A

MINI PROJECT REPORT

ON

TRAFFIC SIGNS RECOGNITION

Submitted in partial fulfillment of the requirements

For the award of Degree of

BACHELOR OF ENGINEERING

IN

CSE (AI ML)

Submitted By

CH.SRI VARSHA 245321748075

E.SATHVIKA 245321748079

I.SAI MAYANK 245321748084

Under the guidance

Of

P.NAGESWARA RAO

ASSISTANT PROFESSOR

Department of CSE(AIML)

NEIL GOGTE INSTITUTE OF TECHNOLOGY

Kachavanisingaram Village, Hyderabad, Telangana 500058.

FEBRUARY 2024

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

CERTIFICATE

This is to certify that the Mini project work entitled "TRAFFIC SIGNS RECOGNITION" is bonafide work carried out by CH.SRI VARSHA (245321748075),E.SATHVIKA (245321748079) and I.SAI MAYANK (245321748084) of III-year V semester Bachelor of Engineering in CSE(AIML) by Osmania University, Hyderabad during the academic year 2023-2024 is a record of bonafide work carried out by them. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree

Internal Guide

Mr. P.NAGESWARA RAO

Assistant Professor

Head of Department

Dr. T. PREM CHANDER

Associate Professor

External

NEIL GOGTE INSTITUTE OF TECHNOLOGY
A Unit of Keshav Memorial Technical Education (KMTES)
Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

DECLARATION

We hereby declare that the Mini Project Report entitled, "**TRAFFIC SIGNS RECOGNITION**" submitted for the B.E degree is entirely my work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

Date:

CH.SRI VARSHA 245321748075

E. SATHVIKA 245321748079

I.SAI MAYANK 245321748084

ACKNOWLEDGEMENT

We are happy to express our deep sense of gratitude to the principal of the college **Dr. R. Shyam Sunder, Professor**, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank**, Dr. T.Prem Chander, Head of the Department,** CSE(AIML), Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We would also like to thank our internal guide **Mr. P.Nageswara Rao, Assistant Professor** for his technical guidance & constant encouragement.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering for their timely suggestions, healthy criticism and motivation during this work.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

ABSTRACT

Traffic signs recognition project is built using machine learning which is used to detect and recognize different traffic signs given in the dataset. Traffic Sign Recognition (TSR) is used to regulate traffic signs, warn a driver, and command or prohibit certain actions. A fast real-time and robust automatic traffic sign detection & recognition can support and significantly increase driving safety and comfort. The aim of this project is to create a program that will identify a stop sign in various backgrounds & lighting conditions from static digital images using deep learning concepts. This processing could then output information to an autonomous vehicle, heads-up display, or other driver assistance device in the future. Automatic recognition of traffic signs is also important for automated intelligent driving vehicles such as self-driving cars in which the passenger can fully depend on the car for traveling. So, for achieving accuracy in this technology. The vehicles should be able to interpret traffic signs and make decisions accordingly.

Road traffic accidents are primarily caused by driver error. Safer roads infrastructure and facilities like traffic signs and signals are built to aid drivers on the road. But several factors affect the awareness of drivers to traffic signs including visual complexity, environmental condition, and poor driver education. This study implements a traffic sign detection and recognition system with voice alert using Python. It aims to establish the proper trade-off between accuracy and speed in the design of the system. Four pre-processing and object detection methods in different color spaces are evaluated for efficient, accurate, and fast segmentation of the region of interest. In the recognition phase, ten classification algorithms are implemented and evaluated to determine which will provide the best performance in both accuracy and processing speed for traffic sign recognition. This study has determined that Shadow and Highlight Invariant Method for the pre-processing and color segmentation stage provided the best trade-off between detection success rate (77.05%) and processing speed (31.2ms).

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER – 1
INTRODUCTION

1.1 PROBLEM STATEMENT

Traffic sign recognition is a critical aspect of modern transportation systems, aimed at enhancing road safety and efficiency. However, manual recognition by human drivers is prone to errors, especially in challenging conditions such as poor visibility or driver fatigue. This leads to potential risks on the road due to misinterpretation or oversight of crucial traffic signs.

Existing automated systems face challenges in achieving high accuracy and robustness across diverse environmental conditions and sign variations. Additionally, the integration of these systems into vehicles requires addressing compatibility issues and ensuring seamless functionality. Regulatory standards and guidelines must also be considered to ensure the reliability and effectiveness of automated traffic sign recognition solutions. Furthermore, there is a need to continually improve these systems through advancements in machine learning and sensor technologies. Collaborative efforts among industry stakeholders, researchers, and policymakers are essential to address these challenges and develop more robust and reliable traffic sign recognition solutions. Overall, the problem statement revolves around the imperative to overcome the limitations of manual recognition and enhance the performance and adaptability of automated systems to improve road safety and efficiency.

1.2 MOTIVATION

Developing a robust traffic sign recognition system is not merely a technical endeavor but also holds immense potential for addressing real-world challenges. The accurate identification and classification of traffic signs have far-reaching implications, significantly improving road safety and traffic efficiency while reducing the likelihood of accidents. Furthermore, this solution transcends the realm of traffic sign recognition, serving as a stepping stone for tackling a myriad

of fine-grained classification tasks. Just as the bird species identification system aids in ecological studies, a proficient traffic sign recognition system can revolutionize research in traffic management and urban planning domains. Through this initiative, we aim to not only enhance road safety but also foster collaboration and innovation across various scientific disciplines. By providing a versatile tool for researchers, we envision contributing to a deeper understanding of fine-grained classification challenges and their practical applications in diverse fields. This project represents an opportunity to drive meaningful advancements in both transportation systems and scientific research, ultimately shaping a safer and more efficient future for all.

1.3 SCOPE

This project centers on the development of a robust traffic sign recognition system leveraging Convolutional Neural Networks (CNNs) to accurately classify diverse traffic signs encountered on roads, encompassing regulatory, warning, and informational signs. The primary goal is to achieve precise and real-time recognition of traffic signs under various usage scenarios, enhancing road safety and efficiency. Inspired by successful models like the bird species identification system, this traffic sign recognition system follows a structured approach comprising three key phases: pre-processing, model training, and classification. During the pre-processing phase, the "ImageDataGenerator" module from the "Keras" package will be employed to resize images and partition them into training and testing datasets, ensuring optimal data preparation. Model training entails training the CNN model on the designated datasets, employing multiple epochs and specified steps per epoch to refine the model's predictive capabilities. Finally, the classification phase involves deploying the trained model to accurately classify distinct types of traffic signs, effectively addressing the fundamental challenge of traffic sign recognition. This project aims to advance the field of intelligent transportation systems by developing a sophisticated yet accessible solution for automated traffic sign recognition, contributing to safer and more efficient roadways.

1.4 OUTLINE

This project aims to develop a traffic sign recognition system using the VGG16 CNN model to enhance road safety and efficiency in transportation systems. The project involves collecting and preprocessing a comprehensive dataset of annotated traffic sign images, which is divided into training, testing, and validation datasets. The VGG16 model is trained on the training dataset to learn and extract features from traffic sign images, and its performance is validated using the validation dataset. A user-friendly web application is created using Tkinter, capable of accepting input images up to 200MB in size, providing flexibility to users. The system's accuracy and performance are evaluated using the testing dataset, assessing its ability to classify various types of traffic signs under different environmental conditions

CHAPTER – 2

LITERATURE SURVEY

EXISTING SYSTEM:

Traffic sign recognition (TSR) plays a pivotal role within the framework of Advanced Driver-Assistance Systems (ADAS), which are designed to enhance vehicle safety and driver convenience. By leveraging automated technologies such as sensors and cameras, ADAS systems are capable of detecting and interpreting various types of traffic signs present on roadways, including speed limits, warnings, and directional instructions. TSR functions as a critical component of this system, enabling vehicles to identify and respond to traffic signs in real-time.

The development of TSR technology involves the utilization of sophisticated image processing techniques, which can be broadly categorized into color-based, shape-based, and learning-based methods. While color-based methods rely on the recognition of specific color patterns associated with different types of traffic signs, shape-based methods focus on identifying geometric shapes and contours. Learning-based methods, on the other hand, utilize machine learning algorithms to classify signs based on training data.

Despite significant advancements in TSR technology, challenges persist in achieving a balance between accuracy, processing efficiency, and complexity. For instance, while some approaches, such as those proposed by Yuan et al., demonstrate high accuracy rates, they often require extensive computational resources and exhibit prolonged processing times. In contrast, fusion network approaches aim to extract features from both traffic signs and their surrounding environments to improve accuracy, but their implementation may introduce additional complexity.

Disadvantages of existing system:

> Increased algorithms complexity

> Heavy system hardware requirement

> Different pre-processing for training data is necessary

PROPOSED SYSTEM:

The framework we proposed is categorized into three stages: Detection and feature extraction and recognition. The detection stage is just used to find a road sign. At the point when a vehicle is travelling at a specific speed, the camera catches the road sign in nature, and our calculation verifies whether a sign is available in that outline or not available in that perimeter. Distinguishing the traffic sign depends on shape and color. In the feature extraction stage, the proposed calculation characterizes the distinguished road sign. This is accomplished with the assistance of "Convolutional Neural Network" algorithm which classifies the image into sub classes. Traffic sign recognition and detection is an important part of any autonomous vehicle. However, the real challenge lies in the detection and recognition of these traffic sign from the natural image in real time and with accuracy. This paper gives an overview of the traffic road sign detection and recognition system, we developed and implemented using an artificial neural network which is trained using real-life datasets. This paper presents the usage of convolution neural network along with dataset as an implementation of our project to attain real-time result with accuracy. The system developed based on this methodology can be implemented in public transports, personal cars, and other vehicles in order to keep drivers alert and reduce human errors that lead to accidents. The project has a wide implementation of self-driving vehicles.

ADVANTAGES OF PROPOSED SYSTEM

>Reducing the number of accidents caused by driver distraction.

>Reduce the seriousness of such accidents.

>Improve the drivers' safety on the road.

CHAPTER - 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Overall Description:

The Software Requirements Specification (SRS) serves as a foundational document for the development of the course management system, offering a comprehensive overview of its purpose, functionalities, and operational constraints. Within this document, stakeholders and developers alike gain insight into the system's core features and user interactions. By detailing the system's objectives and capabilities, the SRS fosters clear communication and alignment among all project stakeholders, ensuring a shared understanding of project goals and requirements. Additionally, it provides a roadmap for system development, guiding stakeholders and developers through the design, implementation, and testing phases.

Furthermore, the SRS outlines how the course management system will respond to various stimuli, including user inputs and system errors, to ensure robust performance under diverse conditions. Through meticulous documentation of system interfaces and interactions, the SRS elucidates the relationships between different system components and user roles. This clarity empowers developers to design and implement a system that effectively streamlines educational processes, enhances user experience, and optimizes administrative tasks within specified constraints.

As a critical reference point throughout the development lifecycle, the SRS enables stakeholders to provide feedback for further refinement and ensures that the final product aligns with their expectations. Developers utilize the SRS as a blueprint for system design, implementation, and testing, leveraging its detailed specifications to guide their work. Through collaborative efforts and adherence to the SRS, the course management system is poised to contribute to improved educational outcomes and organizational efficiency, ultimately serving the needs of both educators and learners effectively.

3.2. Operating Environment:

Software Requirements:

Operating System	:	Windows 7 (Min)
Front End	:	GUI Tkinter (python)
Back End	:	Python
Database	:	Microsoft Excel

Hardware Requirements:

Processor	:	Intel Pentium® Dual Core Processor (Min)
Speed	:	2.9 GHz (Min)
RAM	:	2 GB (Min)
Hard Disk	:	2 GB (Min)

Software Requirements:

The software is designed to run on Windows 7 as the minimum supported operating system, ensuring compatibility with a wide range of computer systems. The graphical user interface (GUI) is developed using Tkinter, a standard GUI toolkit for Python, providing users with an intuitive and interactive experience. Python serves as the backend programming language, offering robust functionality and flexibility for implementing various features and functionalities of the software. Additionally, Microsoft Excel is utilized as the database platform, leveraging its familiarity and versatility for storing and managing data efficiently. By adhering to these software requirements, users can expect a seamless and reliable experience while interacting with the software, ensuring compatibility across different hardware configurations and operating environments.

Hardware Requirements:

The software demands a moderate hardware setup to ensure smooth operation and optimal performance. A minimum of an Intel Pentium® Dual Core Processor clocked at 2.9 GHz is required to handle computational tasks efficiently. This processor configuration offers a balance between performance and cost-effectiveness, suitable for running the software without significant lag or slowdowns. Additionally, a minimum of 2 GB of RAM is necessary to support the execution of the software and handle data processing tasks effectively. This RAM capacity ensures sufficient memory allocation for the software to operate smoothly, even when handling multiple concurrent operations. Moreover, a minimum of 2 GB of hard disk space is required to accommodate the installation files and storage of temporary data generated during software operation.

3.3. Functional Requirements:

User Functionality:

Image Upload Capability:

Users will have the functionality to upload images related to traffic signs they wish to identify. This feature enables user interaction by allowing them to provide input to the system.

File Size Limit:

Users will be able to insert images up to 200MB in size, facilitating flexibility and accommodating various image sizes without imposing unnecessary restrictions on users.

Traffic Sign Information:

The system will provide users with the ability to view the meaning or information associated with a particular traffic sign.

3.4. Non-Functional Requirements:

3.4.1. Performance Requirements:

Response Time:

The system's average response time shall not exceed 5 seconds, ensuring prompt feedback and a seamless user experience. This requirement emphasizes the importance of efficient processing and quick turnaround times to meet user expectations.

Recovery Time:

In the event of a system failure, the redundant system will resume operations within 30 seconds, with an average repair time of less than 45 minutes. This requirement emphasizes system reliability and resilience, ensuring minimal downtime and prompt recovery in case of disruptions.

Start-Up/Shutdown Time:

The system shall be operational within 1 minute of starting up, optimizing user productivity by minimizing wait times during system initialization. This requirement ensures that users can quickly access and utilize the system without significant delays.

3.4.2. Safety Requirements:

There are no specific safety requirements identified for this project, as it primarily focuses on providing user functionality and ensuring system performance. However, it's essential to maintain data privacy and security measures to protect user information and ensure safe usage of the application.

3.4.3. Security Requirements:

User authentication will be implemented to ensure that only authorized individuals can access the system, safeguarding sensitive data and functionalities.

Data encryption will be employed to protect user information and uploaded images from unauthorized access or interception during transmission.

Secure file handling mechanisms will be implemented to prevent malicious files from being uploaded and to ensure the integrity of the system.

3.4.4. Software Quality Attributes:

Accuracy:

The software should accurately recognize traffic signs from images with a high degree of precision. The accuracy of the recognition system is crucial for ensuring road safety and preventing accidents.

Availability:

The website will be available to all its users round the clock i.e., they can access the website at any time.

Security:

The software should be secure and protect sensitive data, such as images or user information, from unauthorized access or manipulation. It should implement authentication, encryption, and other security measures to ensure data integrity and confidentiality.

Maintainability:

The software should be easy to maintain and update, with well-organized code that is easy to understand and modify. It should follow coding best practices and be thoroughly documented to facilitate future enhancements or bug fixes.

Usability:

The software should be user-friendly and easy to use, with an intuitive interface that allows users to interact with the system effectively. The UI should be designed to minimize user errors and provide clear feedback.

Scalability:

The software should be scalable to accommodate future growth and increasing demands for traffic sign recognition. It should be able to handle a growing number of users, images, and traffic sign variations without a significant decrease in performance.

CHAPTER-4

SYSTEM DESIGN

Use case Diagram:

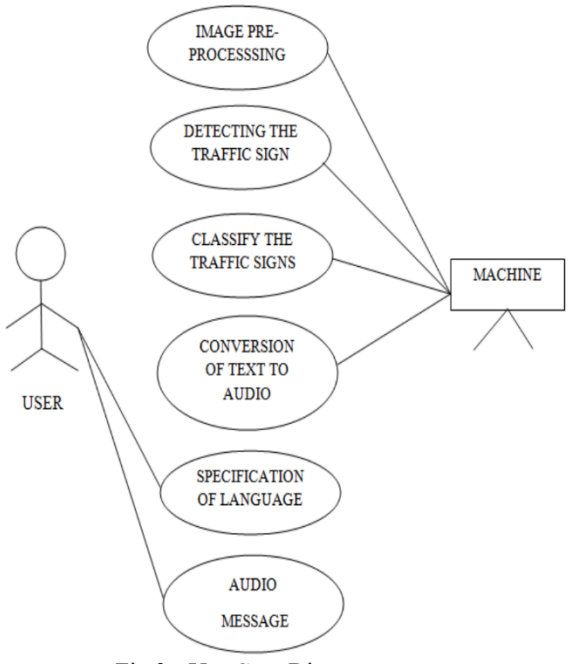

Fig 4.1: Use case diagram for User

Class Diagram:

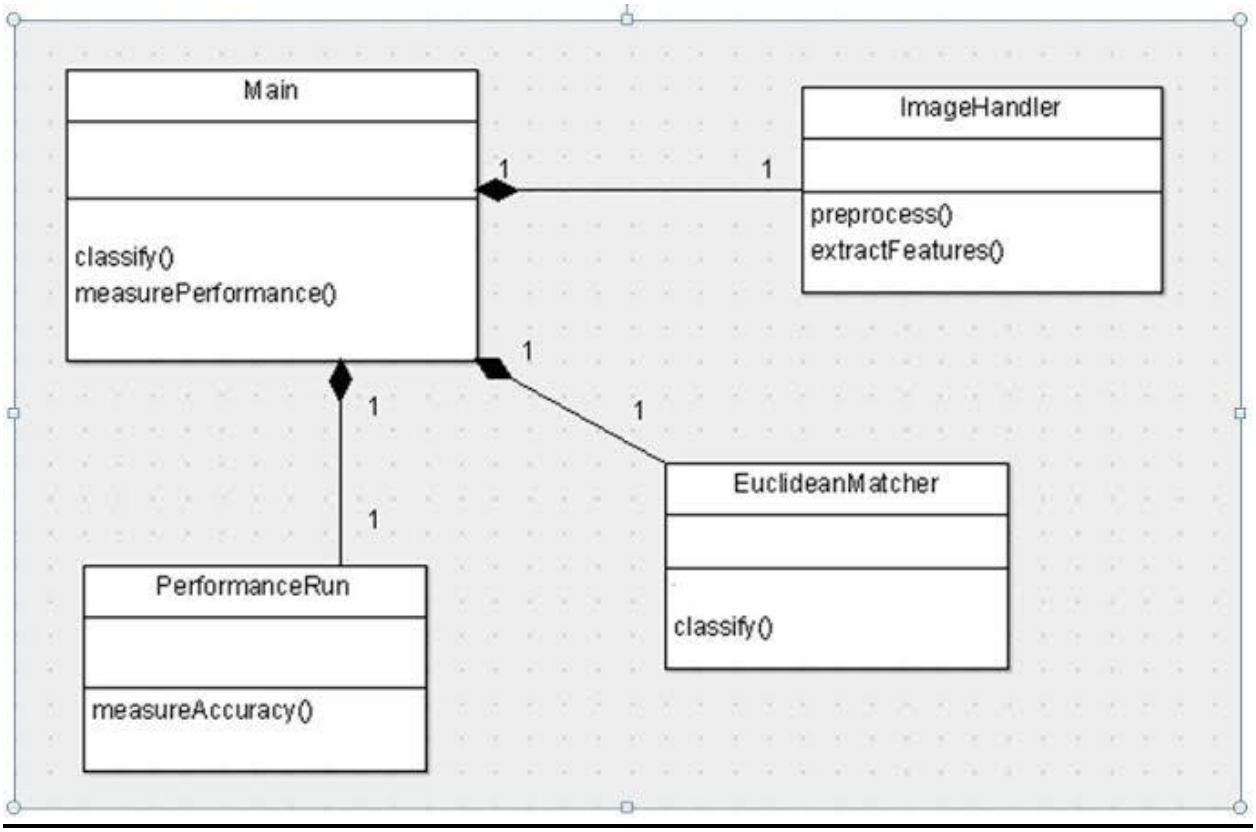

Fig 4.2:Class Diagram

Sequence Diagram:

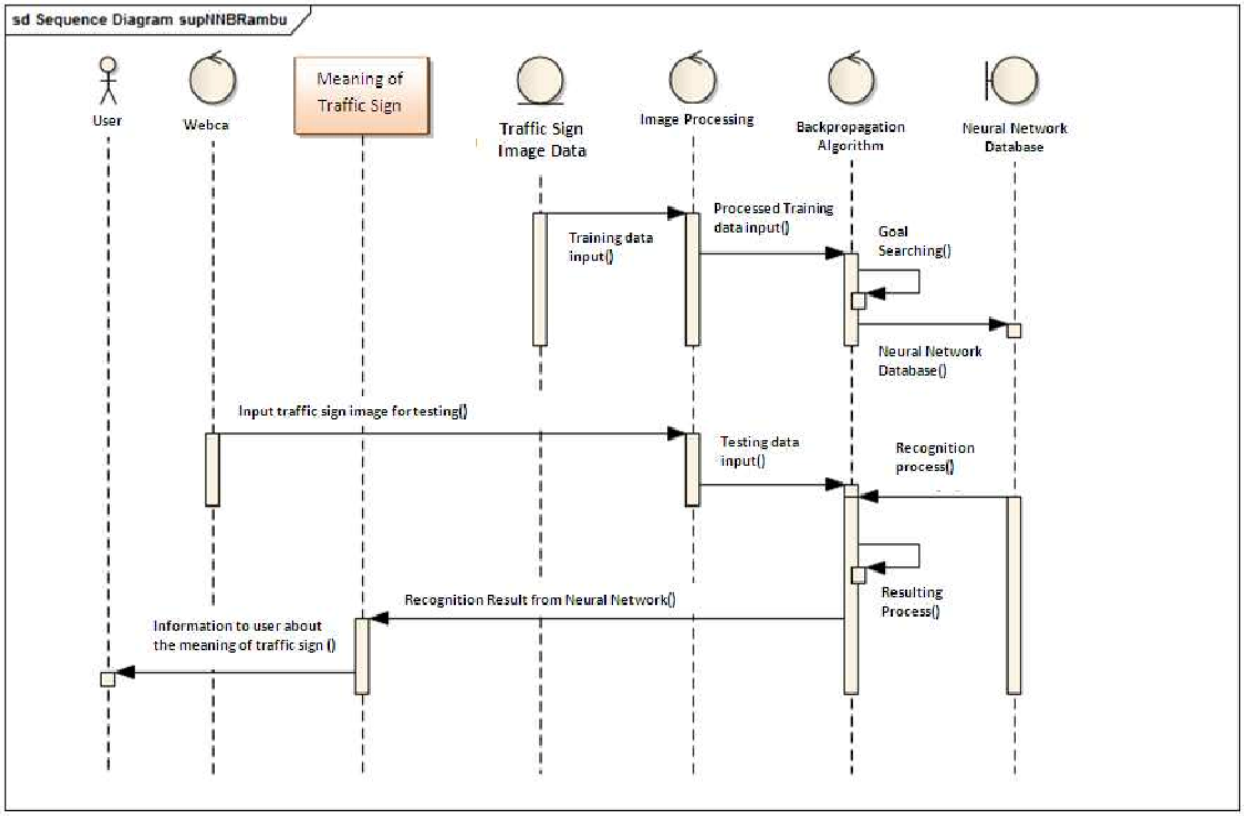

Fig.4.2 : Sequence diagram for Traffic Sign Recognition.

CHAPTER – 5

IMPLEMENTATION

5.1 SAMPLE CODE

```
import os

import numpy as np

import cv2

from keras.models import load_model

from tkinter import Tk, filedialog, Label, Button, ttk

from PIL import Image, ImageTk

MODEL_PATH ='C://Users//mayan//OneDrive//Desktop//SEM V//MINI PROJECT//traffic_sign_recognition//model.h5'

model = load_model(MODEL_PATH)

class TrafficSignRecognitionApp:

    def __init__(self, master):

        self.master = master

        master.title("Traffic Sign Recognition")

        master.geometry("600x400")

        master.configure(bg="#f0f0f0")

        self.label_trafficlights = Label(master)

        self.label_trafficlights.pack(pady=10)
```

```
        self.label_select = Label(master, text="Click below to select an image for prediction:", bg="#f0f0f0", fg="#333333", font=("Arial", 12))

        self.label_select.pack()

        self.select_button = Button(master, text="Select Image", command=self.select_image, bg="#4CAF50", fg="white", font=("Arial", 12))

        self.select_button.pack()

        self.label_selected_image = Label(master)

        self.label_selected_image.pack(pady=10)

        self.result_label = Label(master, text="", bg="#f0f0f0", fg="#333333", font=("Arial", 12))

        self.result_label.pack(pady=10)

        self.quit_button = Button(master, text="Quit", command=master.quit, bg="#f44336", fg="white", font=("Arial", 12))

        self.quit_button.pack()

        self.load_trafficlights_image()

    def load_trafficlights_image(self):

        img_path = "C://Users//mayan//OneDrive//Desktop//SEM V//MINI PROJECT//traffic_sign_recognition//image.jpg"  # Path to your traffic lights image

        img = Image.open(img_path)

        img = img.resize((200, 200), Image.LANCZOS)

        photo = ImageTk.PhotoImage(img)

        self.label_trafficlights.config(image=photo)
```

```
        self.label_trafficlights.image = photo
    def grayscale(self, img):
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        return img
    def equalize(self, img):
        img = cv2.equalizeHist(img)
        return img
    def preprocessing(self, img):
        img = self.grayscale(img)
        img = self.equalize(img)
        img = img / 255
        return img
    def getClassName(self, classNo):
        class_names = [
            'Speed Limit 20 km/h', 'Speed Limit 30 km/h', 'Speed Limit 50 km/h',
            'Speed Limit 60 km/h', 'Speed Limit 70 km/h', 'Speed Limit 80 km/h',
            'End of Speed Limit 80 km/h', 'Speed Limit 100 km/h', 'Speed Limit 120
km/h',
            'No passing', 'No passing for vehicles over 3.5 metric tons',
            'Right-of-way at the next intersection', 'Priority road', 'Yield', 'Stop',
            'No vehicles', 'Vehicles over 3.5 metric tons prohibited', 'No entry',
```

```
            'General caution', 'Dangerous curve to the left', 'Dangerous curve to the right',
            'Double curve', 'Bumpy road', 'Slippery road', 'Road narrows on the right',
            'Road work', 'Traffic signals', 'Pedestrians', 'Children crossing',
            'Bicycles crossing', 'Beware of ice/snow', 'Wild animals crossing',
            'End of all speed and passing limits', 'Turn right ahead', 'Turn left ahead',
            'Ahead only', 'Go straight or right', 'Go straight or left', 'Keep right',
            'Keep left', 'Roundabout mandatory', 'End of no passing',
            'End of no passing by vehicles over 3.5 metric tons'
        ]
        return class_names[classNo]
    def model_predict(self, img):
        img = cv2.resize(img, (32, 32))
        img = self.preprocessing(img)
        img = img.reshape(1, 32, 32, 1)
        predictions = model.predict(img)
        classIndex = np.argmax(predictions)
        preds = self.getClassName(classIndex)
        return preds
    def select_image(self):
        file_path = filedialog.askopenfilename(title="Select Image",
filetypes=[("Image files", "*.jpg;*.jpeg;*.png")])
```

```
        if file_path:
            img = cv2.imread(file_path)
            if img is not None:
                preds = self.model_predict(img)
                self.result_label.config(text=f"Predicted Road Sign: {preds}")
                # Display selected image
                img = cv2.cvtColor(cv2.imread(file_path), cv2.COLOR_BGR2RGB)
                img = Image.fromarray(img)
                img = img.resize((200, 200), Image.LANCZOS)
                photo = ImageTk.PhotoImage(img)
                self.label_selected_image.config(image=photo)
                self.label_selected_image.image = photo
            else:
                self.result_label.config(text=f"Error reading image: {file_path}")
        else:
            self.result_label.config(text="No image selected.")
root = Tk()
app = TrafficSignRecognitionApp(root)
root.mainloop()
```

CHAPTER – 6

TESTING

6.1 TEST CASES

Test Case to check whether the required Software is installed on the systems

Test Case ID:	1
Test Case Name:	Required Software Testing
Purpose:	To check whether the required Software is installed on the systems
Input:	Enter python command
Expected Result:	Should Display the version number for the python
Actual Result:	Displays python version
Failure	If the python environment is not installed, then the Deployment fails

Table 6.1.1 python Installation verification

Test Case to check Program Integration Testing

Test Case ID:	2
Test Case Name:	Programs Integration Testing
Purpose:	To ensure that all the modules work together
Input:	All the modules should be accessed.
Expected Result:	All the modules should be functioning properly.
Actual Result:	All the modules should be functioning properly.
Failure	If any module fails to function properly, the implementation fails.

Table 6.1.2 python Programs Integration Testing

Test Case to Collect Dataset and Load the Dataset

Test Case ID:	3
Test Case Name:	Collect Dataset and Load the Dataset
Purpose:	Check Dataset is collected, and the data is stored
Input:	Provide Dataset as input
Expected Result:	Dataset is collected and view the Dataset and store the Dataset
Actual Result:	Load the Dataset and view the Dataset and store
Failure	If the dataset is not loaded, it will throw an error.

Table 6.1.3 Collect Dataset and Load the Dataset

Test Case to check whether the Traffic Sign is recognized

Test Case ID:	4
Test Case Name:	Traffic Signs Recognition
Purpose:	Traffic Signs Recognition using CNN
Input:	Provide dataset and input an image
Expected Result:	After Evaluation I get the Meaning of Traffic Sign
Actual Result:	After Evaluation I get the Meaning of Traffic Sign
Failure	If the data is not Evaluated, it does not display the gesture

Table 6.1.4 Traffic Sign Recognition

CHAPTER – 7

SCREENSHOTS

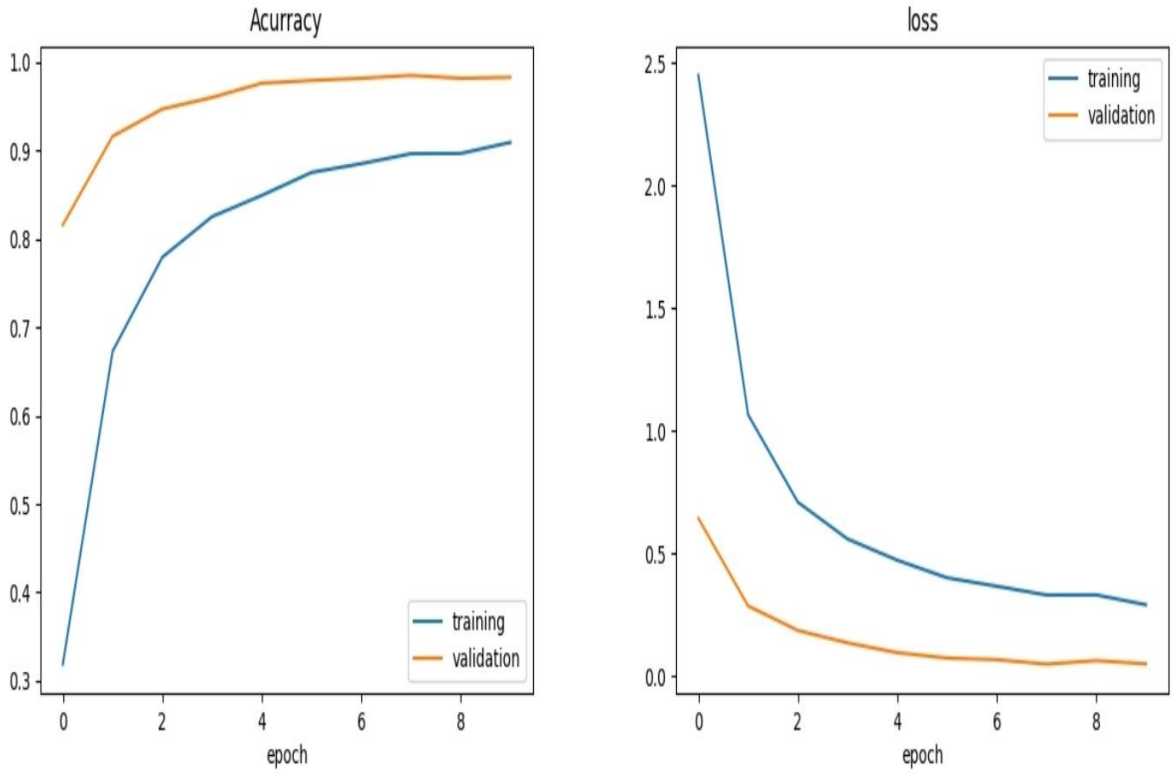

Figure 7.1 Accuracy and loss of the model

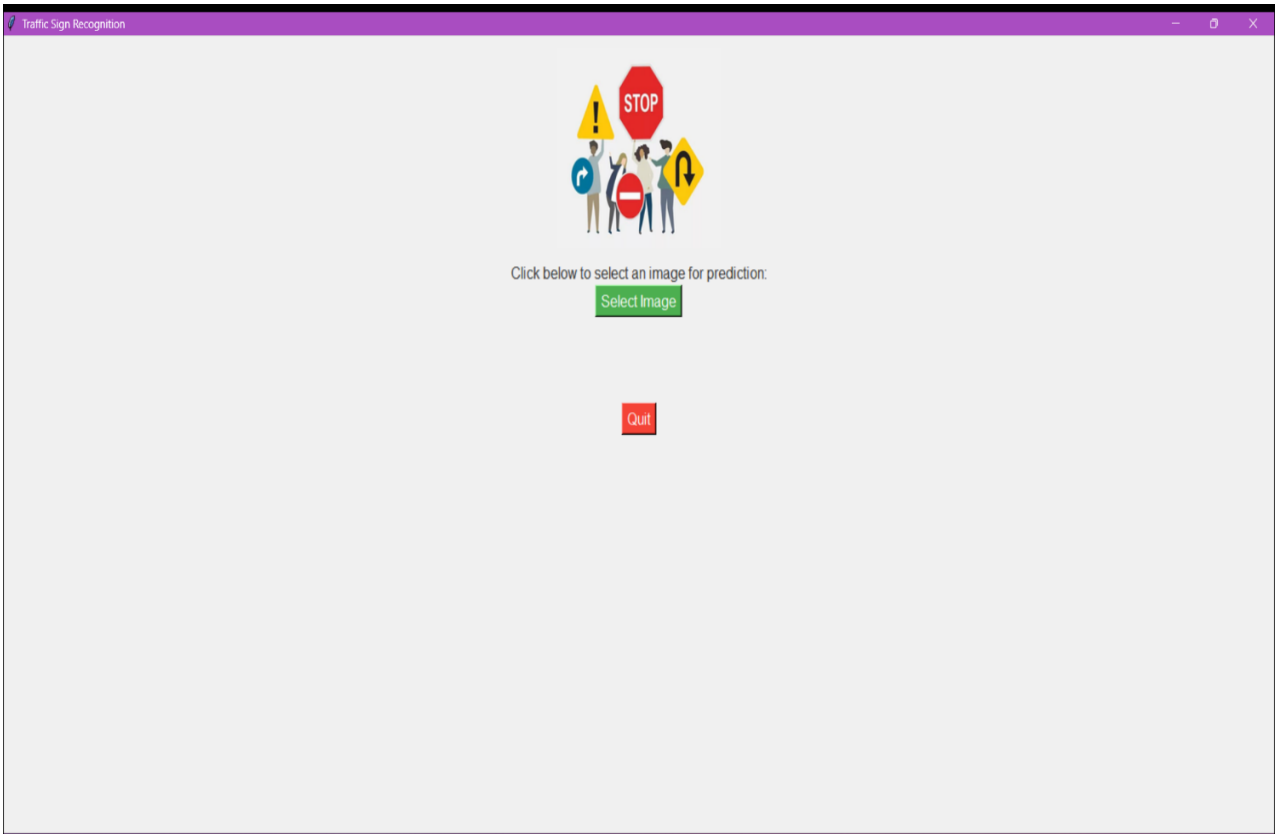

Figure 7.2 HOMEPAGE of the application

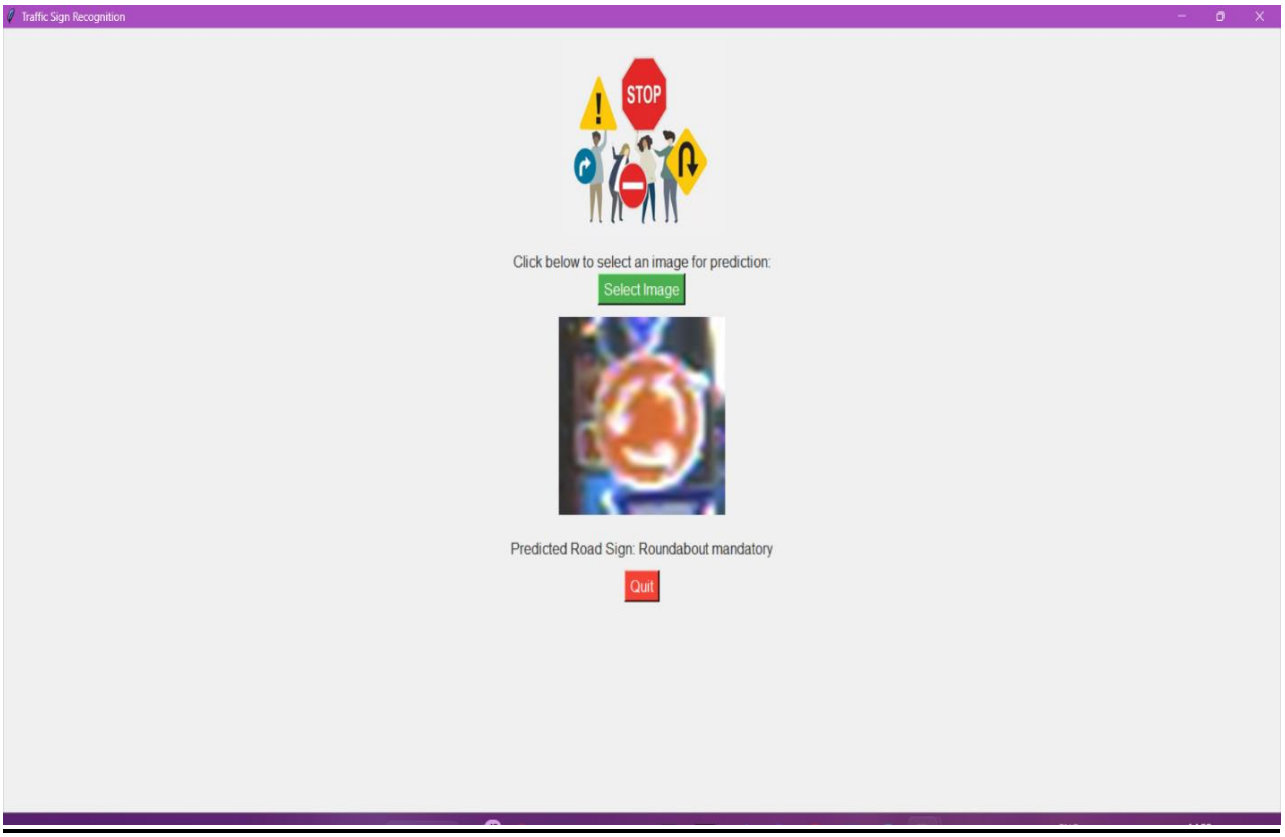

Figure 7.3 Sample Recognition of Traffic Sign

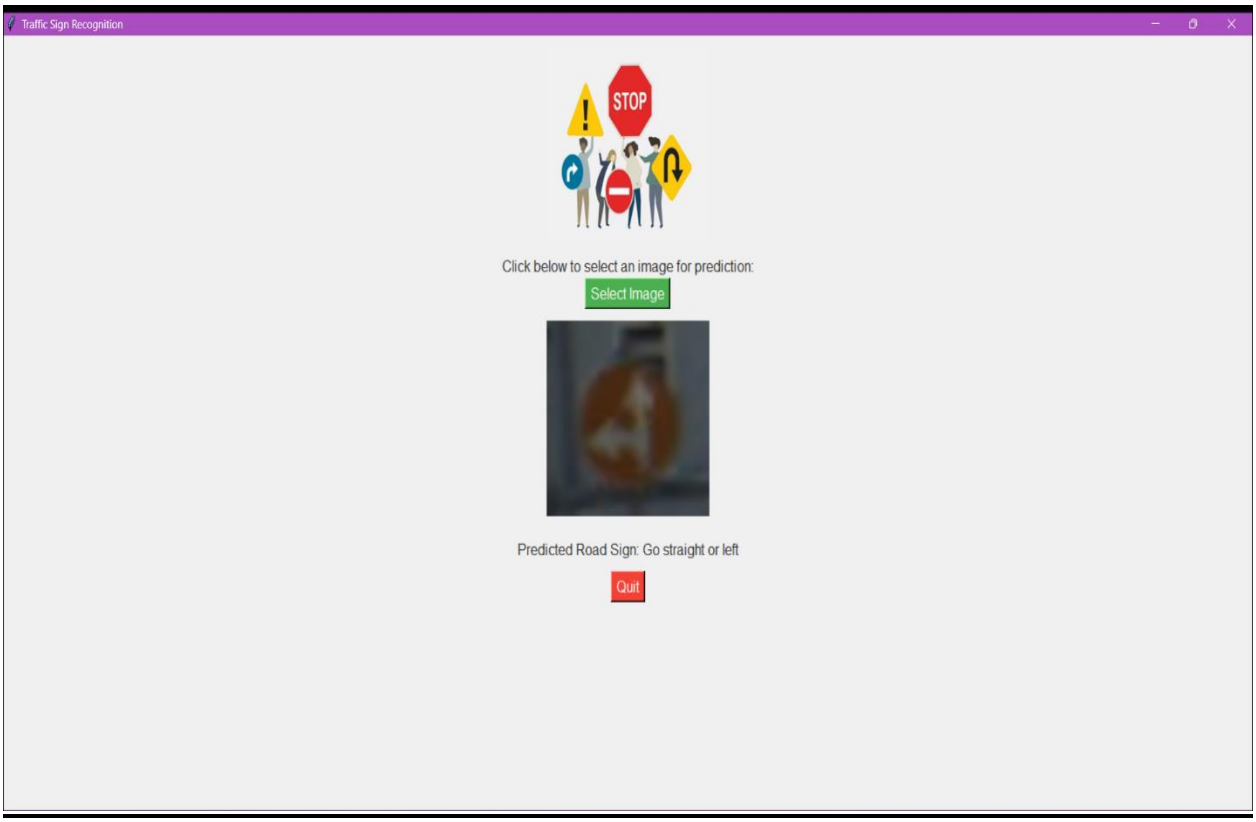

Figure 7.4 Sample Recognition of Traffic Sign

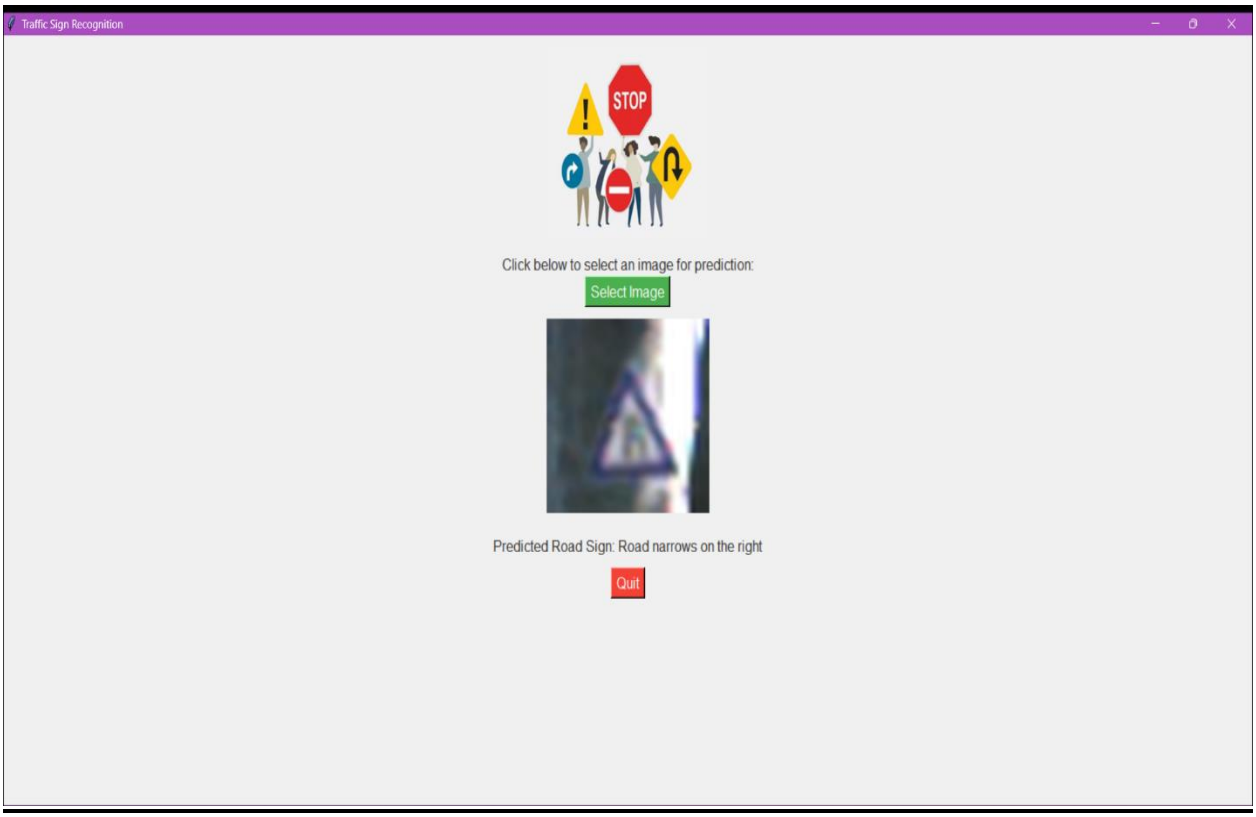

Figure 7.5 Sample Recognition of Traffic Sign

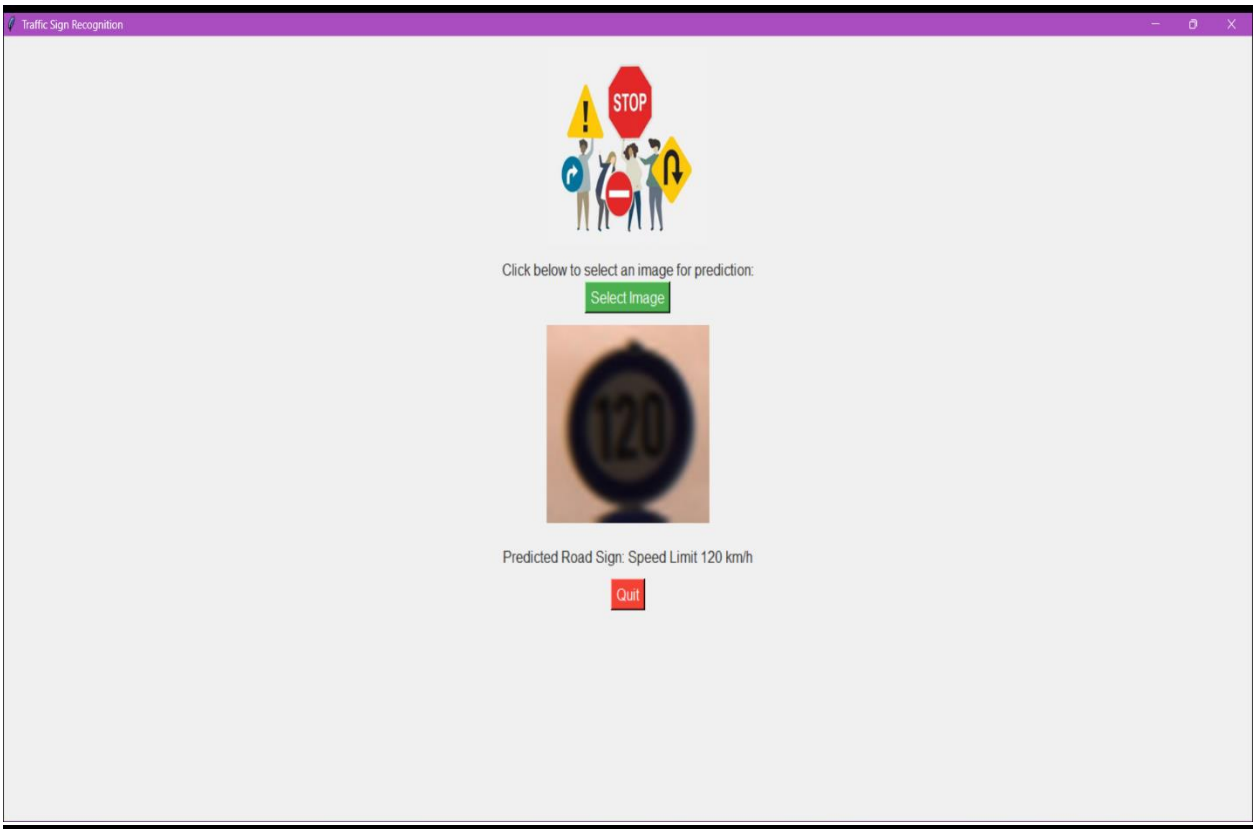

Figure 7.6 Sample Recognition of Traffic Sign

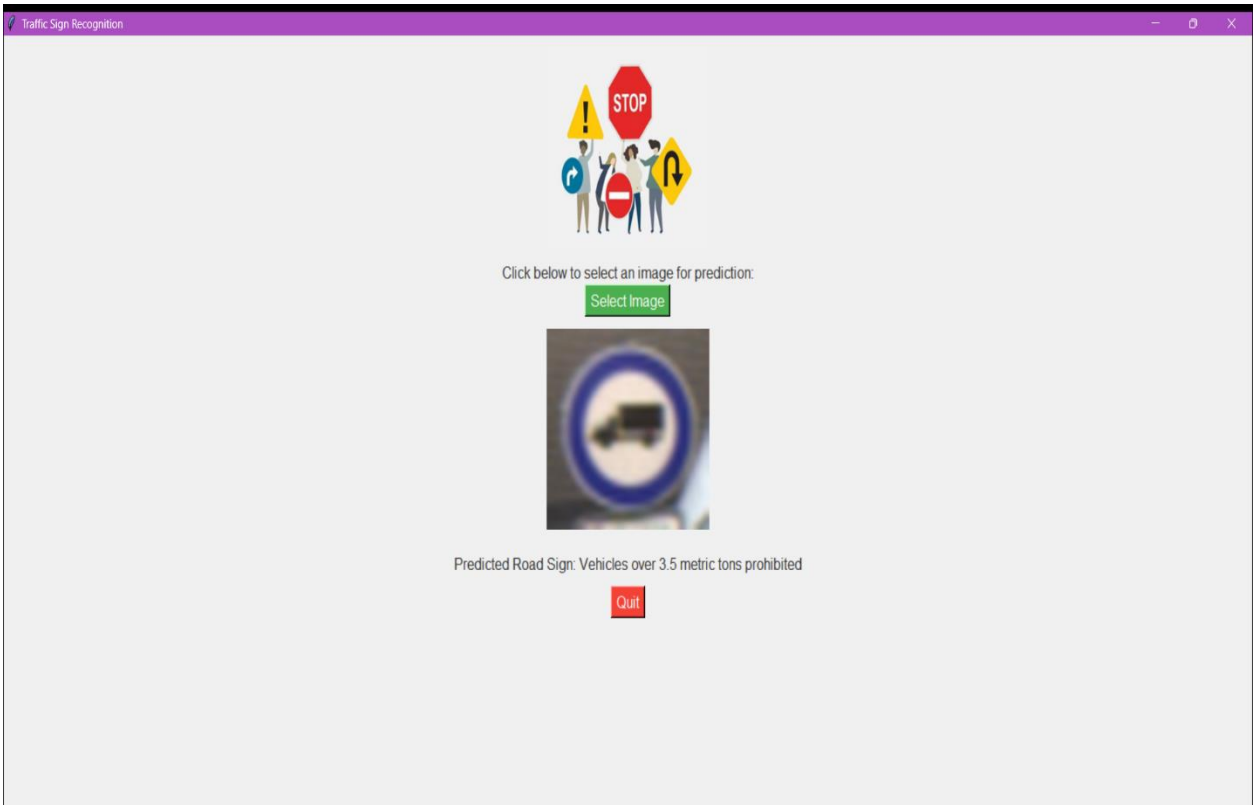

Figure 7.7 Sample Recognition of Traffic Sign

CHAPTER - 8

CONCLUSION AND FUTURE SCOPE

The traffic sign recognition project aims to improve road safety and efficiency. Using CNN technology, it simplifies traffic sign identification for drivers. Although the current model shows promise, there's room for improvement and future development.

However, ongoing development and refinement are crucial to further enhance the system's performance and reliability, ensuring its effectiveness across diverse real-world scenarios. Future efforts will focus on integrating advanced machine learning techniques, such as recurrent neural networks (RNNs) and reinforcement learning, to improve accuracy and adaptability. Additionally, the implementation of real-time object detection algorithms and expansion of the dataset to encompass a wider range of traffic sign variations and environmental conditions will be essential for comprehensive system optimization. Furthermore, sensor fusion techniques and model refinement strategies will be explored to enhance system robustness and adaptability in dynamic traffic environments, ensuring the project's continued relevance and impact in enhancing road safety and efficiency

BIBLIOGRAPHY

[1] Man, W., Ji, Y., & Zhang, Z. (2018). Image classification based on improved random forest algorithm. In 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA) (pp. 346-350).

[2] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR.

[3] Amato, G., & Falchi, F. (2010). kNN based image classification relying on local feature similarity. In Proceedings of the Third International Conference on Similarity Search and Applications - SISAP '10 (p. 101).

[4] Dave, S. (2017). Image Classification Algorithm based on Multi-Feature Extraction and KNN Classifier. Int. J. Adv. Eng. Res. Dev., 4(6).

[5] Rathi, D., Jain, S., & Indu, S. (2017). Underwater Fish Species Classification using Convolutional Neural Network and Deep Learning. In 2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR) (pp. 1-6).

[6] Chen, Y., Lin, Z., & Zhao, X. (2013). Riemannian manifold learning based k-nearest-neighbour for hyperspectral image classification. In 2013 IEEE International Geoscience and Remote Sensing Symposium - IGARSS (pp. 1975-1978).

[7] Blanzieri, E., & Melgani, F. (2008). Nearest Neighbor Classification of Remote Sensing Images with the Maximal Margin Principle. IEEE Trans. Geosci. Remote Sens., 46(6), 1804-1811.

[8] Jia, X., & Richards, J. (2005). Fast K-NN Classification Using the Cluster-Space Approach. IEEE Geoscience and Remote Sensing Letters, 2(2), 225-228.

[9] Varoquaux, G., Buitinck, L., Louppe, G., Grisel, O., Pedregosa, F., & Mueller, A. (2015). Scikit-learn. GetMobile Mob. Comput. Commun., 19(1), 29-33.

[10] Mercier, G., & Lennon, M. (2003). Support vector machines for hyperspectral image classification with spectral-based kernels. In IGARSS 2003. 2003 IEEE.

APPENDIX A: TOOLS AND TECHNOLOGIES

- Python v3: Python is a versatile programming language with a rich ecosystem of libraries and frameworks, making coding efficient and saving time.

- OpenCV (cv2): OpenCV is a popular open-source computer vision library used for image processing tasks such as reading, writing, and manipulating images.

- Keras: Keras is an open-source software library providing a Python interface for building artificial neural networks. It simplifies the process of building and training neural network models.

- Tkinter: Tkinter is a standard GUI (Graphical User Interface) toolkit in Python used for creating desktop applications. It provides a set of tools for building user interfaces with widgets such as buttons, labels, and entry fields.

- PIL (Python Imaging Library) / Pillow: PIL is a library in Python used for opening, manipulating, and saving many different image file formats. Pillow is a modernized version of PIL that provides additional functionality and support for Python 3.

- NumPy: NumPy is a Python library used for working with arrays, along with functions for linear algebra, Fourier transforms, and matrix operations.

- These tools and technologies were utilized to develop the traffic sign recognition application, providing a robust and efficient framework for image processing, neural network modeling, and user interface development.