An

Industry Oriented Mini Project Report

On

**TRAFFIC SIGN RECOGNITION USING DEEP LEARNING**

A report submitted in partial fulfilment of the requirements for the award of Degree of

Bachelor of Technology

By

G. Sanjay Deep

(20EG105414)

V. Hemanth

(20EG105450)

D. Nithin Reddy



(20EG105408)



Under The Guidance Of

E. Radha Krishnaiah

Assistant Professor, CSE

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**ANURAG UNIVERSITY VENKATAPUR– 500088 TELANGANA**

**YEAR 2023-2024**

**DECLARATION**

We hereby declare that the project titled “ **TRAFFIC SIGN RECOGNITION USING DEEP LEARNING ”** submitted for the award of Bachelor of technology Degree is our own work and the Report has not formed the basis for the award of any degree , diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.



G. Sanjay Deep

(20EG105414)

V. Hemanth

(20EG105450)

D. Nithin Reddy

(20EG105408)



**CERTIFICATE**

This is to certify that the Report / dissertation entitled **Traffic Sign Recognition Using Deep Learning** that is being submitted by **Mr. G. Sanjay Deep** bearing the Hall ticket number **20EG105414**, **Mr. V. Hemanth** bearing the Hall ticket number **20EG105450, Mr. D. Nithin Devaram** bearing the Hall ticket number **20EG105408,** in partial fulfilment for the award of B. Tech in Computer Science And Engineering to the Anurag University is a record of bonafide work carried out by them under our guidance and supervision.

The results embodied in this Report have not been submitted to any other university or Institute for the award of any degree or diploma.

Signature of The Supervisor Head of The Department External Examiner

E. Radha Krishnaiah

Assistant Professor

**ACKNOWLEDGEMENT**

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **E. Radha Krishnaiah** for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.



We would like to acknowledge our sincere gratitude for the support extended by

**Dr. G. Vishnu Murthy** , Dean, Dept. of CSE ,Anurag University. We also express our deep sense of gratitude to **Dr. V V S S S Balaram**, Acadamic co-ordinator. **Dr. Pallam Ravi** Project Co-ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage our project work.

We would like express our special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for his encouragement and timely support in our B.Tech program.

G. Sanjay Deep

(20EG105414)

V. Hemanth

(20EG105450)

D. Nithin Reddy

(20EG105408)

**ABSTRACT**

A road sign detection and recognition system is providing a comprehensive assistance to the driver to follow the traffic signs by developing a system that will automatically detect and recognize traffic sign, thus providing accurate information to the driving system. This system comprises of capturing signboards using camera installed in the vehicle. This captured image will undergo for analysis, processing and identifying the signboard. The detected sign board will be notified to the user by sending a notification. This paper proposes a system which will classifies different types of traffic signs in real time video.

A deep learning based road traffic signs recognition method is developed which is very promising in the development of Advanced Driver Assistance Systems (ADAS) and autonomous vehicles. The system architecture is designed to extract main features from images of traffic signs to classify them under different categories. It is constituted of a Convolutional Neural Network (CNN) modified by connecting the output of all convolutional layers to the Multilayer Perceptron (MLP). The training is conducted using the German Traffic Sign Dataset and achieves good results on recognizing traffic signs.

Traffic sign recognition system (TSRS) is a significant portion of intelligent transportation system (ITS). Being able to identify traffic signs accurately and effectively can improve the driving safety. This paper brings forward a traffic sign recognition technique on the strength of deep learning, which mainly aims at the detection and classification of circular signs. Firstly, an image is pre-processed to highlight important information. Secondly, Hough Transform is used for detecting and locating areas. Finally, the detected road traffic signs are classified based on deep learning.

In this article, a traffic sign detection and identification method on account of the image processing is proposed, which is combined with convolutional neural network (CNN) to sort traffic signs. On account of its high recognition rate, CNN can be used to realize various computer vision tasks. A series of warnings about the route are conveyed by traffic signs. They keep traffic going by aiding travellers in reaching their destinations and providing them with advance notice of arrival, exit, and turn points. Road signs are placed in specific positions to ensure the safety of travellers. They also have guidance for when and where drivers can turn or not turn. In this paper, we proposed a system for traffic sign detection and recognition, as well as a method for extracting a road sign from a natural complex image, processing it, and alerting the driver through voice command.

It is applied in such a way that it helps drivers make fast decisions. In real-time situations, factors like shifting weather conditions, changing light directions, and varying light intensity make traffic sign identification challenging. The reliability of the machine is influenced by a number of factors such as noise, partial or absolute underexposure, partial or complete overexposure, and significant variations in colour saturation, wide variety of viewing angles, view depth, and shape/colour deformations of traffic signs (due to light intensity).

**Keywords:** Traffic sign detection, traffic sign classification, traffic sign recognition, Traffic Signs Dataset.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No** | **CONTENT** | **Page No.** |
| 1. | **INTRODUCTION**  1.1 PURPOSE, AIM AND OBJECTIVES  1.2 SCOPE OF PROJECT | **01**  01  02 |
| 2. | **LITERATURE SURVEY** | **03** |
| 3. | **SYSTEM ANALYSIS**  3.1 HARDWARE AND SOFTWARE REQUIREMENTS  3.2 SOFTWARE REQUIREMENTS SPECIFICATION | **04**  04  04 |
| 4. | **TECHNOLOGIES USED**  4.1 PYTHON  4.2 CONVOLUTIONAL NEURAL NETWORK(CNN)  4.3 DEEP LEARNING  4.4 TENSOR FLOW | **06**  06  07  09  10 |
| 5. | **SYSTEM DESIGN**  5.1 SOFTWARE DESIGN  5.2 ARCHITECTURE  5.3 UNIFIED MODELING LANGUAGE  5.3.1 BUILDING BLOCKS OF UML  5.3.1.1 THINGS IN THE UML  5.3.1.2 RELATIONSHIPS IN THE UML  5.3.2 UML DIAGRAMS  5.3.2.1 CLASS DIAGRAM  5.3.2.2 USE CASE DIAGRAM  5.3.2.3 SEQUENCE DIAGRAM  5.3.2.4 ACTIVITY DIAGRAM  5.3.2.5 STATE CHART DIAGRAM | **12**  12  12  13  14  14  16  17  17  18  19  19  20 |
| 6. | **INPUT/OUTPUT DESIGN**  6.1 INPUT DESIGN  6.2 OUTPUT DESIGN | **21**  21  21 |
| 7. | **IMPLEMENTATION** | **22** |
| 8. | **SYSTEM TESTING**  8.1 TESTING OBJECTIVES  8.2 USER TRAINING  8.3 MAINTAINENCE  8.4 TESTING STRATEGY  8.5 TEST CASES | **29**  29  30  30  30  31 |
| 9. | **EXECUTION SCREENS** | **37** |
| 10. | **CONCLUSION & FUTURE SCOPE** | **41** |
| 11. | **BIBILOGRAPHY** | **42** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure. Name** | **Page No.** |
| 4.2 | Example for Convolutional Neural Networks. | 07 |
| 4.4 | Example for TensorFlow Graphs | 10 |
| 5.2 | System flow diagram | 13 |
| 5.3.2.1 | Class diagram | 17 |
| 5.3.2.2 | Use case diagram | 18 |
| 5.3.2.3 | sequence diagram | 19 |
| 5.3.2.4 | Activity Diagram | 20 |
| 5.3.2.5 | State chart Diagram | 20 |
| 8.5.2.1 | User Interface | 33 |
| 8.5.2.2 | Uploading an image | 35 |
| 8.5.2.3 | Classify the image | 36 |

**LIST OF OUTPUTS**

|  |  |  |
| --- | --- | --- |
| **Fig.No** | **NAME** | **Page No.** |
| 9.1 | Uploading Image | 37 |
| 9.2 | General caution sign | 38 |
| 9.3 | No entry sign | 39 |
| 9.4 | No passing sign | 39 |
| 9.5 | Speed limit(100km/h) sign | 40 |
| 9.6 | Speed limit(70km/h) sign | 40 |

# 1. INTRODUCTION

This chapter gives an overview about the purpose, aim, objectives, background and operation environment of the system.

**1.1 PURPOSE, AIM AND OBJECTIVES:**

With the rapid development in societal and economical section, automobiles have become almost one of the convenient modes of transportation in each household. This makes the road traffic environment more complicated, and people expect to have intelligent Vision assisted system that provide drivers with traffic sign information, regulate operations of driver, or assist in vehicle control to ensure road safety. As one of the more important functions, traffic sign detection and recognition system, has become a hot research direction of researchers at home and abroad.

In the current traffic system there may be the probability of drivers missing the traffic sign because of traffic or even the drivers might ignore the traffic sign. With the continuous growth of urbanization this problem is only expected to grow worse.

A traffic sign detection and recognizing system can be applied to the vehicles where the system capture the traffic sign, detects them and recognizes the significance of the sign and informs the driver about the sign. The efficiency of traffic sign detection and recognition is influenced by any factors like

* Colour fading: This occurs due to long exposure to sun and rain.
* Vehicle motion: the vehicle motion may lead to camera jitter or blur the image.
* Weather conditions: The clarity of the image is varied by due to weather conditions such as heavy rain etc. Traffic sign recognition is divided into two parts detection and recognition. The traffic signs have distinct color and particular shapes, which can be easily observed by drivers. Traffic sign detection is usually based on the inherent Characteristics of traffic signs such as shape and colour. The colour characteristics of traffic signs are more common, they usually mainly are red, yellow, and blue. Colour enhancement method is used to extract red, yellow and blue colour spots, which focuses on the pixels of a given colour are dominant over the other two colour channels in colour space of RGB. In the Lab and HSI colour spaces are used to extract candidate signs. Meanwhile, the detected signs will help to extract more information, discarding the uninteresting or irrelevant area, connecting the scattered signs, and separate signs at the same location.

Detection methods based on the colour characteristics have low computing, good robustness and other characteristics, which can improve the detection performance to a certain extent, but they depend on the corresponding threshold. Shape-based detection researches are generally based on the specific colour and shape of the traffic signs, mainly are triangles, circles, rectangles and square. These approaches have a certain degree of robustness.

The overall aim is to develop a system that can be used for traffic sign inventory. This system can assist local or national authorities in the task of maintaining and updating their road and traffic signs by automatically detecting and classifying one or more traffic signs from a complex scene (like the one shown in Figure 1.2) when captured by a camera from a vehicle.

This research focuses on the following objectives

1. The use of traffic sign recognition system is very challenging. Since rain, fog, snow etc. affect the whole system. Another thing is the light variation i.e., shadows, sun, clouds etc.
2. The geometrical shape of the object and the perspective is also a big concern. So, we must be able to come up with a system which can work under light variation and geometrical transformation of the objects in a scene. If we narrow down the scope, we are particularly interested in detection of these signs under low light condition.
   1. **SCOPE OF PROJECT:**

Our system mainly focus on the security that must be provided to the drivers in case of any emergency situations or normal situations where they need immediate help.

**2.LITERATURE SURVEY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No** | **Author (s)** | **Method** | **Advantages** | **Disadvantages** |
| 1 | Kaoutar Sefrioui  Boujemaa, Afaf  Bouhoute, Karim  Boubouh and Ismail  Berrada | Relational Convolutional  Neural Networks | High accuracy in image classification tasks , making them  well-suited for traffic sign recognition. | Facing issues in resource-constrained environments, such as in embedded systems. |
| 2 | Prashengit Dhar,  Md. Zainal Abedin,  Tonoy Biswas,  Anish Datta | Convolutional Neural Networks | Real-time alerts about speed limits, warnings. | Adverse weather conditions like heavy rain, snow, or fog can affect the performance of the recognition system, reducing its reliability. |
| 3 | Faming Shao,  Xinqing Wang,  Fanjie Meng, Ting Rui, Dong Wang, and Jian Tang | Simplified Gabor Wavelets | Gabor wavelets helping to differentiate between different sign types. | It focus on individual features of traffic signs. |
| 4 | Shopa, Sumitha, & Patra | OpenCV (Open Source Computer Vision Library) | Real-time image processing. | Daily implementation required. |
| 5 | Faming Shao,  Xinqing Wang,  Fanjie Meng, Ting Rui, Dong Wang, and Jian Tang | Simplified Gabor Wavelets and CNNs | It handle variations in lighting conditions, weather, and different angles of view, making them robust for real-world scenarios. | It doesn’t show variations in traffic sign designs, colours, and shapes. |

**3. SYSTEM ANALYSIS**

In this chapter, we will discuss and analyse about the developing process of Audit Control including software requirement specification (SRS) and comparison between existing and proposed system. The functional and non-functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out. Besides that, existing vs. proposed provides a view of how the proposed system will be more efficient than the existing one.

* 1. **HARDWARE AND SOFTWARE REQUIREMENTS**

**3.1.1 HARDWARE REQUIREMENTS:**

**Processor** : Intel or AMD.

**Ram** : 1 GB or above.

**Hard disk** : 40GB or above.

* + 1. **SOFTWARE REQUIREMENTS:**

**Technology/Language** : python, html

##### Operating System : windows

**Libraries** : keras, CV2, sklearn

**3.2 SOFTWARE REQUIREMENT SPECIFICATION:**

**3.2.1 SRS:**

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software

project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

**3.2.3 EXIXSTING SYSTEM**

* Traditionally, standard computer vision methods were employed to detect and classify traffic signs, but these required considerable and time-consuming manual work to handcraft important features in images.

**3.2.3.1 DRAWBACK OF EXISTING SYSTEM**

* **Low accuracy:** Existing systems are providing low accuracy for the solution provided by those systems.
* **Time consuming**: Existing systems are providing the solution for the problem, but they are taking very long time. So, the time complexity is also increasing which may result in serious impact on the users.

**3.2.3 PROPOSED SYSTEM**

* By applying deep learning to this problem, we create a model that reliably classifies traffic signs, learning to identify the most appropriate features for this problem by itself.
* A deep learning based road traffic signs recognition method is developed which is very promising in the development of Advanced Driver Assistance Systems (ADAS) and autonomous vehicles.

The system architecture is designed to extract main features from images of traffic signs to classify them under different categories. It is constituted of a Convolutional Neural Network

(CNN) modified by connecting the output of all convolutional layers to the Multilayer Perceptron (MLP).The training is conducted using the German Traffic Sign Dataset and achieves good results on recognizing traffic signs.

**4.TECHNOLOGIES USED**

**4.1 PYTHON**

Python is a high-level, general-purpose and a very popular programming language.

Python programming language (latest Python 3) is being used in web development, Machine

Learning applications, along with all cutting edge technology in Software Industry. Python Programming Language is very well suited for Beginners, also for experienced programmers with other programming languages like C++ and Java.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.
* The biggest strength of Python is huge collection of standard library which can be used for the following:
* Machine Learning
* GUI Applications (like kivy, Tkinter, PyQt etc. )
* Web frameworks like Django (used by YouTube, Instagram, Dropbox)
* Image processing (like OpenCV, Pillow)
* Web scraping (like Scrapy, Beautiful Soup, Selenium)
* Test frameworks

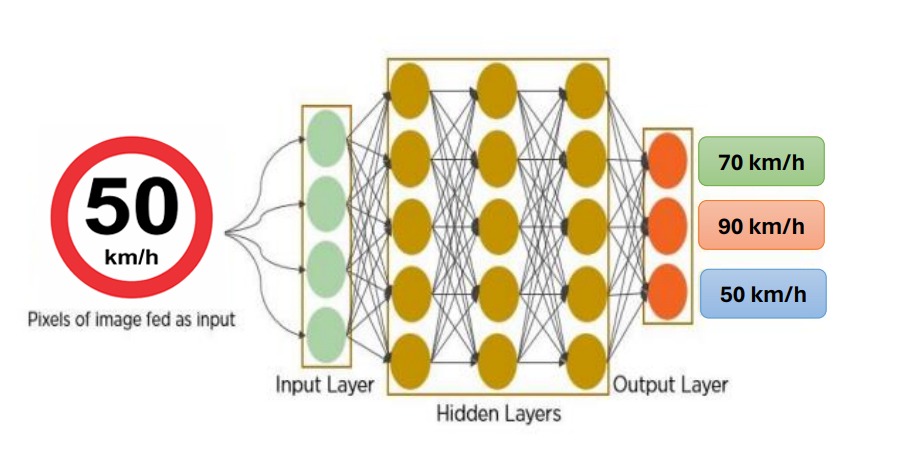
## 4.2 CONVOLUTIONAL NEURAL NETWORKS(CNN):

It is assumed that the reader knows the concept of Neural networks. When it comes to

Machine Learning, Artificial Neural Networks perform really well. Artificial Neural

Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN. Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks.



**Fig:4.2: example for Convolutional Neural Networks.**

Background of CNN : CNN’s were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. This was a major drawback for CNNs at that period and hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.

1. **Input Layer**: It’s the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feedforward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

## 4.3 DEEP LEARNING

Deep learning is a branch of Machine Learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don’t need to explicitly program everything. The concept of deep learning is not new.

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours. The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

The advantage of deep learning is the program builds the feature set by itself without supervision. Unsupervised learning is not only faster, but it is usually more accurate.

It will simply look for patterns of pixels in the digital data. With each iteration, the predictive model becomes more complex and more accurate.

To achieve an acceptable level of accuracy, deep learning programs require access to immense amounts of training data and processing power, neither of which were easily available to programmers until the era of big data and cloud computing. Because deep learning programming can create complex statistical models directly from its own iterative output, it is able to create accurate predictive models from large quantities of unlabelled, unstructured data.

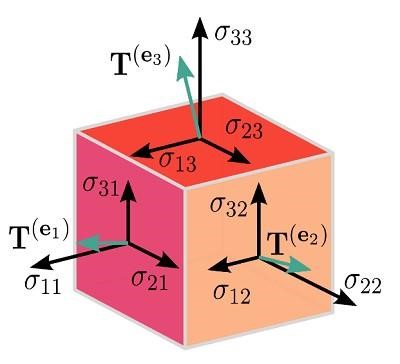
**4.4 TENSOR FLOW :**

TensorFlow has APIs available in several languages both for constructing and executing a TensorFlow graph. The Python API is at present the most complete and the easiest to use, but other language APIs may be easier to integrate into projects and may offer some performance advantages in graph execution.

TensorFlow tutorial is designed for both beginners and professionals. Our tutorial provides all the basic and advanced concept of machine learning and deep learning concept such as deep neural network, image processing and sentiment analysis.

TensorFlow is one of the famous deep learning framework, developed by Google Team. It is a free and open source software library and designed in Python programming language, this tutorial is designed in such a way that we can easily implement deep learning project on TensorFlow in an easy and efficient way.

**Graphs**

TensorFlow makes use of a graph framework. The chart gathers and describes all the computations done during the training. 

##### Fig.4.4:example for TensorFlow Graph

**Advantages**

o It was fixed to run on multiple CPUs or GPUs and mobile operating systems.

* The portability of the graph allows to conserve the computations for current or later use.

The graph can be saved because it can be executed in the future.

* All the computation in the graph is done by connecting tensors together.

Consider the following expression a= (b+c)\*(c+2)

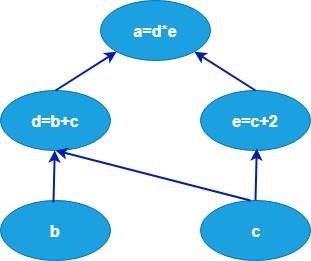
We can break the functions into components given below:

d=b+c

e=c+2

a=d\*e

Now, we can represent these operations graphically below:



# 5. SYSTEM DESIGN

System design is transition from a user oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

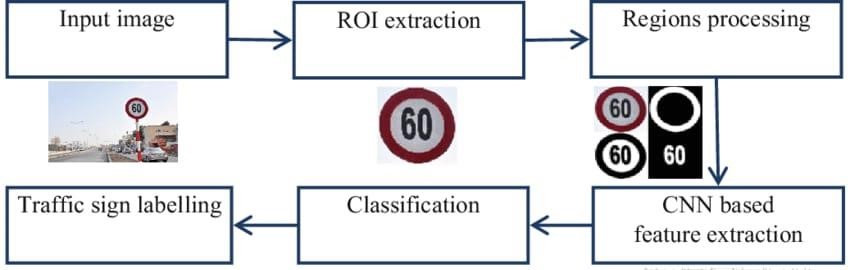
## 5.1 SOFTWARE DESIGN:

In designing the software following principles are followed:

1. **Modularity and partitioning:** Software is designed such that, each system should consists of hierarchy of modules and serve to partition into separate function.
2. **Coupling:** Modules should have little dependence on other modules of a system.
3. **Cohesion:** Modules should carry out in a single processing function.
4. **Shared use:** Avoid duplication by allowing a single module be called by other that need the function it provides.

## 5.2. ARCHITECTURE

Architecture diagram is a diagram of a system, in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.



**Figure 5.2. System Flow diagram**

The traffic detection and recognition system is consisting a user interface that Is where the user can just upload an image of the traffic sign board and then the system will preprocess the uploaded image of the traffic sign board and classifies that then gives result.

## 5.3 UNIFIED MODELING LANGUAGE (UML):

The unified modeling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure and control information about the systems.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management.

### 5.3.1 BUILDING BLOCKS OF UML:

The vocabulary of the UML encompasses three kinds of building blocks:

* Things.
* Relationships.
* Diagrams.

#### 5.3.1.1 Things in the UML:

Things are the abstractions that are first-class citizens in a model; relationship stie these things together; diagrams group interesting collections of things.

There are four kinds of things in the UML:

Structural things.

Behavioral things.

Grouping things.

Annotational things.

**1. Structural things** are the nouns of UML models. The structural things used in the project design are:

i)First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

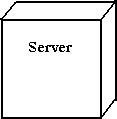
|  |
| --- |
| Window |
| origin size |
| open() close() move() display() |

**Fig: Classes**

1. **Second**, a use case is a description of set of sequence of actions that a system performs that yields an observable result of value to particular actor.

  **Fig: Use Cases**

1. **Third**, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability



**Fig: Nodes**

**2. Behavioral things** are the dynamic parts of UML models. The behavioral thing used is:

i) Interaction: An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links (the connection between objects).



**Fig: Messages**

**5.3.1.2 Relationships in the UML:**

There are four kinds of relationships in the UML:

Dependency.

Association.

Generalization.

Realization.

A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).



**Fig: Dependencies**

An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.

**Fig: Association**

A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).



**Fig: Generalization**

A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.

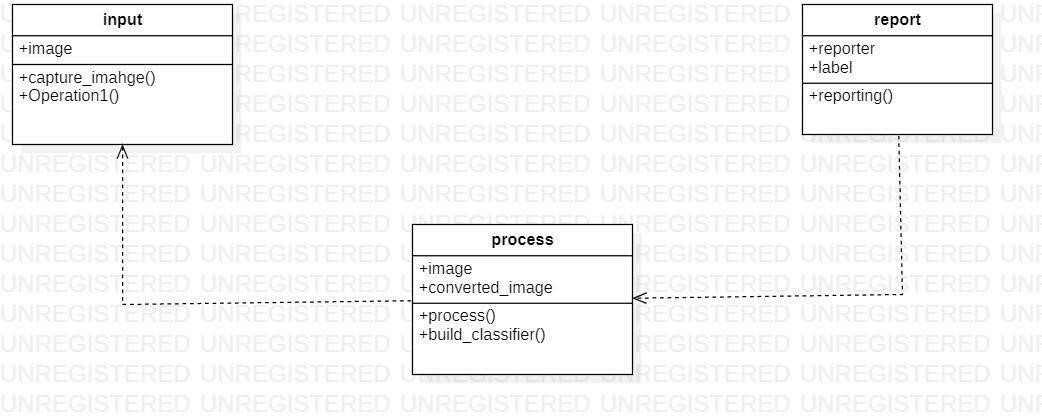


**Fig: realization**

### 5.3.2 UML DIAGRAMS:

#### 5.3.2.1 CLASS DIAGRAM:

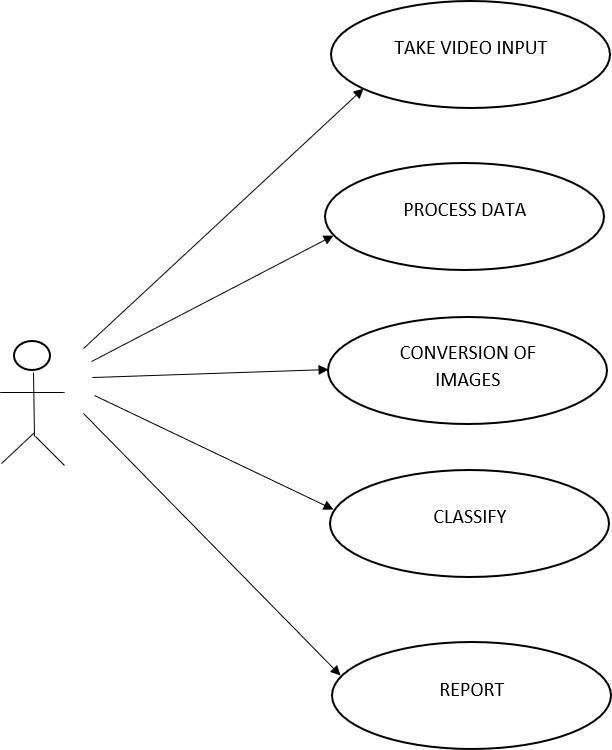
A class is a representation of an object and, in many ways; it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application. Although thousands of students attend the university, you would only model one class, called Student, which would represent the represent the entire collection of students.



**Figure 5.3.2.1 Class diagram**

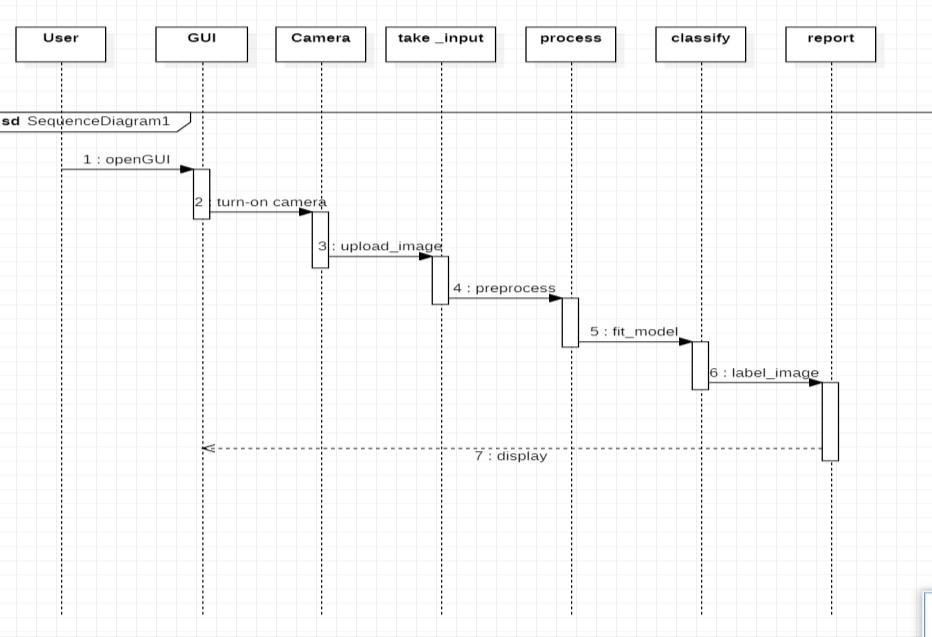
#### 5.3.2.2 USE CASE DIAGRAM:

A use case diagram is a graph of actors set of use cases enclosed by a system boundary, communication associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system’s behavior



**Figure 5.3.2.2 Use case diagram**

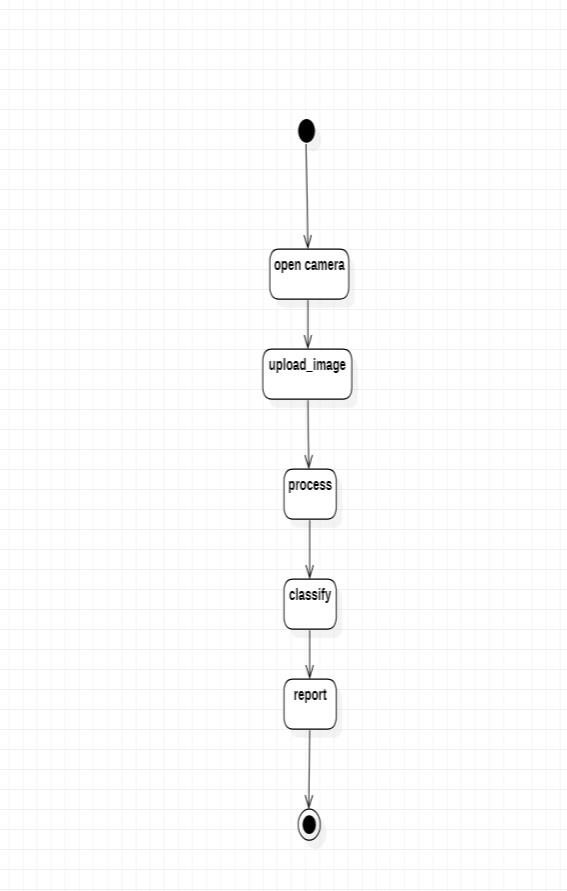
#### 5.3.2.3 SEQUENCE DIAGRAM:



##### Figure 5.3.2.3. sequence diagram

**5.3.2.4 ACTIVITY DIAGRAM:**

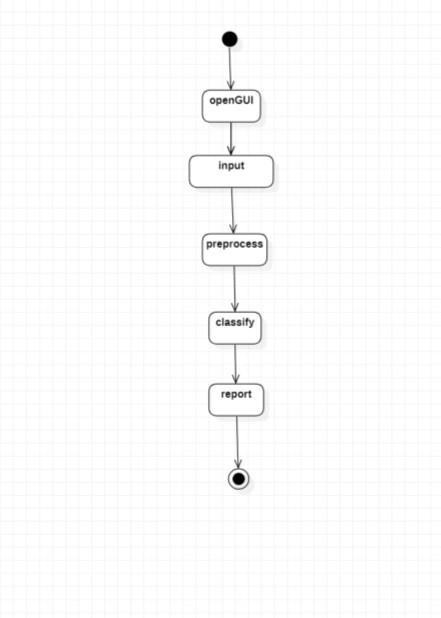
Activity diagram represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state . So, what is the importance of an Activity diagram, as opposed to a State diagram? A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. These transitions depict the activities causing these transitions, shown by arrows.



#### Figure 5.3.2.4. Activity Diagram

#### 5.3.2.5 STATE CHART DIAGRAM:

State chart diagram is used to describe the states of different objects in its life cycle. So the emphasis is given on the state changes upon some internal or external events.



**Figure 5.3.2.5. State chart Diagram**

# 6. INPUT/OUTPUT DESIGN

## 6.1 INPUT DESIGN:

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations. Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

## 6.2 OUTPUT DESIGN:

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User

authentication procedures are maintained at the initial stages itself.

# 7. IMPLEMENTATION

**Model Building**

import numpy as np

import pandas as pd

import matplotlib.pyplot as pl t#to plot accuracy

import cv2

import tensorflow as tf

from PIL import Image

import os

from sklearn.model\_selection import train\_test\_split #to split training and testing data from tensorflow.keras.utils import to\_categorical

#to convert the labels present in y\_train and t\_test into one-hot encoding

from keras.models import Sequential, load\_model

from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout#to create CNN data = []

labels = []

classes = 43

cur\_path = os.getcwd()

#Retrieving the images and their labels 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) #sim = Image.fromarray(image) data.append(image) labels.append(i) except:

print("Error loading image") #Converting lists into numpy arrays data = np.array(data) labels = np.array(labels) print(data.shape, labels.shape)

#Splitting training and testing dataset

X\_t1, X\_t2, y\_t1, y\_t2 = train\_test\_split(data, labels, test\_size=0.2, random\_state=42) print(X\_t1.shape, X\_t2.shape, y\_t1.shape, y\_t2.shape) #Converting the labels into one hot encoding y\_t1 = to\_categorical(y\_t1, 43) y\_t2 = to\_categorical(y\_t2, 43) #Building the model model = Sequential()

model.add(Conv2D(filters=32,kernel\_size=(5,5),activation='relu',input\_shape=X\_t1.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)) model.add(Dense(43, activation='softmax'))

#Compilation of the model model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy']) epochs = 5 history = model.fit(X\_t1, y\_t1, batch\_size=32, epochs=eps, validation\_data=(X\_t2, y\_t2)) model.save("my\_model.h5") #plotting graphs for 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() plt.figure(1) 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()

#testing accuracy on test dataset from sklearn.metrics import accuracy\_score y\_test = pd.read\_csv('Test.csv') labels = 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) predict\_x=model.predict(X\_test) classes\_x=np.argmax(predict\_x,axis=1) #Accuracy with the test data from sklearn.metrics import accuracy\_score print(accuracy\_score(labels, classes\_x)) model.save("traffic\_classifier.h5") **#User interface** import numpy

import tkinter as tk from tkinter import filedialog from tkinter import \* from PIL import ImageTk, Image

#load the trained model to classify sign from keras.models import load\_model model = load\_model('traffic\_classifier.h5') #dictionary to label all traffic signs class. classes = { 1:'Speed limit (20km/h)',

2:'Speed limit (30km/h)',

3:'Speed limit (50km/h)',

4:'Speed limit (60km/h)',

5:'Speed limit (70km/h)',

6:'Speed limit (80km/h)',

7:'End of speed limit (80km/h)',

8:'Speed limit (100km/h)',

9:'Speed limit (120km/h)',

10:'No passing',

11:'No passing veh over 3.5 tons',

12:'Right-of-way at intersection',

13:'Priority road',

14:'Yield',

15:'Stop',

16:'No vehicles',

17:'Veh > 3.5 tons prohibited',

18:'No entry',

19:'General caution',

20:'Dangerous curve left',

21:'Dangerous curve right',

22:'Double curve',

23:'Bumpy road',

24:'Slippery road',

25:'Road narrows on the right',

26:'Road work',

27:'Traffic signals',

28:'Pedestrians',

29:'Children crossing',

30:'Bicycles crossing',

31:'Beware of ice/snow',

32:'Wild animals crossing', 33:'End speed + passing limits',

34:'Turn right ahead',

35:'Turn left ahead',

36:'Ahead only',

37:'Go straight or right',

38:'Go straight or left',

39:'Keep right',

40:'Keep left',

41:'Roundabout mandatory',

42:'End of no passing',

43:'End no passing vehicle with a weight greater than 3.5 tons' }

#initialise GUI top=tk.Tk() top.geometry('800x600') top.title('Traffic sign classification')

top.configure(background='#CDCDCD') label=Label(top,background='#CDCDCD', font=('arial',15,'bold')) sign\_image = Label(top) def classify(file\_path): global label\_packed image = Image.open(file\_path) image = image.resize((30,30)) image = numpy.expand\_dims(image, axis=0)

image = numpy.array(image) predict\_x=model.predict([image])[0] classes\_x=numpy.argmax(predict\_x,axis=0) print(predict\_x) classes\_x=numpy.argmax(predict\_x,axis=0) sign = classes[classes\_x+1] print(sign) label.configure(foreground='#011638', text=sign) def show\_classify\_button(file\_path):

classify\_b=Button(top,text="Classify Image",command=lambda: classify(file\_path),padx=10,pady=5) classify\_b.configure(background='#364156', foreground='white',font=('arial',10,'bold')) classify\_b.place(relx=0.79,rely=0.46) def upload\_image():

try:

file\_path=filedialog.askopenfilename() uploaded=Image.open(file\_path) uploaded.thumbnail(((top.winfo\_width()/2.25),(top.winfo\_height()/2.25))) im=ImageTk.PhotoImage(uploaded) sign\_image.configure(image=im) sign\_image.image=im

label.configure(text='') show\_classify\_button(file\_path) except: pass upload=Button(top,text="Upload an image",command=upload\_image,padx=10,pady=5) upload.configure(background='#364156', foreground='white',font=('arial',10,'bold')) upload.pack(side=BOTTOM,pady=50) sign\_image.pack(side=BOTTOM,expand=True) label.pack(side=BOTTOM,expand=True)

heading = Label(top, text="check traffic sign",pady=20, font=('arial',20,'bold')) heading.configure(background='#CDCDCD',foreground='#364156') heading.pack() top.mainloop()

# 8. SYSTEM TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

## 8.1 TESTING OBJECTIVES:

* To ensure that during operation the system will perform as per specification.
* To make sure that system meets the user requirements during operation .
* To make sure that during the operation, incorrect input, processing and output will be detected.
* To see that when correct inputs are fed to the system the outputs are correct.
* To verify that the controls incorporated in the same system as intended.
* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has a high probability of finding an as yet undiscovered error.

The software developed has been tested successfully using the following testing strategies and any errors that are encountered are corrected and again the part of the program or the procedure or function is put to testing until all the errors are removed. A successful test is one that uncovers an as yet undiscovered error.

Note that the result of the system testing will prove that the system is working correctly. It will give confidence to system designer, users of the system, prevent frustration during implementation process etc.

## 8.2 USER TRAINING:

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

## 8.3 MAINTAINENCE:

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

## 8.4 TESTING STRATEGY :

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements. Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding.

**8.4.1 SYSTEM TESTING:**

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

**8.4.2 UNIT TESTING:**

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself.

## 8.5 TEST CASES:

**8.5.1 Model Building:**

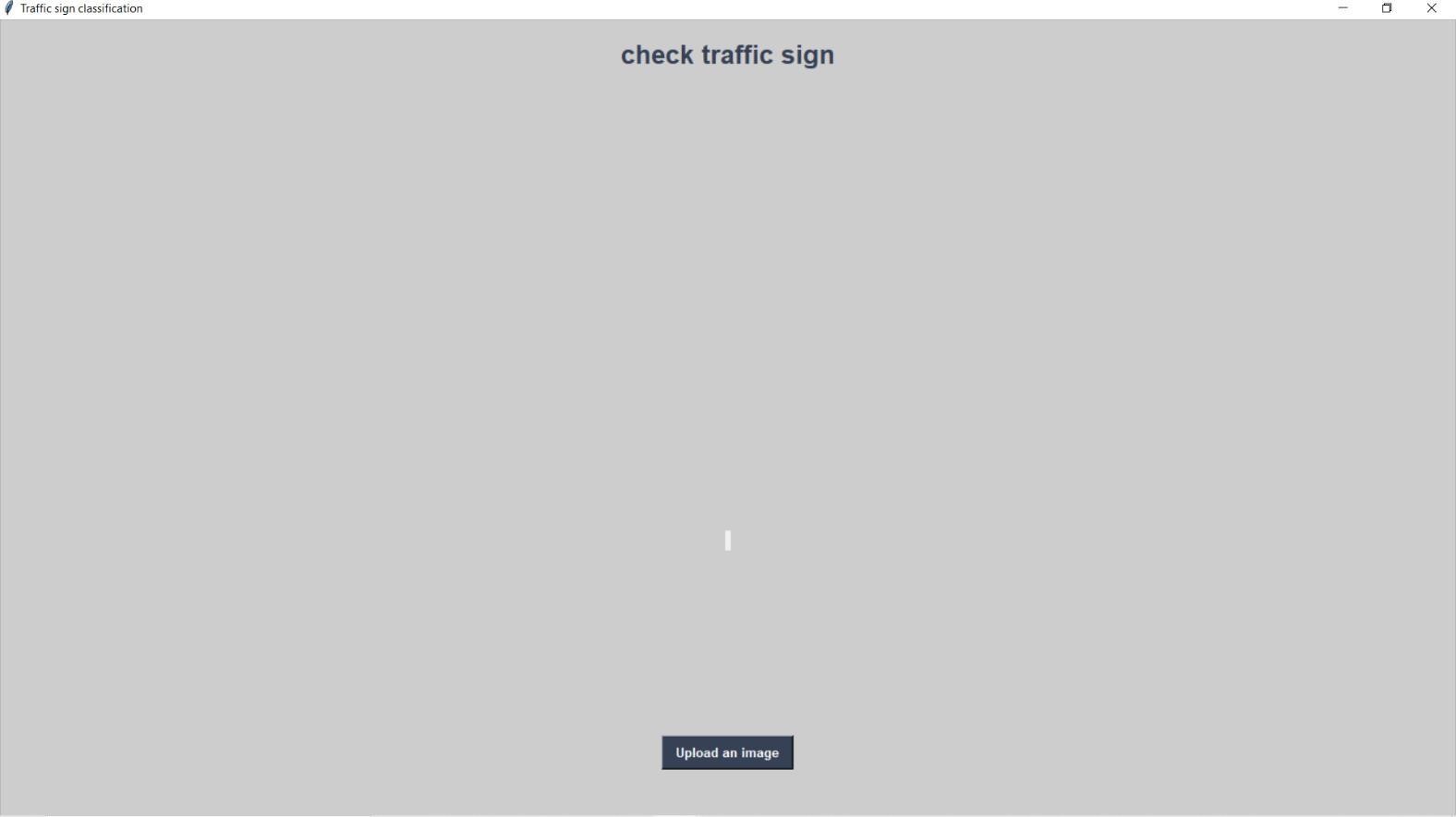
|  |  |  |
| --- | --- | --- |
| **Test Case 1: M**odel **Building** | | **Priority (H, L):** High |
| **Test Objective:** Model building recursively improves accuracy | | |
| **Test Description:** Model building recursively improves accuracy | | |
| **Requirements Verified:** Yes | | |
| **Test Environment:** Python IDLE | | |
| **Test Setup/Pre-Conditions:** | | |
| Actions | Expected Results | |

Built successfully Built successfully

|  |  |  |
| --- | --- | --- |
| **Pass: Yes** | **Conditions pass: Yes** | **Fail**: No |
| **Problems / Issues:** NIL |  |  |

**8.5.2 User Interface**

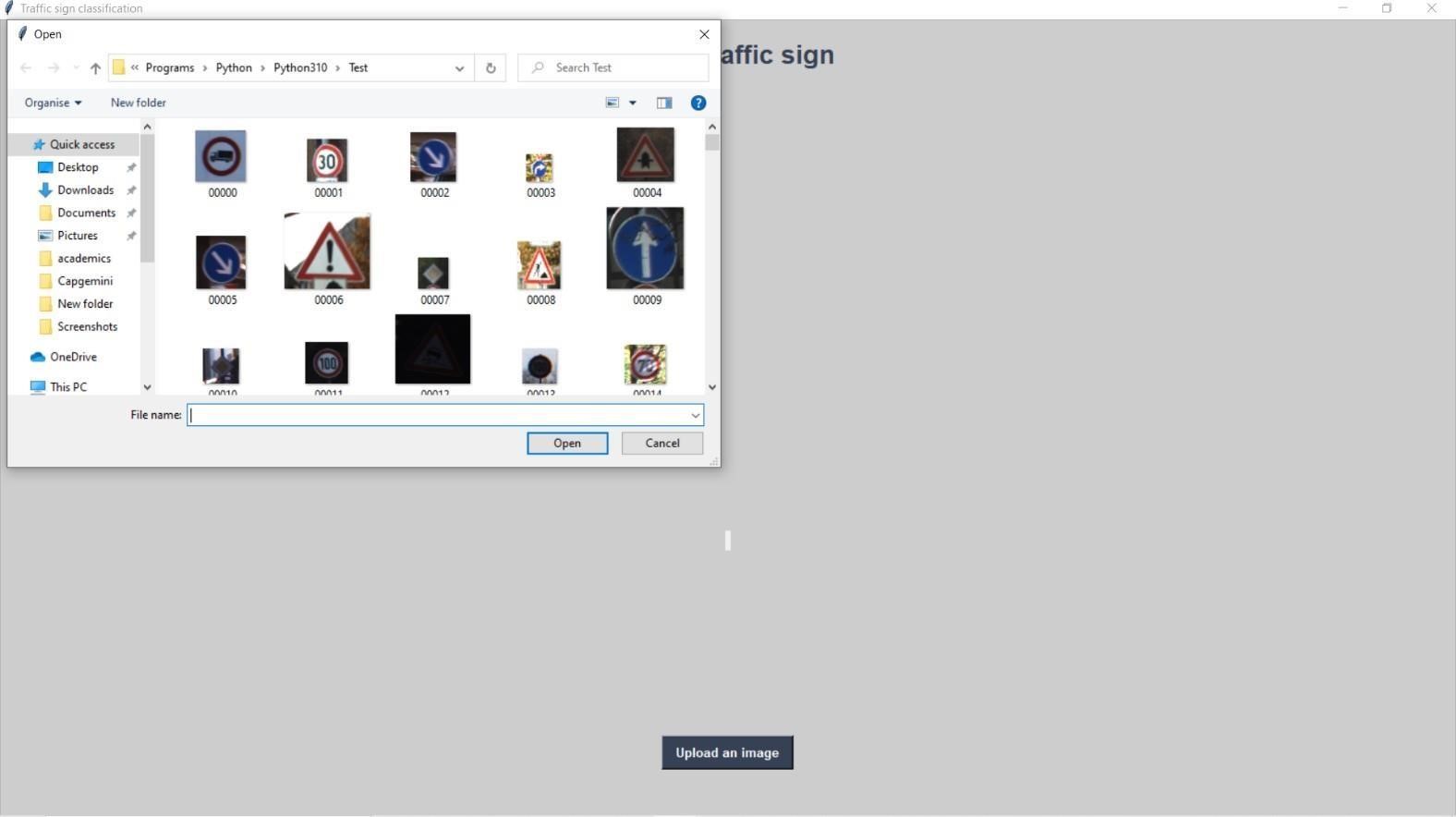
|  |  |  |
| --- | --- | --- |
| **Test Case 2: User Interface building** | | **Priority (H, L):** High |
| **Test Objective:** User should see a user interface | | |
| **Test Description:** User should be able to click on button to add an image | | |
| **Requirements Verified:** Yes | | |
| **Test Environment:** Python IDLE | | |
| **Test Setup/Pre-Conditions:** | | |
| Actions | Expected Results | |
| User interface built | User interface built | |
| **Pass: Yes Conditions pass: Yes Fail**: No | | |
| **Problems / Issues:** NIL | | |
| **Notes**: Successfully Executed | | |



**Fig. 8.5.2.1 : User Interface**

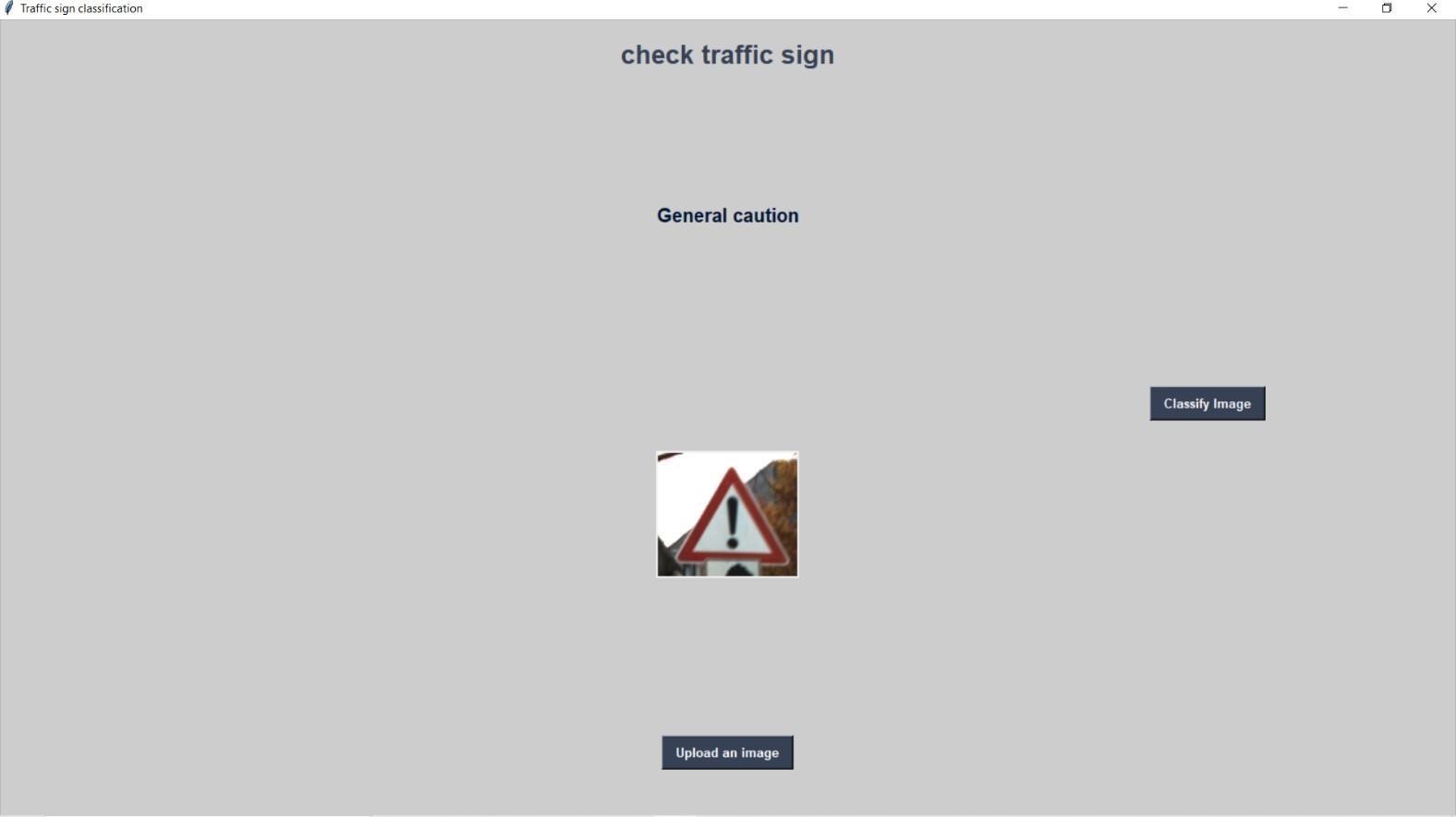
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case 3: User Interface building** | | | **Priority (H, L):** High |
| **Test Objective:** User should be able to upload an image | | |  |
| **Test Description:** User clicks button to upload image | | |  |
| **Requirements Verified:** Yes | | |  |
| **Test Environment:** Python IDLE | | |  |
| **Test Setup/Pre-Conditions:** | | | |
| Actions |  | Expected Results | |
| Image uploaded |  | Image uploaded | |
| **Pass: Yes** | **Conditions pass: Yes** | **Fail**: No | |
| **Problems / Issues:** NIL |  |  | |
| **Notes**: Successfully Executed |  |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Pass: Yes** | **Conditions pass: Yes** |  | **Fail**: No |
| **Problems / Issues:** NIL |  |  |  |
| **Notes**: Successfully Executed |  |  |  |



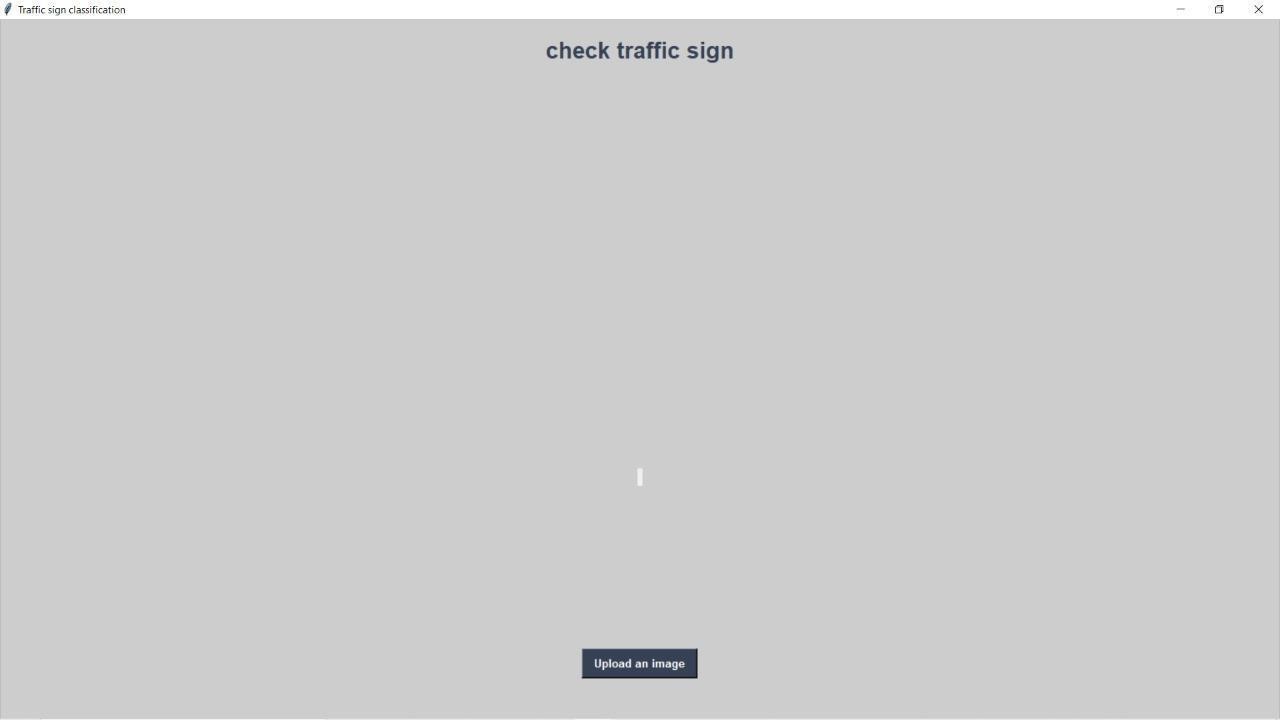
**Fig. 8.5.2.2 : Uploading an image**

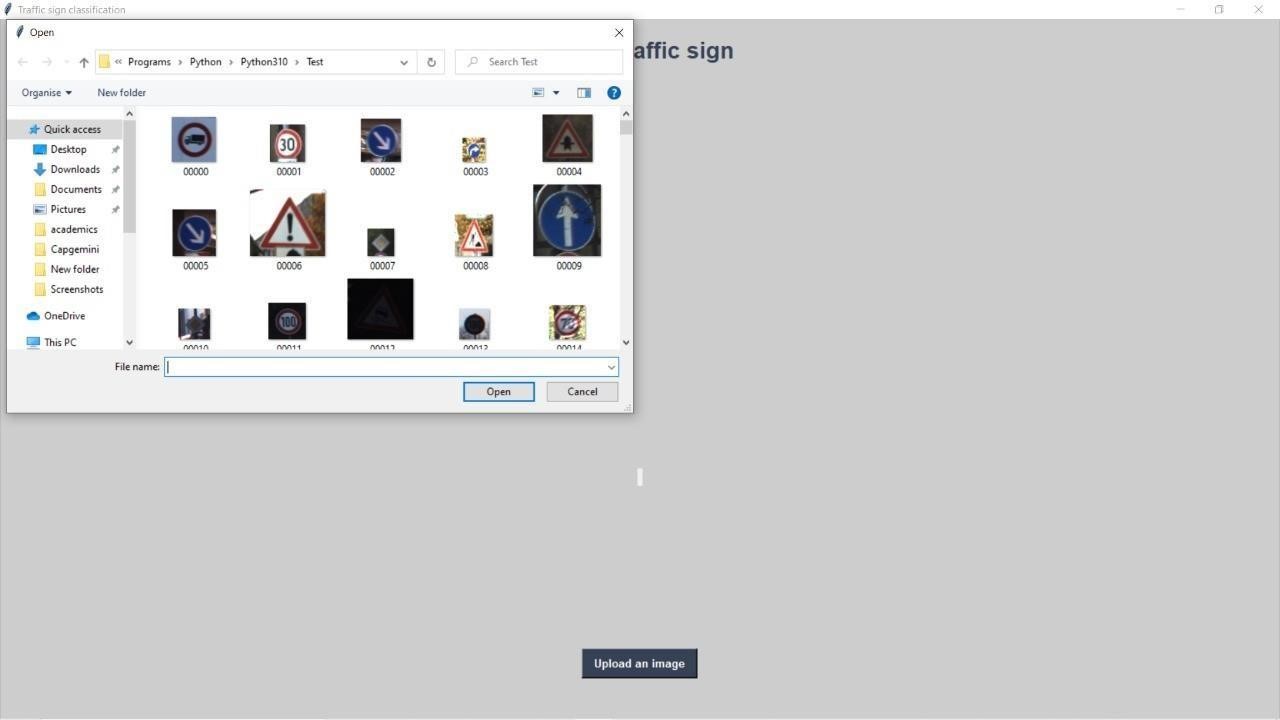
|  |  |  |
| --- | --- | --- |
| **Test Case 3: Classify the image** |  | **Priority (H, L):** High |
| **Test Objective:** Classifying the image |  | |
| **Test Description:** Classifying the image |  | |
| **Requirements Verified:** Yes |  | |
| **Test Environment:** Python IDLE |  | |
| **Test Setup/Pre-Conditions:** |  | |
| Actions | Expected Results | |
| Image classified | Image classified | |
| **Pass: Yes Conditions pass: Yes** | **Fail**: No | |
| **Problems / Issues:** NIL |  | |
| **Notes**: Successfully Executed |  | |



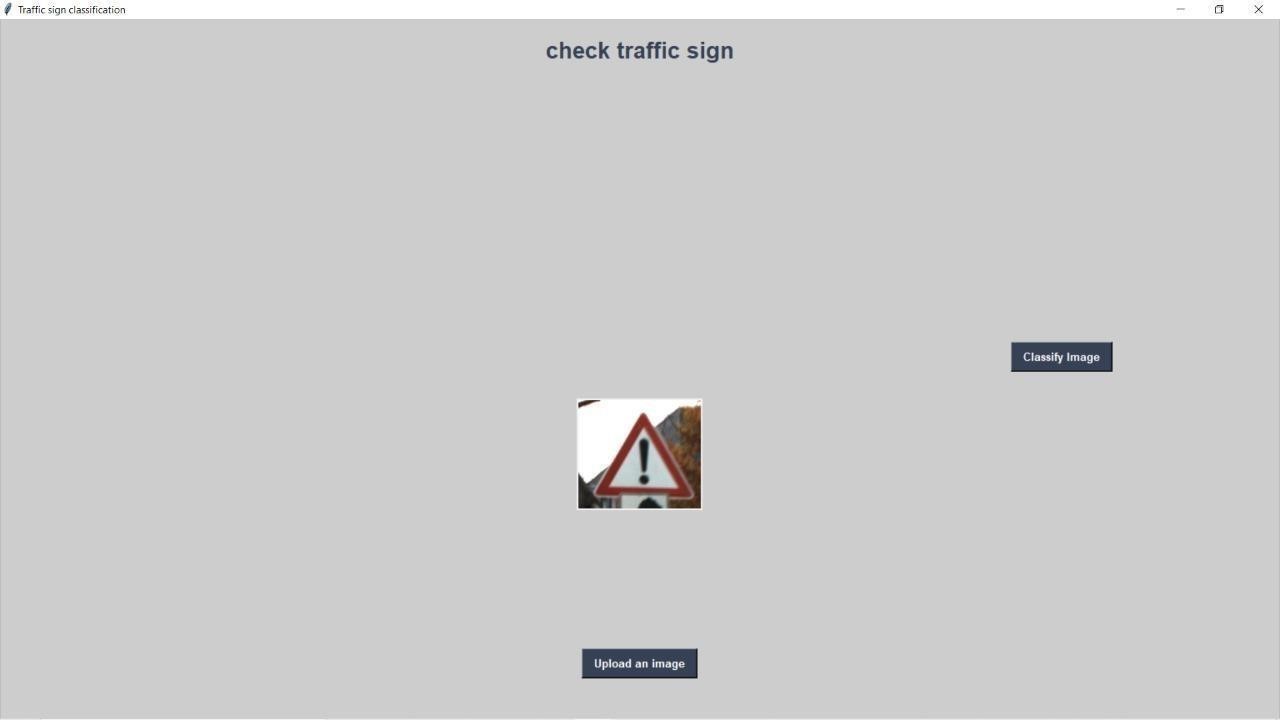
**Fig. 8.5.2.3: Classify the image**

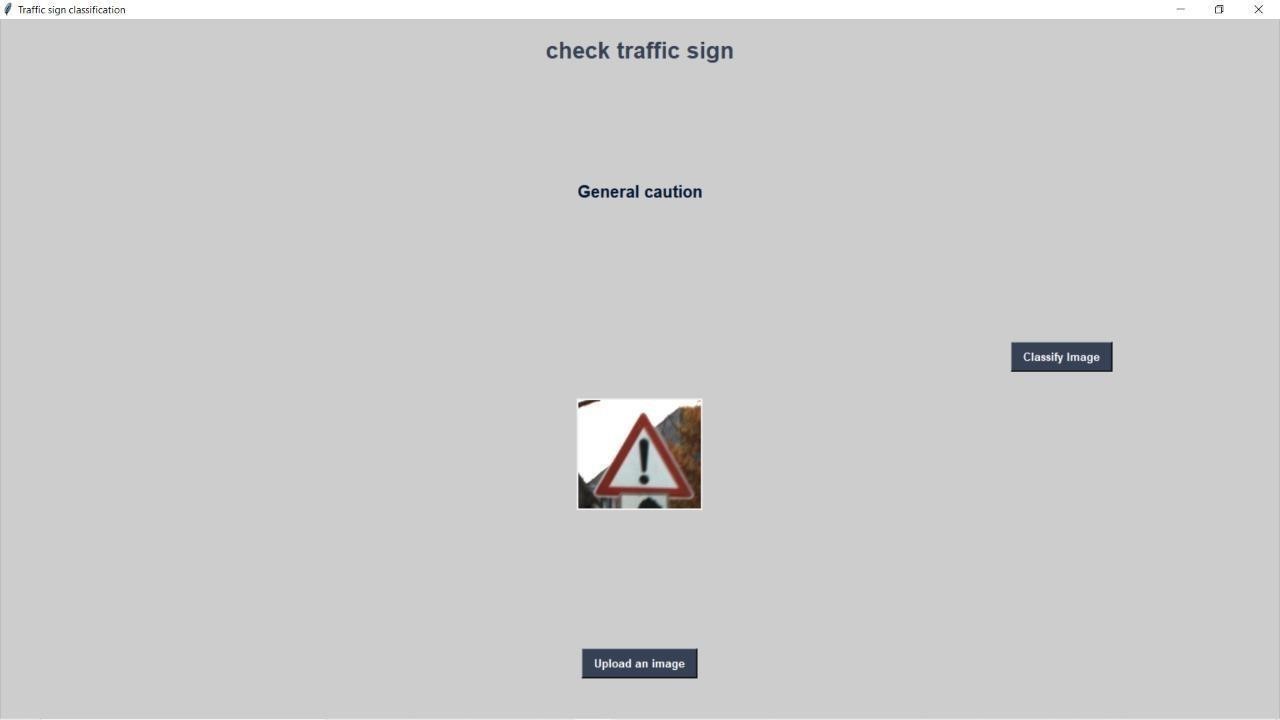
# 9. EXECUTION SCREENS



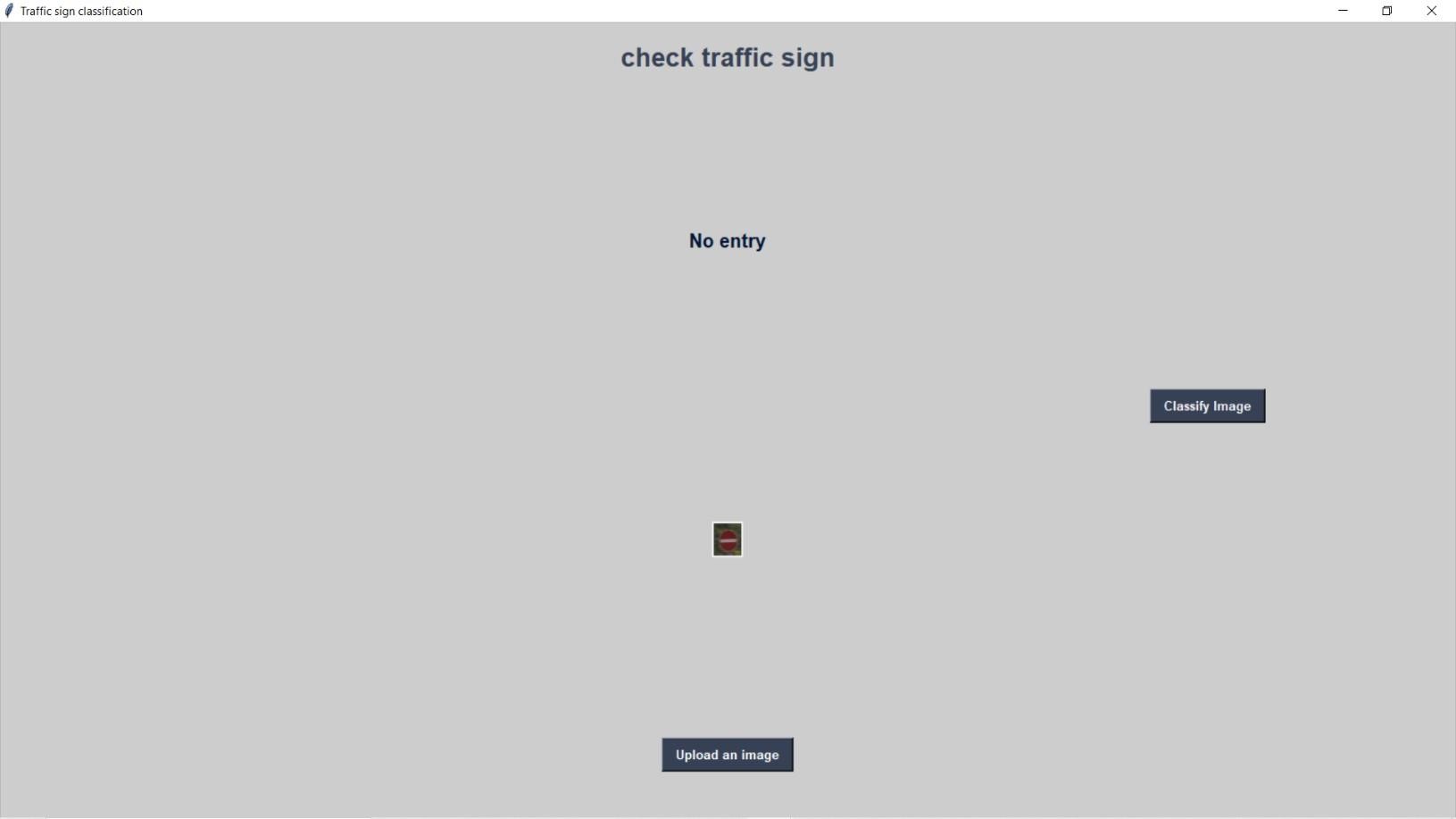


##### Fig 9.1:uploading image

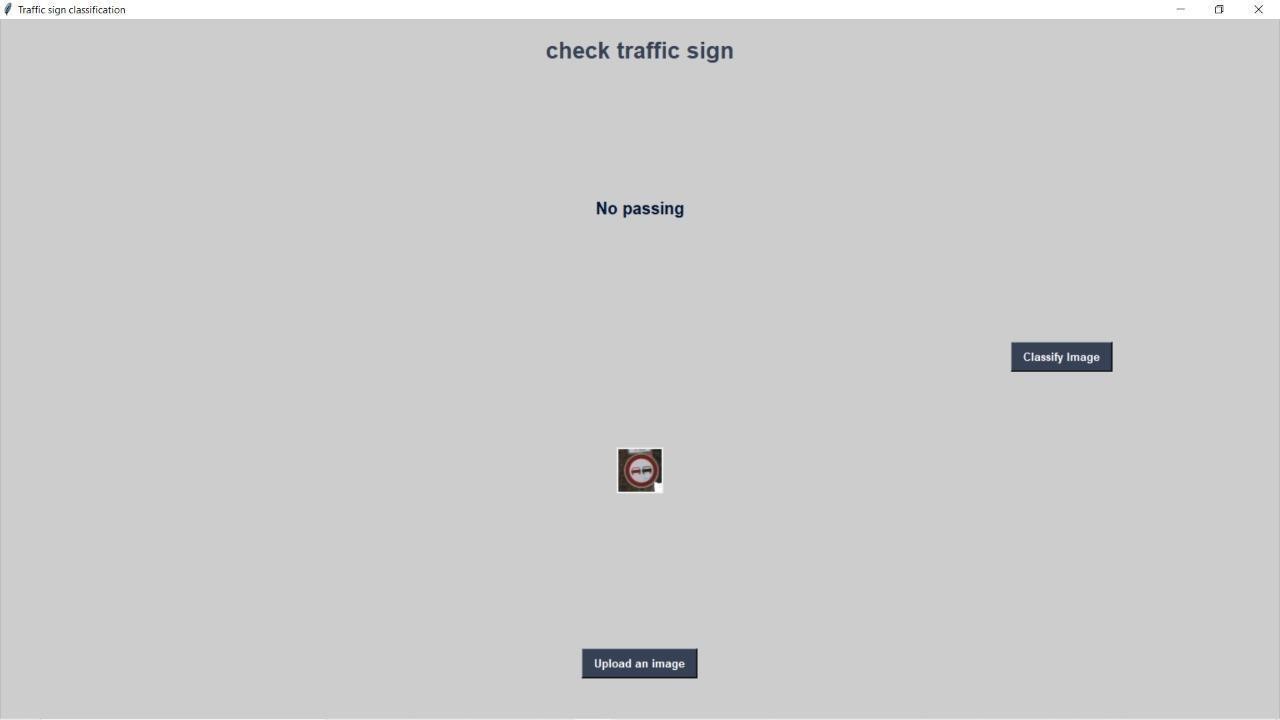




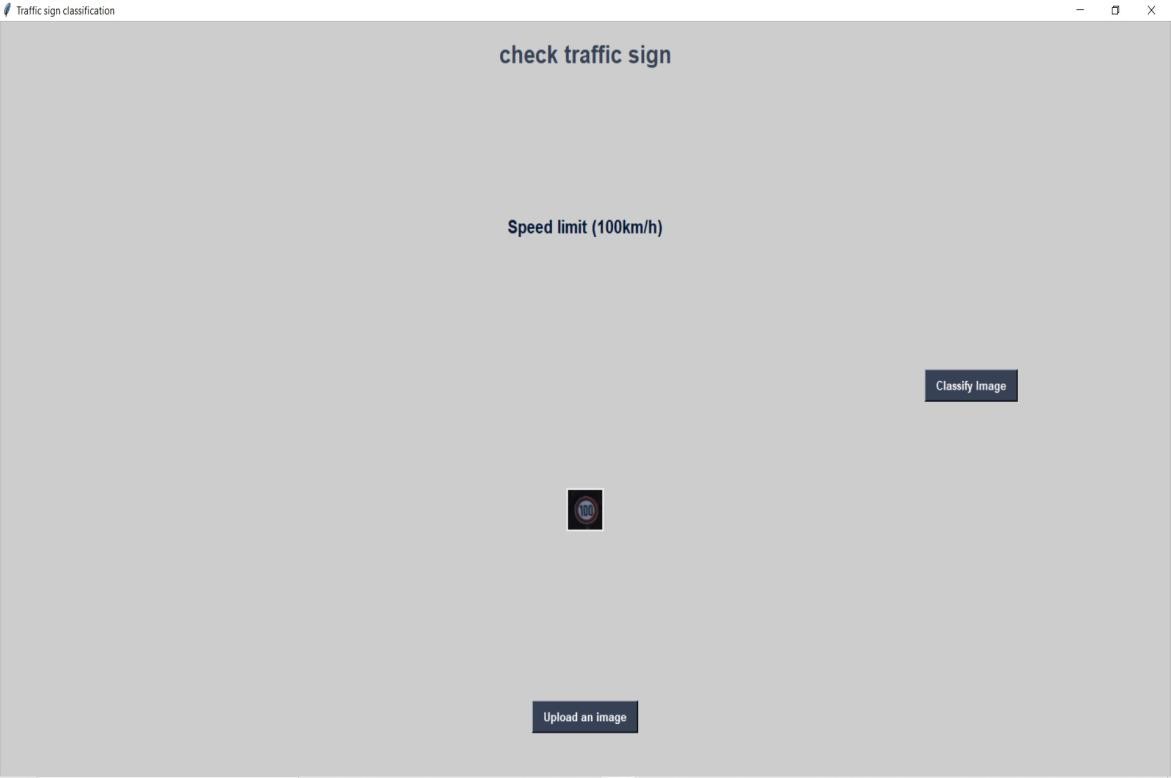
**Fig 9.2:General caution sign**



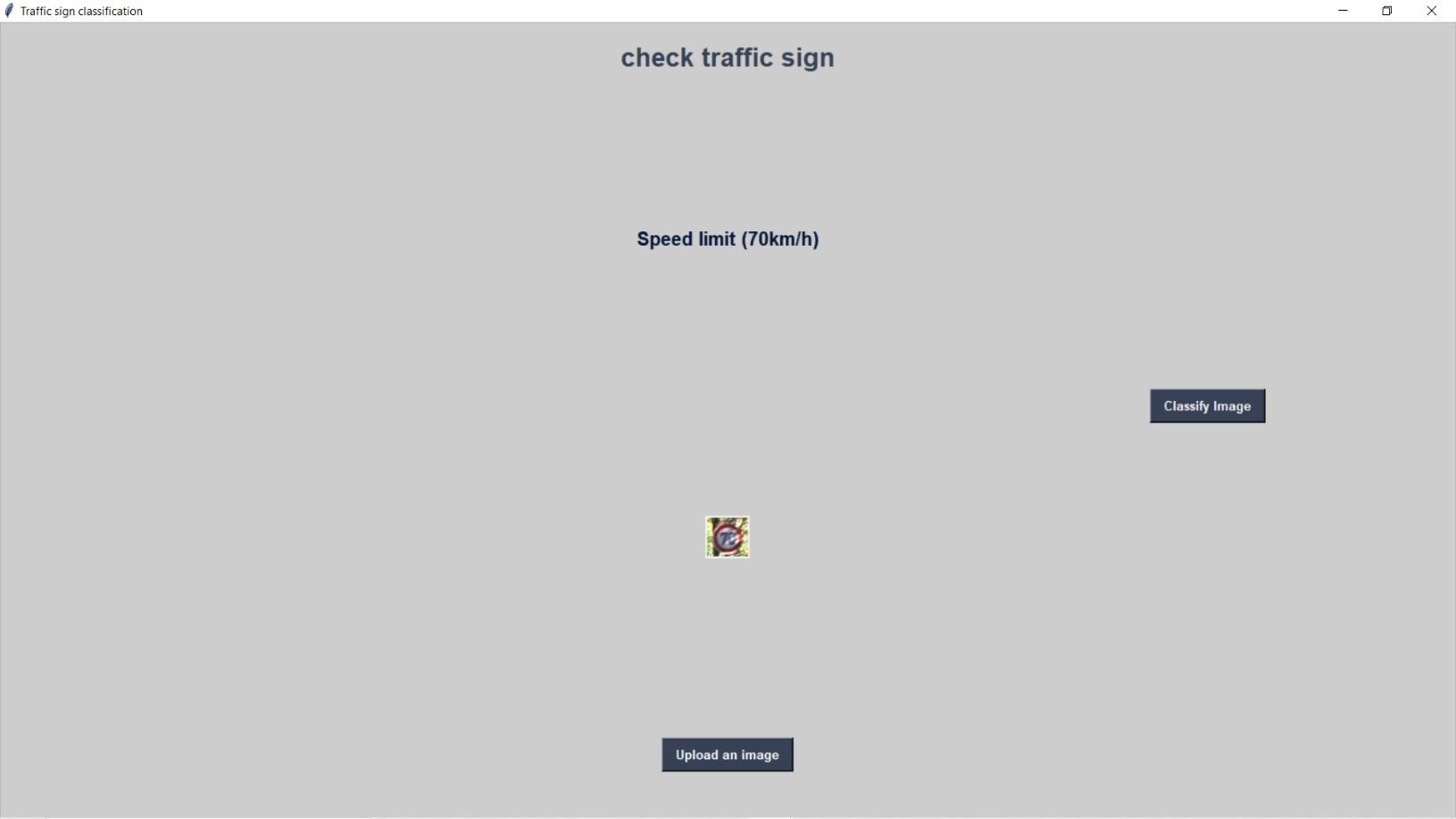
##### Fig 9.3:No entry sign



**Fig 9.4:No passing sign**



##### Fig 9.5:Speed limit(100km/h) sign



**Fig 9.6:Speed limit(70km/h) sign**

# 10. CONCLUSION & FUTURE SCOPE

**10.1 CONCLUSION:**

We have successfully implemented a Convolutional Neural Network to the Traffic Sign Recognition task with greater than 90% accuracy on average. We have covered how deep learning can be used to classify traffic signs with high accuracy, employing a variety of preprocessing and visualization techniques and trying different model architectures. We built a simple easy to understand CNN model to recognize road traffic signs accurately. Our model reached close to close to 90% accuracy on the test set which is good considering limitation of computational power and with a fairly simple architecture. There is still much work to be done, link including modern Deep Learning systems which use more recent and more complicated architectures like Google Net or ResNet. But obviously this comes in more computational cost, on the other hand.

**10.2 FUTURE SCOPE**

In Future we can add this ADAS feature to all vehicles.so that every person can save their life. we can also add some blink or sounding system for recognition purpose.so that driver may respond more quickly and take appropriate decisions based on traffic sign.

# 11. BIBLIOGRAPHY

1. Rajesh Kannan Megalingam, Kondareddy thanigundala, Sreevatsava Reddy Musani, Hemanth Nidamanuru, Lokesh Gadde 2023 “ Traffic sign recognition using Deep Learning” in International Journal of Transportation Science and Technology ( TJTST ) 2023.
2. Kaoutar Sefrioui Boujemaa, Afaf Bouhoute, Karim Boubouh and Ismail Berrada,

“Traffic sign recognition using convolutional neural networks” in International Conference on Wireless Networks and Mobile Communications (WINCOM) 2022 IEEE.

1. Amal Bouti, Mohamed Adnane Mahraz, Jamal Riffi, Hamid Tairi, “Road sign

recognition with Convolutional Neural Network” in International Conference on Intelligent Systems and Computer Vision (ISCV). 2023 IEEE

1. Prashengit Dhar, Md. Zainal Abedin, Tonoy Biswas, Anish Datta, “Traffic Sign Detection- A New Approach and Recognition Using Convolution Neural Network” in IEEE Region 10 Humanitarian Technology Conference (R10-HTC) 2022 IEEE.
2. Yann Lecun, Leon Bottou, Yoshua Bengio, Patrick Haffner, “Gradient-Based Learning Applied to Document Recognition” in PROC, Of the IEEE, November 2021.
3. Faming Shao, Xinqing Wang, \* Fanjie Meng, Ting Rui, Dong Wang, and Jian TangReal- Time Traffic Sign Detection and Recognition Method Based on Simplified Gabor Wavelets and CNNs . 2020
4. Shopa, P., Sumitha, N., & Patra, “Traffic sign detection and recognition using OpenCV” in International Conference on Information Communication and Embedded Systems (ICICES2021).