

**A**  
**MINI PROJECT REPORT**  
**ON**  
**“Indian Traffic Sign Recognition System”**

**Submitted by**  
**Pitroda Vivek (20IT456)**  
**Shinde Krunal (20IT461)**  
**Shah Manav (20IT483)**

**Dr. Zankhana Shah**  
**Faculty Guide**



**Information Technology Department**  
**Birla Vishvakarma Mahavidyalaya Engineering College**  
**(An Autonomous Institution)**  
**AY: 2022-23, Semester II**



**Birla Vishvakarma Mahavidyalaya Engineering College**

**(An Autonomous Institution)**

**Information Technology Department**

**AY: 2022-23, Semester II**

## **CERTIFICATE**

This is to certify that project entitled with **“Indian Traffic Sign Recognition System”** has been successfully carried out by **Pitroda Vivek(20IT456), Shinde Krunal(20IT461) and Shah Manav(20IT483)** for the subject of **3IT31-Mini Project** under my guidance during the academic year 2022-23, Semester II. The Mini Project work carried out by the students of 6<sup>th</sup> semester is satisfactory.

**Date:**

**Dr. Zankhana Shah**

Faculty Guide  
IT Department  
BVM

**Dr. Keyur Brahmabhatt**

Head of the Department  
IT Department  
BVM

# ACKNOWLEDGEMENT

We are incredibly appreciative to Dr. Keyur Brahmbhatt, Head of the Information Technology Department, and Dr. Indrajit Patel, Principal of Birla Vishvakarma Mahavidyalaya Engineering College, for giving all the resources necessary for the successful completion of our project.

We really appreciate Dr. Zankhana Shah, an assistant professor of information technology and our project leader, for her insightful advice and leadership in the creation of the progress report and project execution.

We would like to thank the project coordinator, Dr. Kanu Patel, as well as the whole team and friends for their assistance and coordination in completing this project on schedule and effectively.

If we miss out on recognition where acknowledgment is required by the authors of the references and other works mentioned in this project, we shall be in breach of our responsibility.

Finally, we want to express our gratitude to our parents for being there for us at every turn.

Thanking You

Pitroda Vivek (20IT456)  
Shinde Krunal (20IT461)  
Shah Manav (20IT483)

# **ABSTRACT**

An Indian traffic sign detection system is an artificial intelligence-based technology that uses computer vision algorithms to detect and recognize traffic signs on Indian roads. This system has the potential to improve road safety by providing real-time information to drivers about traffic signs, speed limits, and other important road regulations. The system uses deep learning models to accurately identify traffic signs in various weather and lighting conditions. The main objective of this system is to reduce road accidents and ensure safe driving on Indian roads. The system has been developed by combining the latest advancements in computer vision and machine learning technologies, making it a promising solution for the Indian traffic management system.

# TABLE OF CONTENTS

<b>Chapter 1: Introduction.....</b>	<b>1</b>
1.1 Brief overview of the work in INDIAN TRAFFIC SIGN RECOGNITION SYSTEM....	1
1.2 Aim of the Project.....	1
1.3 Project Scope.....	2
1.4 Project Objective.....	2
1.5 Project Modules.....	2
1.6 Project basic requirements.....	3
<b>Chapter 2: Analysis, Design Methodology, and Implementation Strategy.....</b>	<b>4</b>
2.1 Comparison of Existing Applications with your Project.....	4
2.2 Project Feasibility Study.....	4
2.3 Project Timeline chart.....	5
2.4 Detailed Modules Description.....	6
2.5 Project SRS.....	7
2.5.1 Use Case Diagrams.....	7
2.5.2 Data Flow Diagrams.....	8
2.5.3 Activity Diagram.....	9
2.5.4 State diagram.....	10
2.6 Database design and Normalization(Old).....	11
2.7 Database design and Normalization(New).....	12
2.8 Database segmentation.....	13
2.9 Sample Images of each class with description.....	16
2.10 Template Design.....	23
<b>Chapter 3: Implementation and Testing.....</b>	<b>25</b>
3.1 Importing required library.....	25
3.2 Preprocessing of Images.....	27
3.3 Build the model.....	29
3.4 Training the CNN Model(Before preprocessing the dataset).....	30
3.5 Training the CNN Model(After preprocessing the dataset).....	32
3.6 Save the Model.....	34
3.7 Implementation on Frontend side.....	34
<b>Chapter 4: Conclusion and Future work .....</b>	<b>38</b>
4.1 Conclusion.....	38
4.2 Future work.....	38
<b>Chapter 5: References.....</b>	<b>40</b>

# LIST OF FIGURES

Fig 1 Project Timeline Chart.....	5
Fig 2 Use Case Diagram.....	7
Fig 3 Data Flow Diagram Level 0.....	8
Fig 4 Activity Diagram.....	9
Fig 5 State Diagram.....	10
Fig 6 Class name.....	13
Fig 7 Class name.....	14
Fig 8 Class folders.....	15
Fig 9 Implementation.....	23
Fig 10 Implementation.....	24
Fin 11 Implementation.....	24
Fig 12 Implementation.....	27
Fig 13 Implementation.....	28
Fig 14 Implementation.....	28
Fig 15 Implementation.....	35
Fig 16 Implementation.....	36
Fig 17 Implementation.....	36
Fig 18 Implementation.....	37

## LIST OF TABLES

Table 1 Categories of signs and there average percent of occurrence in our dataset(old).....	14
Table 2 Categories of signs and there average percent of occurrence in our dataset(new).....	15

# Chapter 1: Introduction

## 1.1 Brief overview of the work in INDIAN TRAFFIC SIGN RECOGNITION SYSTEM

The Indian Traffic Sign Recognition System (ITSRS) is a computer vision-based system that aims to recognize and classify traffic signs used in India. The system can automatically identify and interpret traffic signs, providing valuable information to drivers in real-time.

The system typically involves the following steps:

**Image Acquisition:** Here image is uploaded from the page.

**Preprocessing:** The acquired images are preprocessed to remove noise, adjust contrast, and enhance the image quality.

**Feature Extraction:** A set of features is extracted from the preprocessed images. These features represent the unique characteristics of the traffic signs, such as shape, color, and texture.

**Classification:** A machine learning algorithm is used to classify the traffic signs based on the extracted features. The algorithm can be trained on a dataset of labeled traffic sign images to learn the different patterns and characteristics of the signs.

**Post-processing:** Once the traffic signs are classified, post-processing techniques can be used to refine the results and improve the accuracy of the system.

Overall, the Indian Traffic Sign Recognition System is an important tool for enhancing road safety and reducing the risk of accidents caused by human error or lack of awareness.

## 1.2 Aim of the project

As placement of traffic sign board do not follow any international standard, it may be difficult for non-local residents to recognize and infer the signs easily. So, this project mainly focuses on demonstrating a system that can help facilitate this inconvenience. This can be achieved by interpreting the traffic sign in readable form.

TSR system is developed, mainly to decrease the probability of missing some important traffic signs on the road.



## 1.3 Project Scope

The scope of the project is to implement a traffic sign recognition which is based on the traffic signs for Indian system.

Traffic sign recognition mainly includes two stages: the first stage is traffic sign detection, which concerns the location and size of the traffic signs in the traffic scene images, and the second stage of the process is traffic sign recognition, which pays close attention to the classification of what exact class the traffic signs belong to.

## 1.4 Project Objective

The objectives are:

- To understand the properties of road and traffic signs and their implications for image processing for the recognition task.
- To understand color, color spaces and color space conversion.
- To develop an efficient road sign classification algorithm.

## 1.5 Project Modules

### 1.5.1 Traffic sign Data set creation

Creating a dataset that is healthy to provide our user with accurate result of their uploaded image.

### 1.5.2 Preprocessing of the images

The image or videos which is scanned is pre-processed. The image which has higher resolution is scaled down to small resolution, blur images are smoothened dark images are illuminated and RGB image is converted into greyscale format so that the image can be easily processed.

### 1.5.3 The detection module

The detection module receives images from the user and finds out all the regions in the images that may contain traffic signs.

### 1.5.4 The classification module

The classification module determines the category of traffic sign in each region. The information provided by the traffic signs is encoded in their visual properties: color, shape, and symbol.

Home page in which user can upload their image and get the accurate result.

### 1.5.5 User Interface

The user interface module provides a graphical user interface for interacting with the ITRS system, displaying the recognized traffic signs, and providing feedback to the user.

## **1.6 Project basic requirements**

### **1.5.1 Hardware**

None required.

### **1.5.2 Software**

Python and its version 3.9 ,flask.

## Chapter 2: Analysis, Design Methodology, and Implementation Strategy

### 2.1 Comparison of Existing Applications with your Project

#### 2.1.1 Mobileye:

Mobileye is a leading provider of Advanced Driver Assistance Systems (ADAS) and offers a Traffic Sign Recognition (TSR) feature as part of its offering. The system uses cameras and machine learning algorithms to detect and recognize traffic signs in real-time.

#### 2.1.2 TomTom:

TomTom provides navigation and mapping solutions and has developed a TSRS that uses artificial intelligence and machine learning algorithms to recognize traffic signs accurately.

#### 2.1.3 Bosch:

Bosch is a multinational engineering and technology company that has developed a TSRS based on deep learning algorithms. The system uses cameras and sensors to recognize traffic signs and provide real-time information to drivers.

#### 2.1.4 NVIDIA:

NVIDIA is a leading provider of graphics processing units (GPUs) and has developed a deep learning-based TSRS. The system uses cameras and sensors to detect and recognize traffic signs and can operate in real-time.

#### 2.1.5 Continental:

Continental is a German automotive manufacturing company that provides solutions for the automotive industry. It has developed a TSRS based on machine learning algorithms that can recognize traffic signs accurately.

### 2.2 Project Feasibility Study

Yes as it will be a website based system it will work in all the smart system.

#### 2.2.1 Technical feasibility :

The system will require upgrades. The industry demands for new features, and the desire for things to run faster are just necessary reasons why system might need upgrades timely.

The development process of the system would be advantageous as we are using the currently available resources.

### 2.2.2 Operational Feasibility :

Operationally it would be feasible as it is user friendly. Also no core technical knowledge is required to operate it. As the service is not being provided.

### 2.2.3 Economical Feasibility :

Indian Traffic Sign Recognition System (ITRS) is a system that can improve road safety and traffic management in India. It consists of several interconnected modules, including image acquisition, preprocessing, segmentation, feature extraction, classification, post-processing, and user interface. The economic feasibility of ITRS depends on factors such as the cost of hardware and software, maintenance costs, implementation costs, and potential cost savings from reducing accidents caused by driver error. While the initial costs of implementing ITRS may be high, the potential cost savings and improved road safety make it economically feasible in the long run.

## 2.3 Project Timeline Chart

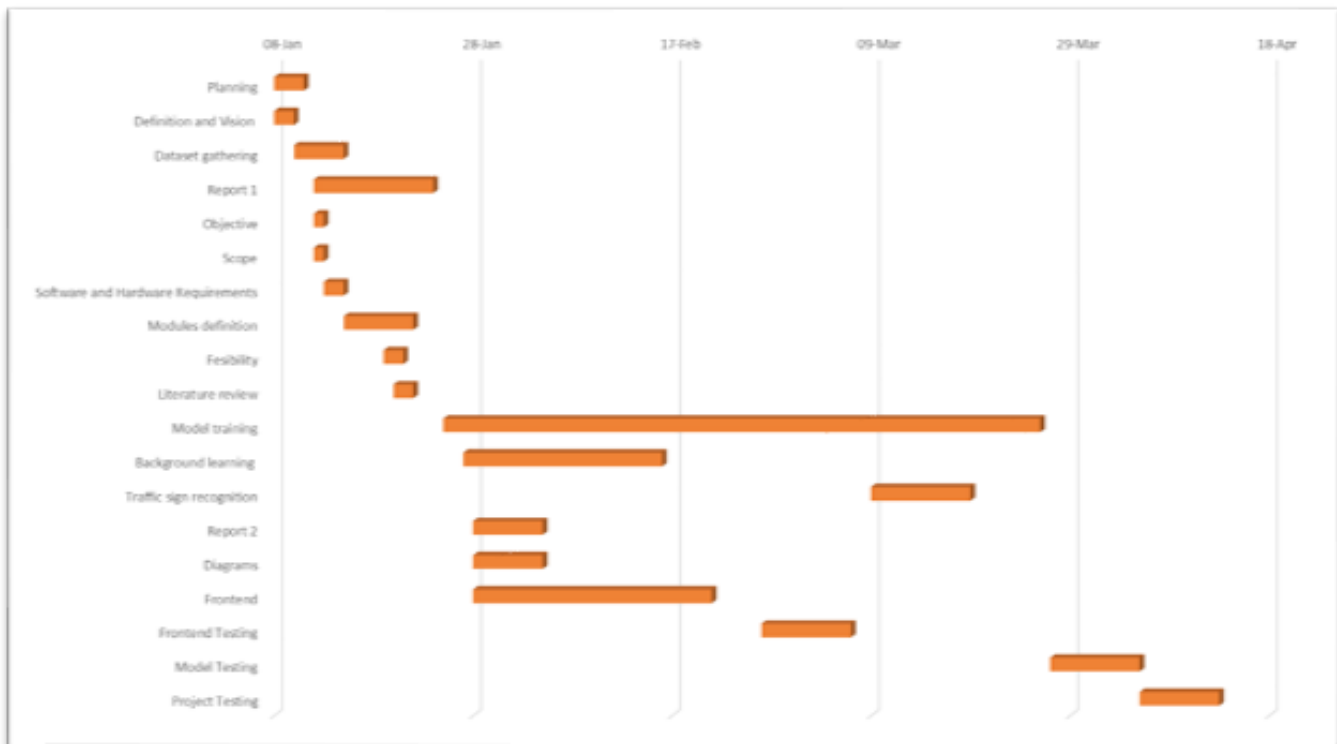


Fig 1. Project Timeline Chart

## 2.4 Detailed Modules Description

### 2.4.1 Traffic sign Data set creation

Traffic sign data set creation involves collecting and labeling images of traffic signs from various sources, such as photos taken from roads, highways, and cities. The images are then processed to extract the traffic signs and remove any background noise.

The next step is to label each image with the corresponding traffic sign class, which can include stop signs, speed limit signs, pedestrian crossing signs, and more. This labeling process can be done manually or through automated techniques, such as object detection algorithms.

The labeled images are then used to train machine learning models, such as deep neural networks, to recognize and classify traffic signs. These models can be used in various applications, such as self-driving cars, traffic flow monitoring, and road safety analysis.

Creating a high-quality traffic sign data set is crucial for developing accurate and reliable machine learning models. It requires careful curation and labeling of images, as well as ensuring that the data set is diverse and representative of different traffic sign classes and environmental conditions.

### 2.4.2 Preprocessing of the images

The image or videos which is scanned is pre-processed. The image which has higher resolution is scaled down to small resolution, blur images are smoothened dark images are illuminated and RGB image is converted into greyscale format so that the image can be easily processed.

### 2.4.3 The detection module

The detection module in a traffic sign detection system uses machine learning algorithms to identify the location and boundaries of traffic signs in an image or video frame. It involves preprocessing the image, extracting features, proposing regions, classifying traffic sign regions, and refining the detections. The output is the location and boundaries of traffic signs, which can be used in various applications.

- The performance of the detection module depends on the quality of the input image, the complexity of the scene, and the design of the machine learning algorithms used. Therefore, it is essential to have a diverse and representative dataset to train and validate the detection module.
- Some approaches to enhance the performance of the detection module include using ensemble models, which combine multiple models to improve accuracy and reduce false positives, and incorporating contextual information, such as road geometry and speed limits, to improve the accuracy of the detection.
- The detection module can be optimized for real-time performance by using techniques like hardware acceleration, model quantization, and network pruning, which reduce the computational cost and memory usage of the algorithms.

- The detection module is a crucial component of traffic sign detection systems, but it is often complemented by other modules, such as segmentation, tracking, and recognition, to provide a complete solution for traffic sign detection and recognition.

- The use of traffic sign detection systems is becoming increasingly important in the development of autonomous vehicles, as they enable the vehicles to perceive and respond to traffic signs and signals accurately and efficiently.

#### 2.4.4 The classification module

The classification module determines the category of traffic sign in each region. The information provided by the traffic signs is encoded in their visual properties: color, shape, and symbol.

Home page in which user can upload their image and get the accurate result.

The classification module in a traffic sign detection system identifies the specific traffic sign class of a detected region using machine learning algorithms. It involves data preparation, feature extraction, training, testing, and real-time inference. The output is the traffic sign class, which can be used in various applications. The performance of the classification module depends on the quality and diversity of the training data and the design of the machine learning algorithms used. Having a large and diverse dataset is crucial to train and validate the classification module.

#### 2.4.5 User Interface

The user interface module provides a graphical user interface for interacting with the ITRS system, displaying the recognized traffic signs, and providing feedback to the user.

## 2.5 Project SRS

### 2.5.1 Use Case Diagrams

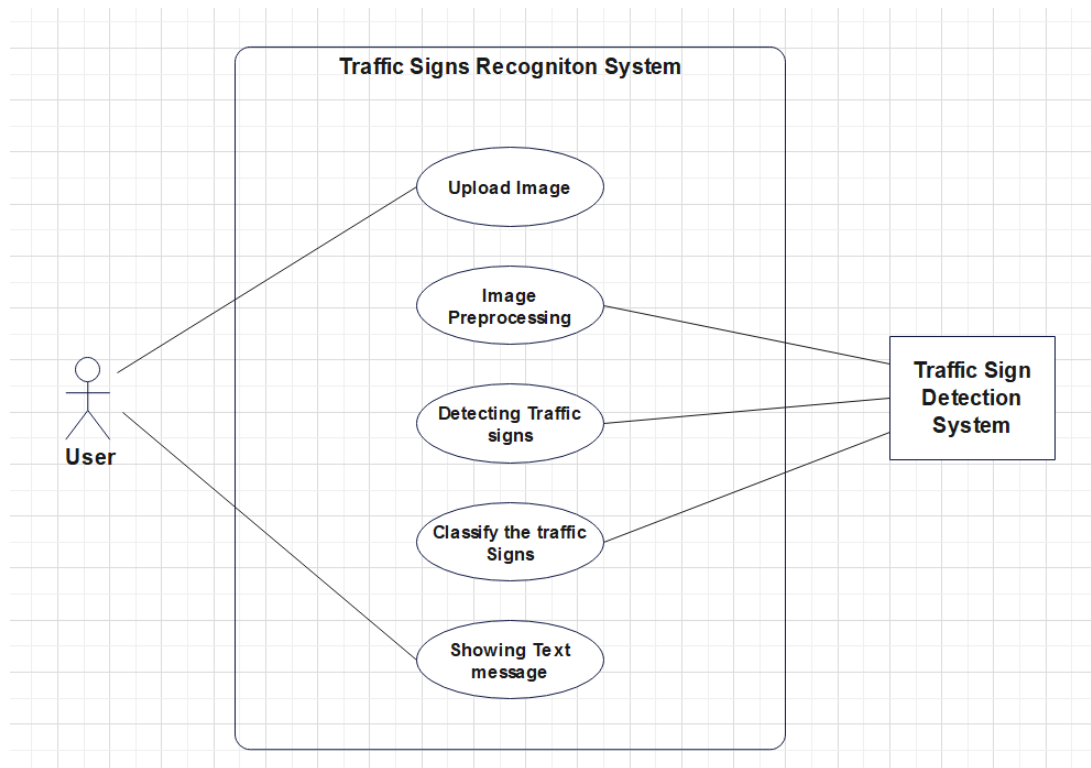


Fig 2. Use Case Diagram

## 2.5.2 Data Flow Diagrams

(Level-0)

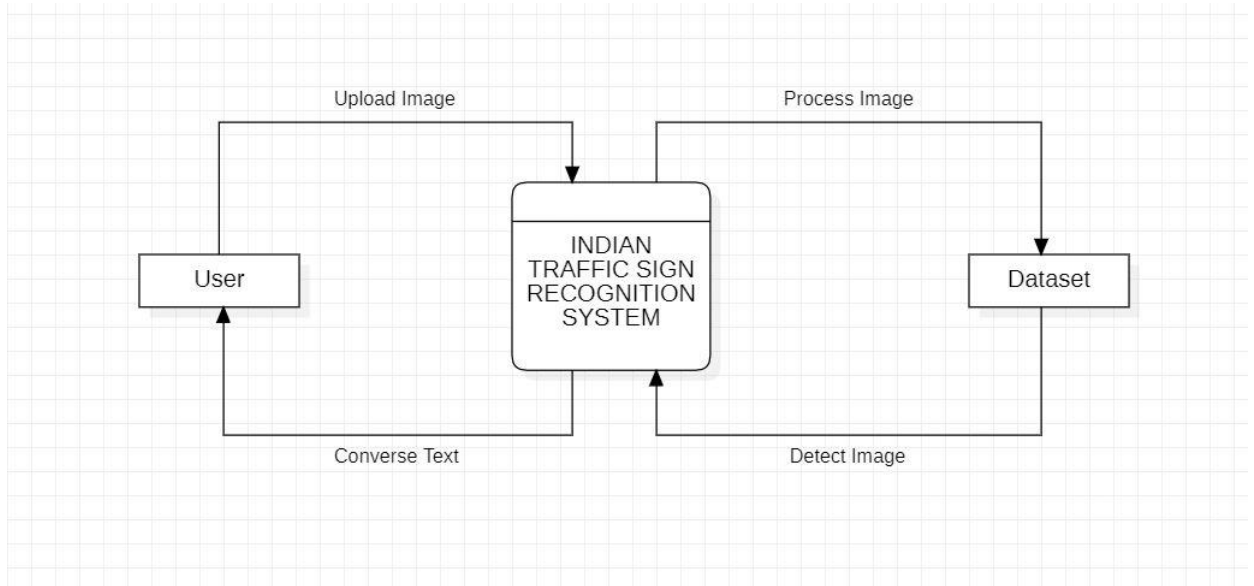
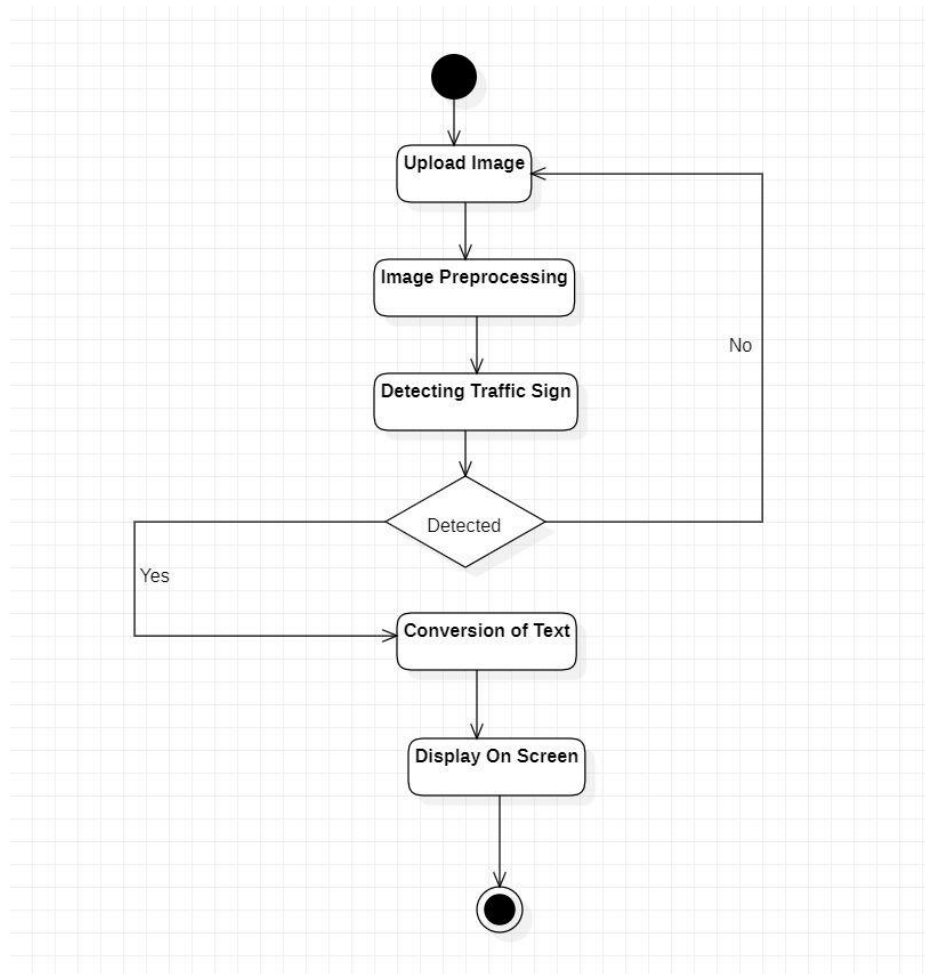


Fig 3. Level 0 DFD

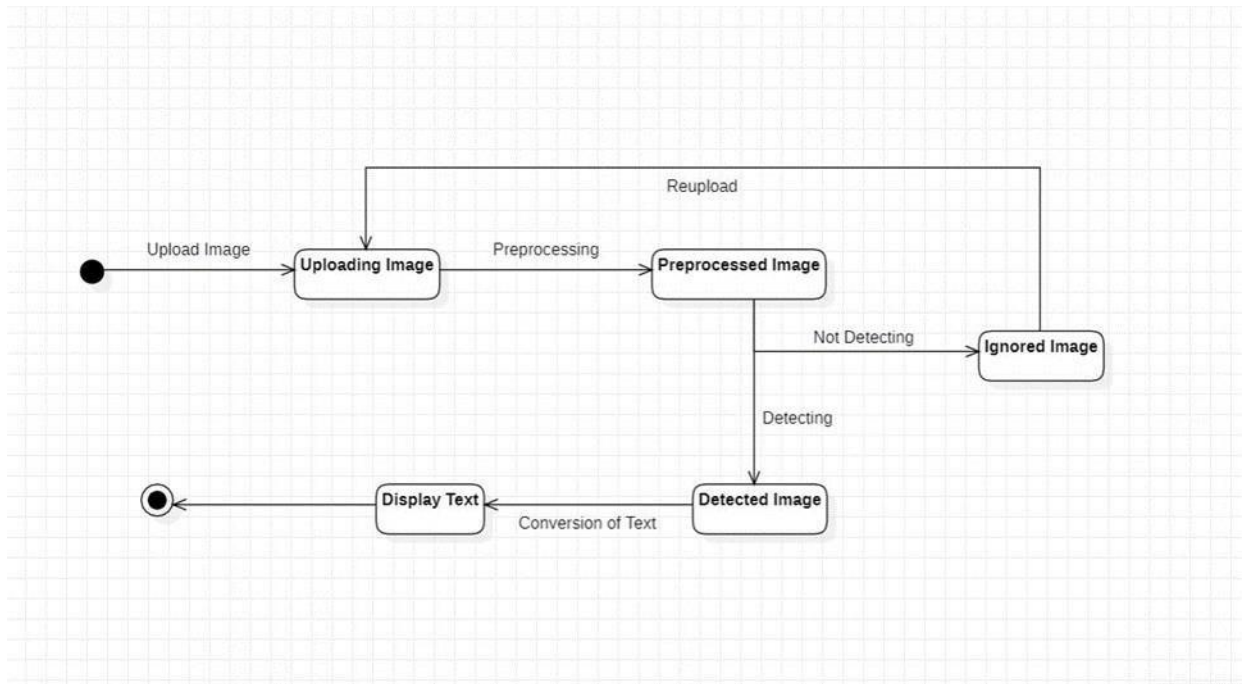
### 2.5.3 Activity Diagram



**Fig 4. Activity Diagram**



## 2.5.4 State Diagram



**Fig 5. State Diagram**

## 2.6 Database Design and Normalization(Old)

This dataset contains total of 14206 signs.

There are 3 types of traffic signs

- Mandatory (Regulatory) signs
- Cautionary (Warning) signs
- Guide signs

Our dataset contains total of 59 different signs.

We have the sets of images of the types of signs as follows

- Mandatory sign – 25 signs
- Cautionary signs – 26 signs
- Guide signs – 8 signs

### 2.6.1 Categories of signs and there average percent of occurrence in our dataset:

Sr.No.	Name	Percent
1	Normal	13%
2	Reverse	11%
3	Mirrored	16%
4	Reversed Mirrored	12%
5	Black and White	14%
6	Dark	13%
7	Edge cut	12%
8	Blur/Unclear	9%

**Table 1: Categories of signs and their average percentage of occurrence in dataset**

## Database Design and Normalization(New)

This dataset contains total of 14206 signs.

There are 3 types of traffic signs

- Mandatory (Regulatory) signs
- Cautionary (Warning) signs
- Guide signs

Our dataset contains total of 59 different signs.

We have the sets of images of the types of signs as follows

- Mandatory sign – 25 signs
- Cautionary signs – 26 signs
- Guide signs – 8 signs

### 2.6.2 Categories of signs and there average percent of occurrence in our dataset:

Sr.No.	Name	Percent
1	Normal	70%
2	Reverse	0%
3	Mirrored	0%
4	Reversed Mirrored	0%
5	Black and White	15%
6	Dark	1%
7	Edge cut	8%
8	Blur/Unclear	6%

**Table 2: Categories of signs and their average percentage of occurrence in dataset (New)**

## 2.7 Database segmentation




















ClassId	Name
0	Give way
1	No entry
2	One-way traffic
3	One-way traffic
4	No vehicles in both directions
5	No entry for cycles
6	No entry for goods vehicles
7	No entry for pedestrians
8	No entry for bullock carts
9	No entry for hand carts
10	No entry for motor vehicles
11	Height limit
12	Weight limit
13	Axle weight limit
14	Length limit
15	No left turn
16	No right turn
17	No overtaking
18	Maximum speed limit (90 km/h)
19	Maximum speed limit (110 km/h)
20	Horn prohibited
21	No parking
22	No stopping
23	Turn left
24	Turn right
25	Steep descent
26	Steep ascent
27	Narrow road
28	Narrow bridge
29	Unprotected quay
30	Road hump

Fig 6. Class names

31	Dip
32	Loose gravel
33	Falling rocks
34	Cattle
35	Crossroads
36	Side road junction
37	Side road junction
38	Oblique side road junction
39	Oblique side road junction
40	T-junction
41	Y-junction
42	Staggered side road junction
43	Staggered side road junction
44	Roundabout
45	Guarded level crossing ahead
46	Unguarded level crossing ahead
47	Level crossing countdown marker
48	Level crossing countdown marker
49	Level crossing countdown marker
50	Level crossing countdown marker
51	Parking
52	Bus stop
53	First aid post
54	Telephone
55	Filling station
56	Hotel
57	Restaurant
58	Refreshments

Fig 7. Class names

Class IDs:

 0	30-01-2023 09:13 PM	File folder
 1	30-01-2023 09:13 PM	File folder
 2	30-01-2023 09:13 PM	File folder
 3	30-01-2023 09:13 PM	File folder
 4	30-01-2023 09:13 PM	File folder
 5	30-01-2023 09:13 PM	File folder
 6	30-01-2023 09:13 PM	File folder
 7	30-01-2023 09:13 PM	File folder
 8	30-01-2023 09:14 PM	File folder
 9	30-01-2023 09:14 PM	File folder
 10	30-01-2023 09:13 PM	File folder
 11	30-01-2023 09:13 PM	File folder
 12	30-01-2023 09:13 PM	File folder
 13	30-01-2023 09:13 PM	File folder
 14	30-01-2023 09:13 PM	File folder
 15	30-01-2023 09:13 PM	File folder
 16	30-01-2023 09:13 PM	File folder
 17	30-01-2023 09:13 PM	File folder
 18	30-01-2023 09:13 PM	File folder

**Fig 8. Class folders**

## 2.8 Sample Images of each class with description

**ClassID : 0**

**Name : Give way**

This sign indicates that Driver should immediately stop. Usually Police, traffic and toll authorities use this signs at check posts.



**ClassID : 1**

**Name : No entry**

These signs are located at places where the vehicles are not allowed to enter.



**ClassID : 2**

**Name : One-way traffic**

This indicates that the traffic flow is allowed in only one direction. The way beyond this sign restricts entry of the traffic however, the oncoming traffic flow remains normal.



**ClassID : 3**

**Name : One-way traffic**

This indicates that the traffic flow is allowed in only one direction. The way beyond this sign restricts entry of the traffic however, the oncoming traffic flow remains normal.

**ClassID : 4**

**Name : No vehicles in both directions**

This indicates that the traffic flow is allowed in only one direction. The way beyond this sign restricts entry of the traffic however, the oncoming traffic flow remains normal. This sign directs that the demarcated area beyond it is prohibited for traffic flow from both sides.



**ClassID : 5**

**Name : No entry for cycles**

This sign signifies that there should no movement of traffic in the designated area either from outside or within.



**ClassID : 6**

**Name : No entry for goods vehicles**

As sign itself speaks the area designated is a no entry zone for Trucks or HMV.



**ClassID : 7**

**Name : No entry for pedestrians**

This sign restricts the movement of pedestrian on road or the adjoining area.



**ClassID : 8**

**Name : No entry for bullock carts**

This sign indicates that the road has been prohibited for plying of Bullock & Hand Carts.



**ClassID : 9****Name : No entry for hand carts**

This sign is erected on each entry to the road where all types of slow moving vehicles except cycles are to be prohibited.

**ClassID : 10****Name : No entry for motor vehicles**

This sign is used at places where entry to all types of motor vehicles is prohibited. .

**ClassID : 11****Name : Height limit**

The diameter of a mandatory sign disc must be 60 cm and height from ground level should be 2.8 m.

**ClassID : 12****Name : Weight limit**

This road sign limits the load of the vehicle, which should ply on the road further.

**ClassID : 13****Name : Axle weight limit**

This sign specifies the amount of axle load that is allowed to be carried on the over bridge road.

**ClassID : 14****Name : Length limit**

This sign on road indicates that length of the vehicle, which can be maneuvered through that passage

**ClassID : 15****Name : No left turn**

The no left turn sign is a turn prohibition sign that is designed to prevent an accident from occurring by informing drivers that turning left is prohibited.

**ClassID : 16****Name : No right turn**

This sign indicates that the driver should drive in left lane for smooth traffic flow.





**ClassID : 17****Name : No overtaking**

The no overtaking road sign is circular with a red border meaning that it is giving drivers an order.

**ClassID : 18****Name : Maximum speed limit (90 km/h)**

Speed limits are generally indicated on a traffic sign reflecting the maximum permitted speed – expressed as kilometres per hour (km/h)

**ClassID : 19****Name : Maximum speed limit (110 km/h)**

Speed limits are generally indicated on a traffic sign reflecting the maximum permitted speed – expressed as kilometres per hour (km/h)

**ClassID : 20****Name : Horn prohibited**

This sign is used on stretches of the road where sounding of horn is not allowed, near hospitals and in silence zones.

**ClassID : 21****Name : No parking**

This sign is very significant in major cities. It prohibits parking of a vehicle in designated area.

**ClassID : 22****Name : No stopping**

This sign instructs drivers not to park or stop their vehicles at the marked location.

**ClassID : 23****Name : Turn left**

This sign directs the traffic to either move straight or take left turn.

**ClassID : 24****Name : Turn right**

This sign directs the driver to turn right only, there could be any reason for it but obeying this signal would lead to safety and hassle free drive.



**ClassID : 25****Name : Steep descent**

This road sign indicates that there is steep ascent ahead and driver should get ready to climb and put the vehicle in relevant gear.

**ClassID : 26****Name : Steep ascent**

This road sign indicates that there is steep ascent ahead and driver should get ready to climb and put the vehicle in relevant gear.

**ClassID : 27****Name : Narrow road**

Road narrows signs indicate that the road ahead will not be as wide as the road you're currently on.

**ClassID : 28****Name : Narrow bridge**

This sign is erected before such bridges which are narrower than the road.

**ClassID : 29****Name : Unprotected quay**

The Quayside warning sign warns of harbour walls, quays and so on which are unprotected by barriers.

**ClassID : 30****Name : Road hump**

This sign cautions the driver that he should reduce the speed to cross the hump comfortably.

**ClassID : 31****Name : Dip**

Dip signs indicate that there is a dip or low place in the road.

**ClassID : 32****Name : Loose gravel**

This sign indicates that loose pebbles and stones could be lying on the road.

**ClassID : 33****Name : Falling rocks**

The falling rocks warning sign warns of the possibility of boulders tumbling onto the driver's way.



**ClassID : 34**

**Name : Cattle**

This sign shows that there is some repair/cleaning etc. being undertaken on the road and labour is involved in it.



**ClassID : 35**

**Name : Crossroads**

There is another road ahead that crosses the road you are on. Watch carefully for cross traffic in your path



**ClassID : 36**

**Name : Side road junction**

This side road sign warns drivers of an upcoming intersection so they can be aware of traffic entering or exiting the highway.



**ClassID : 37**

**Name : Side road junction**

This side road sign warns drivers of an upcoming intersection so they can be aware of traffic entering or exiting the highway.



**ClassID : 38**

**Name : Oblique side road junction**

Provide clear instructions to motorists, cyclists and walkers.



**ClassID : 39**

**Name : Oblique side road junction**

Provide clear instructions to motorists, cyclists and walkers.



**ClassID : 40**

**Name : T-junction**

The T- Junction sign means there is a right or left turn and the end of the current road.



**ClassID : 41**

**Name : Y-junction**

Indicates a Y-shaped intersection, where through traffic approaches from the stem of the "Y" and continues along the branch of the "Y" with the arrowhead.



**ClassID : 42**

**Name : Staggered side road junction**

This sign warns drivers that a side road that is broken up joins and leaves the



current road.

**ClassID : 43**

**Name : Staggered side road junction**

This sign warns drivers that a side road that is broken up joins and leaves the current road.

**ClassID : 44**

**Name : Roundabout**

There are not traffic signals or stop signs in a modern roundabout.



**ClassID : 45**

**Name : Guarded level crossing ahead**

This sign indicates that there is a Railway crossing which is guarded by a person.



**ClassID : 46**

**Name : Unguarded level crossing ahead**

This is a sign to advise drivers that there is a gap in the divider or the median ahead without there being any intersection.



**ClassID : 47**

**Name : Level crossing countdown marker**

When you see countdown markers with a white background and red distance marker stripes, then you should be prepared to stop as they indicate you are approaching a concealed level crossing.



**ClassID : 48**

**Name : Level crossing countdown marker**

These signs give you an early warning that you may need to stop at a barrier for a level crossing that you cannot yet see.



**ClassID : 49**

**Name : Level crossing countdown marker**

When you see countdown markers with a white background and red distance marker stripes, then you should be prepared to stop as they indicate you are approaching a concealed level crossing.

**ClassID : 50**

**Name : Level crossing countdown marker**

These signs give you an early warning that you may need to stop at a barrier for a level crossing that you cannot yet see.



**ClassID : 51**

**Name : Parking**

Parking sign means a sign, within the public right-of-way or adjacent thereto, that directs motorists to parking facilities.



**ClassID : 52**

**Name : Bus stop**

Bus Stop sign indicates that there is a Bus Stop ahead.



**ClassID : 53**

**Name : First aid post**

If you require basic medical attention, this sign indicates the presence of a first aid facility nearby.



**ClassID : 54**

**Name : Telephone**

Phone Symbol Sign. Reflective Sheeting on Aluminum. Radius Corners. 3/8" Prepunched Holes.



**ClassID : 55**

**Name : Filling station**

Petrol pump informatory sign indicates that there is a Petrol Pump ahead.



**ClassID : 56**

**Name : Hotel**

It's vital to create a unique identity for your hotel that sets it apart from every other hotel in the area.



**ClassID : 57**

**Name : Restaurant**

EATING PLACE sign indicates that there is an eating-place in the vicinity. This sign is common on highways and long stretches of road.



**ClassID : 58**

**Name : Refreshments**

This sign indicates the presence of a light refreshment facility nearby where you can grab a quick bite or drink



## 2.9 Template Design

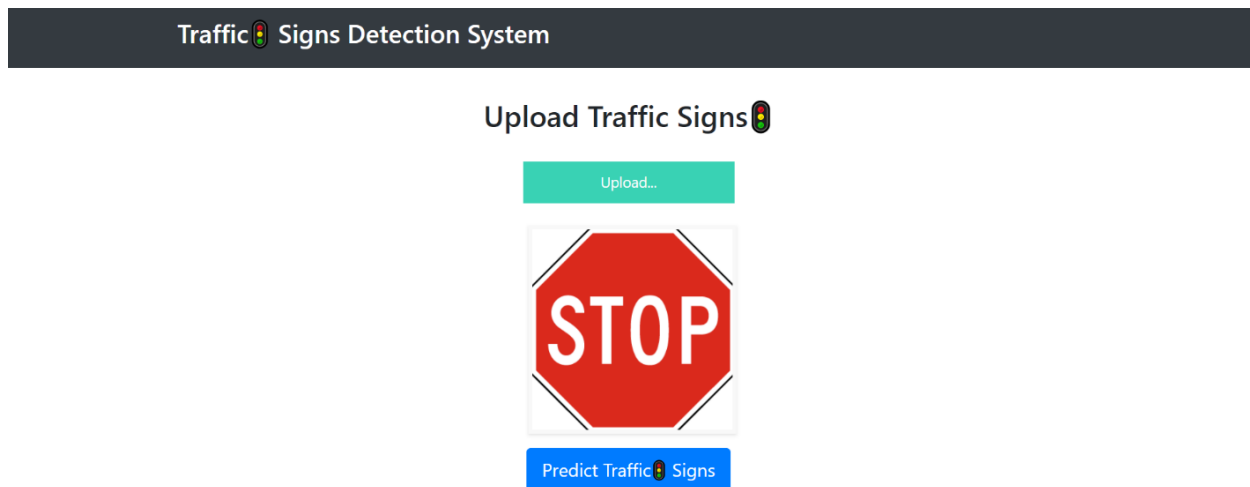
### 2.9.1 Before input a traffic sign (Input):

First we upload our image using the upload button.



**Fig 9**

Then image preview is shown to verify your image.



**Fig 10**

### 2.9.2 After input a traffic sign (Output):

When we click on the predict button we get the answer of the image  
To which class it belongs i.e. what the image is trying to convey by the means of  
traffic sign.



**Fig 11**

## Chapter 3: Implementation and Testing

### 3.1 Importing required library

#### Import Required Library

```
[ ] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
```

The required libraries are imported here.

They are:-

1. Numpy
  2. Pandas
  3. Matplotlib
  4. Opencv
  5. Tensorflow
  6. Sklearn
  7. Keras
- NumPy: NumPy is a library for Python programming language that adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The requirements to use NumPy are Python 3.x and a C compiler such as GCC.
  - Pandas: Pandas is a library for data manipulation and analysis. It provides data structures and functions to manipulate numerical tables and time series data. The requirements to use Pandas are Python 3.x and NumPy.
  - Matplotlib: Matplotlib is a data visualization library. It provides a variety of tools for creating static, animated, and interactive visualizations in Python. The requirements to use Matplotlib are Python 3.x, NumPy, and a backend such as Tk, GTK, or Qt.



- **OpenCV:** OpenCV (Open Source Computer Vision) is a library for computer vision and machine learning algorithms. It provides tools for image and video processing, object detection, face recognition, and more. The requirements to use OpenCV are Python 3.x, NumPy, and a C++ compiler such as GCC.
- **TensorFlow:** TensorFlow is a library for machine learning and deep learning algorithms. It provides tools for building and training neural networks, and for deploying machine learning models in production. The requirements to use TensorFlow are Python 3.x, NumPy, and a C++ compiler such as GCC.
- **Scikit-learn:** Scikit-learn is a library for machine learning algorithms. It provides tools for classification, regression, clustering, and dimensionality reduction, among other things. The requirements to use Scikit-learn are Python 3.x, NumPy, and SciPy.
- **Keras:** Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It provides tools for building and training deep learning models, and for deploying them in production. The requirements to use Keras are Python 3.x, NumPy, and a backend such as TensorFlow or Theano.

### 3.2 Preprocessing of Images

#### 3.2.1 Image without Preprocessing

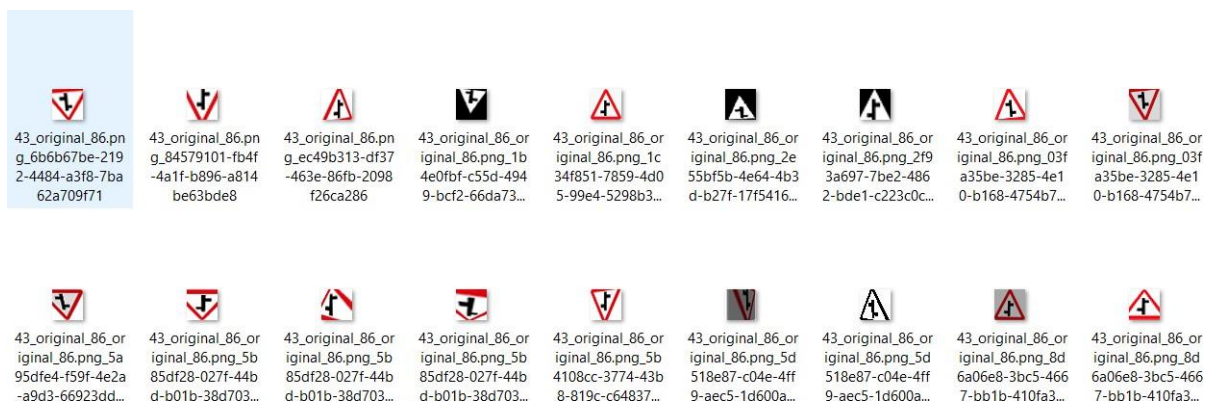
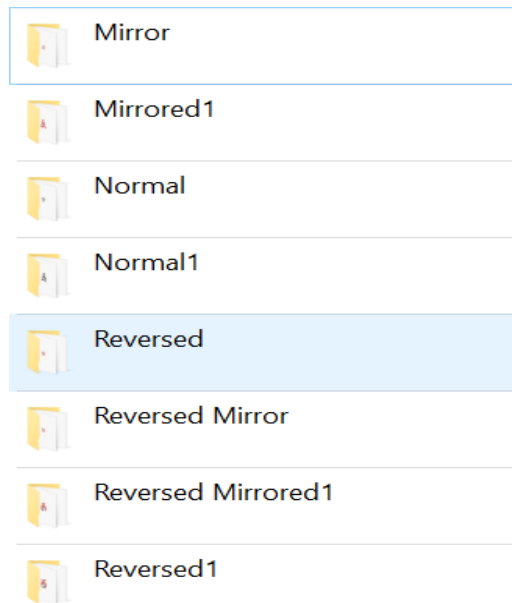


Fig 12

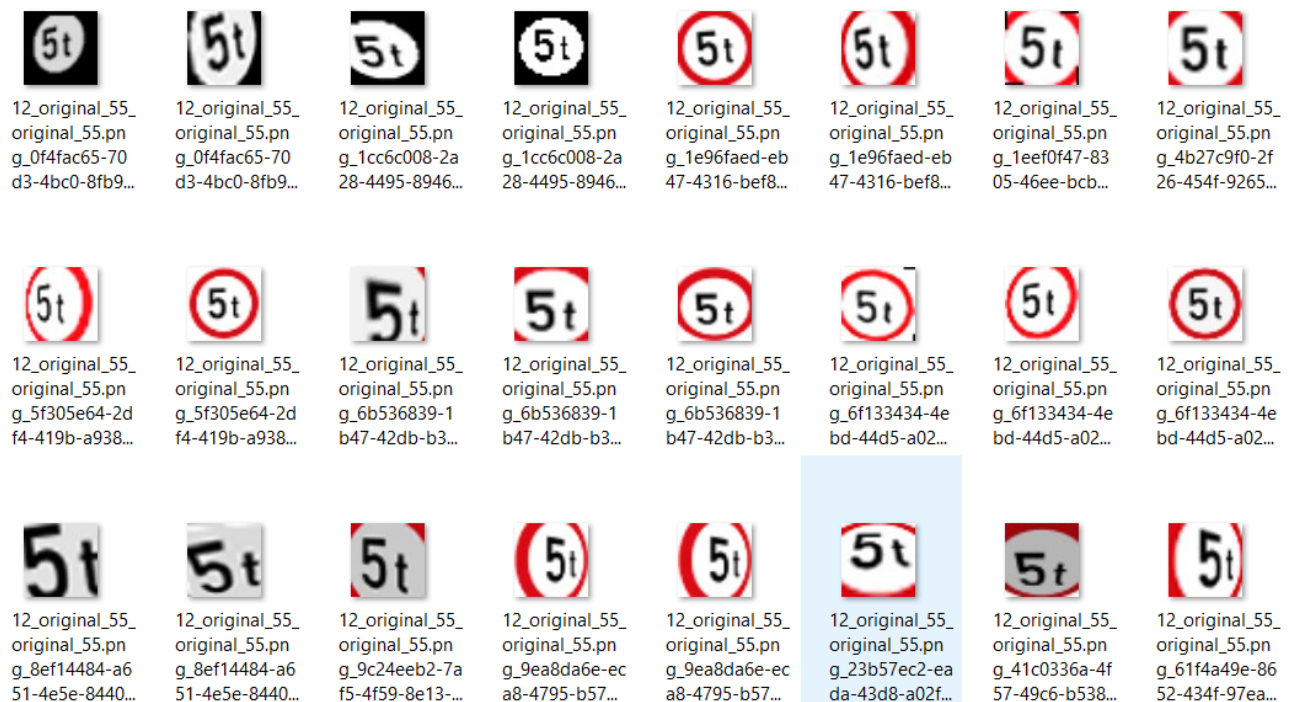
Here we did not preprocess the image.

We directly used this dataset for model training. And we got 78.1% accuracy

### 3.2.2 Image after Preprocessing



**Fig 13**



**Fig 14**

Here this are the preprocessed images.

We first divided images into different folders such as

- Normal
- Reversed Mirror
- Mirror
- Reversed etc.

Then we preprocessed this folders using certain algorithms.

We directly used this dataset for model training. And we got 91.1% accuracy.

### 3.3 Build the Model

#### Build the Model

```

model = Sequential()
model.add(Conv2D(filters=16, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=16, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.10))
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.10))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.20))
model.add(Dense(59, activation='softmax'))

```

This code snippet is defining a convolutional neural network (CNN) model using the Keras library.

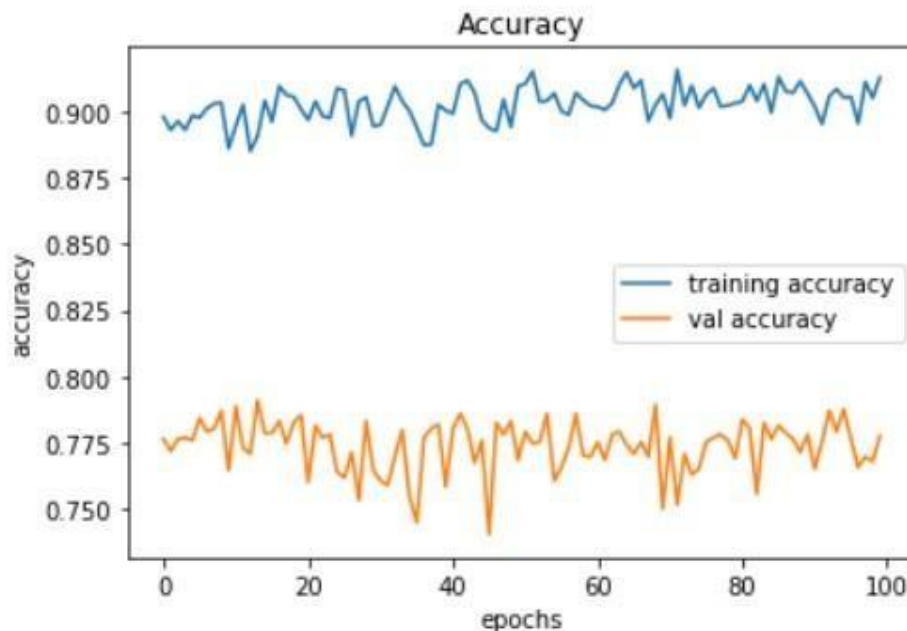
- The `Sequential()` function creates a new empty model object to which layers will be added sequentially.
- The `Conv2D()` function adds a 2D convolutional layer with 16 filters, each with a size of 5x5 pixels, and using the 'relu' activation function. The `input_shape` parameter is set to the shape of the input data, which in this case is `X_train.shape[1:]` (excluding the batch size).
- Another `Conv2D()` layer is added with the same number of filters and kernel size, followed by a `MaxPool2D()` layer that reduces the spatial dimensions of the output by a factor of 2.
- A `Dropout()` layer is added to prevent overfitting by randomly setting a fraction (10%) of the input units to 0 during training.
- Two more `Conv2D()` layers with 32 filters each and a size of 3x3 pixels are added, followed by another `MaxPool2D()` layer and a `Dropout()` layer.
- The output of the last convolutional layer is flattened into a 1D array using the `Flatten()` layer.
- A fully connected `Dense()` layer with 256 units and 'relu' activation function is added, followed by another `Dropout()` layer.
- The final `Dense()` layer with 59 units (corresponding to the number of classes) and 'softmax' activation function is added to output the class probabilities.

### 3.4 Training the CNN Model (Before preprocessing the dataset)

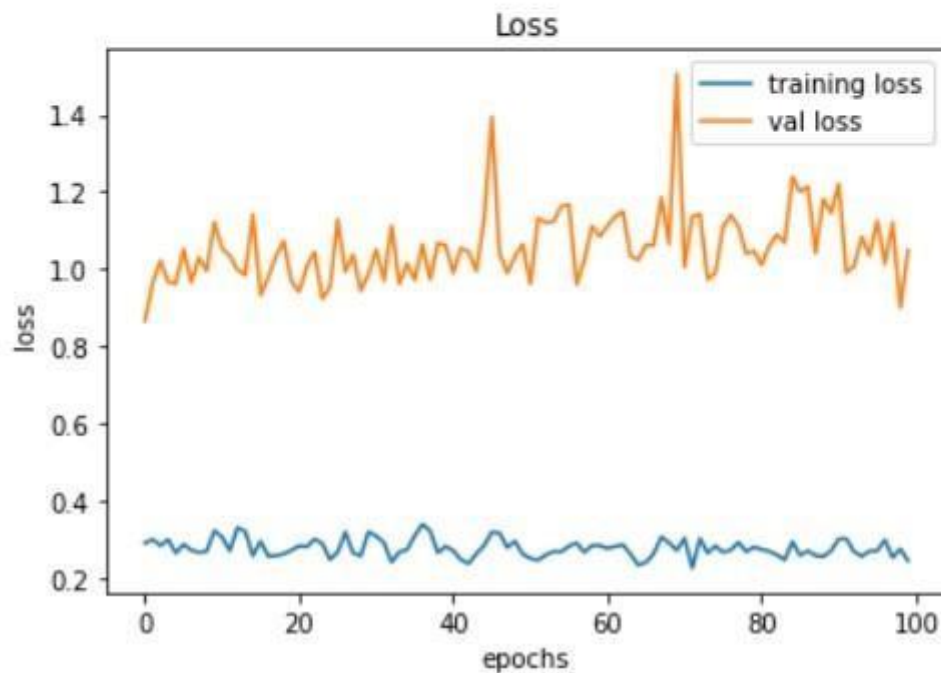
```
epochs = 100
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))

Epoch 1/100
224/224 [=====] - 1s 7ms/step - loss: 0.2913 - accuracy: 0.8978 - val_loss: 0.8671 - val_accuracy: 0.7766
Epoch 2/100
224/224 [=====] - 1s 6ms/step - loss: 0.3007 - accuracy: 0.8929 - val_loss: 0.9643 - val_accuracy: 0.7721
Epoch 3/100
224/224 [=====] - 1s 6ms/step - loss: 0.2844 - accuracy: 0.8962 - val_loss: 1.0221 - val_accuracy: 0.7766
Epoch 4/100
224/224 [=====] - 2s 8ms/step - loss: 0.3002 - accuracy: 0.8929 - val_loss: 0.9668 - val_accuracy: 0.7772
Epoch 5/100
224/224 [=====] - 1s 6ms/step - loss: 0.2989 - accuracy: 0.8954 - val_loss: 1.0142 - val_accuracy: 0.7660
Epoch 6/100
224/224 [=====] - 2s 7ms/step - loss: 0.2536 - accuracy: 0.9109 - val_loss: 1.1201 - val_accuracy: 0.7699
Epoch 7/100
224/224 [=====] - 1s 6ms/step - loss: 0.2761 - accuracy: 0.9049 - val_loss: 0.9010 - val_accuracy: 0.7682
Epoch 8/100
224/224 [=====] - 1s 6ms/step - loss: 0.2455 - accuracy: 0.9126 - val_loss: 1.0490 - val_accuracy: 0.7777
```

```
# accuracy
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```



```
# loss
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



## Accuracy with Test data

```
[ ] from sklearn.metrics import accuracy_score
    print(accuracy_score(label, pred_classes))
```

```
0.7814998096688237
```

### 3.5 Training the CNN Model (After preprocessing the dataset)

```
In [17]: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

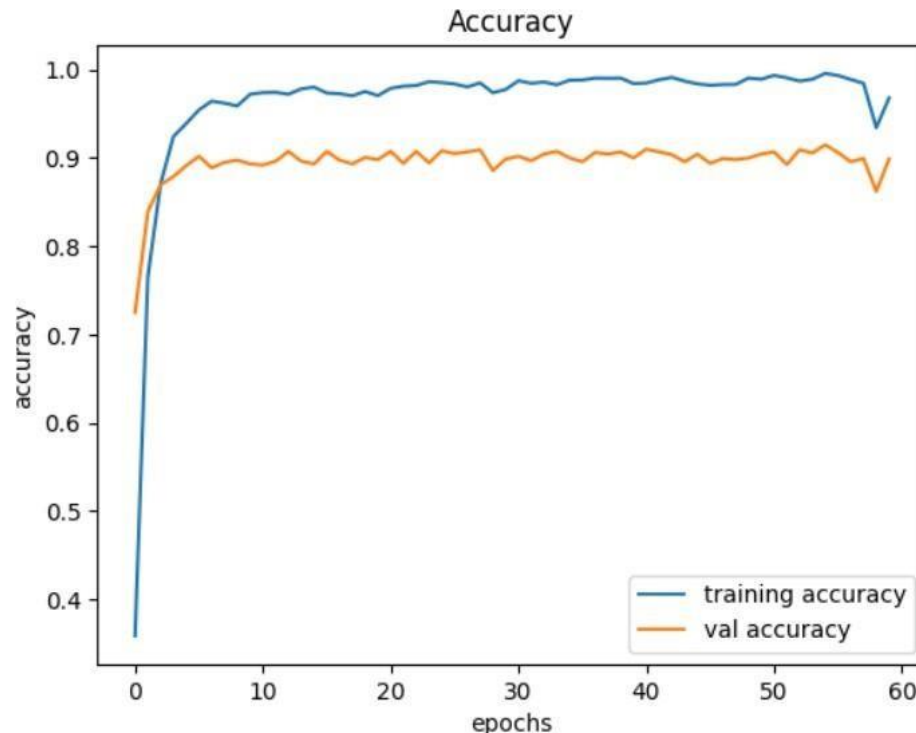
```
In [18]: epochs = 60
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
```

```
Epoch 1/60
202/202 [=====] - 67s 312ms/step - loss: 3.8634 - accuracy: 0.3585 - val_loss: 1.1664 - val_accuac
y: 0.7250
Epoch 2/60
202/202 [=====] - 62s 308ms/step - loss: 0.9025 - accuracy: 0.7654 - val_loss: 0.6509 - val_accuac
y: 0.8392
Epoch 3/60
202/202 [=====] - 62s 307ms/step - loss: 0.4524 - accuracy: 0.8709 - val_loss: 0.5569 - val_accuac
y: 0.8696
Epoch 4/60
202/202 [=====] - 56s 277ms/step - loss: 0.2716 - accuracy: 0.9243 - val_loss: 0.5571 - val_accuac
y: 0.8790
Epoch 5/60
202/202 [=====] - 52s 258ms/step - loss: 0.2082 - accuracy: 0.9390 - val_loss: 0.4925 - val_accuac
y: 0.8914
Epoch 6/60
202/202 [=====] - 41s 202ms/step - loss: 0.1480 - accuracy: 0.9545 - val_loss: 0.4890 - val_accuac
y: 0.9019
Epoch 7/60
202/202 [=====] - 41s 202ms/step - loss: 0.1480 - accuracy: 0.9545 - val_loss: 0.4890 - val_accuac
y: 0.9019
```

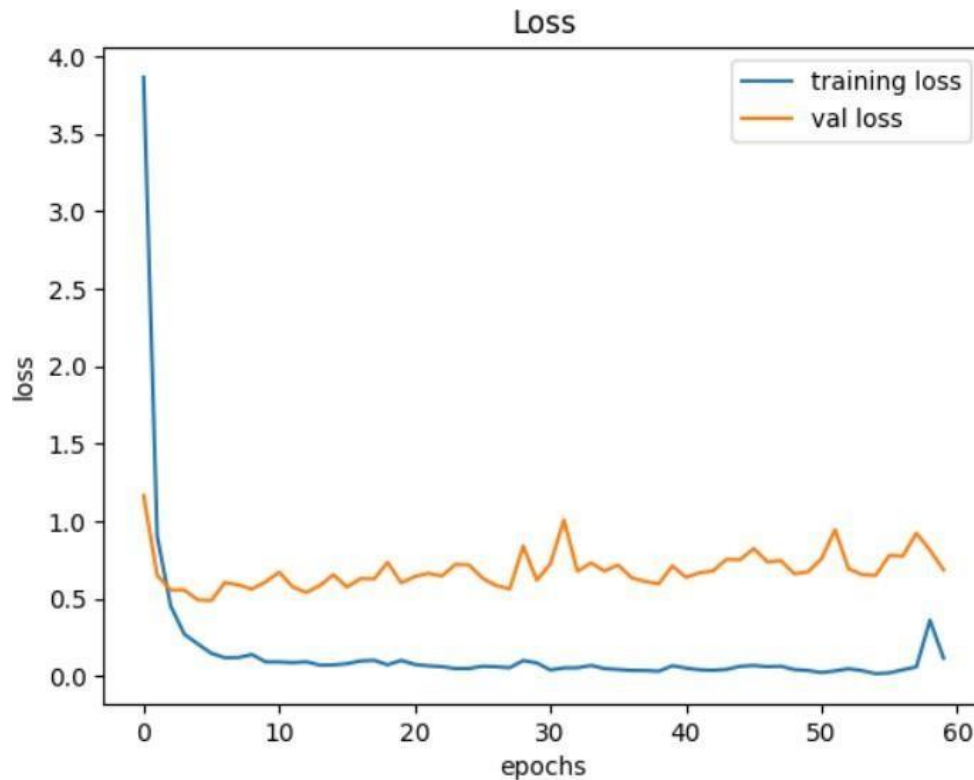
```
In [17]: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
In [18]: epochs = 60
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
```

```
y: 0.9056
Epoch 55/60
202/202 [=====] - 36s 175ms/step - loss: 0.0169 - accuracy: 0.9958 - val_loss: 0.6507 - val_accuac
y: 0.9150
Epoch 56/60
202/202 [=====] - 36s 177ms/step - loss: 0.0221 - accuracy: 0.9935 - val_loss: 0.7793 - val_accuac
y: 0.9063
Epoch 57/60
202/202 [=====] - 36s 181ms/step - loss: 0.0417 - accuracy: 0.9890 - val_loss: 0.7751 - val_accuac
y: 0.8957
Epoch 58/60
202/202 [=====] - 36s 179ms/step - loss: 0.0611 - accuracy: 0.9846 - val_loss: 0.9230 - val_accuac
y: 0.8994
Epoch 59/60
202/202 [=====] - 37s 183ms/step - loss: 0.3611 - accuracy: 0.9344 - val_loss: 0.8139 - val_accuac
y: 0.8622
Epoch 60/60
202/202 [=====] - 37s 181ms/step - loss: 0.1177 - accuracy: 0.9680 - val_loss: 0.6871 - val_accuac
y: 0.8988
```







## Accuracy with the Test dataset

```
In [28]: from sklearn.metrics import accuracy_score
print(accuracy_score(label, pred_classes))

0.9125844594594594
```

This code snippet is training the CNN model defined in the previous code snippet using the fit() method in Keras.

- epochs is an integer that specifies the number of times the entire dataset will be iterated over during training.
- The model.fit() method is called with the following parameters:
- X\_train and y\_train are the training data and labels, respectively.
- batch\_size is an integer that specifies the number of samples per gradient update.
- epochs is the number of epochs to train the model.
- validation\_data is a tuple containing the validation data and labels (X\_test, y\_test).
- The fit() method returns a history object, which contains information about the training process such as the loss and accuracy values for the training and validation data at each epoch. This object is assigned to the history variable.

After training the model, it can be used for making predictions on new data using the predict() method.

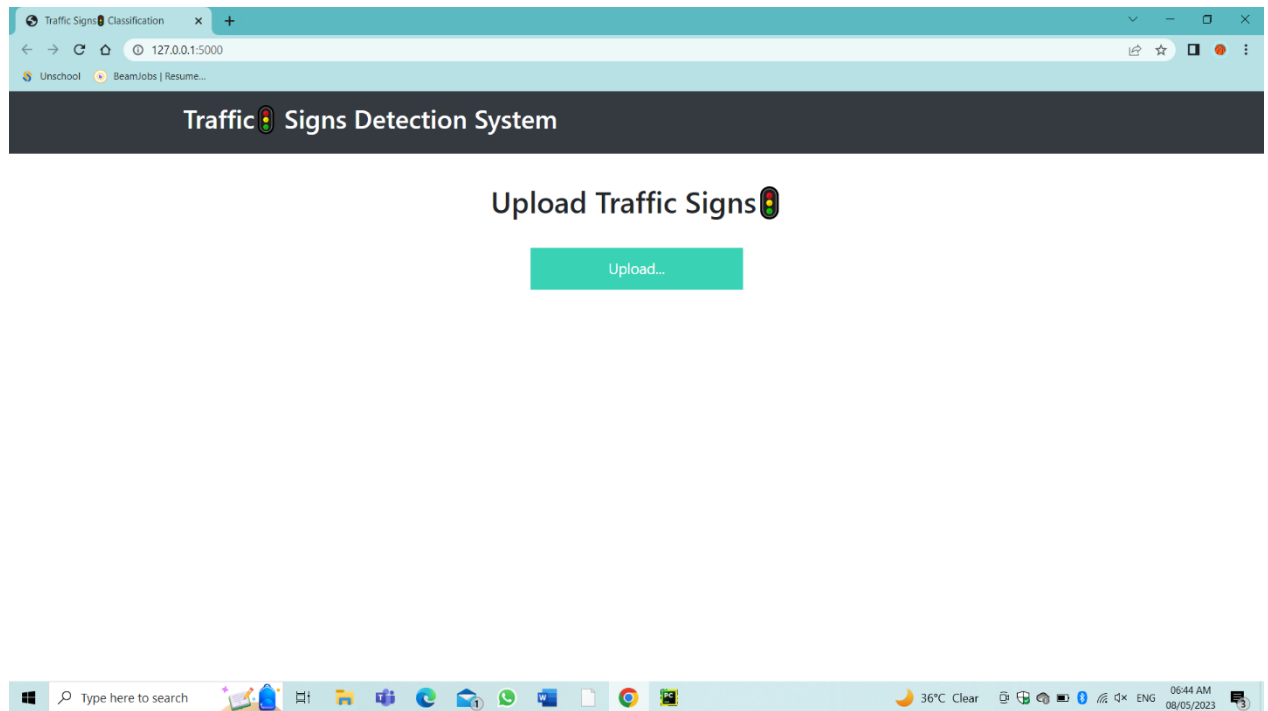
### 3.6 Save the Model

## Save the Model

```
model.save("D:\TrafficSignDetectionSystem\Web App\model\TSDR2.h5")
```

Save the trained model for further testing.

### 3.7 Implementation on Frontend side



**Fig 15**

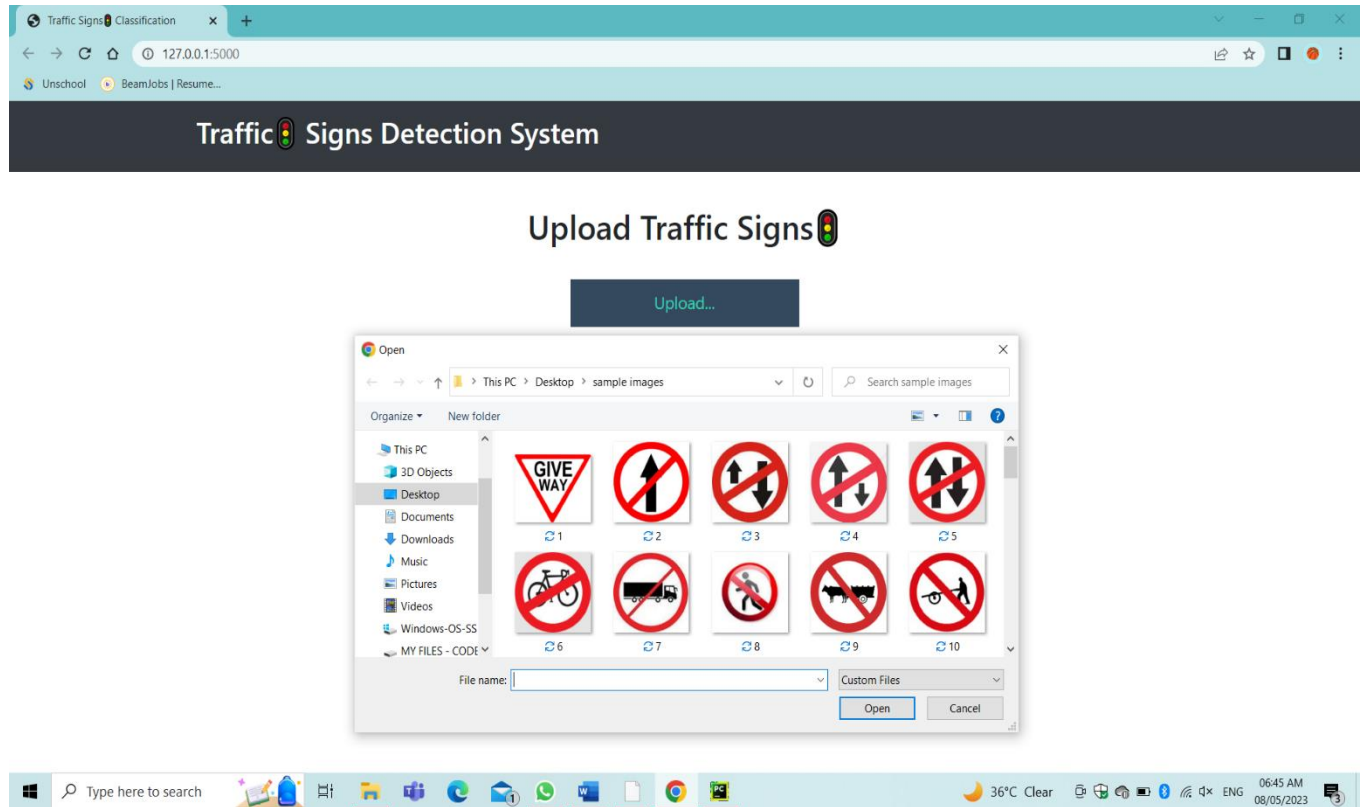


Fig 16

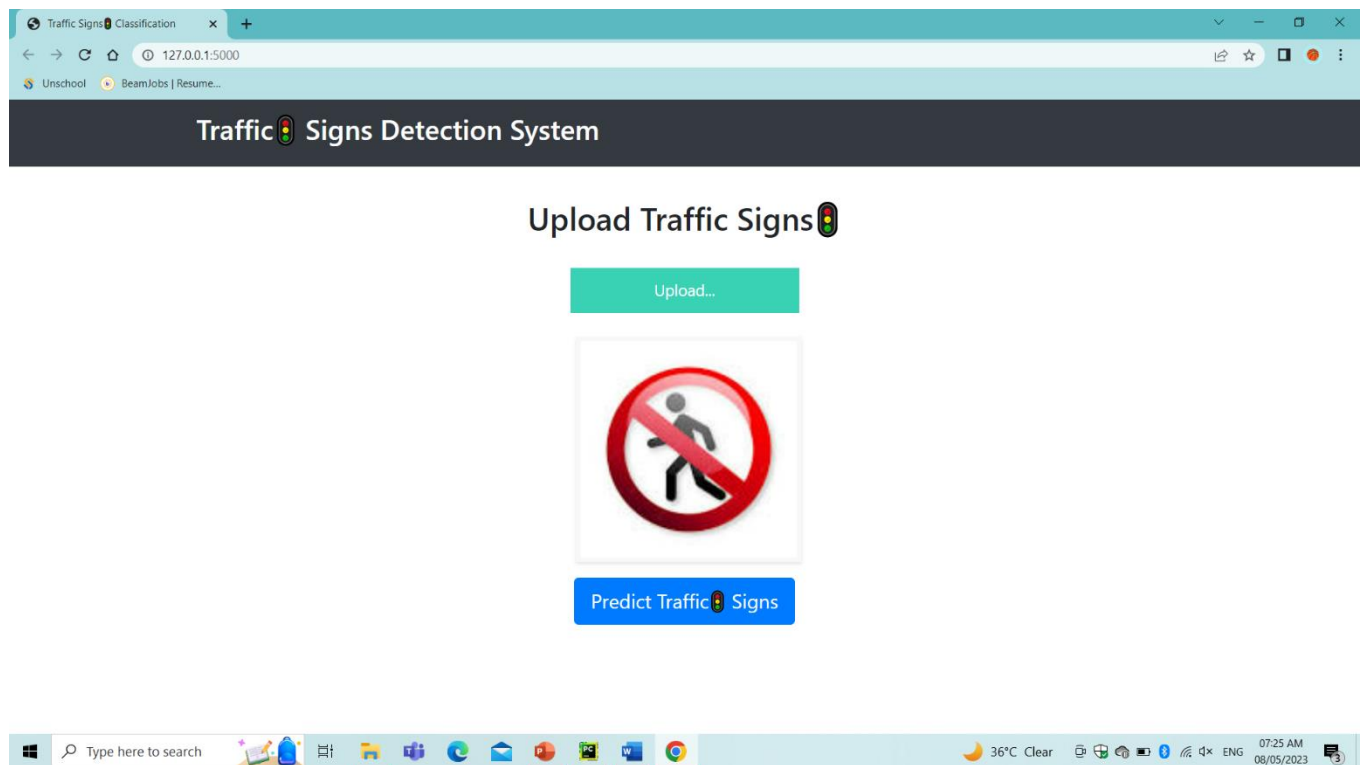
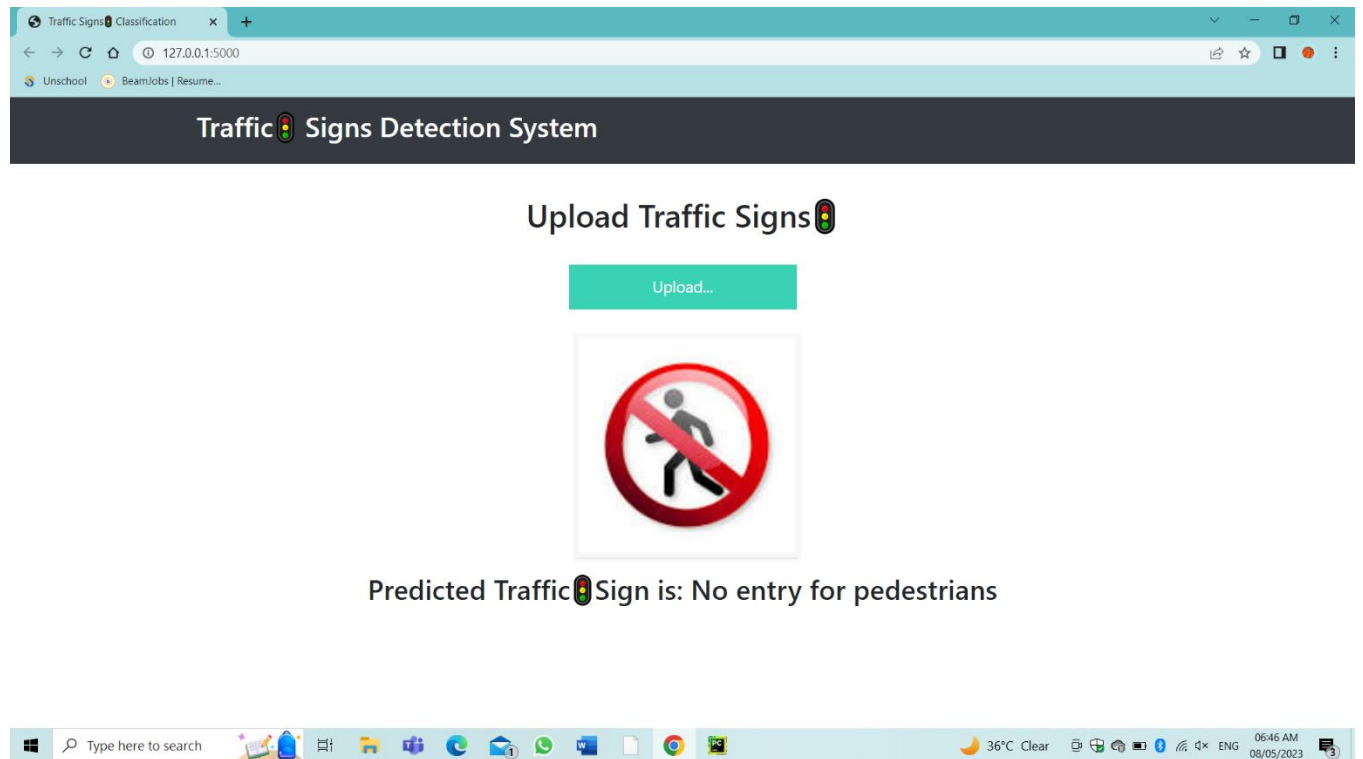


Fig 17



**Fig 18**

## Chapter 4: Conclusion and Future work

### 4.1 Conclusion

The conclusion of a report on an Indian traffic sign recognition system would depend on the specific details and results of the project. However, some general observations and conclusions could include:

- The Indian traffic sign recognition system is a complex and challenging problem that requires accurate and reliable detection and classification of traffic signs.
- Various techniques and approaches can be used to solve this problem, including machine learning, deep learning, computer vision, and image processing.
- The performance of the system can be evaluated based on various metrics such as accuracy, precision, recall, and F1-score.
- The accuracy and reliability of the system can be affected by various factors such as lighting conditions, weather conditions, camera quality, and the quality and variability of the traffic signs.
- The system can be optimized and improved by using various techniques such as data augmentation, transfer learning, fine-tuning, and model ensembling.
- The effectiveness of the system can be demonstrated by comparing it with other existing systems and benchmark datasets and by evaluating its performance in real-world scenarios.

### 4.2 Future Work

Here are some potential areas for future work on an Indian traffic sign recognition system:

**1. Improving the accuracy and reliability of the system:** One of the main goals of future work could be to improve the accuracy and reliability of the system by using more advanced techniques and algorithms for image processing, computer vision, and machine learning. For example, techniques such as transfer learning, fine-tuning, and model ensembling could be used to improve the performance of the system.

**2. Extending the system to handle new types of signs:** Another area for future work could be to extend the system to handle new types of signs that may be introduced in the future, such as signs for electric vehicle charging stations or autonomous vehicle lanes. This could involve updating the existing dataset or collecting new data and retraining the model.

**3. Adapting the system to different environments and conditions:** Another potential area for future work could be to adapt the system to different environments and conditions, such as rural or urban areas, day or night time, or different weather conditions. This could involve developing new models or algorithms that are robust to different types of noise and variability in the data.

**4. Integrating the system with other traffic management systems:** Another possible area for future work could be to integrate the traffic sign recognition system with other traffic management systems, such as traffic lights or speed cameras. This could help to improve overall traffic safety and efficiency by providing more accurate and timely information to drivers.

**5. Developing a real-time implementation of the system:** Finally, a key goal of future work could be to develop a real-time implementation of the system that can be deployed in a variety of settings, such as in-vehicle systems or roadside cameras. This would require optimizing the algorithms and models to run efficiently in real-time and ensuring that the system can handle large volumes of data and traffic.

## Chapter 5: References

- General idea is taken from Internet.
- For model and cnn :  
[https://www.youtube.com/watch?v=qahpZkPI'IRM&ab\\_channel=MachineLearningHub](https://www.youtube.com/watch?v=qahpZkPI'IRM&ab_channel=MachineLearningHub)
- Dataset : <https://www.kaggle.com/search?q=traffic+signs>
- GitHub : [https://github.com/Spidy20/Traffic\\_Signs\\_WebApp](https://github.com/Spidy20/Traffic_Signs_WebApp)
- Learning CNN :  
[https://www.youtube.com/watch?v=zfiSAzpy9NM&ab\\_channel=codebasics](https://www.youtube.com/watch?v=zfiSAzpy9NM&ab_channel=codebasics)  
<https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9>