riders not wearing helmets are recognized by the object detection algorithm and their photos are sent to an online database using IOT.

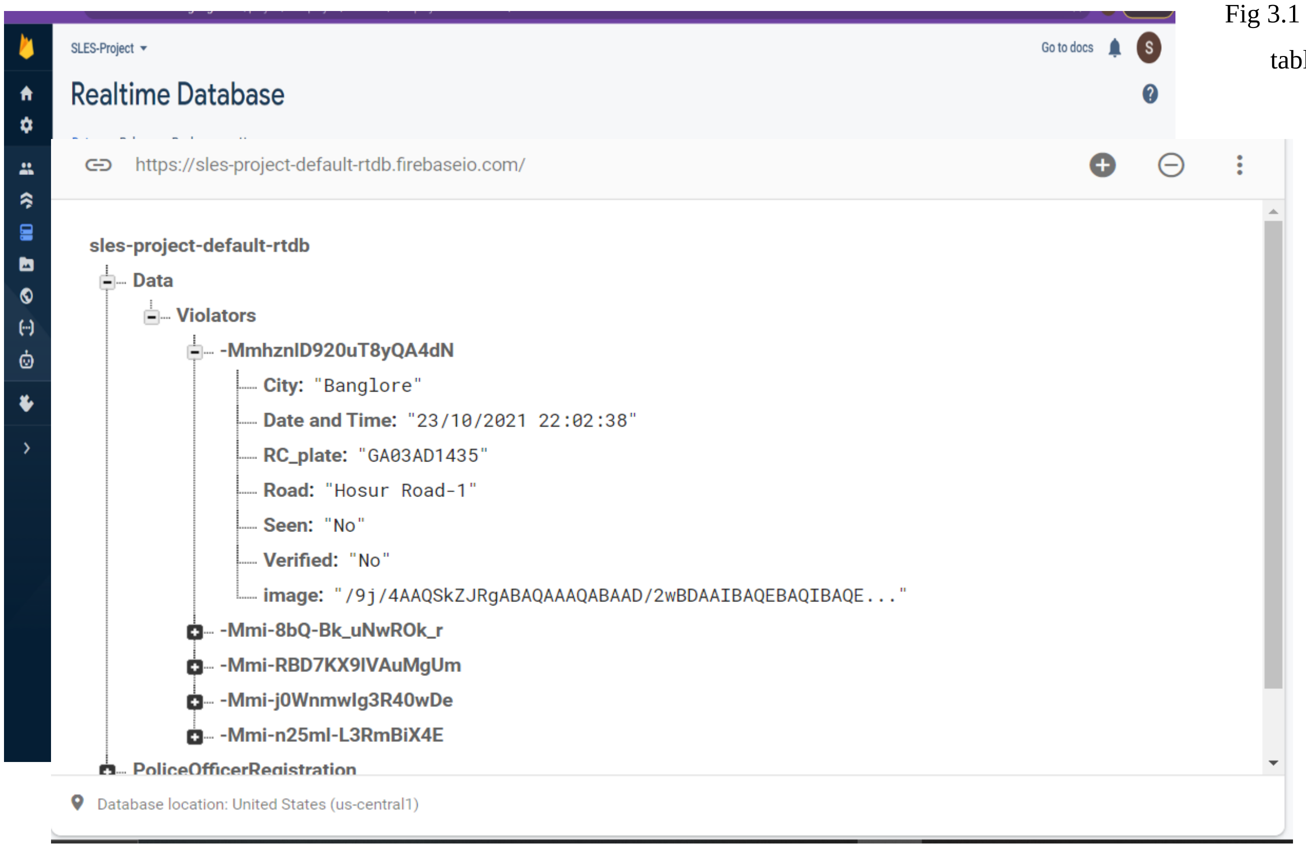
4. View violators:

   Law enforcement authorities can view the violators in real time from different locations. Their time and date and relevant information will also be displayed.

5. Send Messages:

The main functionality of this module is to send a challan message to rider of the bike if approved by the law enforcement officer.

### 3.3.1 TABLES AND RELATIONSHIPS



SLES-Project

Realtime Database

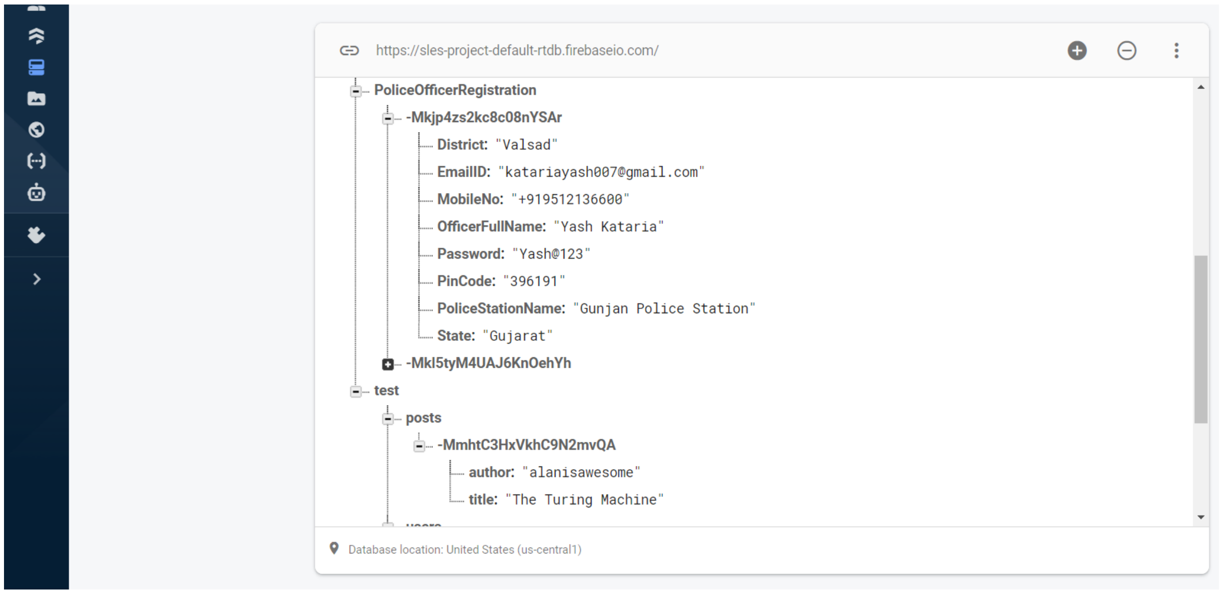
https://sles-project-default-rtdb.firebaseio.com/

```
sles-project-default-rtdb
  Data
    Violators
      -MmhznlD920uT8yQA4dN
        City: "Banglore"
        Date and Time: "23/10/2021 22:02:38"
        RC_plate: "GA03AD1435"
        Road: "Hosur Road-1"
        Seen: "No"
        Verified: "No"
        image: "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAIBAQEBAQIBAQE..."
      -Mmi-8bQ-Bk_uNwROk_r
      -Mmi-RBD7KX9lVAuMgUm
      -Mmi-j0Wnmwlg3R40wDe
      -Mmi-n25ml-L3RmBiX4E
  PoliceOfficerRegistration
```

Database location: United States (us-central1)

Fig 3.1 List of tables

Fig 3.3: Violator Details



```
https://sles-project-default-rtdb.firebaseio.com/
PoliceOfficerRegistration
  -Mkjp4zs2kc8c08nYSAr
    District: "Valsad"
    EmailID: "katariayash007@gmail.com"
    MobileNo: "+919512136600"
    OfficerFullName: "Yash Kataria"
    Password: "Yash@123"
    PinCode: "396191"
    PoliceStationName: "Gunjan Police Station"
    State: "Gujarat"
  -MkI5tyM4UAJ6KnOehYh
test
  posts
    -MmhtC3HxVkhC9N2mvQA
      author: "alanisawesome"
      title: "The Turing Machine"
Database location: United States (us-central1)
```

Fig 3.3: Registration Details and Test Data

### 3.3.2 DATA INTEGRITY AND CONSTRAINTS

Integrity constraints are a set of rules. It is used to maintain the quality of information. Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.

In the database we have ensured that the Integrity Constraint types are well maintained:

**Domain Constraints** - All attributes are given a valid set of values based on their data types.

**Entity Integrity Constraints** - Primary for each table is well defined and not null as they are used to identify a row.

**Referential Integrity Constraints -** All the foreign keys are well defined in the necessary tables.

**Key Constraints -** These are the set of attributes that are capable of uniquely identifying the table among which one is chosen as the primary key.

## 3.4 INTERFACE AND PROCEDURAL DESIGN

The application generates a report in the App with two columns i.e the name and timestamp of login for each user.

# 4: IMPLEMENTATION

## 4.1 Implementation Approaches

### 4.1.1 Block Diagram of Project:

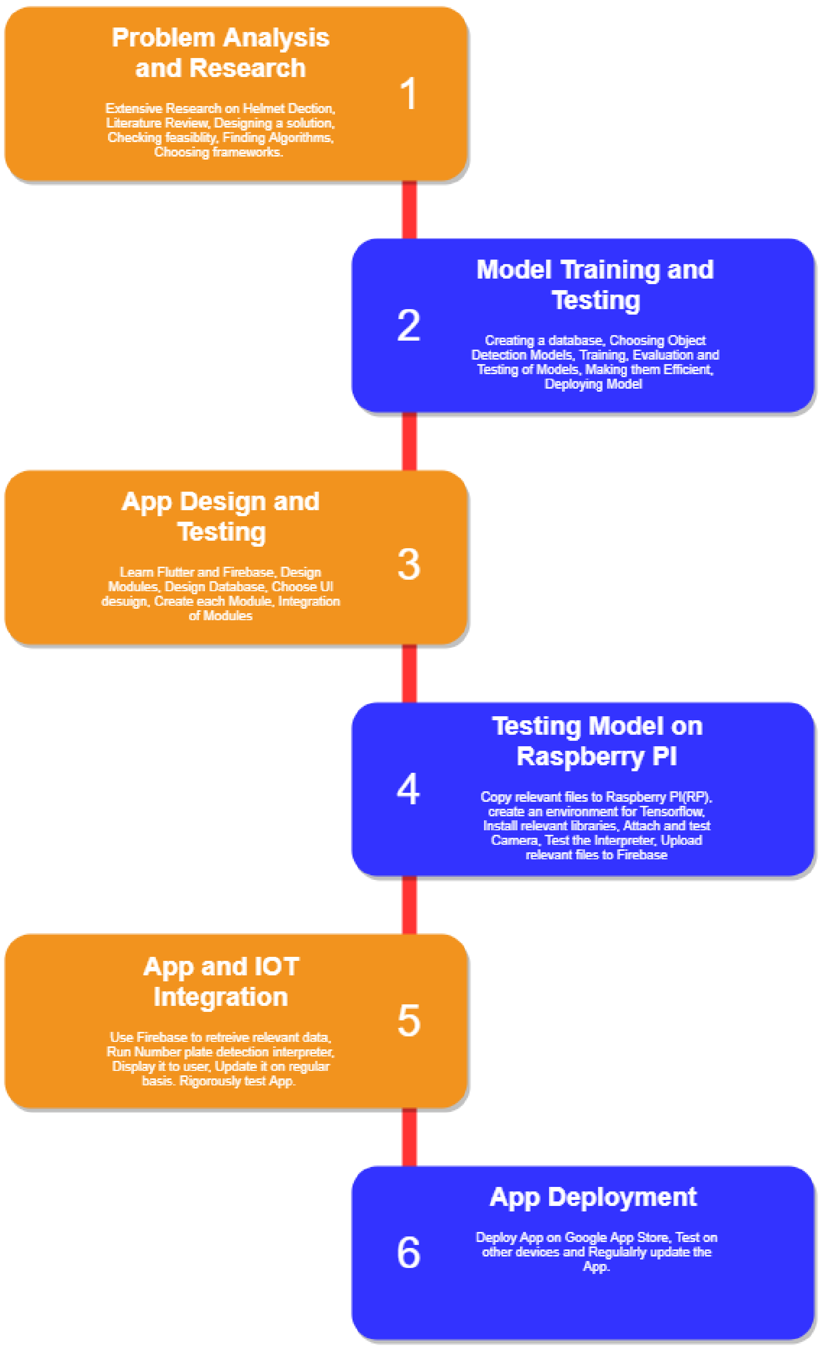

Problem Analysis and Research
Extensive Research on Helmet Dection, Literature Review, Designing a solution, Checking feasiblity, Finding Algorithms, Choosing frameworks.
1
Model Training and Testing
Creating a database, Choosing Object Detection Models, Training, Evaluation and Testing of Models, Making them Efficient, Deploying Model
2
App Design and Testing
Learn Flutter and Firebase, Design Modules, Design Database, Choose UI desuign, Create each Module, Integration of Modules
3
Testing Model on Raspberry PI
Copy relevant files to Raspberry PI(RP), create an environment for Tensorflow, Install relevant libraries, Attach and test Camera, Test the Interpreter, Upload relevant files to Firebase
4
App and IOT Integration
Use Firebase to retreive relevant data, Run Number plate detection interpreter, Display it to user, Update it on regular basis. Rigorously test App.
5
App Deployment
Deploy App on Google App Store, Test on other devices and Regulalrly update the App.
6

Fig-4.1:DATA FLOW BLOCK DIAGRAM

### 4.1.2 Data Collection and Refinement:

### 4.1.2 Data Collection and Refinement:

#### Data Collection:

The search for a dataset for the problem i.e., Helmet Detection began online. Only one dataset was found which was relevant to the problem, which was from IIT Hyderabad. But downloading that dataset needed special permission and therefore it was not available to us. So, a custom dataset had to be made. In total 3 datasets were prepared, out of which dataset 3 was used due to the Object detection model's excellent performance after using it for training.

The first dataset was basically a set of 1080p videos at 30fps which were recorded on the Oppo Reno Smart Phone. The footage was recorded at a single location from 2 different angles. After recording the footage, screenshots were clicked which contained the required information as shown in Figure 2. This made up the first dataset.

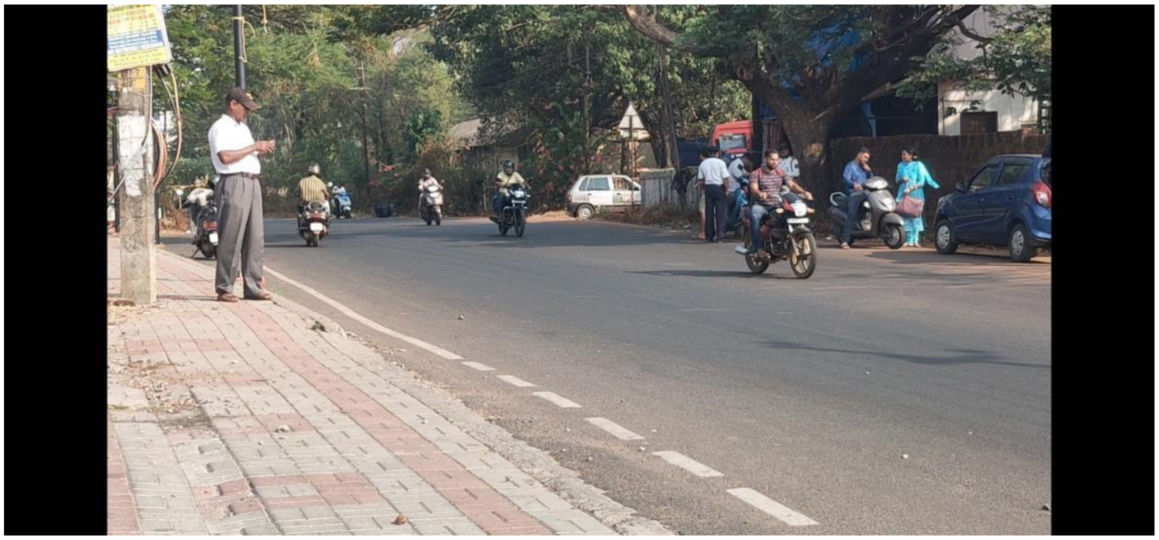


Figure 2 : Sample from Oppo Reno Video of Dataset 1

The second dataset was made using the Nikon D5500 DSLR. Each photo has a resolution of 24.2 MP. Photos were clicked from 3 different locations. Only on the first location photos were clicked from 2 different angles. From the other 2 angles photos were clicked from one angle. A sample image is shown in Figure 3. For the third dataset, pictures were clicked from a single location and 2 different angles a sample of which is shown in Figure 4.

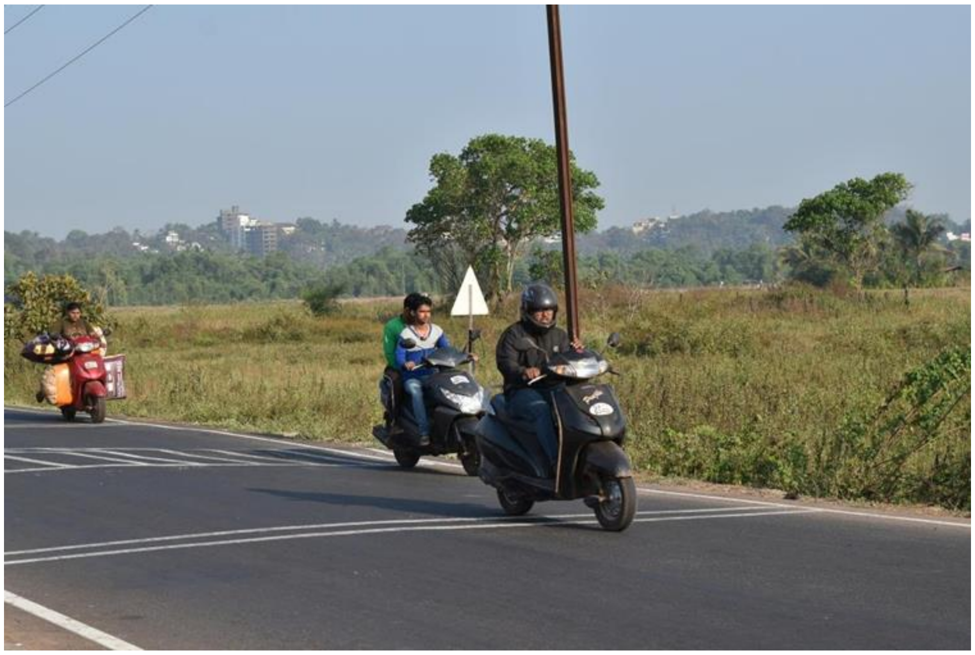


Figure 3: Sample from Dataset 2

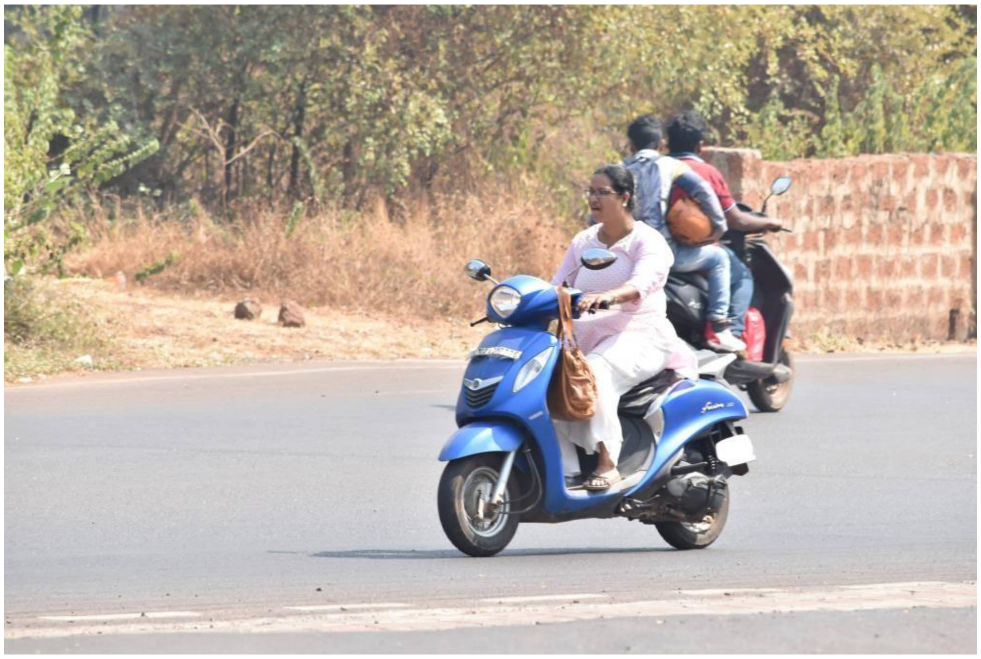


Figure 4: Sample from Dataset 3

For the final working model 302 training images and 71 test images were used.

## Dataset Labelling

For this step a python script developed by Darrenl tzutalin [1] was used. It’s free and open-sourced code designed to label Image data for Object Detection. It has a user-friendly UI with which we need to tag the objects in the image as shown in Figure 5 and 6. Using this software around 800 images were labelled. Each image having anywhere from 1 to 20 labels depending on the approach used. LabelImg saves a .xml file containing the label data for each image. These .xml files will be later used to generate TFRecords, which are one of the inputs to the TensorFlow trainer.

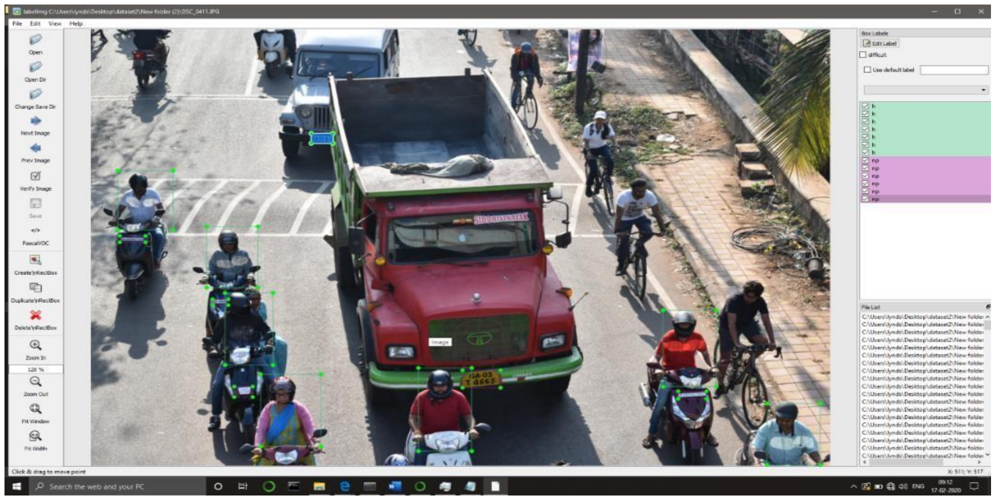


Figure 5: Labelling a sample from a Dataset 2

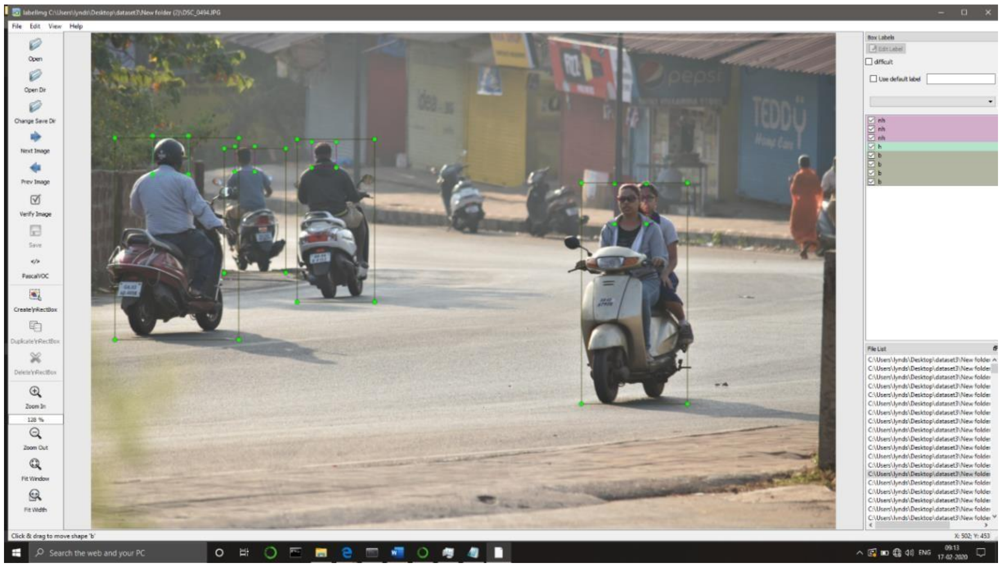


Figure 6: Labelling a sample from Dataset 3

**Dataset Pre-processing**

First the Images were resized using resizer.py written by Evan [2] so training becomes much easier and all the data is normalized. To generate the TFRecords that serve as input data to the TensorFlow training model, we use xml_to_csv.py and generate_tfrecord.py scripts from Dat Tran's Raccoon Detector [3]. The image .xml data will be used to create .csv files containing all the data for the train and test images. The generate_tfrecord.py was then edited with the required labels and the appropriate ids. After running the script, we should have all the necessary TFRecords files for training themodel.

## 4.2 Coding Standard

The coding standards were well maintained throughout the project. Naming conventions, adding comments, maintaining modularity and avoiding repetition of codes was taken into consideration while programming. We also ensured that security was taken care of and the code is bug free and no logical errors are present. We also ensured that there is no lag in performance and computation to be as quick as possible.

## 4.3 Coding Details

**Login Screen**

```
class LoginScreen extends StatefulWidget {
  LoginScreen({required this.themeBloc});

  final ThemeBloc themeBloc;

  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  GlobalKey key = GlobalKey();
  final FbDatabase=FirebaseDatabase.instance.reference();
  …………..…………….
RaisedButton(
                padding: const EdgeInsets.all(Sizes.PADDING_0),
                elevation: Sizes.ELEVATION_8,
                onPressed: ()
                {
                  //        _scaffoldKey.currentState!.showSnackBar(SnackBar(content:        Text("Invalid
Credentials!")));
```

```
                 // ExtendedNavigator.root.push(Routes.homescreen);

                 // if((validateMobileNumberStructure(Username.text) == true) && ) {

                   if(MobileNo.text.isNotEmpty && Password.text.isNotEmpty)
                   {
                     if(validateMobileNumberStructure(MobileNo.text) == true)
                     {
                       if(validatePasswordStructure(Password.text) == true)
                       {

                         // String MobileNoVar="+91 9512136600".replaceAll(' ', '');
                         String MobileNoVar=MobileNo.text.toString();


FbDatabase.child("PoliceOfficerRegistration").orderByChild("MobileNo").equalTo(MobileNoVa
r.replaceAll(' ', ''))
                             .once().then((snapshot){

                           if(snapshot.value == null)
                           {
                             print("--> Mobile No. not registered.");

                             ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:
Text("Mobile No. Not Registered.")));

                           }
                           else
                           {

FbDatabase.child("PoliceOfficerRegistration").orderByChild("Password").equalTo(Password.text
)
                                 .once().then((snapshot){

                               if(snapshot.value == null)
                               {
                                 print("--> Invalid Password.");

                                 ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:
Text("Invalid Password.")));

                               }
                               else   // Everything is correct move to homepage
                               {
                                 ExtendedNavigator.root.push(Routes.Masterpage);
                               }
                             });
                           }
                         });
                       }
                       else
                       {
                         ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:   Text("Provide
A Valid Password")));
                       }
                     }
                     else
                     {
                       ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("Provide A
Valid Mobile No.")));
                     }
                 }
                 else
                 {
                   _scaffoldKey.currentState!.showSnackBar(SnackBar(content: Text("Please fill all the
fields.")));
                 }
               },
bool validatePasswordStructure(String value){
    String  pattern = r'^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[!@#\$&*~]).{8,}$';
    RegExp regExp = new RegExp(pattern);
    return regExp.hasMatch(value);
  }

  bool validateMobileNumberStructure(String value){
    String  pattern = r'^(\+91[\-\s]?)?[0]?(91)?[6789]\d{9}$';
    RegExp regExp = new RegExp(pattern);
    return regExp.hasMatch(value);
  }
```

**Verify Gmail Screen**

```
class VerifyGmailScreen extends StatefulWidget
{
  VerifyGmailScreen({required this.themeBloc});
  final ThemeBloc themeBloc;
  @override
  _VerifyGmailScreenState createState() => _VerifyGmailScreenState();
}
class _VerifyGmailScreenState extends State<VerifyGmailScreen> {
```

```
………………
void signInWithCredential() async
  {
    final googleSignIn=GoogleSignIn();

    setState(() {
      showLoading=true;
    });

    try
    {
      final Gmailuser= await googleSignIn.signIn();

      if(Gmailuser == null)
      {
        showLoading=false;
        _scaffoldKey.currentState!.showSnackBar(SnackBar(content: Text("User is not recognized as
a valid google user.")));

        //                                        ExtendedNavigator.root.push(Routes.gmailVerify,arguments:
GmailVerifyArguments(themeBloc: _themeBloc));
      }
      else
      {
        final googleAuth=await Gmailuser.authentication;

        final gmailUserCredential=GoogleAuthProvider.credential(
          accessToken: googleAuth.accessToken,
          idToken: googleAuth.idToken,
        );
        final authCredential = await _auth.signInWithCredential(gmailUserCredential);

        setState(() {
          showLoading=false;
        });

        if(authCredential.user != null)
        {
          // Storing the value of mobile number in global scope shared preference

          final Preference = await SharedPreferences.getInstance();
          Preference.setString('EmailId', authCredential.user.email);
          Preference.setString('Username', authCredential.user.displayName);
          //Logging Out from Google
          googleSignIn.disconnect();
          ExtendedNavigator.root.push(Routes.VerifyMobile,arguments: 
VerifyMobileArguments(themeBloc: _themeBloc));
        }
      }
    }
    on FirebaseAuthException catch (e)
    {
      setState(() {
        showLoading=false;
      });
      _scaffoldKey.currentState!.showSnackBar(SnackBar(content: Text(e.message)));
    }
  }
………………
```

**Verify Mobile Number Screen**

```
enum MobileVerificationState
{
  SHOW_MOBILE_FORM_STATE,
  SHOW_OTP_FORM_STATE,
}
class VerifyMobileScreen extends StatefulWidget {
  VerifyMobileScreen({required this.themeBloc});
  final ThemeBloc themeBloc;
  @override
  _VerifyMobileScreenState createState() => _VerifyMobileScreenState();
}
class _VerifyMobileScreenState extends State<VerifyMobileScreen> {
  MobileVerificationState 
currentState=MobileVerificationState.SHOW_MOBILE_FORM_STATE;
……………..
void signInWithPhoneAuthCredential(AuthCredential phoneAuthCredential) async
  {
    setState(() {
      showLoading=true;
    });
    try
    {
      final authCredential = await _auth.signInWithCredential(phoneAuthCredential);
      setState(() {
        showLoading=false;
      });
      if(authCredential.user != null)
      {
        // Storing the value of mobile number in global scope shared preference

        final Preference = await SharedPreferences.getInstance();
        Preference.setString('MobileNumber', authCredential.user.phoneNumber);
        ExtendedNavigator.root.push(Routes.signUpScreen,arguments:
SignUpArguments(themeBloc: _themeBloc));
      }
    }
    on FirebaseAuthException catch (e)
    {
      setState(() {
        showLoading=false;
      });
      _scaffoldKey.currentState!.showSnackBar(SnackBar(content: Text(e.message)));
    }
  }
………………..
onPressed: () async
                  {
                    if(!validateMobileNumberStructure(MobileNumber.text))
                    {
                      _scaffoldKey.currentState!.showSnackBar(SnackBar(content: Text("Provide a valid
Mobile No. (Eg : +91 9234567890)")));
                    }
                    else
                    {
                      if(_FormKey.currentState!.validate())
                      {
                        setState(() {
                          showLoading=true;
                        });
                        await _auth.verifyPhoneNumber(
                          phoneNumber: MobileNumber.text,
                          verificationCompleted: (phoneAuthCredential) async
                          {
                            setState(() {
                              showLoading=false;
                            });
                            // signInWithPhoneAuthCredential(phoneAuthCredential);
                          },
                          verificationFailed: (verificationFailed) async
                          {
```

```
                    setState(() {
                      showLoading=false;
                    });
                    _scaffoldKey.currentState!.showSnackBar(SnackBar(content:
Text(verificationFailed.message)));
                  },
                  codeSent: (verficationId, resendingToken) async
                  {
                    setState(() {
                      showLoading=false;
                      currentState=MobileVerificationState.SHOW_OTP_FORM_STATE;
                      this.verificationId=verficationId;
                    });
                  },
                  codeAutoRetrievalTimeout: (verificationId) async
                  {                    },
                );
              }
              else
              {
                _scaffoldKey.currentState!.showSnackBar(SnackBar(content:   Text("Mobile   No.
field can't be empty.")));
              }
            }
          },
…………….
```

**Sign Up Screen**

```
class SignUpScreen extends StatefulWidget {
  SignUpScreen({required this.themeBloc});
  final ThemeBloc themeBloc;
  @override
  _SignUpScreenState createState() => _SignUpScreenState();
}
class _SignUpScreenState extends State<SignUpScreen> {
…………..
void LoadAllPreferences() async
  {
    print("Coming here");
    SharedPreferences Preference = await SharedPreferences.getInstance();
  // Try reading data from the counter key. If it doesn't exist, return null.
    setState(()
    {
      FetchedEmailId = Preference.getString('EmailId') ?? "";
      FetchedUsername = Preference.getString('Username') ?? "";
```

```
     FetchedMobileNumber = Preference.getString('MobileNumber') ?? "";
     print("Email id : "+FetchedEmailId + " Username : "+FetchedUsername+" Mobile Number : "+FetchedMobileNumber);
   });
  }
…………………..
onPressed: () {

                  if((ValidatePassword() == true) && (Pincode.text.isNotEmpty) && (PoliceStationName.text.isNotEmpty)) {

                    // Logging out Firebase instance from Mobile Screen User
                    UserFromMobileScreen.signOut();

                    // Regsitering Police Officer in the Firebse Database


FbDatabase.child("PoliceOfficerRegistration").orderByChild("EmailID").equalTo(EmailId.text)
                       .once().then((snapshot){

                         if(snapshot.value != null)
                         {
                           print("-->"+snapshot.value.toString());
                           print("--> Email ID Already Registered.");
                           //  _scaffoldKey.currentState!.showSnackBar(SnackBar(content:  Text("Email ID Already Registered.")));

                           ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("Email ID Already Registered.")));

                           ExtendedNavigator.root.push(Routes.loginScreen,
                               arguments: LoginScreenArguments(
                                   themeBloc: _themeBloc));
                         }
                         else
                         {

FbDatabase.child("PoliceOfficerRegistration").orderByChild("MobileNo").equalTo(MobileNumber.text)
                               .once().then((snapshot){

                             if(snapshot.value != null)
                             {
                               print("-->"+snapshot.value.toString());
                               print("--> Mobile No. Already Registered.");
                               //              _scaffoldKey.currentState!.showSnackBar(SnackBar(content:
Text("Mobile No. Already Registered.")));

                               ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:
Text("Mobile No. Already Registered.")));

                               ExtendedNavigator.root.push(Routes.loginScreen,
                                   arguments: LoginScreenArguments(
                                       themeBloc: _themeBloc));

                             }
                             else
                             {
                               final NewPoliceOfficerRegistry=<String,dynamic>
                               {
                                 'OfficerFullName' : Fullname.text,
                                 'EmailID' : EmailId.text,
                                 'MobileNo' : MobileNumber.text,
                                 'PoliceStationName' : PoliceStationName.text,
                                 'PinCode' : Pincode.text,
                                 'District' : District.text,
                                 'State' : State.text,
                                 'Password' : Password.text
                               };

                               FbDatabase
                                   .child("PoliceOfficerRegistration")
                                   .push()
                                   .set(NewPoliceOfficerRegistry)
                                   .then((_){
                                     print("Police Officer Has Been Successfully Registered.!");
                                     //         _scaffoldKey.currentState!.showSnackBar(SnackBar(content:
Text("Registered Successfully.")));
                                     ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:
Text("Registered Successfully.")));

                                   })
                                   .catchError((error) => print("Error Message : $error"));

                               ExtendedNavigator.root.push(Routes.loginScreen,
                                   arguments: LoginScreenArguments(
                                       themeBloc: _themeBloc));
                             }
```

```
                                });
                              }
                        });
                      }
                      else
                      {
                        _scaffoldKey.currentState!.showSnackBar(SnackBar(content: Text("Please fill all
the fields.")));
                      }
                    },
…………..
```

**Homepage Screen**

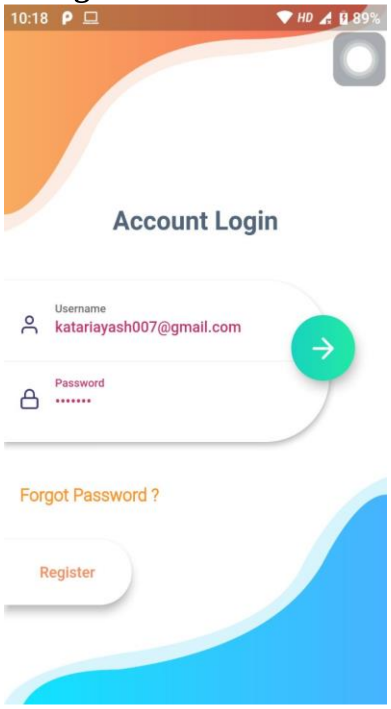
```
class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
  _HomePageState createState() => _HomePageState();
}
class _HomePageState extends State<HomePage> {
  final List<String> RoadLocations=[
    'assets/images/kormanagala.jpg',
    'assets/images/hsr_layout.jpg',
    'assets/images/mg_road.jpg',
    'assets/images/shivaji_nagar.jpg',
  ];
………………
```

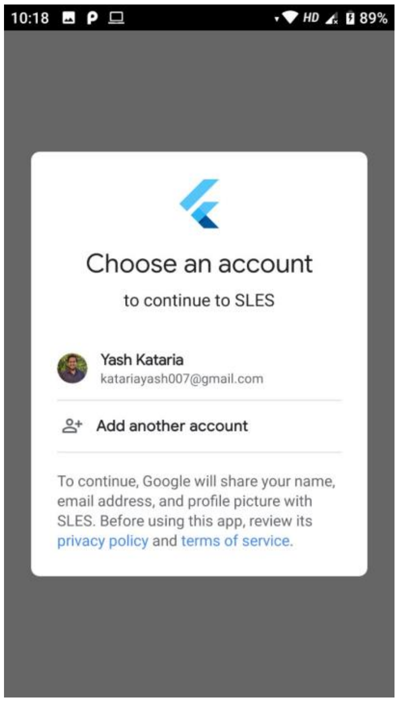
**Violators Screen**

```
 class Violators extends StatefulWidget {
  const Violators({Key? key}) : super(key: key);
  @override
  _ViolatorsState createState() => _ViolatorsState();
}
class _ViolatorsState extends State<Violators> {
  String LocationChoosed = "Bangalore";
```

## 4.4 SCREENSHOTS

### a. Login

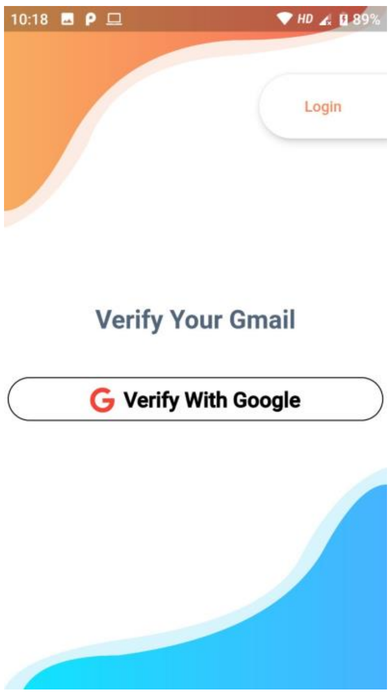


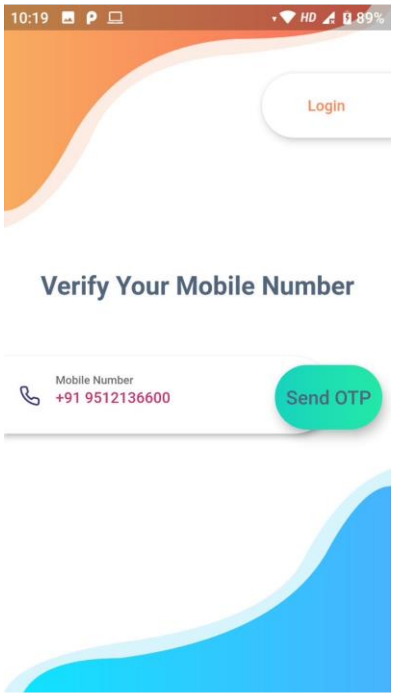


To continue, Google will share your name, email address, and profile picture with SLES. Before using this app, review its privacy policy and terms of service.

Fig 4.1:login page

### b. Verify User





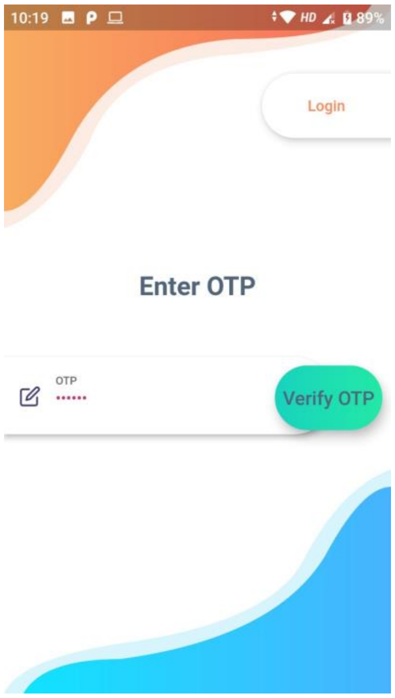
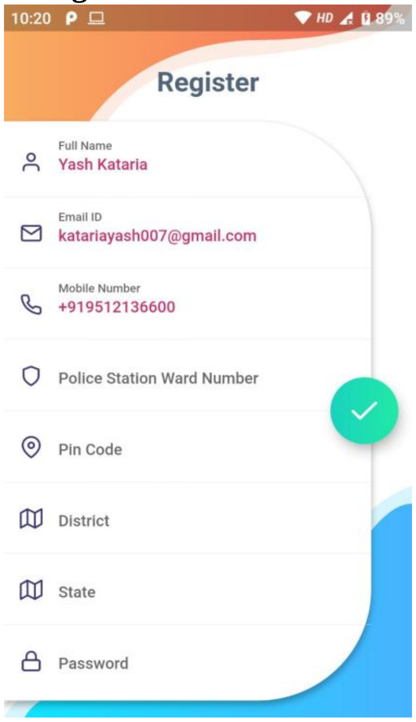


Fig 4.2: Verification page

## c. Registration



Full Name
Yash Kataria

Email ID
katariayash007@gmail.com

Mobile Number
+919512136600

Police Station Ward Number

Pin Code

District

State

Password

Fig 4.3: Registration page

## d. View Location

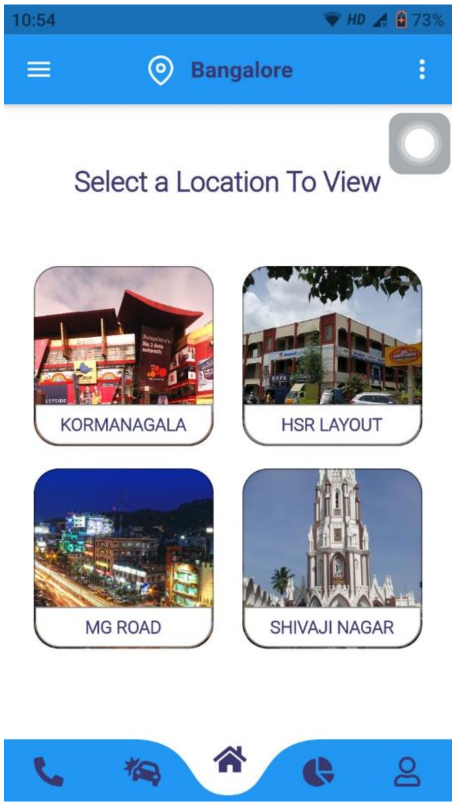


Fig 4.4: View Location

## e. Verify Violator

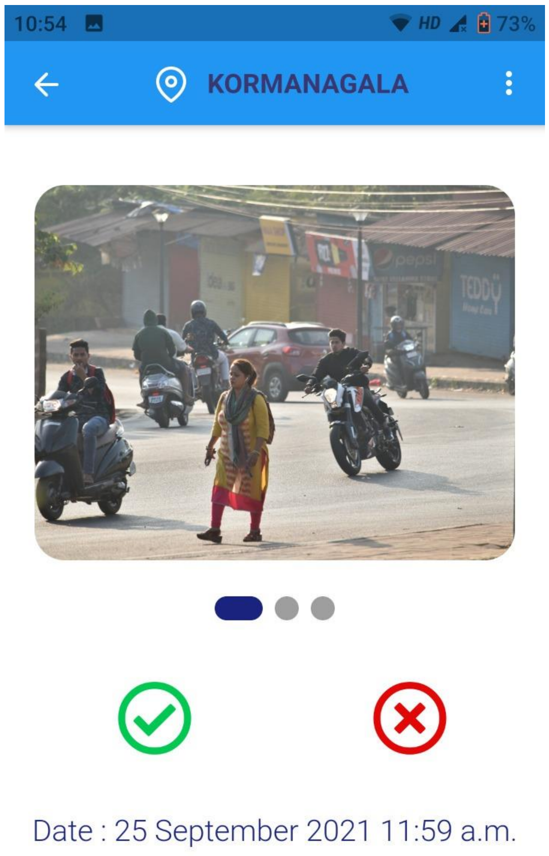


Fig 4.5: Verify violators by police officer

# REFERENCES

[1] J. Chiverton, "Helmet presence classification with motorcycle detection and tracking," in IET Intelligent Transport Systems, vol. 6, no. 3, pp. 259-269, September 2012.

[2] B. Duan, W. Liu, P. Fu, C. Yang, X. Wen, and H. Yuan, “Real-time onroad vehicle and motorcycle detection using a single camera,” in Procs. of the IEEE Int. Conf. on Industrial Technology (ICIT), pp. 1–6, 10-13 Feb 2009.

[3] R. V. Silva, K. Aires, T. Santos, K. Abdala, R. Veras, and A. Soares, “Automatic detection of motorcyclists without helmet,” in Proc. Latin American Computing Conf. (CLEI), Puerto Azul, Venezuela, pp. 1–7, 4–6 October 2013.

[4] R. V. Silva, T. Aires, and V. Rodrigo, “Helmet detection on motorcyclists using image descriptors and classifiers,” in Proc. Graphics, Patterns and Images (SIBGRAPI), Rio de Janeiro, Brazil, pp. 141–148, 27–30 August 2014.

[5] R. E. Kalman, “A new approach to linear filtering and prediction problems,” Journal of Basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.

[6] J. Mistry, A. K. Misraa, M. Agarwal, A. Vyas, V. M. Chudasama and K. P. Upla, "An automatic detection of helmeted and non-helmeted motorcyclist with license plate extraction using convolutional neural network," 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA), Montreal, QC, pp. 1-6, 2017.

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

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

[9] K. C. D. Raj, A. Chairat, V. Timtong, M. N. Dailey and M. Ekpanyapong, "Helmet violation processing using deep learning", 2018 International Workshop on Advanced Image Technology (IWAIT), Chiang Mai, pp. 1-4, 2018

[10] K. Dahiya, D. Singh, and C. K. Mohan, “Automatic detection of bikeriders without helmet using surveillance videos in real-time,” in Proc. Int. Joint Conf. Neural Networks (IJCNN), Vancouver, Canada, pp. 3046–3051, 24–29 July 2016.

[11] D.G.Lowe, “Distinctive image features from scale-invariant key points,” Int. Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.

[12] D. Navneet and B. Triggs, “Histograms of oriented gradients for human detection,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), San Diego, California, pp. 886– 893, 20–26 June 2005.

[13] Z. Guo, D. Zhang, and L. Zhang, “A completed modeling of local binary pattern operator for texture classification,” IEEE Trans. Image Processing, vol. 19, no. 6, pp. 1657–1663, 2010.

[14] C. Cortes and V. Vapnik, “Support vector networks,” Machine Learning (Springer), vol. 20, no. 3, pp. 273–297, 1993.

[15] D. Singh, D. Roy, and C. K. Mohan, “Dip-svm:distribution preserving kernel support vector machine for big data,” IEEE Trans. on Big Data, 2017.

[16] D. Singh and C. K. Mohan, “Distributed quadratic programming solver for kernel SVM using genetic algorithm,” in Proc. IEEE Congress on Evolutionary Computation, Vancouver, pp. 152–159, July 24–29 2016.

[17] 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), Anchorage, AK, pp. 3036-3041, 2017.

[18] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam,” MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, Computer Vision and Pattern Recognition (cs.CV), arXiv:1704.04861, 17 Apr 2017