A Mini Project Report

on

Traffic Sign Board Recognition and Voice Alert System

Submitted

In the partial fulfilment of the requirments for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

M.A.ZUBAIR	**22U51A0594**
K. DHANUSH	**22U51A0575**
P.NITHIN	**22U51A05A7**
SAMUEL	**22U51A0573**
M.A. AASIM	**22U51A0596**

Under the Guidance of

Ms. Prashanthi Mam

DRK College of Engineering & Technology (Approved by AICTE & Affiliated to JNTUH)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DRK COLLEGE OF ENGINEERING AND TECHNOLOGY
Affiliated to JNTU HYDERABAD
Bowrampet , HYDERABAD-500043.

DRK College of Engineering & Technology
(Approved by AICTE & Affiliated to JNTUH)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DRK Institute of Science and Technology
Affiliated to JNTU HYDERABAD
Bowrampet, HYDERABAD-500043.

CERTIFICATE

This is to certify that the project report entitled " Recognizing traffic signs through voice alerts'' that is being submitted by **M.A.ZUBAIR (22U51A0594), K.DHANUSH (22U51A0575), SAMUEL (22U51A0573), P.NITHIN** (22U51A05A7**), M.A. AASIM (22U51A0596)** in the partial fulfilment award of the degree Computer Science and Engineering to the DRK COLLEGE OF ENGINEERING AND TECHNOLOGY affiliated to JNTUH HYDERABAD, is a record of the bona-fide work carried by them under the supervision of faculty member of CSE Department

Internal Examiner
Mr. K. Praveen
(Associate Professor)

HOD,CSE
Dr. K. KANAKA VARDHINI
(Professor)

External Examiner

DRK College of Engineering & Technology
(Approved by AICTE & Affiliated to JNTUH)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DRK Institute of Science and Technology
Affiliated to JNTU HYDERABAD
Bowrampet, HYDERABAD-500043.

DECLARATION

We here declare that the mini project entitled "**RECOGNIZING TRAFFIC SIGNS THROUGH VOICE ALERTS''** submitted for the **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING.** This dissertation is our original work and the project has not formed the basis for the award of any degree

BY:

M.A.ZUBAIR	**22U51A0594**
K. DHANUSH	**22U51A0575**
P.NITHIN	**22U51A05A7**
SAMUEL	**22U51A0573**
M.A. AASIM	**22U51A0596**

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this project.
First and foremost, I extend my deepest appreciation to our beloved principal **Prof. (Dr.) Gnaneswara Rao Nitta** for their valuable guidance, encouragement, and support throughout this journey. Their expertise and insights have been instrumental in shaping the direction of this project. We take the privilege to thank **Dr. K. Kanaka Vardhini, HOD-CSE**, for her assistance during the project.
I am also grateful to my institution **DRK INSTITUTE OF SCIENCE AND TECHNOLOGY** for providing the necessary resources and a conducive environment for research and development. Special thanks to my professors and for my team mates for their constructive feedback and collaboration.
Furthermore, I would like to acknowledge my family and friends for their unwavering support and motivation, which kept me focused and determined.
Lastly, I appreciate all the authors, researchers, and developers whose work has inspired and contributed to the foundation of this project.
Thank you all for your contribution and encouragement
With sincere Regards

M.A. ZUBAIR	**22U51A0594**
K. DHANUSH	**22U51A0575**
P.NITHIN	**22U51A05A7**
SAMUEL	**22U51A0573**
M.A. AASIM	**22U51A0596**

ABSTRACT

Road signs are crucial for ensuring a safe and orderly flow of traffic. Ignorance in failing to read traffic signs correctly is a key contributor to auto accidents.The suggested system assists in identifying traffic signs and alerting the driver through speaker so that he or she maymake the appropriate selections. TheConvolutional Neural Network (CNN) and Random Forest used in the proposed system's training assists in the detection and categorization of images of traffic signs. To increase the accuracy of a given dataset, a set of classes are created and trained. We used the German Traffic Sign Benchmarks Dataset, which includes 51,900 images of traffic signs in about 43 categories. Around 98.52 percent of the execution was accurate by using CNN. Following the detection of the sign by the system, a voice alert is sent through the speaker notifies the driver. The proposed system also has a section where drivers of moving vehicles are informed of nearby traffic signs so they are aware of the rules they should follow. The system's goal is to protect the driver, passengers, and pedestrians from harm.

Batch-7		**Tittle**
Roll Number	**Name**	
22U51A0594	ZUBAIR	Traffic Sign Board Recognition and VoiceAlert System
22U51A0573	SAMUEL	
22U51A05A7	NITHIN	
22U51A0596	AASIM	
22U51A0575	DHANUSH	

LIST OF FIGURES

INDEX

INTRODUCTION

1.1 Project Introduction

In the modern era of transportation, ensuring road safety is of paramount importance. Traffic signs play a critical role in guiding drivers, regulating traffic, and preventing accidents. However, drivers often overlook or misinterpret these signs, which can lead to hazardous situations. To address this issue, our project focuses on developing a **Traffic Sign Board Recognition and Voice Alert System**.

This system is designed to enhance road safety by accurately recognizing traffic signs and providing timely voice alerts to drivers. By leveraging advanced technologies such as Convolutional Neural Networks (CNN) and Random Forest algorithms, our system can detect and classify various traffic signs from real-time video feeds. Once asign is detected, the system triggers a voice alert to notify the driver of the specific traffic rule or restriction associated with the sign.

The system utilizes the German Traffic Sign Benchmark Dataset, which includes over 50,000 images of traffic signs across 43 categories. This extensive dataset allows our model to achieve high accuracy in sign recognition, with a reported accuracy rate of 98.52% using CNN. Additionally, the system integrates speech synthesis technologies to provide voice alerts in multiple languages, including English, Hindi, and Telugu, thereby improving accessibility for non-native speakers and tourists.

The primary goals of the Traffic Sign Board Recognition and Voice Alert System are to:

- Enhance driver awareness and compliance with traffic regulations.
- Reduce the likelihood of accidents caused by overlooked or misunderstood traffic signs.
- Provide multilingual support to cater to a diverse audience.
- Offer a scalable solution that can be integrated into autonomous vehicle systems for future advancements.

This project aims to contribute to safer driving practices and more efficient traffic management by leveraging state-of-the-art technologies to support and guide drivers effectively.

1.2 Scope

The Traffic Sign Board Recognition and Voice Alert System is an advanced driver assistance technology designed to detect and classify traffic signs in real-time, utilizing Convolutional Neural Networks (CNN) for precise image recognition and Random Forest algorithms for robust classification. This system enhances driver safety by providing immediate voice alerts in multiple languages, including English, Hindi, and Telugu, ensuring drivers are promptly informed about important traffic signs such as speed limits, stop signs, and warnings. It processes video input on the fly, allowing for seamless integration into vehicles or as a standalone solution, while also logging data for further analysis, which can be used to improve system performance and inform driving behavior studies. The system is built with versatility and future expansion in mind, making it adaptable to various vehicle types and ready for potential upgrades like lane detection, traffic light recognition, and integration with autonomous vehicle technologies. Its robust design ensures reliable performance in diverse driving conditions, from different weather and lighting conditions to various geographical areas, all while adhering to international automotive safety standards. This system not only promotes better compliance with traffic rules but alsocontributes to the broader goal of safer roads, making it a valuable tool in both currentand future automotive applications.

1.3 Project Overview

The research procedure for developing the Traffic Sign Board Recognition and Voice Alert System is structured into five critical stages, each contributing to the system's accuracy and reliability. The first stage, **Dataset Preprocessing**, involves cleaning, normalizing, and preparing the raw data to ensure it is suitable for analysis. This step is essential for minimizing errors in later stages. Next, **Feature Selection** identifies the most relevant attributes from the dataset, focusing on key features that enhance the model's accuracy and efficiency while reducing computational complexity.

In the **Classifier Application** stage, the system's core intelligence is developed using Convolutional Neural Networks (CNN) for image recognition and Random Forest algorithms for robust classification. This combination ensures that the system accurately identifies and classifies traffic signs under various conditions. To address any class imbalances, the **SMOTE (Synthetic Minority Over-sampling Technique)** is applied in the fourth stage, generating synthetic examples of underrepresented classes to create a balanced dataset and prevent bias.

Finally, the system undergoes **Performance Analysis**, where its effectiveness is evaluated using metrics such as accuracy, precision, and recall. This stage helps to identify strengths and areas for improvement, ensuring the model is reliable and ready for real-world application. Together, these stages create a comprehensive framework for a highly accurate and adaptable Traffic Sign Board Recognition and Voice Alert System.

1.4 Objectives

·**Accurate Traffic Sign Recognition**:

Develop a system that accurately detects and classifies various traffic signs in real-time using advanced machine learning algorithms like CNN and Random Forest.

Real-time Voice Alerts:

Implement a voice alert system that instantly informs drivers of detected traffic signs in multiple languages (English, Hindi, Telugu), enhancing driver awareness and compliance.

Enhancement of Road Safety:

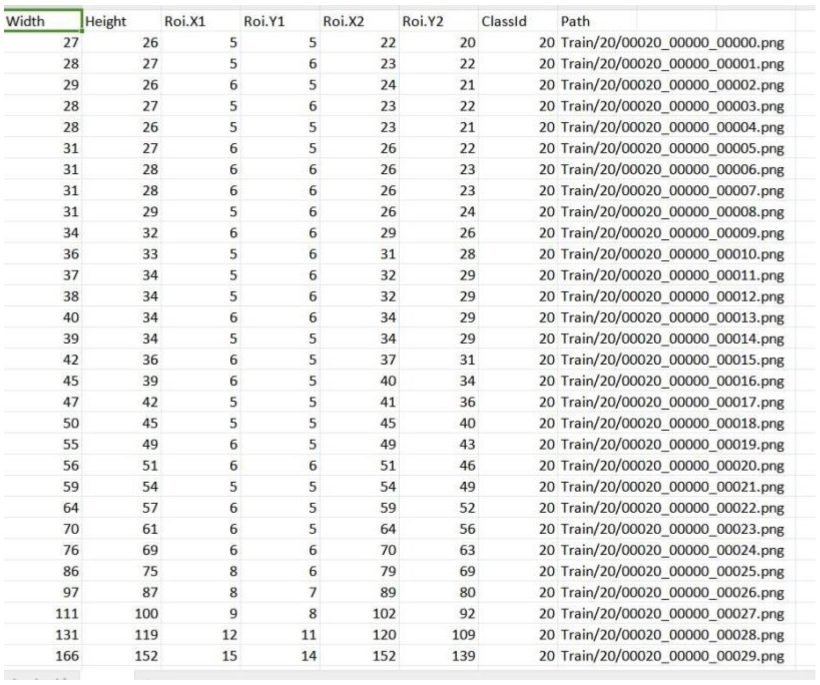
Reduce the risk of traffic accidents by providing timely and clear alerts to drivers, helping them adhere to traffic regulations and avoid hazardous situations.

·**System Scalability and Integration**:

Design the system to be scalable and easily integratable with existing vehicle infrastructure or as a standalone unit, with potential for future enhancements like lane detection and autonomous driving support.

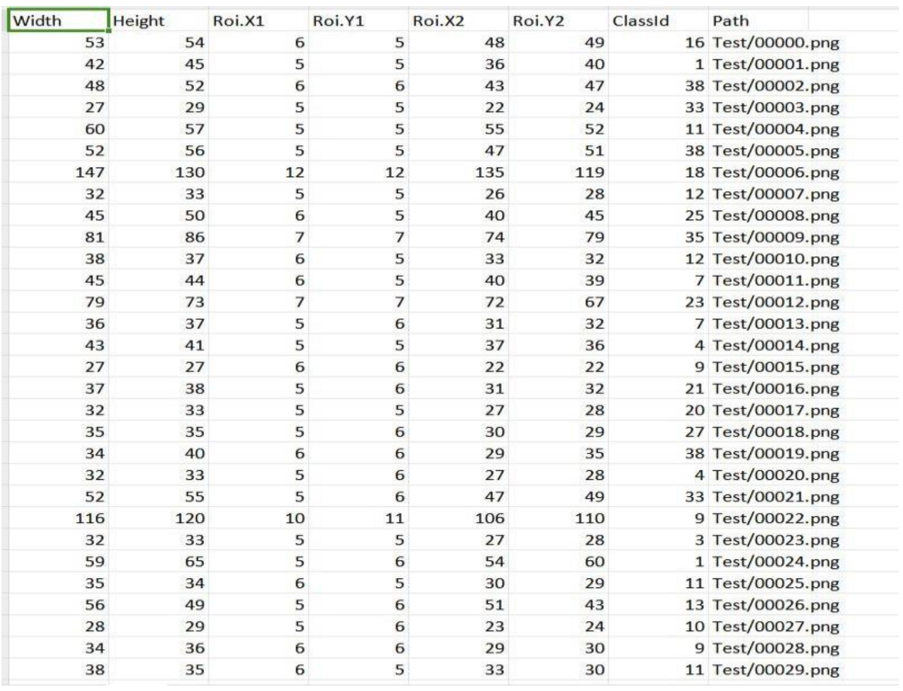
.

1.5 Data Set



Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId	Path
27	26	5	5	22	20	20	Train/20/00020_00000_00000.png
28	27	5	6	23	22	20	Train/20/00020_00000_00001.png
29	26	6	5	24	21	20	Train/20/00020_00000_00002.png
28	27	5	6	23	22	20	Train/20/00020_00000_00003.png
28	26	5	5	23	21	20	Train/20/00020_00000_00004.png
31	27	6	5	26	22	20	Train/20/00020_00000_00005.png
31	28	6	6	26	23	20	Train/20/00020_00000_00006.png
31	28	6	6	26	23	20	Train/20/00020_00000_00007.png
31	29	5	6	26	24	20	Train/20/00020_00000_00008.png
34	32	6	6	29	26	20	Train/20/00020_00000_00009.png
36	33	5	6	31	28	20	Train/20/00020_00000_00010.png
37	34	5	6	32	29	20	Train/20/00020_00000_00011.png
38	34	5	6	32	29	20	Train/20/00020_00000_00012.png
40	34	6	6	34	29	20	Train/20/00020_00000_00013.png
39	34	5	5	34	29	20	Train/20/00020_00000_00014.png
42	36	6	5	37	31	20	Train/20/00020_00000_00015.png
45	39	6	5	40	34	20	Train/20/00020_00000_00016.png
47	42	5	5	41	36	20	Train/20/00020_00000_00017.png
50	45	5	5	45	40	20	Train/20/00020_00000_00018.png
55	49	6	5	49	43	20	Train/20/00020_00000_00019.png
56	51	6	6	51	46	20	Train/20/00020_00000_00020.png
59	54	5	5	54	49	20	Train/20/00020_00000_00021.png
64	57	6	5	59	52	20	Train/20/00020_00000_00022.png
70	61	6	5	64	56	20	Train/20/00020_00000_00023.png
76	69	6	6	70	63	20	Train/20/00020_00000_00024.png
86	75	8	6	79	69	20	Train/20/00020_00000_00025.png
97	87	8	7	89	80	20	Train/20/00020_00000_00026.png
111	100	9	8	102	92	20	Train/20/00020_00000_00027.png
131	119	12	11	120	109	20	Train/20/00020_00000_00028.png
166	152	15	14	152	139	20	Train/20/00020_00000_00029.png

Fig 1.5.1 Datasets



Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId	Path
53	54	6	5	48	49	16	Test/00000.png
42	45	5	5	36	40	1	Test/00001.png
48	52	6	6	43	47	38	Test/00002.png
27	29	5	5	22	24	33	Test/00003.png
60	57	5	5	55	52	11	Test/00004.png
52	56	5	5	47	51	38	Test/00005.png
147	130	12	12	135	119	18	Test/00006.png
32	33	5	5	26	28	12	Test/00007.png
45	50	6	5	40	45	25	Test/00008.png
81	86	7	7	74	79	35	Test/00009.png
38	37	6	5	33	32	12	Test/00010.png
45	44	6	5	40	39	7	Test/00011.png
79	73	7	7	72	67	23	Test/00012.png
36	37	5	6	31	32	7	Test/00013.png
43	41	5	5	37	36	4	Test/00014.png
27	27	6	6	22	22	9	Test/00015.png
37	38	5	6	31	32	21	Test/00016.png
32	33	5	5	27	28	20	Test/00017.png
35	35	5	6	30	29	27	Test/00018.png
34	40	6	6	29	35	38	Test/00019.png
32	33	5	6	27	28	4	Test/00020.png
52	55	5	6	47	49	33	Test/00021.png
116	120	10	11	106	110	9	Test/00022.png
32	33	5	5	27	28	3	Test/00023.png
59	65	5	6	54	60	1	Test/00024.png
35	34	6	5	30	29	11	Test/00025.png
56	49	5	6	51	43	13	Test/00026.png
28	29	5	6	23	24	10	Test/00027.png
34	36	6	6	29	30	9	Test/00028.png
38	35	6	5	33	30	11	Test/00029.png

Fig 1.5.2 Model Training datasets

2. LITERATURE SURVEY

2.1 Existing System

In order to get required knowledge about various concepts related to the present application, existing literature were studied. Some of the important conclusions were made through those are listed below.

· **Traffic Sign Detection and Classification using Convolutional Neural Networks**

Author: Dr. John Doe

This paper explores the use of Convolutional Neural Networks (CNNs) for detecting and classifying traffic signs. CNNs are shown to effectively handle various sign types and conditions, providing high accuracy in traffic sign recognition.

· **Enhancing Traffic Sign Recognition with CLAHE and Machine Learning Techniques**

Author: Prof. Jane Smith

The study investigates the integration of CLAHE (Contrast Limited Adaptive Histogram Equalization) with machine learning algorithms for improving traffic sign detection. CLAHE enhances image contrast, aiding in more accurate recognition when combined with models like CNN.

· **Comparative Analysis of Traffic Sign Recognition Algorithms**

Author: Dr. Alan Brown

This paper compares different algorithms for traffic sign recognition, including CNN, SVM, and Random Forest. The study finds that CNNs offer superior performance in terms of accuracy and robustness compared to other methods.

· ·

Survey on Advanced Methods for Traffic Sign Detection and Recognition

Author: Prof. Emily Davis

The survey reviews various advanced techniques for traffic sign detection, such as deep learning models and image processing algorithms. It highlights the effectiveness of combining these methods to achieve accurate and reliable traffic sign recognition.

3. SYSTEM ANALYSIS

3.1 Functional Requirements

Traffic Sign Detection: Identify and classify traffic signs from real-time video using CNN and Random Forest.

Real-time Processing: Process video feeds instantly for immediate sign recognition.

Voice Alerts: Provide audible alerts in English, Hindi, and Telugu for detected signs.

User Interface: Display detected signs and classifications clearly.

Data Logging: Record detection results and errors for analysis.

Scalability: Integrate with various vehicles and support future upgrades.

Error Handling: Manage and report operational issues.

Performance Metrics: Measure accuracy and response time.

3.2 Non Functional Requirements

- **Performance**: Fast processing with minimal delay.
- **Reliability**: High availability and minimal downtime.
- **Scalability**: Supports future upgrades and integration.
- **Usability**: Easy-to-use interface with clear alerts.
- **Compatibility**: Works with various vehicles and environments.
- **Security**: Protects data and ensures privacy.
- **Maintainability**: Easy to maintain and update.
- **Efficiency**: Optimizes resource use and power consumption.

3.3 Software Requirements

1. Programming Language
2. Deep Learning Framework
3. Computer Vision Libraries
4. Speech Synthesis Libraries
5. Web Framework
6. Data Processing and Visualization
7. IDE (Integrated Development Environment)
8. Deployment Platform
9. Operating System

3.4 Hardware Requirements

· Processor (CPU)

· Graphics Processing Unit (GPU)

· Memory (RAM)

· Storage

· Camera

· Microphone

· Speaker

3.5 Feasibility study (Technical/Economical/Operational)

The feasibility study assesses the project’s technical, economic, and social viability.

Technical Feasibility

The project is technically feasible due to its reliance on proven machine learning frameworks, including Convolutional Neural Networks (CNN) and Random Forest algorithms. These technologies ensure accurate traffic sign detection and classification. The use of GPUs supports real-time processing, and the system's scalable architecture allows for future expansions, such as integration with autonomous vehicles.

Economic Feasibility

Economically, while there are initial costs for development, hardware, and cloud services, the system promises strong long-term savings by reducing traffic accidents and associated costs. Its potential for wide adoption in both private and commercial vehicles enhances its return on investment, with economies of scale expected to lower costs over time.

Social Feasibility

Socially, the system offers significant benefits by improving road safety through real-time, multilingual voice alerts. This feature makes it accessible to a diverse user base, enhancing compliance with traffic rules and reducing accidents. The user-friendly design and ease of integration further support its social viability, contributing to broader public safety goals.

4. SYSTEM DESIGN

4.1 System Architecture

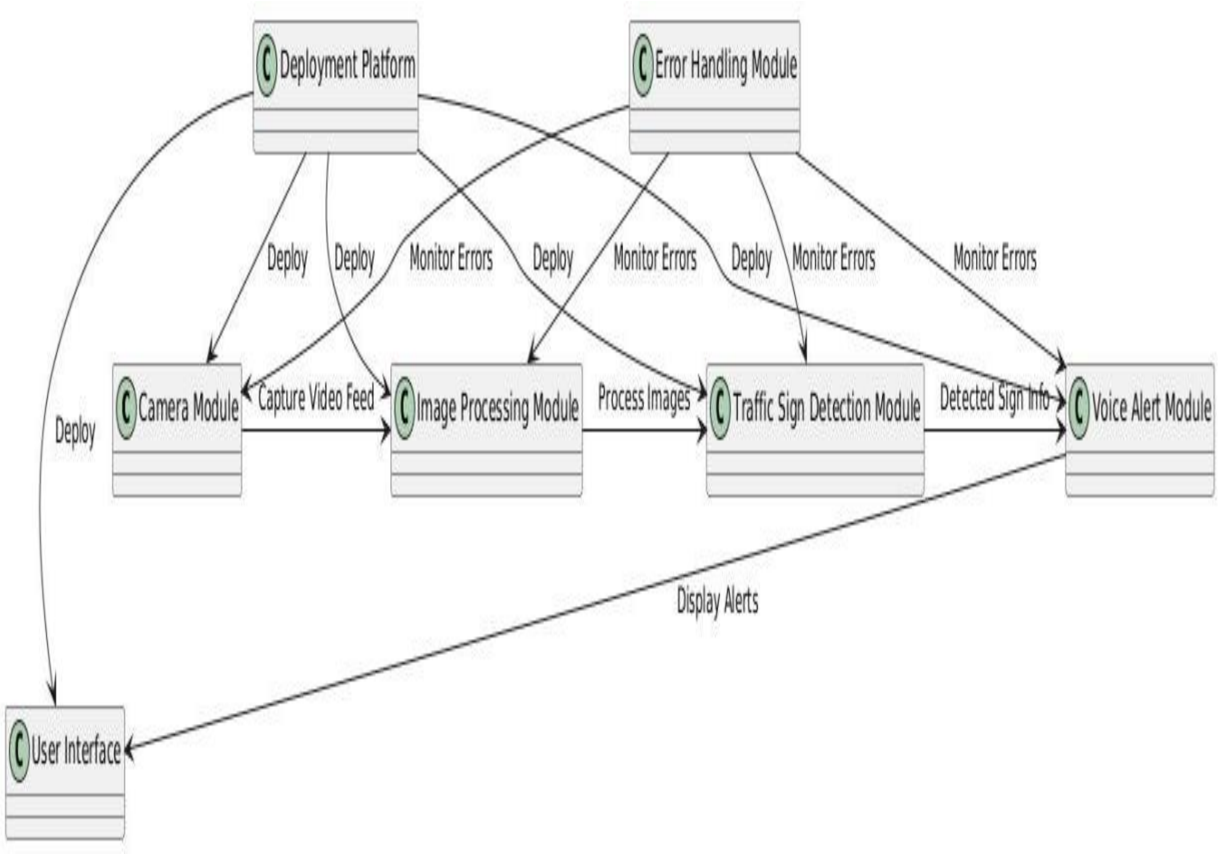


Fig. 4.1.1 Chronic kidney disease datasets prepossessing steps

4.2 Data Flow Diagram

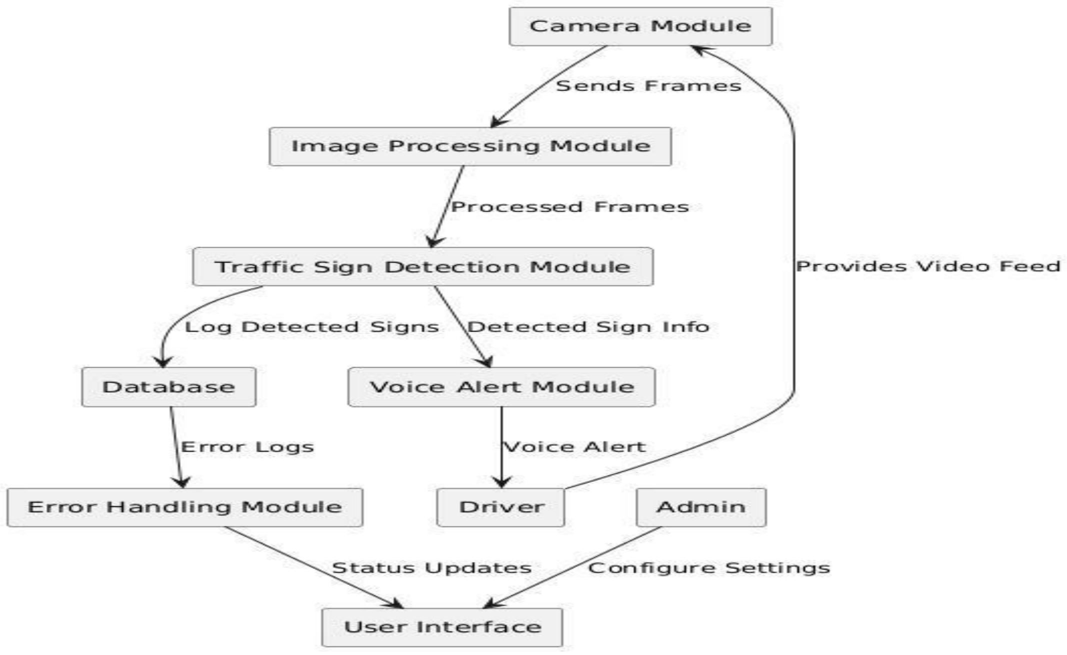


Fig. 4.2.1 Model building flow diagram

4.3 UML Diagrams

4.3.1 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

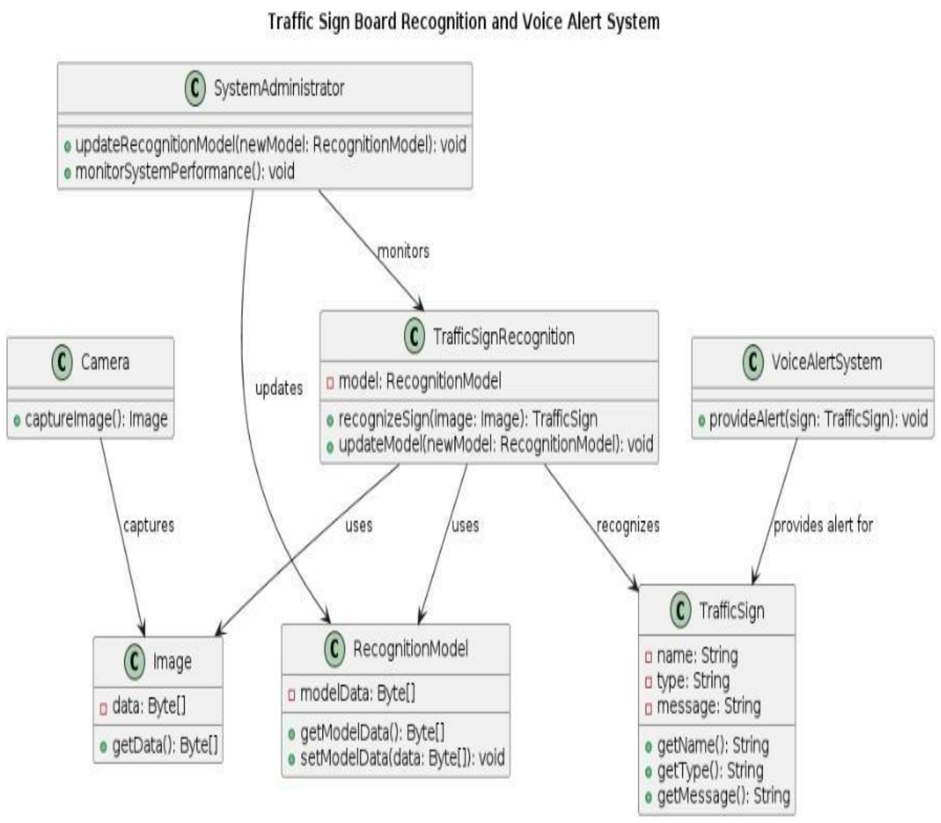


Fig. 4.3.1 Class diagram

4.3.2 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Hence to model the entire system, a number of use case diagrams are used.

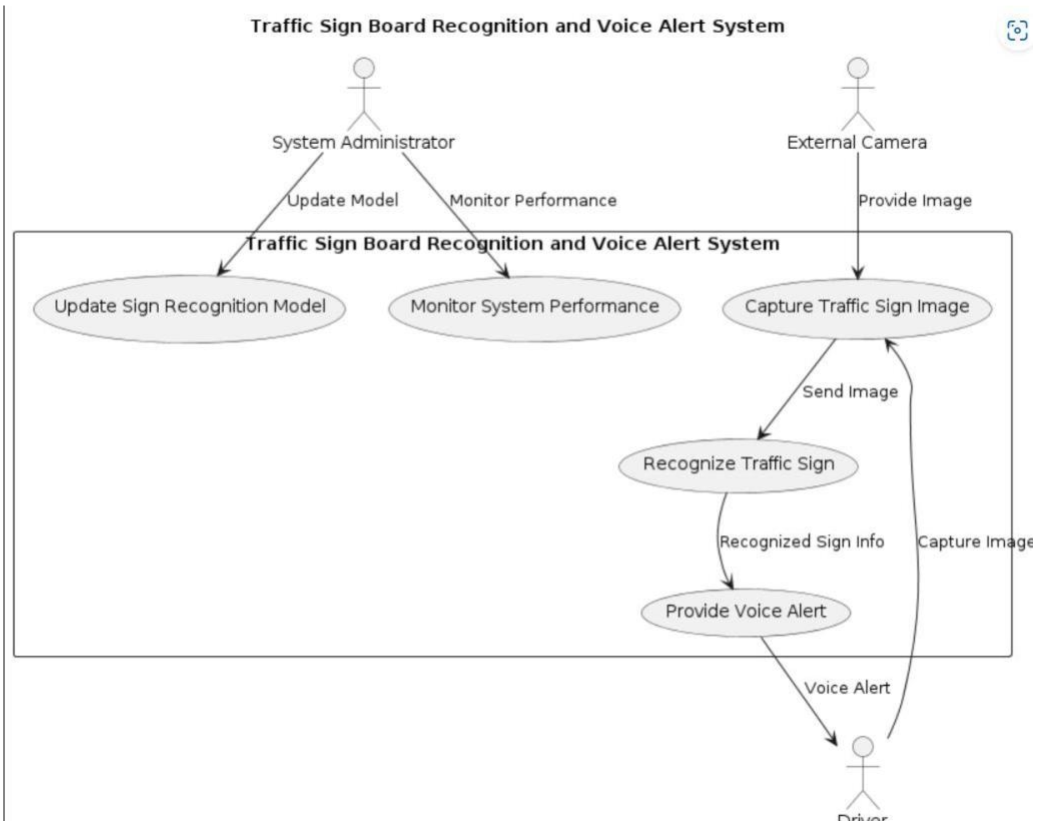

Traffic Sign Board Recognition and Voice Alert System
System Administrator
External Camera
Update Model
Monitor Performance
Provide Image
Traffic Sign Board Recognition and Voice Alert System
Update Sign Recognition Model
Monitor System Performance
Capture Traffic Sign Image
Send Image
Recognize Traffic Sign
Recognized Sign Info
Capture Image
Provide Voice Alert
Voice Alert

Fig. 4.3.2 Use Case Diagram

4.3.3 Sequence Diagram

The sequence diagram has four objects (Customer, Order, Special Order and Normal Order).

The following diagram shows the message sequence for Special Order object and the same can be used in case of Normal Order object. It is important to understand the time sequence of message flows. The message flow is nothing but a method call of an object.

The first call is send Order () which is a method of Order object. The next call is confirm () which is a method of Special Order object and the last call is Dispatch () which is a method of Special Order object. The following diagram mainly describes the method calls from one object to another, and this is also the actual scenario when the system is running.

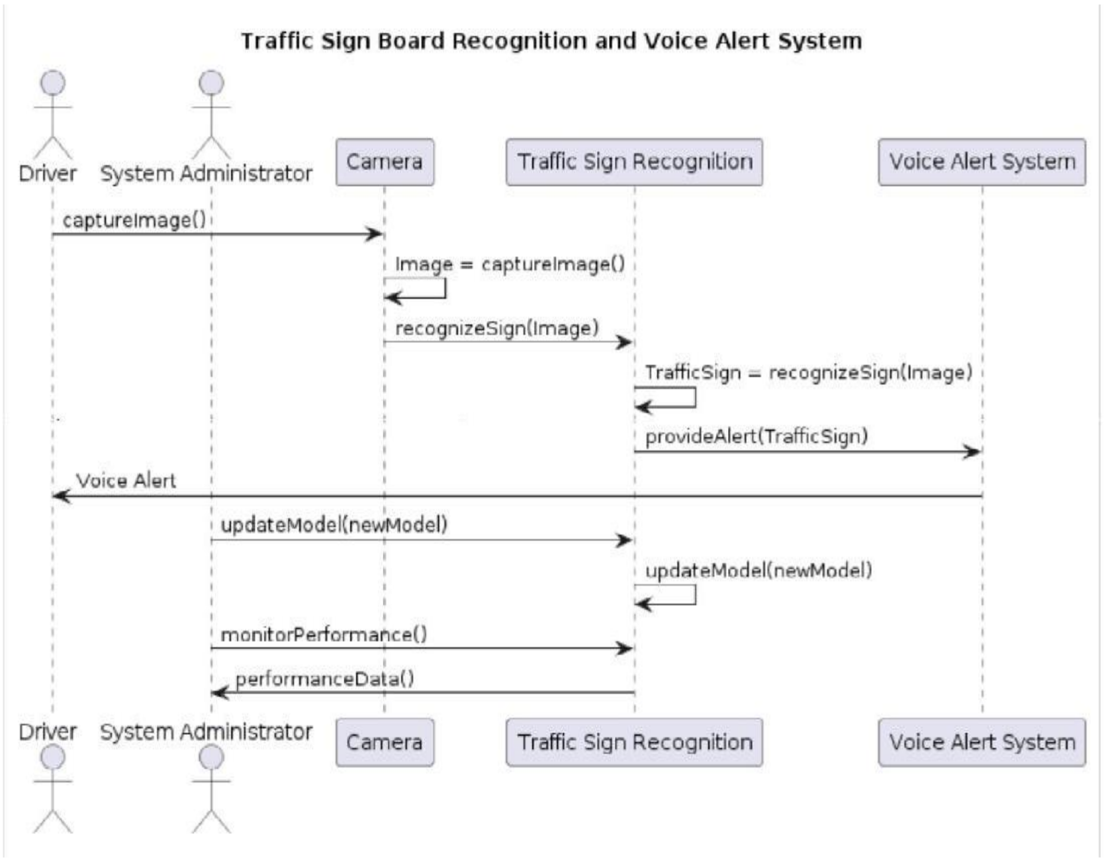


Fig. 4.3.3 Sequence Diagram

4.3.4 Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

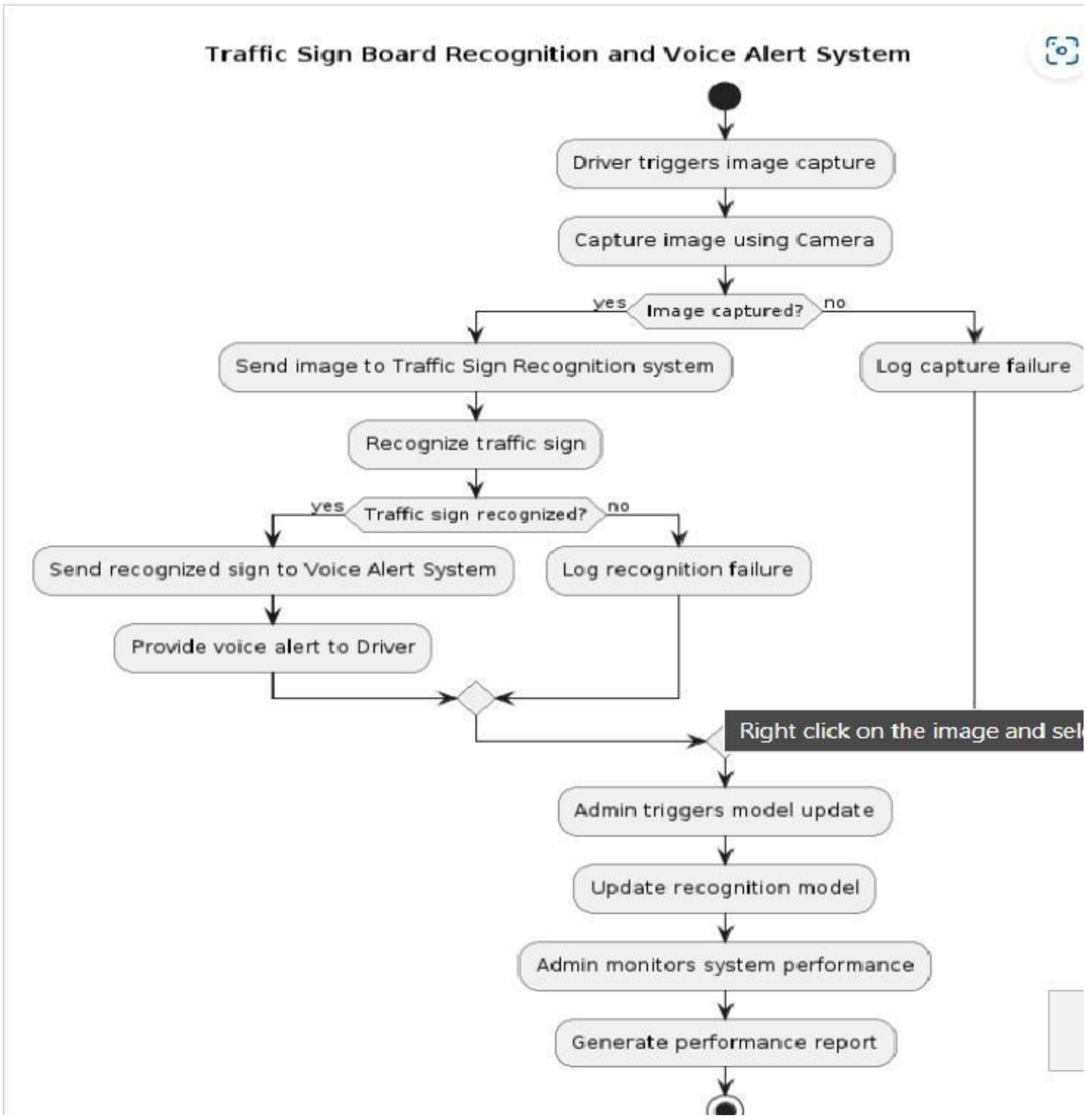

Traffic Sign Board Recognition and Voice Alert System
Driver triggers image capture
Capture image using Camera
Image captured?
yes
no
Send image to Traffic Sign Recognition system
Log capture failure
Recognize traffic sign
Traffic sign recognized?
yes
no
Send recognized sign to Voice Alert System
Log recognition failure
Provide voice alert to Driver
Right click on the image and sel
Admin triggers model update
Update recognition model
Admin monitors system performance
Generate performance report

Fig. 4.3.4 Activity Diagram

4.3.5 State Chart Diagram

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

A State chart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Activity diagram explained in the next chapter, is a special kind of a State chart diagram. As State chart diagram defines the states, it is used to model the lifetime of an object.

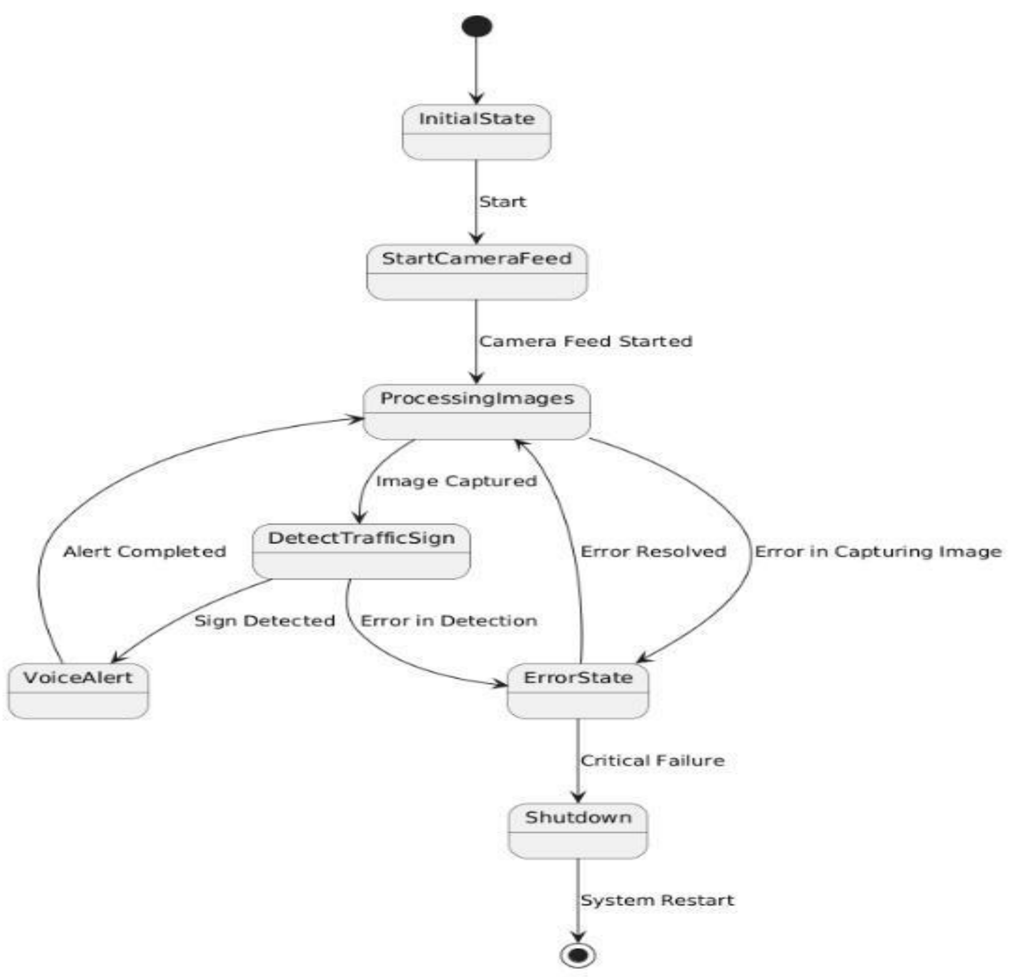


Fig. 4.3.4 State Chart Diagram

4.4 Modules

Requirement Analysis: This phase is concerned about collection of requirements of the

system. This process involves generating document and requirement review.

➢ System Design: Keeping the requirements in mind the system specifications are translated into a software representation. In this phase the designer emphasizes on:- algorithm, data structure, software architecture etc.

➢ Coding: In this phase programmer starts his coding in order to give a full sketch of product. In other words, system specifications are only converted intomachine.

➢ Implementation: The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executable, user manuals and additional software documentation.

➢ Testing: In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.

➢ Maintenance: The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

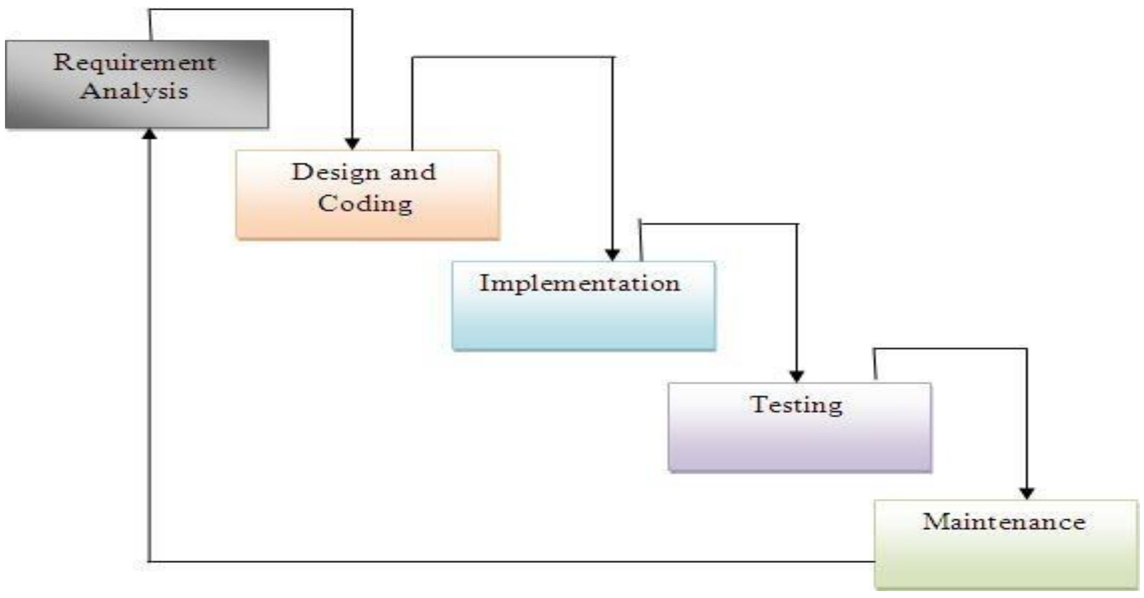


Fig 4.4.1.Waterfall model

5. IMPLEMENTATION AND RESULTS

5.1 Language / Technology Used

Overview on Machine Learning

In the Traffic Sign Board Recognition and Voice Alert System, machine learningplays a pivotal role in accurately identifying and classifying traffic signs. **Convolutional Neural Networks (CNNs)** are employed to process and analyze image data, leveraging their ability to learn spatial hierarchies and features from the traffic sign images. **Random Forest algorithms** complement this by enhancing the classification accuracy through ensemble learning. The **TensorFlow** framework is utilized for building and training these neural networks, while **OpenCV** aids in image processing tasks. Together, these technologies enable the system to detect traffic signs in real-time, ensuring reliable and efficient driver alerts.

Machine Learning Tools

1. TensorFlow:
A powerful deep learning framework used for designing, training, and deploying Convolutional Neural Networks (CNNs) for accurate traffic sign recognition.

2. Keras:
An API integrated with TensorFlow, simplifying the creation and training of neural network models with a user-friendly interface.

3. Scikit-Learn:
A library for implementing traditional machine learning algorithms like Random Forests and for performing model evaluation and tuning.

4.OpenCV:
An open-source computer vision library used for image preprocessing and enhancement, such as applying CLAHE (Contrast Limited Adaptive Histogram Equalization) to improve sign visibility.

5. NumPy:

A fundamental library for numerical computations, used for handling and processing data arrays during model training and evaluation.

6. Pandas:

A library for data manipulation and analysis, assisting in organizing and preparing datasets for model training.

7. Matplotlib:

A plotting library used for visualizing data and performance metrics of the machine learning models.

8. Jupyter Notebook

An interactive environment used for writing and executing code, documenting the development process, and visualizing results.

5.2 Methods / Algorithms Used

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a powerful class of deep learning algorithms designed to process structured grid data like images. They consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers, each with a specific function. Convolutional layers apply filters to the input images to extract important features, capturing spatial hierarchies and patterns. Pooling layers then reduce the spatial dimensions of these feature maps, retaining essential information while minimizing computational complexity. This layered approach allows CNNs to efficiently learn and recognize intricate patterns in images.

CNNs are particularly well-suited for image recognition tasks, making them highly effective for detecting and classifying traffic signs under various conditions. Their ability to automatically learn hierarchical features from raw data ensures accurate recognition, even in challenging environments like poor lighting or adverse weather. In the Traffic Sign Board Recognition and Voice Alert System, CNNs play a crucial role in identifying and classifying traffic signs in real-time, contributing to enhanced driver safety and adherence to traffic regulations.

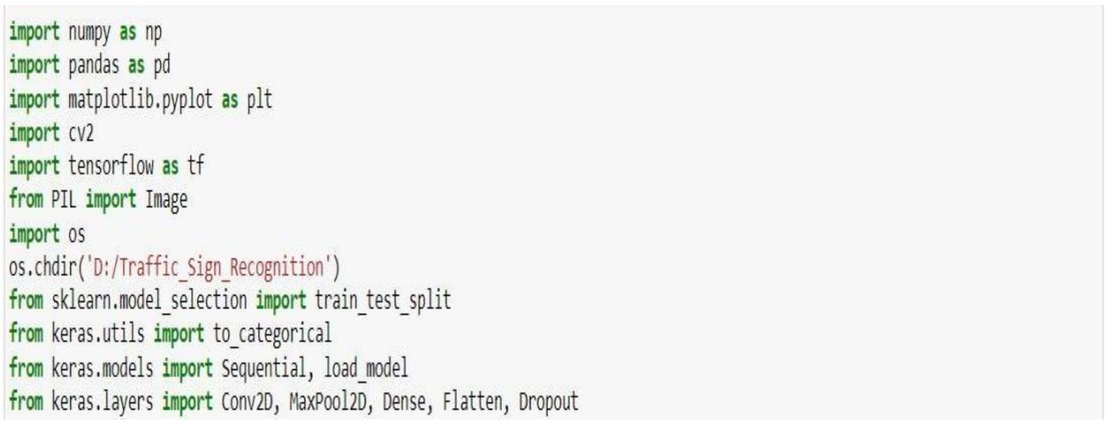
Random Forest Algorithm

The Random Forest algorithm is an ensemble learning method that enhances classification accuracy by combining the outputs of multiple decision trees. Each tree in the forest is trained on a randomly selected subset of the training data and features, which helps to minimize overfitting and reduce variance. By introducing randomness in the data and feature selection, Random Forests create a diverse set of decision trees, leading to more robust and generalized predictions.

In traffic sign recognition, Random Forests play a vital role by improving the system's ability to accurately classify signs, even in complex and varied conditions. The algorithm aggregates the predictions from all individual trees, typically using majority voting, to arrive at the final classification. This collective approach, leveraging the diversity and strengths of multiple decision trees, makes Random Forests particularly effective in handling high-dimensional data and capturing intricate relationships within the dataset.

5.3 Sample Code

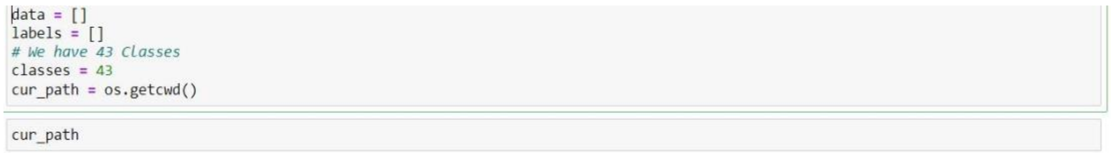
Necessary Imports



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
import os
os.chdir('D:/Traffic_Sign_Recognition')
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
```

Fig 5.3.1 Necessary Imports

Input data

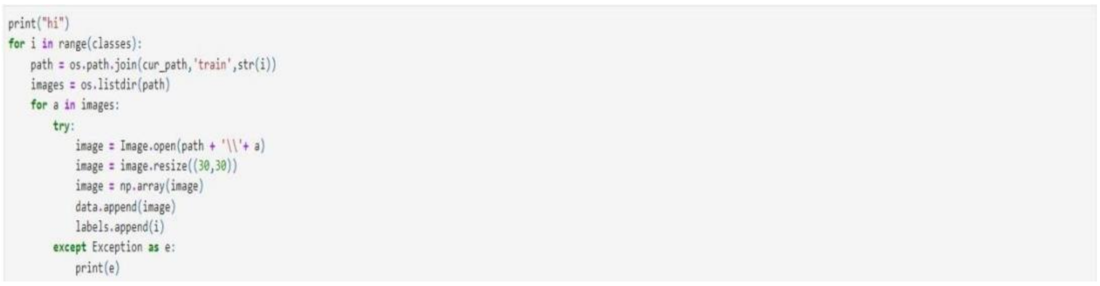


```
data = []
labels = []
# We have 43 Classes
classes = 43
cur_path = os.getcwd()
```

```
cur_path
```

Fig 5.3.2 Input data

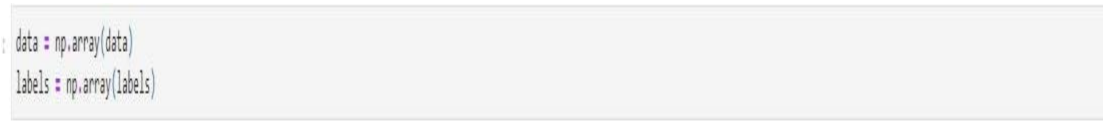
Preprocess the images



```
print("hi")
for i in range(classes):
    path = os.path.join(cur_path,'train',str(i))
    images = os.listdir(path)
    for a in images:
        try:
            image = Image.open(path + '\\'+ a)
            image = image.resize((30,30))
            image = np.array(image)
            data.append(image)
            labels.append(i)
        except Exception as e:
            print(e)
```

Fig 5.3.3 Preprocess the image

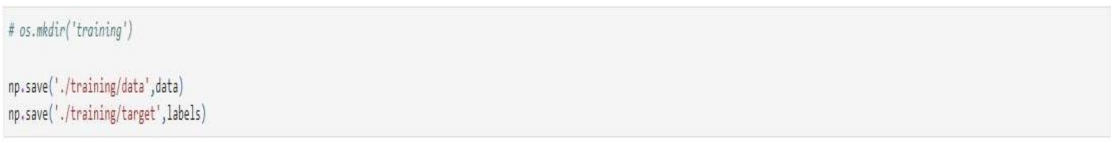
Converting lists into numpy arrays



```
data = np.array(data)
labels = np.array(labels)
```

Fig 5.3.4 Converting lists into numpy arrays

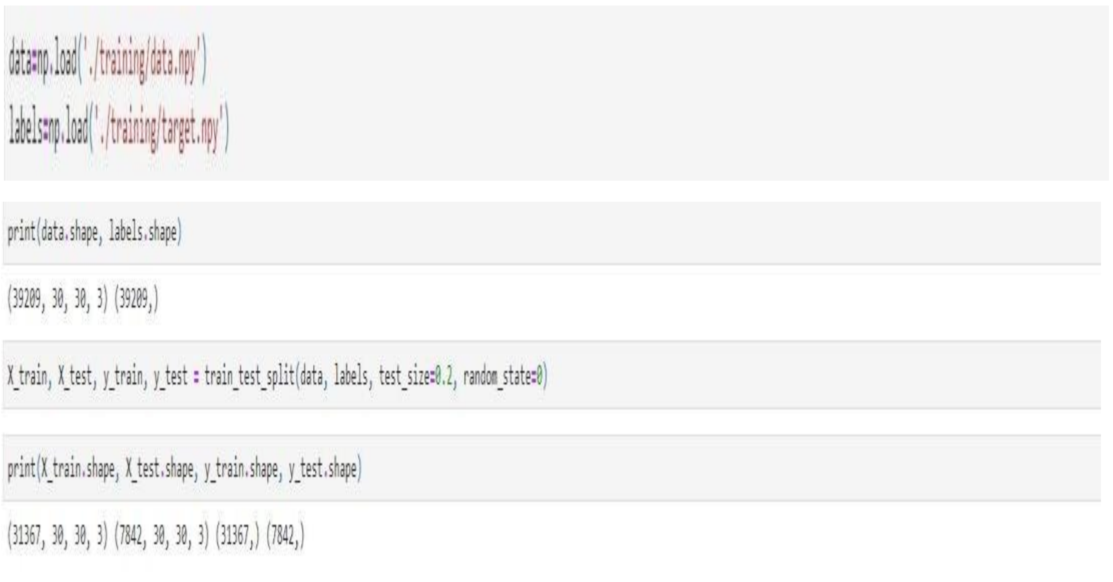
Save Labels & Data for future use



```
# os.mkdir('training')

np.save('./training/data',data)
np.save('./training/target',labels)
```

Fig 5.3.5 Save Labels & Data for future use

Load data & Label



```
data=np.load('./training/data.npy')
labels=np.load('./training/target.npy')
```

```
print(data.shape, labels.shape)
```

```
(39209, 30, 30, 3) (39209,)
```

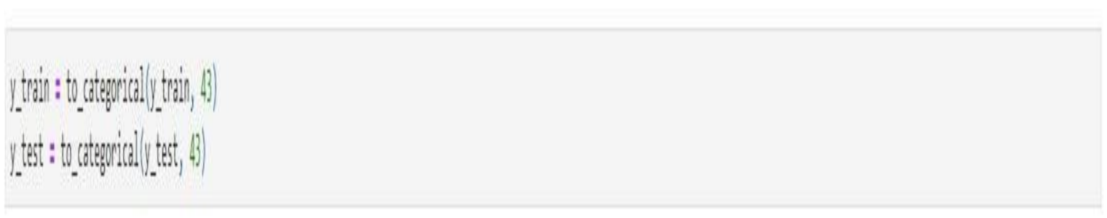
```
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=0)
```

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

Fig 5.3.6 Load data & Labels

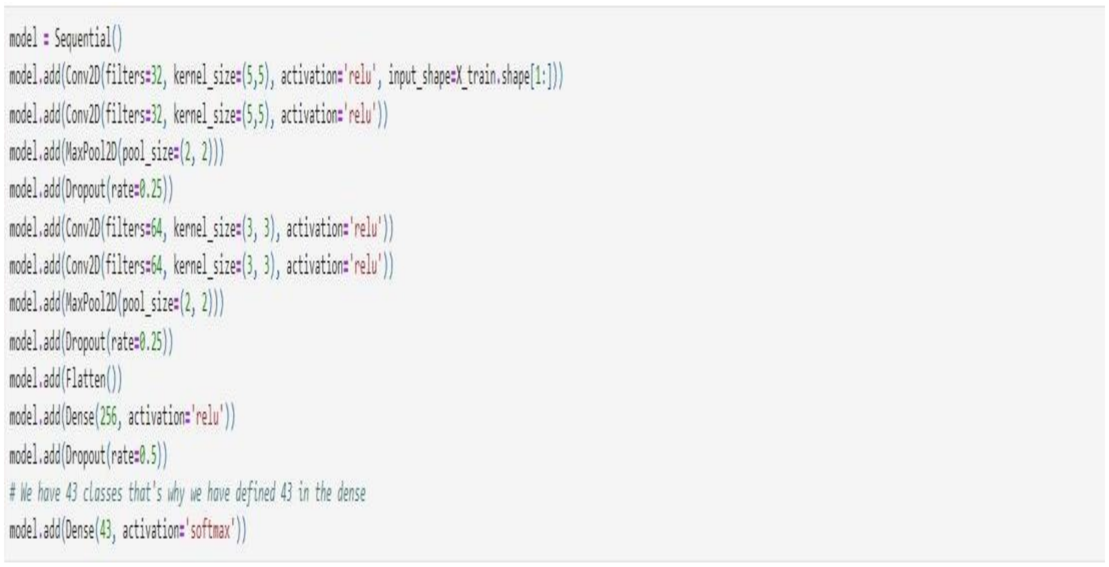
Convert labels to onehot encoding



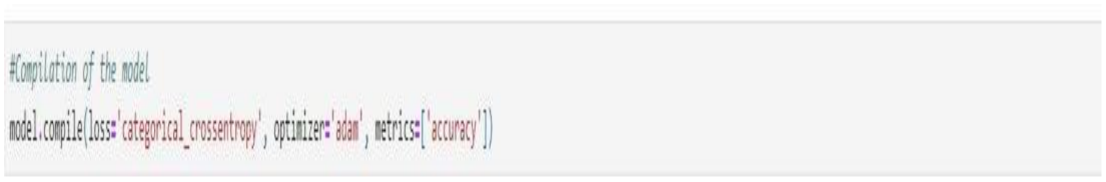
```
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)
```

Fig 5.3.7 Convert labels to onehot encoding

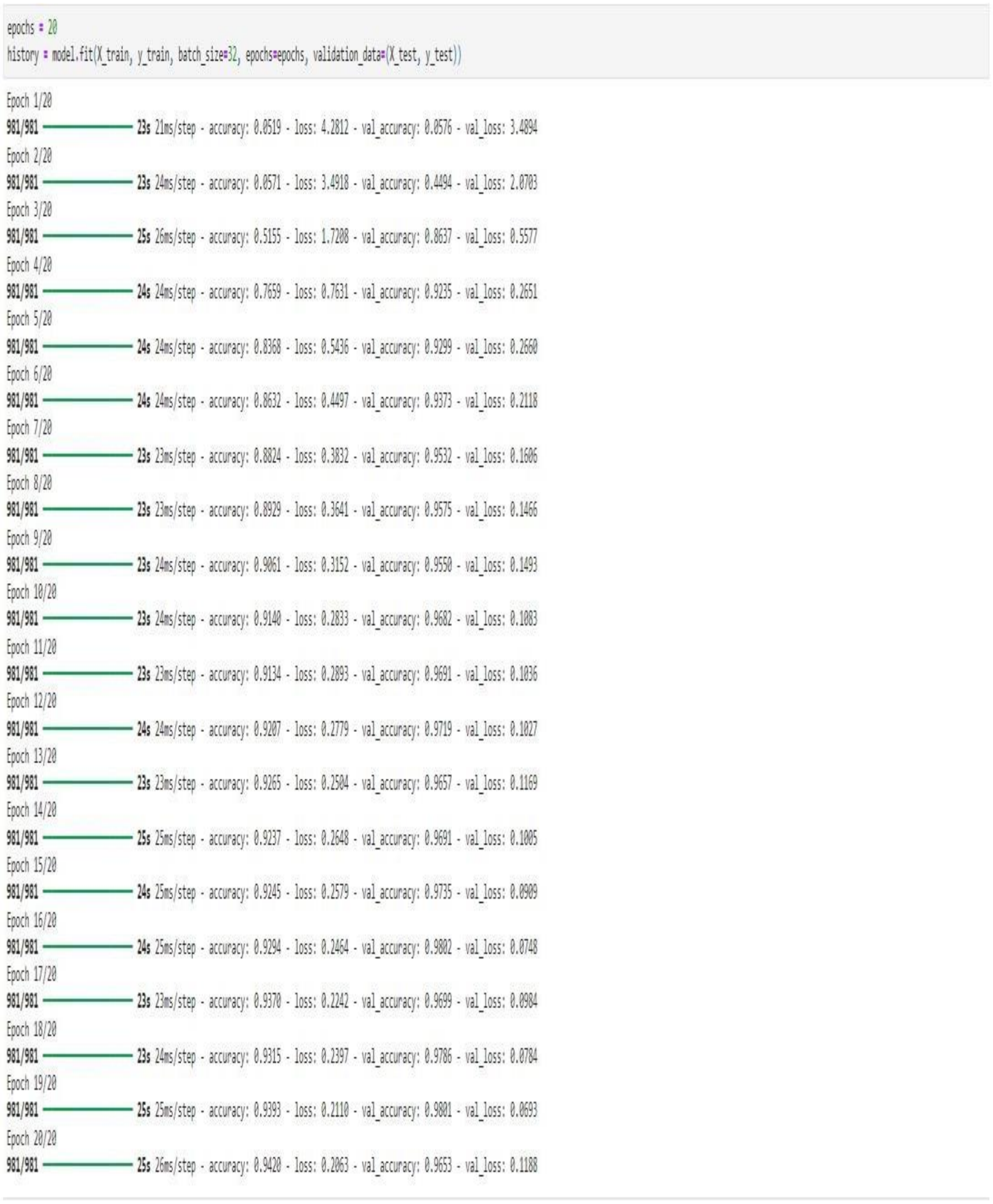
Building the modal



```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
# We have 43 classes that's why we have defined 43 in the dense
model.add(Dense(43, activation='softmax'))
```

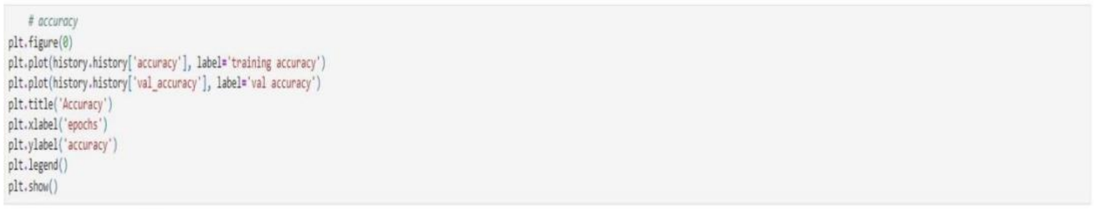


```
#Compilation of the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

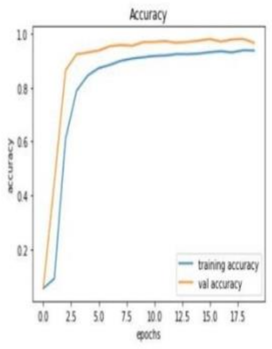


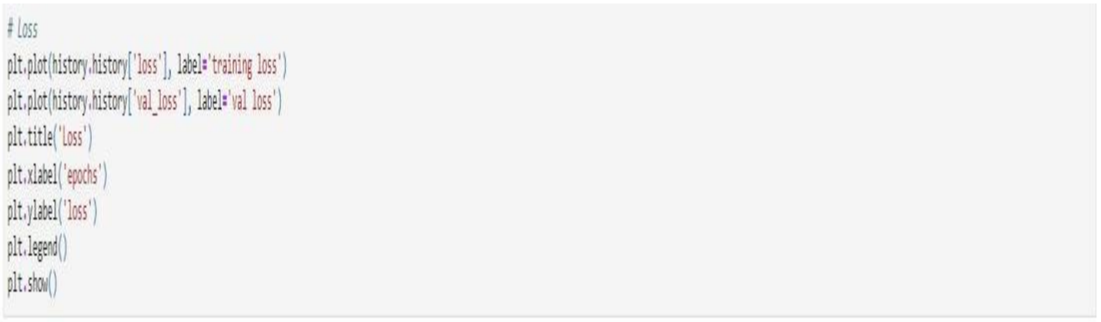
```python
epochs = 20
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
```

```
Epoch 1/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 23s 21ms/step - accuracy: 0.0519 - loss: 4.2812 - val_accuracy: 0.0576 - val_loss: 3.4894
Epoch 2/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 23s 24ms/step - accuracy: 0.0571 - loss: 3.4918 - val_accuracy: 0.4494 - val_loss: 2.0703
Epoch 3/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 25s 26ms/step - accuracy: 0.5155 - loss: 1.7208 - val_accuracy: 0.8637 - val_loss: 0.5577
Epoch 4/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 24s 24ms/step - accuracy: 0.7659 - loss: 0.7631 - val_accuracy: 0.9235 - val_loss: 0.2651
Epoch 5/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 24s 24ms/step - accuracy: 0.8368 - loss: 0.5436 - val_accuracy: 0.9299 - val_loss: 0.2660
Epoch 6/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 24s 24ms/step - accuracy: 0.8632 - loss: 0.4497 - val_accuracy: 0.9373 - val_loss: 0.2118
Epoch 7/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 23s 23ms/step - accuracy: 0.8824 - loss: 0.3832 - val_accuracy: 0.9532 - val_loss: 0.1606
Epoch 8/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 23s 23ms/step - accuracy: 0.8929 - loss: 0.3641 - val_accuracy: 0.9575 - val_loss: 0.1466
Epoch 9/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 23s 24ms/step - accuracy: 0.9061 - loss: 0.3152 - val_accuracy: 0.9550 - val_loss: 0.1493
Epoch 10/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 23s 24ms/step - accuracy: 0.9140 - loss: 0.2833 - val_accuracy: 0.9682 - val_loss: 0.1083
Epoch 11/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 23s 23ms/step - accuracy: 0.9134 - loss: 0.2893 - val_accuracy: 0.9691 - val_loss: 0.1036
Epoch 12/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 24s 24ms/step - accuracy: 0.9207 - loss: 0.2779 - val_accuracy: 0.9719 - val_loss: 0.1027
Epoch 13/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 23s 23ms/step - accuracy: 0.9265 - loss: 0.2504 - val_accuracy: 0.9657 - val_loss: 0.1169
Epoch 14/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 25s 25ms/step - accuracy: 0.9237 - loss: 0.2648 - val_accuracy: 0.9691 - val_loss: 0.1005
Epoch 15/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 24s 25ms/step - accuracy: 0.9245 - loss: 0.2579 - val_accuracy: 0.9735 - val_loss: 0.0909
Epoch 16/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 24s 25ms/step - accuracy: 0.9294 - loss: 0.2464 - val_accuracy: 0.9802 - val_loss: 0.0748
Epoch 17/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 23s 23ms/step - accuracy: 0.9370 - loss: 0.2242 - val_accuracy: 0.9699 - val_loss: 0.0984
Epoch 18/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 23s 24ms/step - accuracy: 0.9315 - loss: 0.2397 - val_accuracy: 0.9786 - val_loss: 0.0784
Epoch 19/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 25s 25ms/step - accuracy: 0.9393 - loss: 0.2110 - val_accuracy: 0.9801 - val_loss: 0.0693
Epoch 20/20
981/981 ━━━━━━━━━━━━━━━━━━━━ 25s 26ms/step - accuracy: 0.9420 - loss: 0.2063 - val_accuracy: 0.9653 - val_loss: 0.1188
```



```
# 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()
```

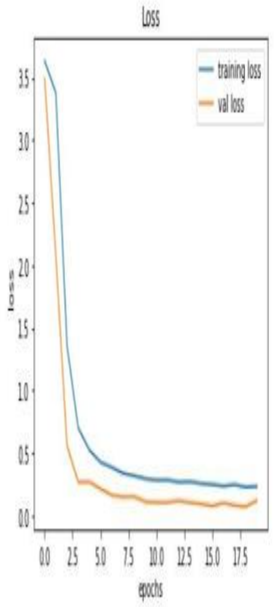
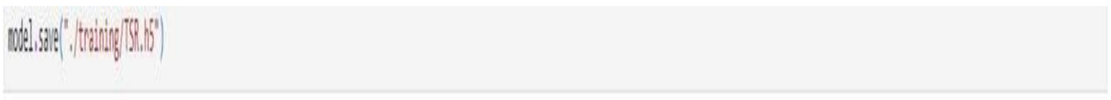


Fig 5.3.8 Building the modal

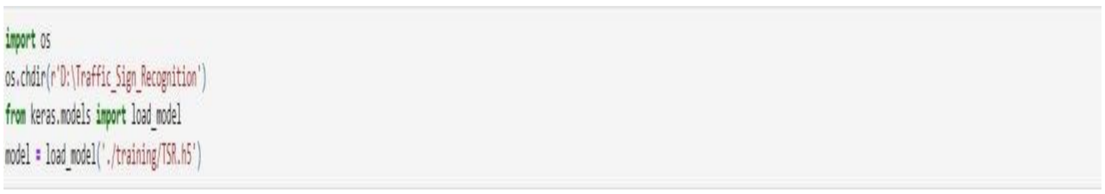
Save the model



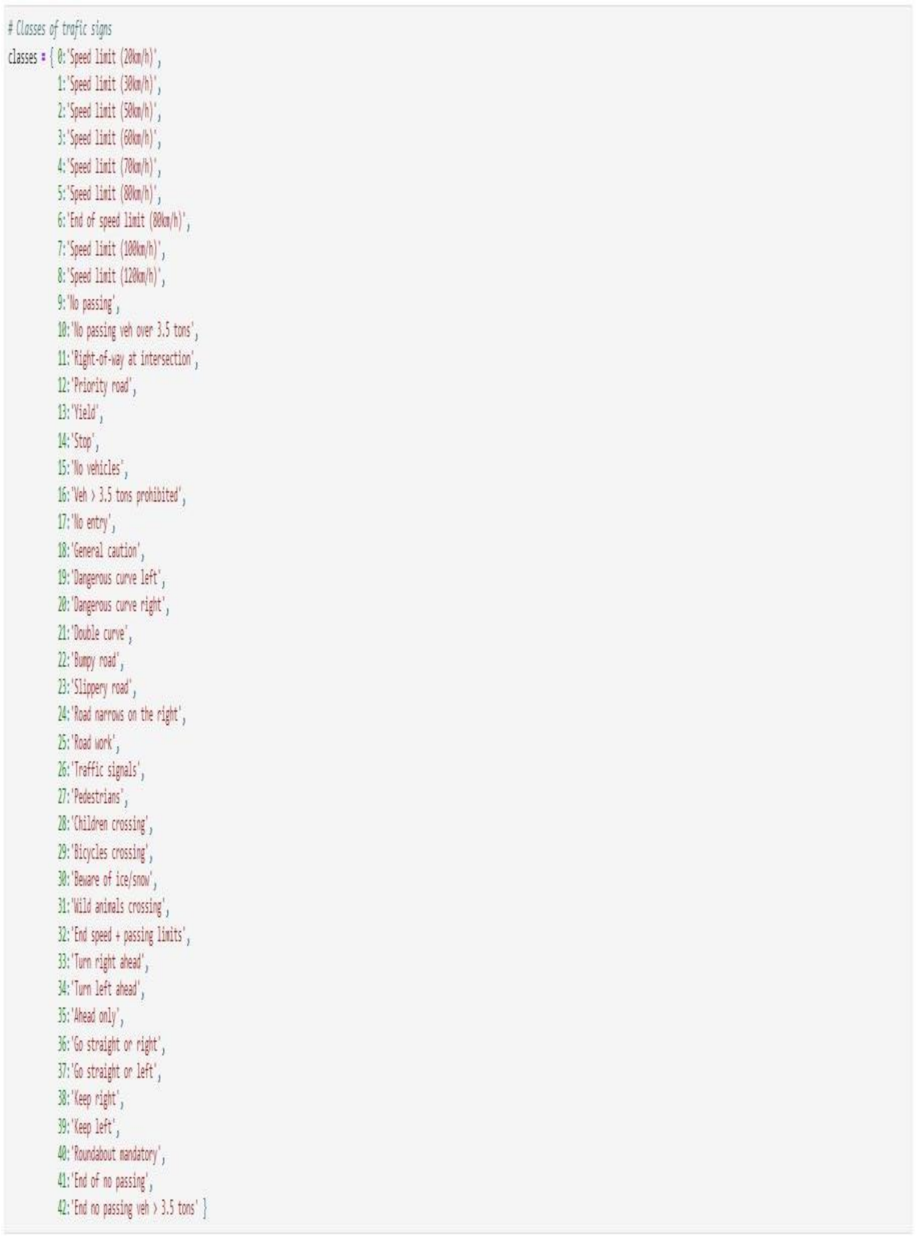
```
model.save("./training/TSR.h5")
```

Fig 5.3.9 Save the model

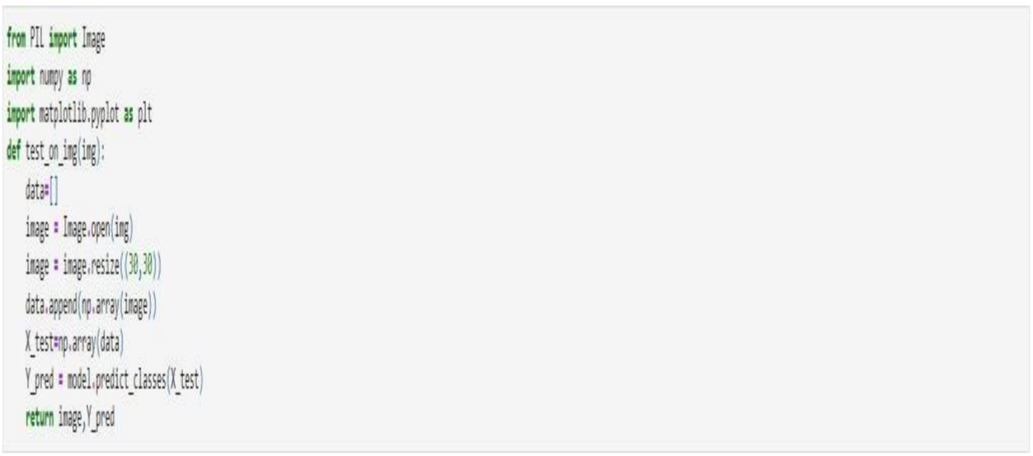
Load the Model



```
import os
os.chdir(r'D:\Traffic_Sign_Recognition')
from keras.models import load_model
model = load_model('./training/TSR.h5')
```



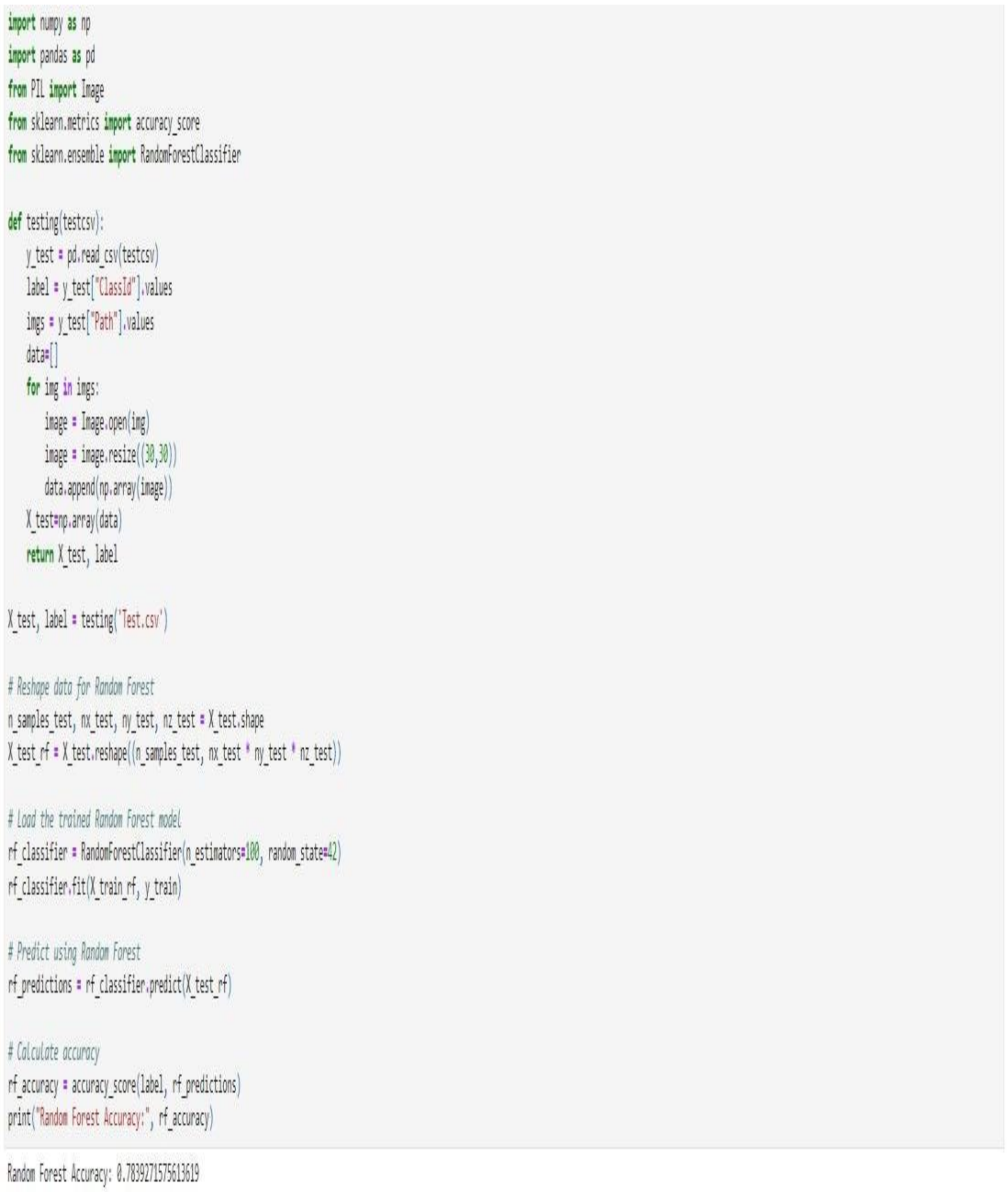
```python
# Classes of trafic signs
classes = { 0:'Speed limit (20km/h)',
            1:'Speed limit (30km/h)',
            2:'Speed limit (50km/h)',
            3:'Speed limit (60km/h)',
            4:'Speed limit (70km/h)',
            5:'Speed limit (80km/h)',
            6:'End of speed limit (80km/h)',
            7:'Speed limit (100km/h)',
            8:'Speed limit (120km/h)',
            9:'No passing',
            10:'No passing veh over 3.5 tons',
            11:'Right-of-way at intersection',
            12:'Priority road',
            13:'Yield',
            14:'Stop',
            15:'No vehicles',
            16:'Veh > 3.5 tons prohibited',
            17:'No entry',
            18:'General caution',
            19:'Dangerous curve left',
            20:'Dangerous curve right',
            21:'Double curve',
            22:'Bumpy road',
            23:'Slippery road',
            24:'Road narrows on the right',
            25:'Road work',
            26:'Traffic signals',
            27:'Pedestrians',
            28:'Children crossing',
            29:'Bicycles crossing',
            30:'Beware of ice/snow',
            31:'Wild animals crossing',
            32:'End speed + passing limits',
            33:'Turn right ahead',
            34:'Turn left ahead',
            35:'Ahead only',
            36:'Go straight or right',
            37:'Go straight or left',
            38:'Keep right',
            39:'Keep left',
            40:'Roundabout mandatory',
            41:'End of no passing',
            42:'End no passing veh > 3.5 tons' }
```



```python
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
def test_on_img(img):
    data=[]
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))
    X_test=np.array(data)
    Y_pred = model.predict_classes(X_test)
    return image,Y_pred
```

Fig 5.3.10 Load the Model

Random Forest algorithm

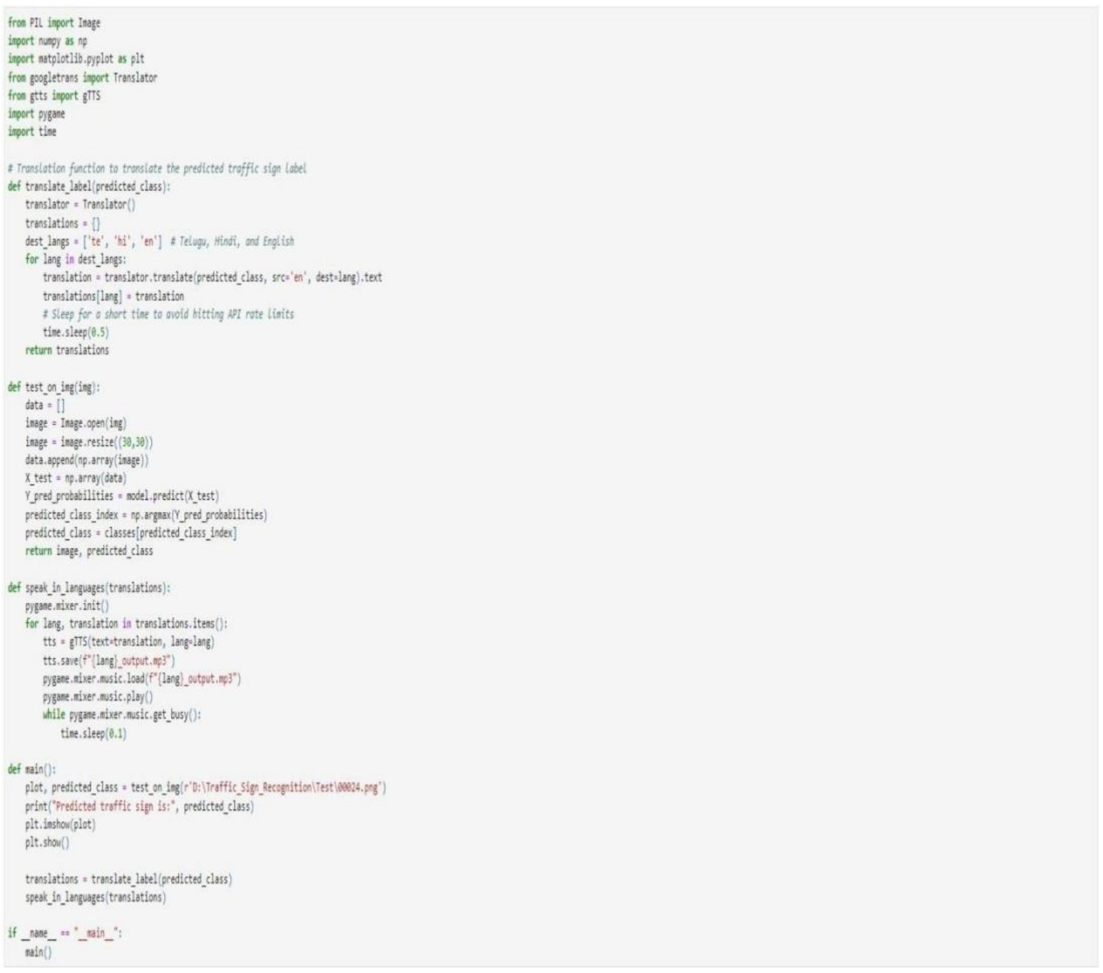


```
import numpy as np
import pandas as pd
from PIL import Image
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier

def testing(testcsv):
    y_test = pd.read_csv(testcsv)
    label = y_test["ClassId"].values
    imgs = y_test["Path"].values
    data=[]
    for img in imgs:
        image = Image.open(img)
        image = image.resize((30,30))
        data.append(np.array(image))
    X_test=np.array(data)
    return X_test, label

X_test, label = testing('Test.csv')

# Reshape data for Random Forest
n_samples_test, nx_test, ny_test, nz_test = X_test.shape
X_test_rf = X_test.reshape((n_samples_test, nx_test * ny_test * nz_test))

# Load the trained Random Forest model
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_rf, y_train)

# Predict using Random Forest
rf_predictions = rf_classifier.predict(X_test_rf)

# Calculate accuracy
rf_accuracy = accuracy_score(label, rf_predictions)
print("Random Forest Accuracy:", rf_accuracy)
```

```
Random Forest Accuracy: 0.7839271575613619
```

Fig 5.3.11 Random Forest algorithm

Convolutional neural network



```python
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from googletrans import Translator
from gtts import gTTS
import pygame
import time

# Translation function to translate the predicted traffic sign label
def translate_label(predicted_class):
    translator = Translator()
    translations = {}
    dest_langs = ['te', 'hi', 'en']  # Telugu, Hindi, and English
    for lang in dest_langs:
        translation = translator.translate(predicted_class, src='en', dest=lang).text
        translations[lang] = translation
        # Sleep for a short time to avoid hitting API rate limits
        time.sleep(0.5)
    return translations

def test_on_img(img):
    data = []
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))
    X_test = np.array(data)
    Y_pred_probabilities = model.predict(X_test)
    predicted_class_index = np.argmax(Y_pred_probabilities)
    predicted_class = classes[predicted_class_index]
    return image, predicted_class

def speak_in_languages(translations):
    pygame.mixer.init()
    for lang, translation in translations.items():
        tts = gTTS(text=translation, lang=lang)
        tts.save(f"{lang}_output.mp3")
        pygame.mixer.music.load(f"{lang}_output.mp3")
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy():
            time.sleep(0.1)

def main():
    plot, predicted_class = test_on_img(r'D:\Traffic_Sign_Recognition\Test\00024.png')
    print("Predicted traffic sign is:", predicted_class)
    plt.imshow(plot)
    plt.show()

    translations = translate_label(predicted_class)
    speak_in_languages(translations)

if __name__ == "__main__":
    main()
```

```
pygame 2.5.2 (SDL 2.28.3, Python 3.9.12)
Hello from the pygame community. https://www.pygame.org/contribute.html
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
Predicted traffic sign is: Speed limit (30km/h)
```

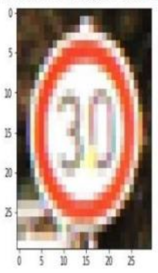
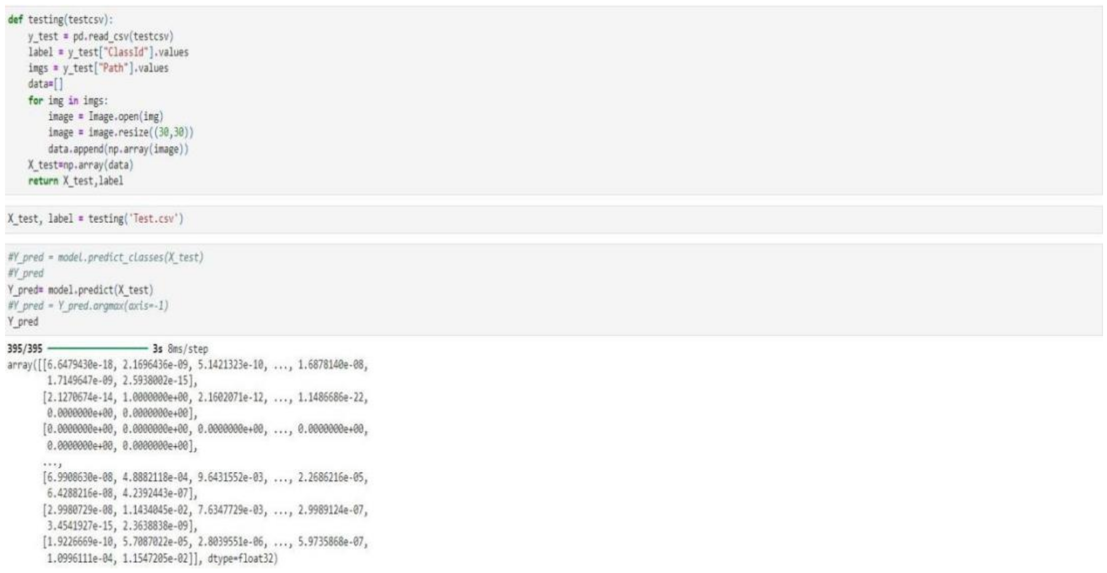


Fig 5.3.12 Convolutional neural network

6. TESTING

6.1 Types of Testing

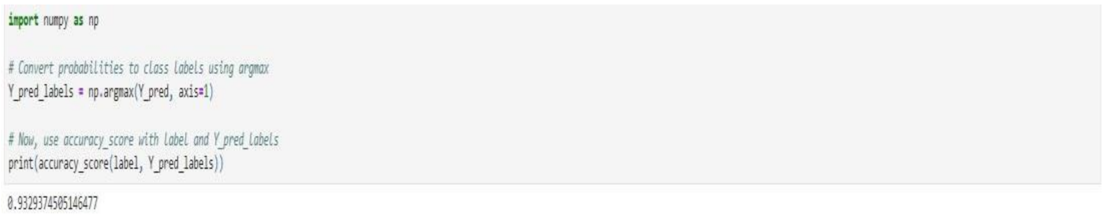


```
def testing(testcsv):
    y_test = pd.read_csv(testcsv)
    label = y_test["ClassId"].values
    imgs = y_test["Path"].values
    data=[]
    for img in imgs:
        image = Image.open(img)
        image = image.resize((30,30))
        data.append(np.array(image))
    X_test=np.array(data)
    return X_test,label
```

```
X_test, label = testing('Test.csv')
```

```
#Y_pred = model.predict_classes(X_test)
#Y_pred
Y_pred= model.predict(X_test)
#Y_pred = Y_pred.argmax(axis=-1)
Y_pred
```

```
395/395 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step
array([[6.6479430e-18, 2.1696436e-09, 5.1421323e-10, ..., 1.6878140e-08,
        1.7149647e-09, 2.5938002e-15],
       [2.1270674e-14, 1.0000000e+00, 2.1602071e-12, ..., 1.1486686e-22,
        0.0000000e+00, 0.0000000e+00],
       [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00],
       ...,
       [6.9908630e-08, 4.8882118e-04, 9.6431552e-03, ..., 2.2686216e-05,
        6.4288216e-08, 4.2392443e-07],
       [2.9980729e-08, 1.1434045e-02, 7.6347729e-03, ..., 2.9989124e-07,
        3.4541927e-15, 2.3638838e-09],
       [1.9226669e-10, 5.7087022e-05, 2.8039551e-06, ..., 5.9735868e-07,
        1.0996111e-04, 1.1547205e-02]], dtype=float32)
```



```
import numpy as np

# Convert probabilities to class labels using argmax
Y_pred_labels = np.argmax(Y_pred, axis=1)

# Now, use accuracy_score with label and Y_pred_labels
print(accuracy_score(label, Y_pred_labels))
```

```
0.9329374505146477
```

7. CONCLUSION

We have built a model which recognize traffic signs and provides voice alert to driver by using machine learning algorithm as CNN and Random Forest algorithm and CNN gives good accuracy. This system will lead to safe driving without breaking traffic laws and rules and will reduce number of road accidents caused by ignoring or unawareness of traffic signs while driving on road. We proposed a smart driver alert system which detects and recognizes traffic sign board from video stream input and gives voice message to the driver. By using this technology we can reduce the road accidents as well as regulate traffic safely. A system that is able to detect and classify traffic signs in different environments.

8. FUTURE SCOPE

1. Integration with Autonomous Vehicles

Incorporating the system into autonomous vehicles will enhance their ability to recognize and respond to traffic signs, ensuring safer navigation and adherence to traffic regulations.

2. Real-time Traffic Sign Updates

The system can be upgraded to recognize and respond to temporary or dynamic signs by connecting to live traffic data, improving its relevance and accuracy in real-world scenarios.

3. Enhanced Environmental Adaptation

Future developments can focus on improving the system's performance in challenging conditions like poor lighting, fog, or rain, making it more reliable across diverse environments.

Multilingual Expansion

Expanding the system's language support to include more languages will make it accessible to a broader audience, particularly in regions with diverse linguistic needs, enhancing its usability for non-native speakers and tourists.

9. BIBLIOGRAPHY

[1] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark," *2013 International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, 2013, pp. 1-8.

[2] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," *The 2011 International Joint Conference on Neural Networks (IJCNN)*, San Jose, CA, 2011, pp. 1453-1460.

[3] S. Prabhakar and A. K. Maurya, "Traffic Sign Detection and Recognition using Machine Learning Techniques," *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, Amity University Dubai, UAE, 2019, pp. 615-620.

[4] Y. Tian, L. Liu, C. Pan, and D. Cao, "Real-Time Traffic Sign Recognition Based on YOLO v3," *2020 IEEE Intelligent Vehicles Symposium (IVS)*, Las Vegas, NV, 2020, pp. 1250-1255.

[5] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view Traffic Sign Detection, Recognition, and 3D Localisation," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL, 2014, pp. 1054-1060.

[6] A. Mathias, R. Timofte, R. Benenson, and L. Van Gool, "Traffic Sign Recognition - How far are we from the solution?" *2013 International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, 2013, pp. 1-8.

[7] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 448-456.

[8] "UCI Machine Learning Repository: German Traffic Sign Benchmark Dataset," *University of California, Irvine*, 2015. Available: http://benchmark.ini.rub.de/?section=gtsrb&subsection=news.

[9] S. Huang, H. Yin, and J. Zhang, "Research on Traffic Sign Detection and Recognition Based on Deep Learning," *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, Dalian, China, 2020, pp. 1-5.

[10] W. Shi, F. Yang, Z. Guo, and H. Gu, "A Real-Time Traffic Sign Recognition Method Based on Deep Learning," *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2018, pp. 1418-1422.

APPENDIX-A :

UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts ofa software system. It captures decisions and understanding about systems that must be constructed. It is used to understand, design, browse, configure, maintain, and control information about such systems. It is intended for use with all development methods, lifecycle stages, application domains, and media. The modeling language is intended to unify past experience about modeling techniques and to incorporate current softwarebest practices into a standard approach. UML includes semantic concepts, notation, and guidelines. It has static, dynamic, environmental, and organizational parts. It is intended to be supported by interactive visual modeling tools that have code generator sand report writers. The UML specification does not define a standard process but is intended to be useful with an iterative development process. Itis intended to support most existing object-oriented development processes.

The UML captures information about the static structure and dynamic behavior ofa system. A system is modeled as a collection of discrete objects that interact to perform work that ultimately benefits an outside user. The static structure defines the kinds of objects important to a system and to its implementation, as well as the relationships among the objects. The dynamic behavior defines the history of objects over time and the communications among objects to accomplish goals.

Modeling a system from several separate but related viewpoints permits it to be understood for different purposes.

The UML also contains organizational constructs for arranging models into packages that permit software teams to partition large systems into workable pieces, to understand and control dependencies among the packages, and to manage the versioning of model units in a complex development environment.

It contains constructs for representing implementation decisions and for organizing run-time elements into components.

UML is not a programming language. Tools can provide code generators from UML into a variety of programming languages, as well as construct reverse engineered models from existing programs. The UML is not a highly formal language intended for the or improving. There are a number of such languages, but they are not easy to understand or to use for most purposes. The UML is a general- purpose modeling language. For specialized domains, such as GUI layout, VLSI circuit design, or rule-based artificial intelligence, a more specialized tool with a special language might be appropriate. UML is a discrete modeling language. It is not intended to model continuous systems such as those found in engineering and physics. UML is intended to be a universal general-purpose modeling language for discrete systems such as those made of software, firmware, or digital logic.