A Project Report on

GUIDEPOST AWARENESS USING CNN

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR, ANANTAPURAM.

In partial fulfillment of the requirement for the award of degree

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

AMEENUDDIN MOHAMMED	**193G1A0503**
C SREE KUMAR	**193G1A0547**
SHAIK TASLEEM	**193G1A0552**
S VARALAKSHMI	**193G1A0554**

Under the Guidance of

Mr. R HENDRA KUMAR, M. Tech.,

Asst. Professor, Department of C.S.E.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KMM INSTITUE OF TECHNOLOGY AND SCIENCES

Affiliated to JNTUA, Anantapuram, Approved by AICTE

Ramireddipalli, Tirupati-515002

Phone:(0877)2287102, www.kmmits.org

2019 – 2023

KMM INSTITUE OF TECHNOLOGY AND SCIENCES

Affiliated to JNTUA, Anantapuram, Approved by AICTE
Ramireddipalli, Tirupati-515002
Phone:(0877)2287102, www.kmmits.org
2019 – 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled "GUIDEPOST AWARENESS USING CNN"**,** is submitted By **AMEENUDDIN MOHAMMED** (193G1A0503), **C SREE KUMAR** (193G1A0547), **SHAIK TASLEEM** (193G1A0552), **S VARALAKSHMI** (193G1A0554) in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering to the Jawaharlal Nehru Technology University Anantapur. The work has been submitted under my Guidance and Supervision during the final year of the academic year 2019-2023.The project work has not submitted for the project conducted earlier by the department to the university or Institute for the award of any Degree or Diploma.

PROJECT GUIDE
Mr. R. HENDRA KUMAR, M. Tech.,
Assistant Professor,
Department of C.S.E., KMMITS

HEAD OF THE DEPARTMENT
Mr. R. HENDRA KUMAR, M. Tech.,
Assistant Professor,
Department of C.S.E., KMMITS

Attended & Submitted for the University Viva-Voice Examination on ……

EXTERNAL EXAMINAR

INTERNAL EXAMINAR

DECLARATION

We hereby **AMEENUDDIN MOHAMMED** (193G1A0503), **C SREE KUMAR** (193G1A0547), **SHAIK TASLEEM** (193G1A0552), **S VARALAKSHMI** (193G1A0554) hereby declare that the major project report entitled, **"GUIDEPOST AWARENESS USING CNN"** done by us under the guidance of DR. Y. RAVIKUMAR, Ph.D., Assistant professor., submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering, KMM Institute of Technology and Science, Tirupati affiliated to Jawaharlal Nehru Technology University, Anantapur during the academic year 2022-2023. The results embedded in this project have not been submitted to any other university or Institute for the award of Degree or Diploma.

Date :

Place:

SIGNATURE OF THE STUDENT:

AMEENUDDIN MOHAMMED	**193G1A0503**
C SREE KUMAR	**193G1A0547**
SHAIK TASLEEM	**193G1A0552**
S VARALAKSHMI	**193G1A0554**

ACKNOWLEDGEMENT

At the outset, we wish to express our sincere gratitude to our beloved and respected Chairman **Sri. S. SREENIVASULU, IRS. (Retd)** for his encouragement and blessings.

We sincerely thank **Mr. R HENDRA KUMAR, Asst. Professor, (Head of the Department Computer Science and Engineering),** for guiding and providing facilities for the successful completion of my project at **KMM INSTITUTE OF TECHNOLOGY AND SCIENCE,** Tirupati.

We express our deep sense of gratitude to **Mr. R HENDRA KUMAR, M. Tech., (Head of the Department Computer Science and Engineering),** for his valuable guidance and constant encouragement given to us during this work.

It is our privilege and pleasure to express our profound sense of respect gratitude and indebtedness to our guide **Dr. Y. RAVIKUMAR, Ph.D., (Professor, Department of Computer Science and Engineering), KMM INSTITUTE OF TECHNOLOGY AND SCIENCE,** for his indefatigable inspiration, guidance, constructive critics and encouragement for us throughout this dissertation work.

We also think all other faculty members of our department for their support and our peers for having stood by us and help to complete this project.

Last but not least, we wish to acknowledgement our parents and friends for giving moral strength and constant supports and encouragement.

MEMBERS:

AMEENUDDIN MOHAMMED	**193G1A0503**
C SREE KUMAR	**193G1A0547**
SHAIK TASLEEM	**193G1A0552**
S VARALAKSHMI	**193G1A0554**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

To be a frontier in computing technologies to produce globally competent computer science engineering graduates with moral values to build a vibrant society and nation.

MISSION

- Providing a strong theoretical and practical background in computer science engineering with an emphasis on software development.

- Inculcating professional behavior, strong ethical values, innovative research capabilities, and leadership abilities.

- Imparting the technical skills necessary for continued learning towards their professional growth and contribution to society and rural communities.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO-1: Graduates will have strong knowledge and skills to comprehend latest tools and techniques of Computer Engineering so that they can analyze, design and create computing products and solutions for real life problems.

PEO-2: Graduates shall have multidisciplinary approach, professional attitude and ethics, communication and teamwork skills, and an ability to relate and solve social issues through their Employment, Higher Studies and Research.

PEO-3: Graduates will engage in life-long learning and professional development to adapt to rapidly changing technology.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-1: Ability to grasp advanced programming techniques to solve contemporary issues.
PSO-2: Have knowledge and expertise to analyze data and networks using latest tools and technologies.
PSO-3: Qualify in national and international competitive examinations for successful higher studies and employment.

PROGRAM OUTCOMES (POS)

PO-1 Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO-2 Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-3 Design/development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations

PO-4 Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO-5 Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitatio

PO-6 The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO-7 Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO-8 Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO-9 Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO-10 Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO-11 Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO-12 Life-Long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

ABSTRACT

The number of accidents caused by failure to observe traffic signs and obey traffic laws has been steadily increasing. When driving, you will notice numerous traffic signs such as traffic indicators, turn left or right, speed restrictions, no passing of heavy vehicles, no entering, children crossing, and so on that you must obey in order to drive safely. By utilizing synthesis training data generated from road sign photos, we can solve the issues associated with road mark identification databases. The primary goal is to create a road mark identification and detection system with high recognition accuracy and performance capability in training and detection processes. We used approaches like as neural networks and feature extraction to solve the shortcomings of existing methods, increase the effectiveness of diagnosing road Hoarding, and reduce roadway casualties.

Keywords- Gesture Recognition, Indication Identification, Hoarding Awareness.

Outcomes:

Our project titled "**GUIDEPOST AWARENESS USING CNN**" is mapped with the following outcomes:

Program Outcomes **:** PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO8, PO9, PO10, PO11, PO12.

Program Specific Outcomes **:** PSO1, PSO2, PSO3.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF OUPUT SCREENSHOTS

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Project Overview

Road and traffic signs considered in this thesis are those that use a visual/symbolic language about the road(s) ahead that can be interpreted by drivers. The terms are used interchangeably in this thesis, and elsewhere might also appear in combination, as “road traffic signs”. They provide the driver with pieces of information that make driving safe and convenient. A type of sign that is NOT considered in this thesis is the direction sign, in which the upcoming directions for getting to named towns or on numbered routes are shown not symbolically but essentially by text.

Road and traffic signs must be properly installed in the necessary locations and an inventory of them is ideally needed to help ensure adequate updating and maintenance. Meetings with the highway authorities in both Scotland and Sweden revealed the absence of but a need for an inventory of traffic signs. An automatic means of detecting and recognizing traffic signs can make a significant contribution to this goal by providing a fast method of detecting, classifying and logging signs. This method helps to develop the inventory accurately and consistently. Once this is done, the detection of disfigured or obscured signs becomes easier for human operator.

Road and traffic sign recognition is the field of study that can be used to aid the development of an inventory system (for which real-time recognition is not required) or aid the development of an in-car advisory system (when real-time recognition is necessary). Both road sign inventory and road sign recognition are concerned with traffic signs, face similar challenges and use automatic detection and recognition. A road and traffic sign recognition system could in principle be developed as part of an Intelligent Transport Systems (ITS) that continuously monitors the driver, the vehicle, and the road in order, for example, to inform the driver in time about upcoming decision points regarding navigation and potentially risky traffic situations. Figure 1.1 depicts these relationships among the three fields.

Road sign recognition has become a major challenging field in academic as well as in industry. The major application of this systems can be mostly used in the

upcoming Artificial Intelligence (AI) world to understand environment and also being one of the most important part in Advance Driver Assistance Systems (ADAS). Majorly, recognition process provides clear visual definition on the fact that every traffic sign may consists of different kinds of shapes, colors and graphics that will represent the environment and road constrains, which enables driver to recognize and prepare, for what he/she may face.

Road Signs are not expected to be highly visible to drivers, there may be little confusion between signs or may be some signs are not recognizable by few drivers. This may lead to confusion and more chance for occurrence of accidents. If confusion occurs in recognizing warning signs, then it may be dangerous. However, because of different environmental situations, there are few cases where the road signs will get completely perverted, making their appearance challenging for humans and machines.

ITS focuses on integrating information technology into transport infrastructure and vehicles. These systems can include road sensors, in-vehicle navigation services, electronic message signs, and traffic management and monitoring. The aim of intelligent transport systems is to increase transportation efficiency, road safety and to reduce the environmental impact with the use of advanced communication technologies [1, 2] . This thesis aims to develop a system to recognise and classify road and traffic signs for the purpose of developing an inventory which could assist the highway authorities to update and maintain the traffic signs. It is based on taking images by a camera from a moving vehicle and invoking colour segmentation, shape recognition, and classification to detect the signs in these images.

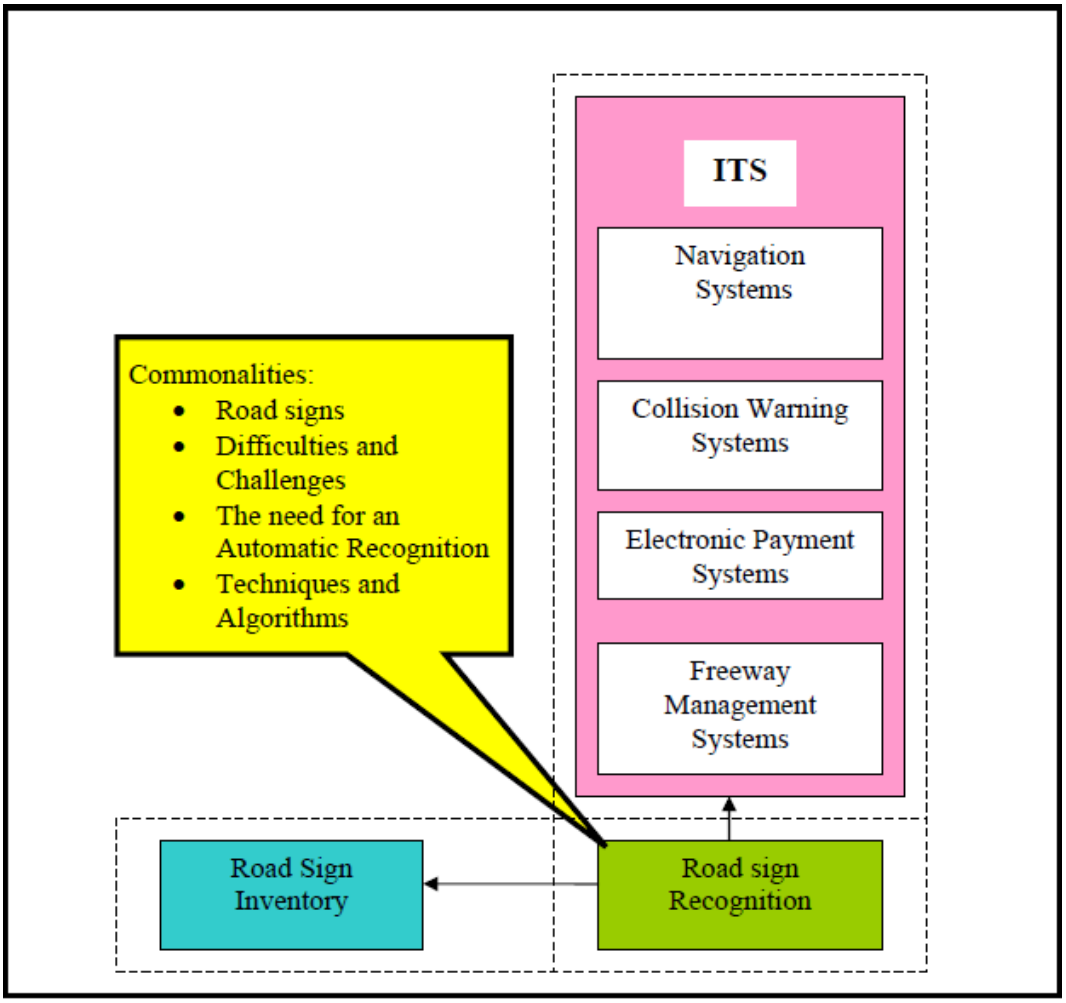


Figure 1.1: The relationship among Road Sign Inventory, Road Sign Recognition and ITS.

ACRONYMS

CNN	Convolution Neural Network
RGB	Red Green Blue
OS	Operating System
PIL	Python Image Library
GUI	Graphical User Interface
ADAS	Advance Driving Assistance
ROI	Region Of Interest
GPU	Graphical Processing Unit

1.2 Project Deliverables

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.

The main strategy is to find the right combination of colors in the scene so that one color is located inside the convex hull of another color and combine this with the right shape.

If a candidate is found, the system tries to classify the object according to the pictogram combination and give the result of this classification.

The fact is that the pattern of sign is always the same, the only change is the view that a recognition camera or the driver sees and also the environment which makes difficult to recognize, this contribute the different variations in recognition, this can be achieved by updating different conditions of traffic signs in the training set.

Our method doesn't require real time data because we used training data set which consists of the photographs that are very similar to the real-world data. The objective of using this data set is to increase our detecting precision as more as possible which should be greater than 90% close to actual observation.

The difficulties found with the Road sign collection could be eliminated by the utilization of different artificially generated data without must burden on the classifier to recognize.

So, whenever a sign is scanned it may not be same as the real-world sign but our method uses CNN to recognize any similarities between them and classifies as the same sign

The objectives are thus:

1. To understand the properties of road and traffic signs and their implications for image processing for the recognition task.
2. To understand color, color spaces and color space conversion.
3. To develop robust color segmentation algorithms that can be used in a wide range of environmental conditions.
4. To develop a recognizer that is invariant to in-plane transformations such as translation, rotation, and scaling based on invariant shape measures.
5. To identify the most appropriate approach for feature extraction from road signs.
6. To develop an appropriate road sign classification algorithm.
7. To evaluate the performance of the aforementioned methods for robustness under different conditions of weather, lighting geometry, and sign.

Fig 1.2 Traffic Signs

1.3 Project Scope

There are several different types of traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc. Traffic signs classification is the process of identifying which class a traffic sign belongs to.

In this Python project example, we will build a deep neural network model that can classify traffic signs present in the image into different categories. With this model, we are able to read and understand traffic signs which are a very important task for all autonomous vehicles.

Road sign recognition has become a major challenging field in industry. The major application of this systems can be mostly used in the upcoming Artificial Intelligence (AI) world to understand environment and also being one of the most important parts in Advance Driver Assistance Systems (ADAS). Majorly, recognition process provides clear visual definition on the fact that every traffic sign may consists of different kinds of shapes, colors and graphics that will represent the environment and road constrains, which enables driver to recognize and prepare, for what he/she may face.

Road Signs are not expected to be highly visible to drivers, there may be little confusion between signs or may be some signs are not recognizable by few drivers. This may lead to confusion and more chance for occurrence of accidents. If confusion occurs in recognizing warning signs, then it may be dangerous. However, because of different environmental situations, there are few cases where the road signs will get completely perverted, making their appearance challenging for humans and machines.

The dataset contains more than 50,000 images of different traffic signs. It is further classified into 43 different classes. The dataset is quite varying, some of the classes have many images while some classes have few images. The size of the dataset is around 300 MB. The dataset has a train folder which contains images inside each class and a test folder which we will use for testing our model.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

In today's world, identification of traffic signs has become an important aspect of our lives. Looking at the increasing traffic, to ensure safety of all and for automatic driving in the future, traffic sign classification is utmost necessary.

Considerable research has been done around recognition of traffic and road signs. In 1987, the first research on the topic "Traffic Sign Recognition" was done by Akatsuka and Imai , where they tried to build a fundamental system that could recognize traffic signs and alert the drivers and ensure his/her safety. But this was used to provide the automatic recognition for only some specific traffic signs.

Traffic sign recognition initially appeared in the form of only speed limit recognition in 2008. These symbols could only detect the circular speed limit signs. On the other hand, later, systems were designed that performed detection on overtaking signs. This technology was available in the Volkswagen Phaeton and in the 2012 in Volvo S80, V70 and many more. But the major drawback of these systems was that they could not detect the city limit signs as they were mostly in the form of direction signs. But nowadays, such systems are expected to be present in the future cars to help drivers while driving.

In [1], the authors used the color processing system to reduce the effect of brightness and shadow on the images. This was the very first research done on this topic by the authors, Akatsuka and Imai.

In [2], the authors have done a survey on traffic sign detection and recognition, where HOG (Histogram of Oriented Gradients) is used for classification purpose.

In [3], a complete study of different traffic sign recognition algorithms has been done, where the highest accuracy (99.46%) was obtained by MCDNN (Multi-column Deep Neural Network).

In [4], the authors have developed a model where they are converting the images to gray scale first and then filter those images using simplified Gabor wavelets. The gabor

filters are used to extract features. These wavelets are important as they help to minimize the product of its standard deviation in both the time and frequency mapping.

Fig 2.1 Traffic Signs

The authors extracted the regions of interest for recognition purpose and classified the signs using “Support Vector Machine” (SVM).

In [5], the authors have extracted the regions of interest in the detection stage and further examined the shapes of such regions. Here, in the classification system, they took the regions of interest and classified them into different classes.

In [6], the authors have created a module which consists of numerous convolutions. They combined the 1*3 kernel and 3*1 kernel and finally linked it with the 1*1 kernel to attain the 3*3 kernel. This was used to extract more features and thus reduce the number of parameters.

In [7], the author has reviewed the traffic sign detection methods and divided them into 3 types of methods: color, shape and learning based methods.

In [8], the author used the number of peaks algorithm to detect and recognize circular shaped traffic signs.

In [9], the authors have tried creating a classification model using the Enhanced LeNet-5 architecture, which consists of two consecutive convolution layers (before the Max Pooling layer) to extract high level features from the image. Also, they have used the data augmentation technique to make the dataset stable.

In [10], the authors have used the technique of color segmentation and the RGB based detection which is used to identify the traffic signs on the road. The optimizer used was "Stochastic Gradient Descent" with Nesterov Momentum. The text to speech system was implemented to alert the driver about the traffic sign. Also, they utilized the GPU (graphical processing unit) system, as part of hardware.

In [11], the authors have tried generating a dataset for the Arabic Road signs and thus develop a CNN model for Arabic sign recognition.

DETECTION USING CNN ENSEMBLE

The method described by Shustanov, P. Yakimov used for Road Sign Detection and Recognition is image processing technique which consist of a group of (CNN) for the recognition called as ensemble. The recognition rate for the CNN is very high, which makes it more desirable for various computer-based vision tasks. The method used for the execution of CNN is TensorFlow. The members of this paper achieved more than 99 percent of accuracy for circular signs on using German data sets.

RECOGNITION USING COLOR SEGMENTATION

Wali etal describes how they have used to implement a novel method for sign recognition. They used advanced ARK-2121 technology which is small computer which they installed this tech on the car. The major techniques in the recognition step of the sign were SVM and HOG. They achieved an accuracy of 91% in detection and about 98% average on the classification process.

THE GERMAN TRAFFIC SIGN RECOGNITION.

R. Qian etal describes the analysis and design process of “German Traffic Sign Recognition Benchmark” dataset. The outputs of this project showed that algorithms of machine learning showed very well in recognition of traffic signs. The participants got a very good percentage of 98.98 recognition rate which is as high as human perfection on these datasets.

Summary Of Literature Survey

The first research on traffic sign recognition can be traced back to 1987; Akatsuka and Imai [2] attempted to make an early traffic sign recognition system. A system capable of automatic recognition of traffic sign could be used as assistance for drivers, alerting them about the presence of some specific sign (e.g., a one-way street) or some risky situation (e.g., driving at a higher speed than the maximum speed allowed).

Traffic sign recognition initially appeared in the form of only speed limit recognition in 2008. These symbols could only detect the circular speed limit signs. On the other hand, later, systems were designed that performed detection on overtaking signs. This technology was available in the Volkswagen Phaeton and in the 2012 in Volvo S80, V70 and many more. But the major drawback of these systems was that they could not detect the city limit signs as they were mostly in the form of direction signs. But nowadays, such systems are expected to be present in the future cars to help drivers while driving.

It also can be used to provide the autonomous unmanned some specific-designed signs. Generally, the procedure of a traffic sign recognition system can be roughly divided of two stages namely detection and classification.

CHAPTER 3

PROBLEM ANALYSIS

3. PROBLEM ANALYSIS

3. 1. Existing System

A traffic sign detection and recognition system can be used to inform a driver about the approaching traffic signs and informs the driver via a visual or auditory aid. A TSDR system can also be used in various other application to reduce manual work and increase efficiency. Some application of such a system include:

1. Highway maintenance: Currently, a human operator has to watch a videotape to check the presence and condition of the signs. It is a tedious & wearisome task because of the amount of manual work to be done and depended on.

2. Sign inventory: Similar to highway maintenance, sign inventory keeps track of positions and placement of traffic signs in cities and towns. In this case, it is more difficult than highways. The signs are not always placed perpendicular to the movement of the vehicles, producing a deformed image of the signs; besides, there are occlusions, and other objects with the same color.

3. Driver Support Systems. Traffic sign detection and classification is one of the main fields of Driver Support Systems. The general idea is to support the driver in some tasks, allowing him or her to concentrate in driving.

Autonomous machines/ vehicles are all set to drive along the road. As roads get denser with other vehicles, it is difficult for drivers as well as autonomous cars to identify all the traffic signs on the path.

Chances of missing out on crucial signs that may lead to fatal accidents cannot be neglected.

To assists drivers and autonomous cars to overcome this problem camera-based traffic sign recognition systems are used as a part of the advanced driver assistance system (ADAS).

Some of the reasons to miss out on these traffic sign

- Poor illumination
- Traffic density
- Lack of concentration

Poor illumination and traffic density, two of the major reasons for missing a traffic sign can be solved by using a robust traffic sign recognition system. However, lack of concentration is something which can be solved by taking two approaches, one is using a driving monitoring solution that monitors the driver for any in-attention and alerts on same, the second approach is similar to other two reasons, wherein a traffic sign recognition system solves the problem.

A traffic sign recognition system is made of a camera that is front facing with a wide field of view covering the entire road for any signs commissioned by traffic regulatory bodies. An embedded platform enabled with algorithms that can recognize any traffic sign.

In general practice, these algorithms take a CNN-based approach rather than a classic image processing technique, which helps the system to identify and classify a diverse set of signs

A self-driving car (sometimes called an *autonomous car* or *driverless car*) is a vehicle that uses a combination of sensors, cameras, radar and artificial intelligence to travel between destinations without a human operator.

To qualify as fully autonomous, a vehicle must be able to navigate without human intervention to a predetermined destination over roads that have not been adapted for its use.

Companies developing and/or testing autonomous cars include Audi, BMW, Ford, Google, General Motors, Tesla, Volkswagen and Volvo. Google's test involved a fleet of self-driving cars -- including Toyota Prii and an Audi TT -- navigating over 140,000 miles of California streets and highways.

How self-driving cars work

AI technologies power self-driving car systems. Developers of self-driving cars use vast amounts of data from image detection systems, along with machine learning and neural networks, to build systems that can drive autonomously.
The neural networks identify patterns in the data, which is fed to the machine learning algorithms. That data includes images from cameras on self-driving cars from which the neural network learns to identify traffic lights, trees, curbs, pedestrians, street signs and other parts of any given driving environment.

For example, Google's self-driving car project, called Waymo, uses a mix of sensors, lidar (light detection and ranging -- a technology similar to RADAR) and cameras and combines all of the data those systems generate to identify everything around the vehicle and predict what those objects might do next. This happens in fractions of a second. Maturity is important for these systems. The more the system drives, the more data it can incorporate into its deep learning algorithms, enabling it to make more nuanced driving choices.

The following outlines how Google Waymo vehicles work:

The driver (or passenger) sets a destination. The car's software calculates a route. A rotating, roof-mounted Lidar sensor monitors a 60-meter range around the car and creates a dynamic three-dimensional (3D) map of the car's current environment.

A sensor on the left rear wheel monitors sideways movement to detect the car's position relative to the 3D map. Radar systems in the front and rear bumpers calculate distances to obstacles.

AI software in the car is connected to all the sensors and collects input from Goggle Street View and video cameras inside the car. The AI simulates human perceptual and decision-making processes using deep learning and controls actions in driver control systems, such as steering and brakes.

The car's software consults google view for advance notice of things like landmarks, traffic signs and lights. An override function is available to enable a human to take control of the vehicle

Cars with self-driving features

Google's Waymo project is an example of a self-driving car that is almost entirely autonomous. It still requires a human driver to be present but only to override the system when necessary. It is not self-driving in the purest sense, but it can drive itself in ideal conditions. It has a high level of autonomy.

Many of the cars available to consumers today have a lower level of autonomy but still have some self-driving features. The self-driving features that are available in many production cars as of 2019 include the following:

Hands-free steering centers the car without the driver's hands on the wheel. The driver is still required to pay attention.

Adaptive Cruise Control (ACC) down to a stop automatically maintains a selectable distance between the driver's car and the car in front.

Lane-Centering Steering intervenes when the driver crosses lane markings by automatically nudging the vehicle toward the opposite lane marking.

Levels of autonomy in self-driving cars

The U.S. National Highway Traffic Safety Administration (NHTSA) lays out six levels of automation, beginning with Level 0, where humans do the driving, through driver assistance technologies up to fully autonomous cars.

Here are the five levels that follow Level 0 automation:

Level 1: An advanced driver assistance system (ADAS) aid the human driver with steering, braking or accelerating, though not simultaneously. An ADAS includes rearview cameras and features like a vibrating seat warning to alert drivers when they drift out of the traveling lane.

Level 2: An ADAS that can steer and either brake or accelerate simultaneously while the driver remains fully aware behind the wheel and continues to act as the driver.

Level 3: An automated driving system (ADS) can perform all driving tasks under certain circumstances, such as parking the car. In these circumstances, the human driver must be ready to retake control and is still required to be the main driver of the vehicle.

Level 4: An ADS can perform all driving tasks and monitor the driving environment in certain circumstances. In those circumstances, the ADS is reliable enough that the human driver needn't pay attention.

Level 5: The vehicle's ADS acts as a virtual chauffeur and does all the driving in all circumstances. The human occupants are passengers and are never expected to drive the vehicle.

Race car -the mini autonomous vehicle kit- was originally developed for a competition by MIT in 2015, then updated in 2016 and used for robotics training. The vehicle was developed on a 1/10 scale based on the Traxxas RC Rally Car racing car. The car kit uses an open-source electronic speed controller called VESC. The main processor is the Nvidia Jetson TX2 developer board with 256 CUDA core

3.1.1 Challenges

The previous methods used for designing traffic sign recognition model are
1) K-means clustering
2) Lidar and vision based
3) Video streaming

DISADVANTAGES OF EXSISTING METHODS
I. False Detection
II. Redundancy (inter pixel redundancy)
III. Less efficiency (compared to our model)
IV. Cost related issues

3.2 Proposed System

Road and traffic signs must be properly installed in the necessary locations and an inventory of them is ideally needed to help ensure adequate updating and maintenance. Meetings with the highway authorities in both Scotland and Sweden revealed the absence of but a need for an inventory of traffic signs. An automatic means of detecting and recognising traffic signs can make a significant contribution to this goal by providing a fast method of detecting, classifying and logging signs. This method helps to develop the inventory accurately and consistently. Once this is done, the detection of disfigured or obscured signs becomes easier for human operator.

Road and traffic sign recognition is the field of study that can be used to aid the development of an inventory system (for which real-time recognition is not required) or aid the development of an in-car advisory system (when real-time recognition is necessary). Both road sign inventory and road sign recognition are concerned with traffic signs, face similar challenges and use automatic detection and recognition.

A road and traffic sign recognition system could in principle be developed as part of an Intelligent Transport Systems (ITS) that continuously monitors the driver, the vehicle, and the road in order, for example, to inform the driver in time about upcoming decision points regarding navigation and potentially risky traffic situations. Figure 1.1 depicts these relationships among the three fields.

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.

CHAPTER 4

SYSTEM ANALYSIS

4. SYSTEM ANALYSIS

In practice, traffic signs are diverse in shapes, images, texts so that recognizing all signs become a very challenging task. Here, supplemental and major signs should be distinguished to reduce the complexity of recognition.

The difficulty of traffic signal recognition is potentially sub signal becoming infinite number of traffic classes by as additional information can be written on nearby major information. In this study, we propose a traffic sign recognition system based on the region proposal using segmentation and deep neural network (DNN).

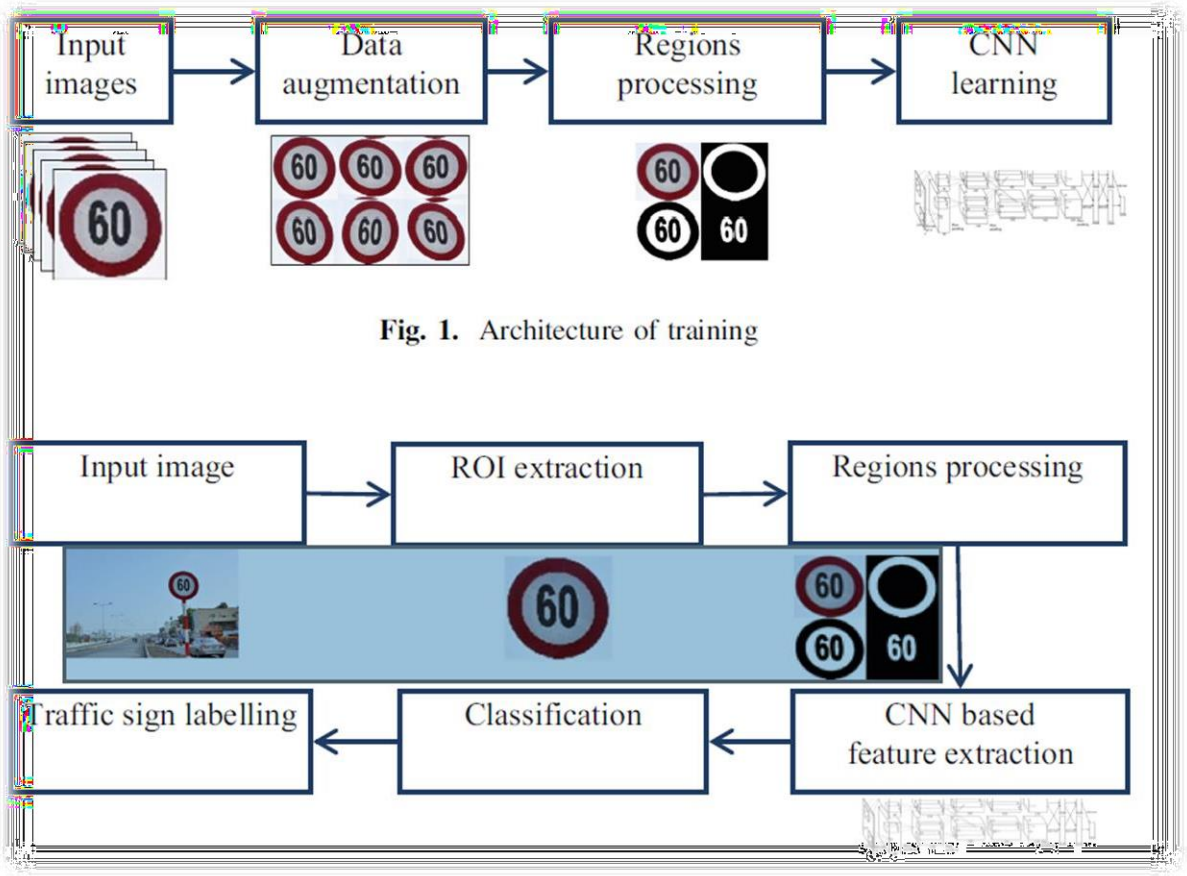


Fig. No 4.1 System Architecture

There is diverse traffic signs of the same types, for example prohibitory of speed limit sign plate usually consists of variable numbers. The traffic plate displays main signal of restriction plate combining with supplement speed signals, e.g., 40, 50, 60, 80 and 100. In traditional approach, the full traffic signs are used to recognize a meaning of traffic signal. This paper presents an approach for interpretation of main signal with supplemental signals for improving accuracy of recognition system.

The method consists of several phases as follows: traffic candidate detection and region of interest Input images Data augmentation Regions processing CNN learning Fig. 1. Architecture of training Input image ROI extraction Regions processing Classification CNN based feature extraction Traffic sign labelling Fig 4.1 Overview of traffic sign recognition architecture 606 V.-D. Hoang et al. (ROI) extraction based on color filter, segmentation of ROI into several regions.

The segmented regions combining original sign plate are feed to the deep neural network for recognition the meaning of traffic signal, as depicted in Fig. 2.

This approach is different to the Region convolution neural network (R-CNN) approach [11], which consists of some tasks as follows: Generating a set of region proposals for bounding boxes, implementing the images within bounding boxes to the pretrained Alex Net model [12], the final classification processing based on the Support vector machine (SVM). In our approach, the classification task was evaluated on two machines of SVM and Full-connection neural network.

An image augmentation task on training data is also applied for efficiency improvements. This task is important to make powerful recognition models since using small data, especially in the experimental data, some traffic sign classes just consist of several image samples. In augmentation task, training images are reproduced to make a larger data, increasing the effective size of the training data set.

4.1 Software Requirement Specification

4. 1. 1. Functional Requirements

Functional requirements describe what the software should do (the functions). Various functional requirements include:

Detection

The goal of traffic sign detection is to locate the regions of interest (ROI) in which a traffic sign is more likely to be found and verify the hypotheses on the sign's presence. The initial detection phase of a traffic sign recognition system offers high costs due to the large scale of detection in a complete single image. In order to reduce the space, prior information of the sign location is supposed to be cropped [3, 4]. This technique called as ROI. ROI locates the traffic sign in the image based on the shape.

The traffic signs are cropped and declared as informative signal. The background image is declared as unwanted signal and removed by defining as black pixel. By these assumptions, a large portion of the image can be ignored. Traffic signs are designed with particular color and shape which make them easier to be recognized.

The detection of traffic signs using only a single frame image has some problems: it is difficult to correctly detect a traffic sign when temporary occlusion occurs; and the correctness of traffic signs is hard to verify.

To increase the speed and accuracy of traffic sign detection in subsequent images by using information about the preceding images, such as the number of the traffic signs and their predicted sizes and positions, can be used. Moreover, information supplied by later images is used to assist in verifying the correct detection of traffic signs, and in this way, those detected and tracked traffic sign can reduce the burden of the processor.

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.

Classification

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze [5]. A binary image classification method is a digital image that has only two possible values for each pixel. The pixels used to represent the object and background is white and black respectively. Based on classification process, the technique used in the classification of traffic sign is binary classification method.

Each traffic sign is grouped based on the amount of white and black pixel. These amounts are matched with the amount of white and black pixel from the template data.

Methodology

Traffic sign recognition algorithm using image processing technique has been developed by combining several methods such as binarization, ROI and pixel matching. Binarization is an early process applied to the traffic sign image. The binarization method ensure the image is in good condition before the implementation of ROI technique.

3.1. Binarization

At the first stage, all images are transformed from RGB color space into black and white color space. Hence, the red regions of each image are detected by the black and white saturation values of pixels. Figure 2 shows the converted image from RGB (Red, Green, and Blue) color into the black and white pixels by using image processing toolbox.

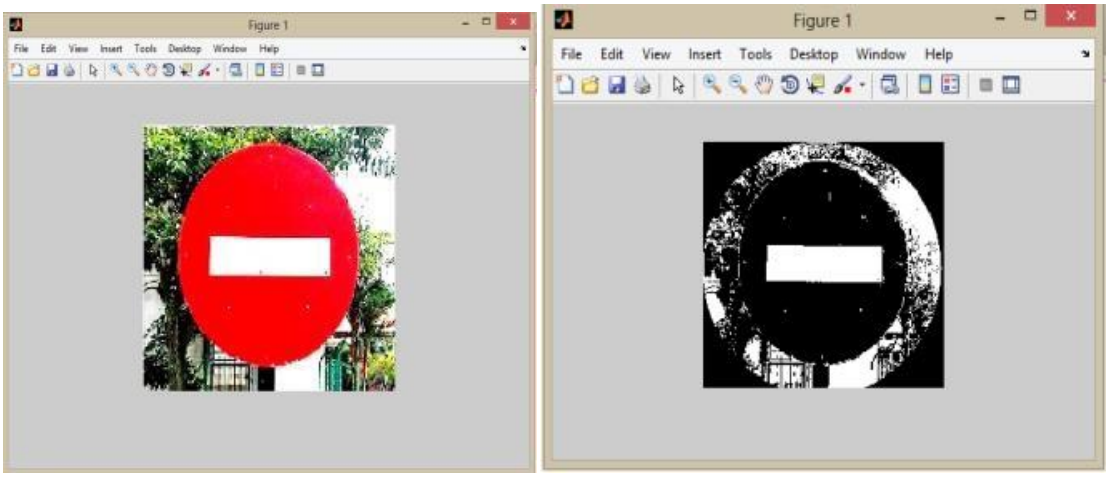


Figure 4.2 RGB **to binary image conversion**

4. 1. 2. Non-Functional requirements

- **Usability**: It defines the user interface of the software in terms of simplicity of understanding. The user interface of Traffic Sign prediction software for any kind of Sign should be simple and understandable.

- **Efficiency**: Maintaining the possible highest accuracy in the Traffic Sign detection in shortest time with available data.

- **Performance**: It is a quality attribute of the Traffic Sign Detection prediction software that describes the responsiveness to various user interactions with it.

4.2 System Requirements

4.2.1 Hardware Requirements

- RAM: 4.00 GB
- Storage: 1 TB
- CPU: Intel-i3 @2.30GHz or faster
- Architecture: 64-bit

4.2.2 Software Requirements

- Python 3.5 or more
- Operating System: Windows 10/ 11
- IDE: Jupyter notebook
- Libraries: NumPy, Matplotlib, Keras, TensorFlow, PIL, Pandas, CV2.

CHAPTER 5

SYSTEM DESIGN

5. SYSTEM DESIGN

5.1. Introduction

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.

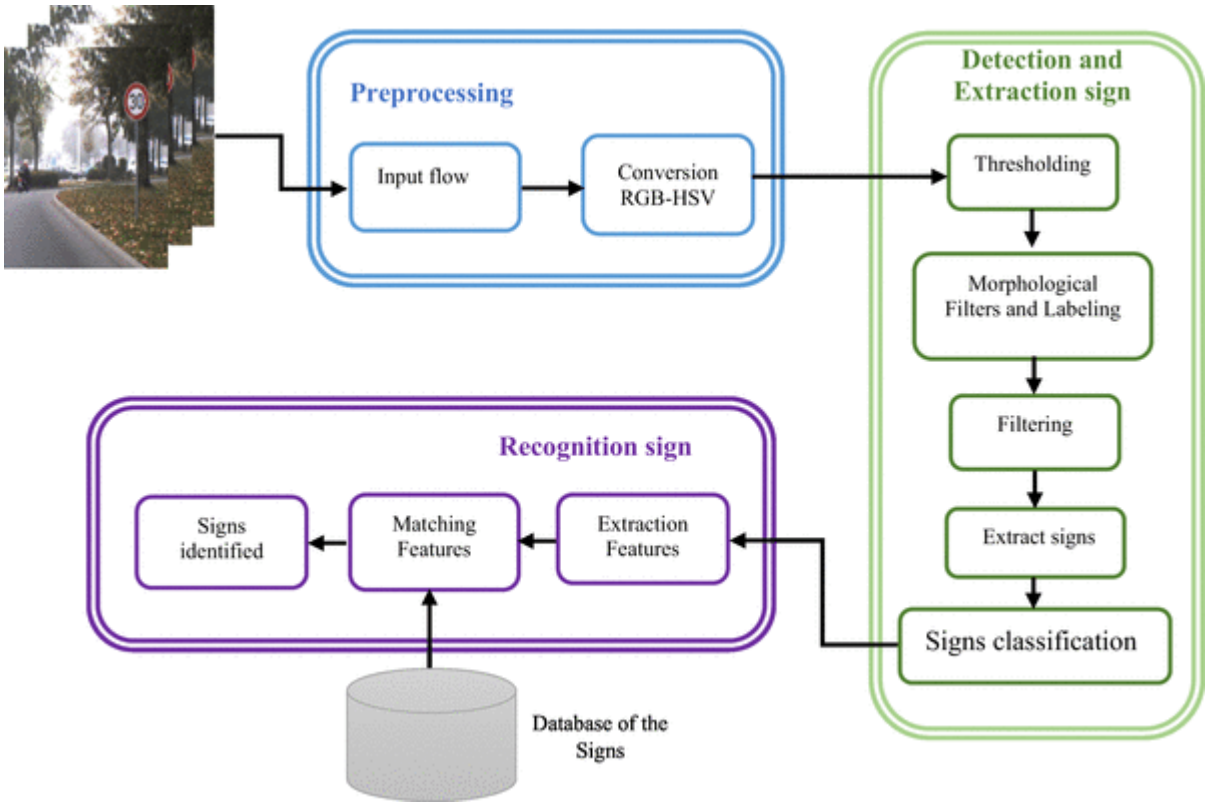


Fig No 5.1 Flow Chart of Traffic Sign Detection

5.2 System Architecture

In practice, traffic signs are diverse in shapes, images, texts so that recognizing all signs become a very challenging task. Here, supplemental and major signs should be distinguished to reduce the complexity of recognition.

The difficulty of traffic signal recognition is potentially sub signal becoming infinite number of traffic classes by as additional information can be written on nearby major information. In this study, we propose a traffic sign recognition system based on the region proposal using segmentation and deep neural network (DNN).

There is diverse traffic signs of the same types, for example prohibitory of speed limit sign plate usually consists of variable numbers. The traffic plate displays main signal of restriction plate combining with supplement speed signals, e.g., 40, 50, 60, 80 and 100. In traditional approach, the full traffic signs are used to recognize a meaning of traffic signal. This paper presents an approach for interpretation of main signal with supplemental signals for improving accuracy of recognition system.

The segmented regions combining original sign plate are feed to the deep neural network for recognition the meaning of traffic signal, as depicted in Fig. 2. This approach is different to the Region convolution neural network (R-CNN) approach [11], which consists of some tasks as follows: Generating a set of region proposals for bounding boxes, implementing the images within bounding boxes to the pretrained Alex Net model [12], the final classification processing based on the Support vector machine (SVM). In our approach, the classification task was evaluated on two machines of SVM and Full-connection neural network.

An image augmentation task on training data is also applied for efficiency improvements. This task is important to make powerful recognition models since using small data, especially in the experimental data, some traffic sign classes just consist of several image samples.

5.3 Data Flow Diagram / UML Design

Activity Diagram

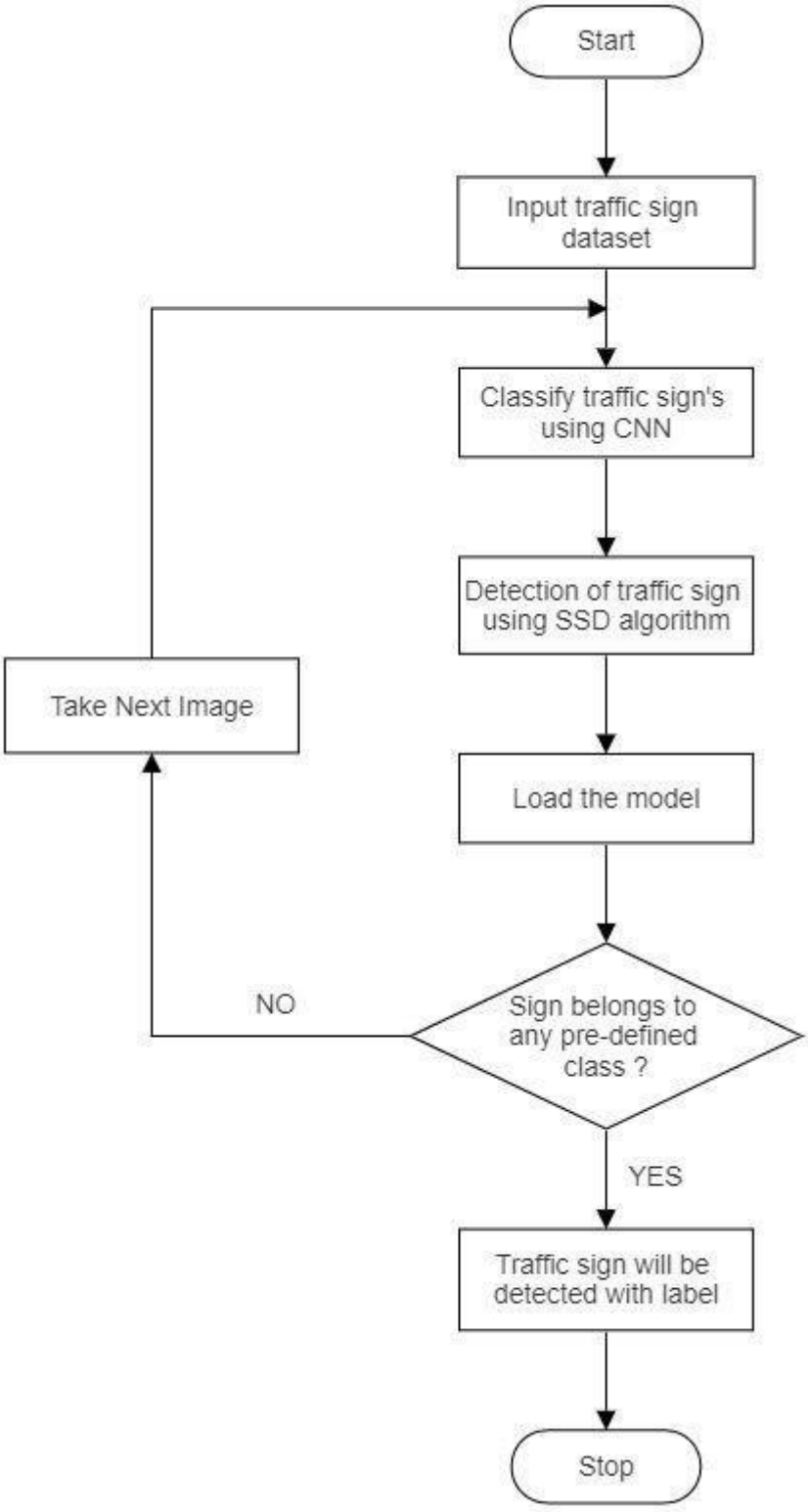


Fig. No 5.2 Activity diagram for Traffic Signs Detection

CHAPTER 6

IMPLEMENTATION

6. ALGORITHM & IMPLEMENTATION

6.1 Technique

CNN is a powerful algorithm for image processing. These algorithms are currently the best algorithms we have for the automated processing of images. Many companies use these algorithms to do things like identifying the objects in an image. Images contain data of RGB combination. Matplotlib can be used to import an image into memory from a file. The computer doesn't see an image, all it sees is an array of numbers. Color images are stored in 3-dimensional arrays. The first two dimensions correspond to the height and width of the image (the number of pixels). The last dimension corresponds to the red, green, and blue colors present in each pixel.

Steps:

- Input Image
- Pre-Processing
- (RGB) Based Detection
- Detection Based on Shape of The Sign
- Recognition Phase

DOUGLAS PEUCKER ALGORITHM

Based on the detection method we calculate the number of edges which is implemented using the algorithm of Douglas-Peucker. We majorly concentrate on two shapes which are circle and triangle because they are most repeated shapes of traffic sign. once using Douglas-Peucker algorithm the number of edges and area of interest is found. Now if the edges found is equal or greater than six and if the main part satisfies minimum condition, then it is considered as a triangle. And if edges are equal and greater than six and moreover if they satisfy minimum condition then the major part of the image is recognized as circle.

After the shapes are recognized the next major step is in detecting the box of the bounding. The bounding box is important because the Region of Interest (ROI) is separated from the environment by the bounding box. Usually, the box touches circle or the triangle of the main region. In a triangular sign, it consists of two triangles they are outer triangle and inner triangle. The outer triangle just touches the box of bounding and the one which does not touch is the inner triangle.

SINGLE-SHOT DETECTOR (SSD)

SSD uses a matching phase while training, to match the appropriate anchor box with the bounding boxes of each ground truth object within an image. Essentially, the anchor box with the highest degree of overlap with an object is responsible for predicting that object's class and its location

Three Layers of CNN

1. Convolutional Layer: In a typical neural network each input neuron is connected to the next hidden layer. In CNN, only a small region of the input layer neurons connects to the neuron hidden layer.

2. Pooling Layer: The pooling layer is used to reduce the dimensionality of the feature map. There will be multiple activation & pooling layers inside the hidden layer of the CNN.

3. Fully-Connected layer: Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

6.2 Pseudo Code

```
import numpy as np
import pandas as pd
 import matplotlib.pyplot as plt
 import tensorflow as tf
from PIL import Image
 import os
from sklearn.model_selection
import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
data = []
labels = []
classes = 43
cur_path = os.getcwd()
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:
        print("Error loading image")
data = np.array(data)
labels = np.array(labels)
```

6.3 Sample Source Code

Installing Libraries:

```
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
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
```

Loading Data

```
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_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

Converting the labels into one hot encoding

```
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)
#Building the model
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))
model.add(Dense(43, activation='softmax'))
```

Compilation of the model

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
epochs = 15
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
```

Saving the Model

```
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 mean_squared_error
from math import sqrt
import pandas as pd
y_test = pd.read_csv('C:/Users/U Sreekanth/dataset/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)
pred = model.predict(X_test).sum(axis=1)
```

Accuracy with the test data

```
from sklearn.metrics import accuracy_score
print(sqrt(mean_squared_error(labels,pred))*0.05)
```

Saving the Model

```
model.save('traffic_classifier.h5')
```

Graphical User Interface

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
from numpy import array
#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 veh > 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 = np.expand_dims(image, axis=0)
    image = np.array(image)
    predict_x= model.predict([image])[0]
    classes_x=np.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="ClassifyImage",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="Know Your Traffic Sign",pady=20, font=('arial',20,'bold'))
heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

6.4 Implementation / Simulation Concepts

Now we are going to build a graphical user interface for our traffic signs classifier with Tkinter. Tkinter is a GUI toolkit in the standard python library. Make a new file in the project folder and copy the below code.

Save it as gui.py and you can run the code by typing python gui.py in the command line. In this file, we have first loaded the trained model 'traffic_classifier.h5' using Keras. And then we build the GUI for uploading the image and a button is used to classify which calls the classify() function.

The classify() function is converting the image into the dimension of shape (1, 30, 30, 3). This is because to predict the traffic sign we have to provide the same dimension we have used when building the model. Then we predict the class, the model.predict_classes(image) returns us a number between (0-42) which represents the class it belongs to. We use the dictionary to get the information about the class. Here's the code for the gui.py file.

When the detection of sign takes place the classification of sign has to be done. By using TensorFlow which is machine learning technique from Google Convolutional Neural Network is designed. In this phase the first thing is to preprocess the image which we received from previous phases. In recognition phase, testing and training of the CNN is considered to be the most significant part.

For the testing and training of the data set we utilized "German Traffic Sign Benchmark and the Belgium Traffic Signs". CNN is considered as the brain as it contains the same features and process that a normal brain does or has. Each neuron receives information and is forwarded to the succeeding neuron. CNN has numerous layers the input layer is the 1st one and output layer is the last. Hidden layer is between the first and the last.

This method has 6 CNN layers. To prevent overfitting in the middle a perfectly connected hidden layer is present. In this model we used "sequential stack" which was developed by Keras that runs above TensorFlow. There is a "Rectified Linear Unit activation" in all layers. In neural networks the main important activation function is ReLu. The input to fully connected layer is the output of sixth conv layer.

The final layer consists of SoftMax activation and the flattened output is given this layer. To improve the processing speed there is a max pooling layer present just after 2 layers. We use a collection or group if CNNs like here we use three to give more perfect results. The output will be more accurate if we use more CNN's rather than one. Things such as optimizer, loss and metric are to be specified. The loss uses values between 0 and 1 rather than percentages. The optimizer utilizes "Stochastic Gradient Descent with Nesterov momentum". Epochs are used in betterment of the training therefore it improves the perfectness in prediction. Finally, a text to speech module is added to the system where the traffic sign is recognized and the output is produced in the form of voice which helps the driver in the car to easily recognize the signs and avoid accidents

Input Screenshots:

Fig 6.1 Images Used for Input

CHAPTER 7

TESTING

7. TESTING

7.1. Introduction

In image-based tests, TestComplete identifies UI elements by their image rather than by object properties.

To simulate user actions over a control specified by its image, TestComplete

1. Searches for the control's image on the screen pixel-by-pixel. For desktop and web applications, it will search for the image on the PC desktop, and for mobile application, it will search for the image on the screen of the connected mobile device.
2. If the image is found, gets its coordinates.
3. Simulates a needed user actions at the image's coordinates.

Image Based Test

- To test uninstrumented Android applications.
- To test desktop or web applications you cannot prepare for testing and make "open" for TestComplete.
- To test desktop, web, Android and iOS applications that include controls that TestComplete cannot recognize (for example, custom-drawn controls).

An image-based test consists of a sequence of operations on UI elements in your tested application: clicks (for desktop and web applications), touches (for mobile applications), drag and drop operations, checkpoints and so on.

For Android applications, the easiest way to create an image-based test is to **record** it. You can then edit and improve the test, if needed. To learn more on recording image-based tests for Android.

For desktop, web, and iOS applications, TestComplete does not support recording image-based tests. Capture images of needed controls and add operations that will simulate user actions over controls to your tests manually.

7.2 Test Cases

Test Case 1

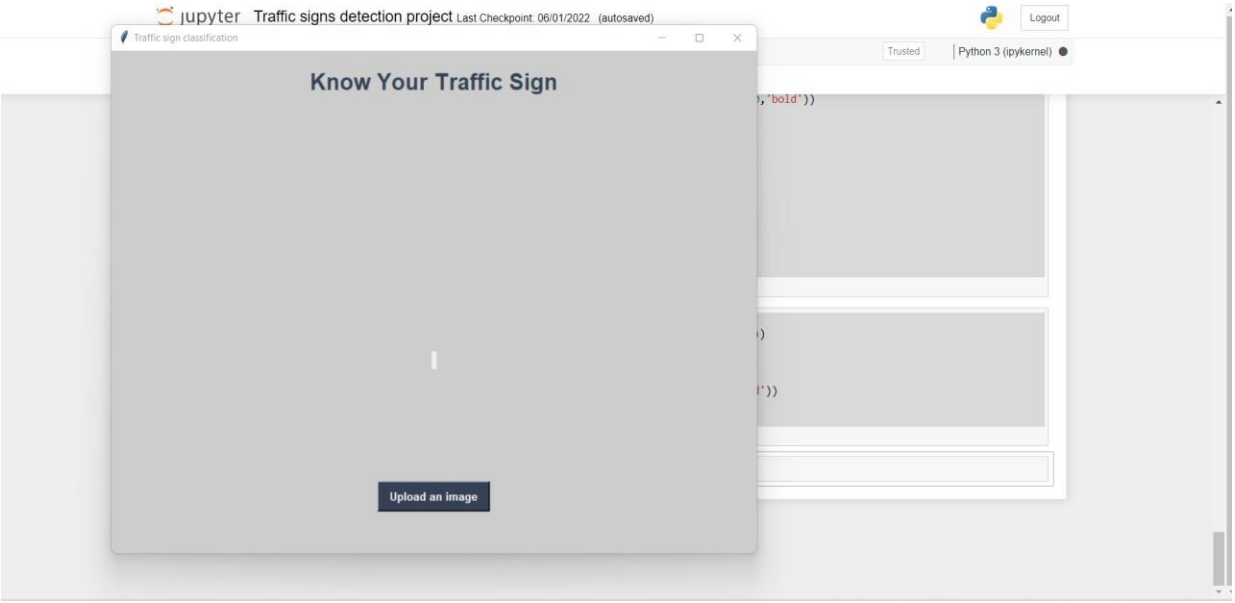


Fig No 7.1 Uploading The Sample Image

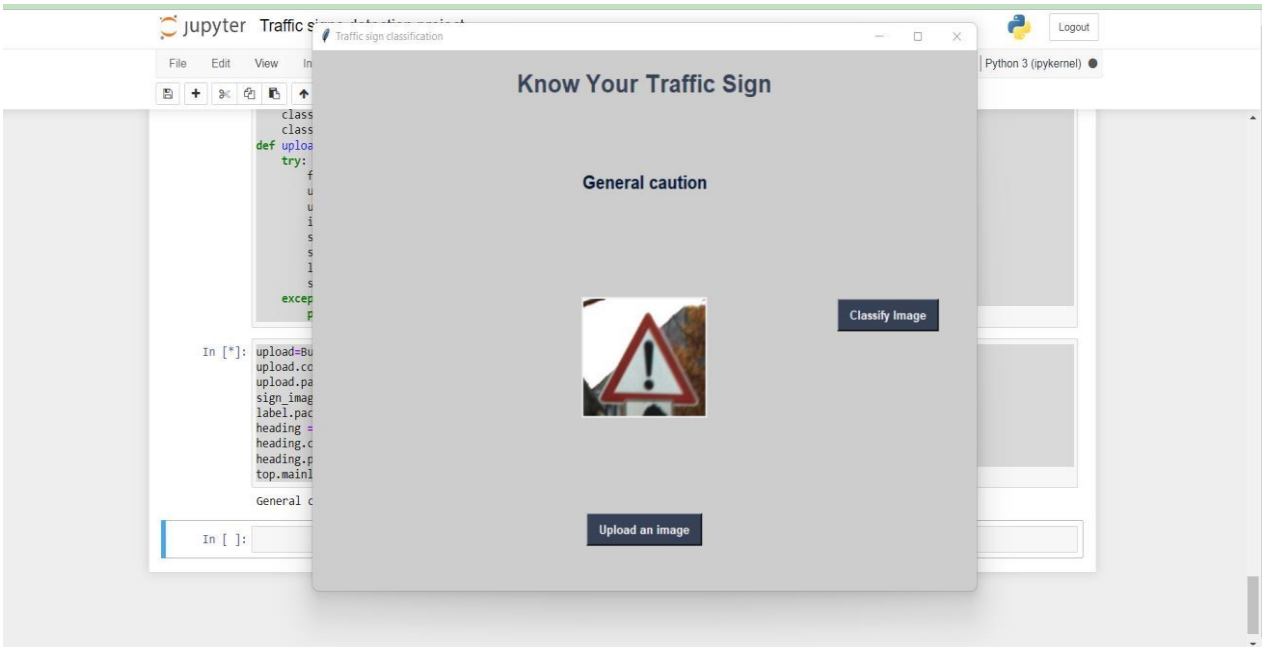


Fig No 7.2 Classifying the Uploaded Image

Test Case 2

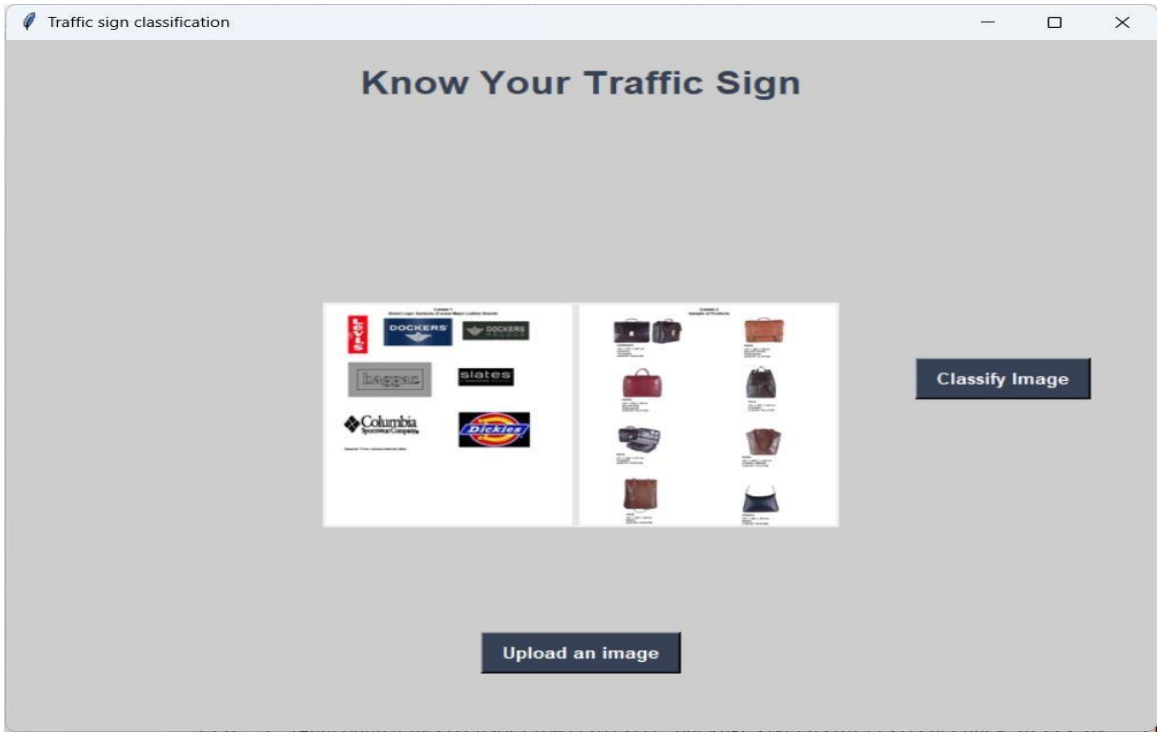


Fig No 7.3 Uploading the Wrong Sample Image

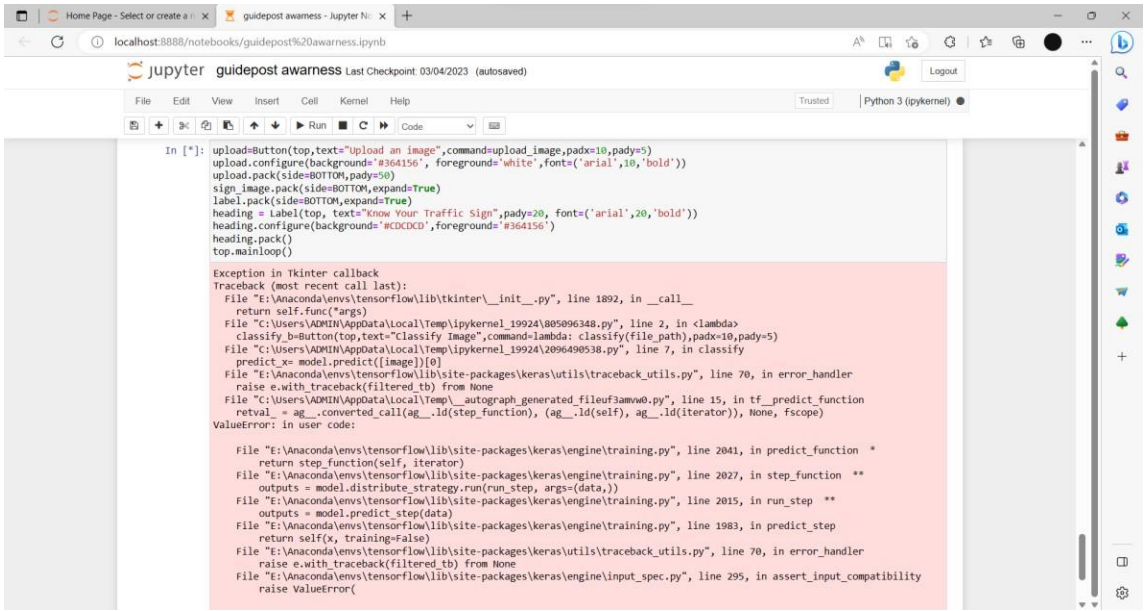


Fig No 7.4 Classifying the Uploaded Wrong Image (Error)

CHAPTER 8

RESULTS

8. RESULTS

```
#Converting lists into numpy arrays
data = np.array(data)
labels = np.array(labels)
print(data.shape, labels.shape)
#Splitting training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

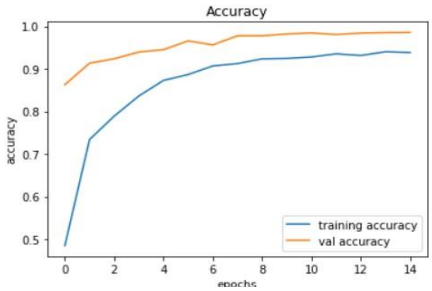
```
(39209, 30, 30, 3) (39209,)
(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

In [5]:
```
#Compilation of the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
epochs = 15
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
```

```
Epoch 1/15
981/981 [==============================] - 39s 40ms/step - loss: 2.0420 - accuracy: 0.4850 - val_loss: 0.5670 - val_accuracy:
0.8629
Epoch 2/15
981/981 [==============================] - 39s 40ms/step - loss: 0.8949 - accuracy: 0.7340 - val_loss: 0.3837 - val_accuracy:
0.9132
Epoch 3/15
981/981 [==============================] - 39s 40ms/step - loss: 0.7019 - accuracy: 0.7890 - val_loss: 0.2569 - val_accuracy:
0.9237
Epoch 4/15
981/981 [==============================] - 40s 41ms/step - loss: 0.5443 - accuracy: 0.8363 - val_loss: 0.2148 - val_accuracy:
0.9396
Epoch 5/15
981/981 [==============================] - 42s 42ms/step - loss: 0.4197 - accuracy: 0.8729 - val_loss: 0.1966 - val_accuracy:
0.9450
Epoch 6/15
981/981 [==============================] - 45s 46ms/step - loss: 0.3853 - accuracy: 0.8867 - val_loss: 0.1167 - val_accuracy:
0.9657
Epoch 7/15
981/981 [==============================] - 43s 44ms/step - loss: 0.3102 - accuracy: 0.9068 - val_loss: 0.1560 - val_accuracy:
0.9563
Epoch 8/15
981/981 [==============================] - 45s 46ms/step - loss: 0.2958 - accuracy: 0.9124 - val_loss: 0.0827 - val_accuracy:
0.9777
Epoch 9/15
981/981 [==============================] - 45s 46ms/step - loss: 0.2631 - accuracy: 0.9236 - val_loss: 0.0710 - val_accuracy:
0.9777
Epoch 10/15
981/981 [==============================] - 43s 44ms/step - loss: 0.2572 - accuracy: 0.9245 - val_loss: 0.0680 - val_accuracy:
```

```
Epoch 11/15
981/981 [==============================] - 44s 45ms/step - loss: 0.2479 - accuracy: 0.9279 - val_loss: 0.0536 - val_accuracy:
0.9843
Epoch 12/15
981/981 [==============================] - 44s 44ms/step - loss: 0.2303 - accuracy: 0.9353 - val_loss: 0.0674 - val_accuracy:
0.9806
Epoch 13/15
981/981 [==============================] - 43s 44ms/step - loss: 0.2414 - accuracy: 0.9315 - val_loss: 0.0562 - val_accuracy:
0.9841
Epoch 14/15
981/981 [==============================] - 44s 45ms/step - loss: 0.2107 - accuracy: 0.9402 - val_loss: 0.0510 - val_accuracy:
0.9852
Epoch 15/15
981/981 [==============================] - 44s 45ms/step - loss: 0.2208 - accuracy: 0.9384 - val_loss: 0.0506 - val_accuracy:
0.9856
```

In [7]:
```
#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()
```



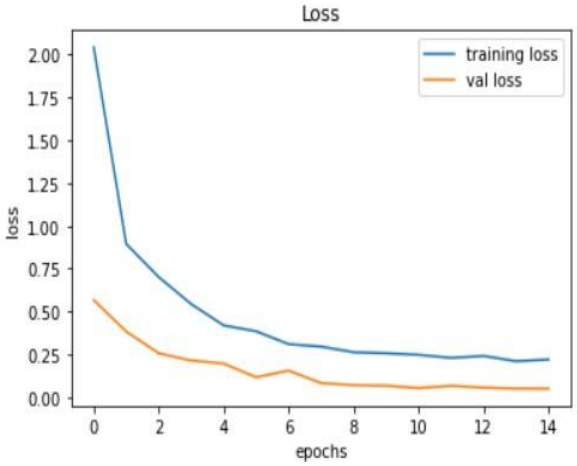


Table 1. Accuracy level of detection

Percentage	Accuracy Group
0 % - 20%	No Detection
20% - 40%	Poor Detection
40% - 60%	Average Detection
60% - 80%	Good Detection
80% - 100%	Excellent Detection

Based on table 1, the accuracy level of detection determines the condition of images captured from the environment. No detection and poor detection levels show the images contain a lot of noises and blur. These noises due to the heavy rain, cloudy and etc. Detection of 60% and above indicates the clear environment.

Besides, blocked sign and different angle of sign's capturing also reflect to the good detection, based on these factors, the result of traffic sign detection as shown as table 2 and figure 6.

Table 2. Detection of traffic sign based on 3 samples

TRAFFIC SIGN	ANGLE (^{0})	DAYTIME	BLOCKED	AFTER RAINING	NIGHT	
					ACTUAL PICTURE	AFTER BEAM LIGHT (A.LIGHT)
Motor Crossing	Left (45^0)	95.1%	35%	80%	0%	79%
	Centre	98.9%	80%	95%	0%	87%
	Right (45^0)	94.5%	30%	85%	0%	90%
No Entry	Left (45^0)	88%	22%	22%	0%	51%
	Centre	99.97%	76%	95%	0%	95%
	Right (45^0)	95%	75%	81%	0%	82%
Stop	Left (45^0)	95%	40%	70%	0%	80%
	Centre	98.7%	35%	90%	0%	94%
	Right (45^0)	75%	30%	56%	0%	88%

The blocked traffic sign during image capturing shows average detection of different angles. The highest and lowest detection in daytime is 80% for “Motor cross” and 22% for “No entry” sign board respectively. Based on the blocked condition, the sign image is blocked in the middle area of image. As an output of the ROI process, the blocked traffic sign affects the percentage of detection.

The blocked image reduces the accuracy of detection in big margin as well as affects the driver to recognize the traffic sign. The detection after the raining shows good detection for 3 traffic sign under different condition and angles. During the raining time, the environment looks so cloudy and darker compared to daytime.

The highest and lowest detection in daytime is 95% for “Motor cross” and “No entry”; and 22% for “No entry” sign board respectively. The detection level percentage is still high due to clear and bright vision after rain. The clear vision helps the detection level reaches the maximum level. The detection at night time shows poor detection from 3 traffic sign under different situation and angles.

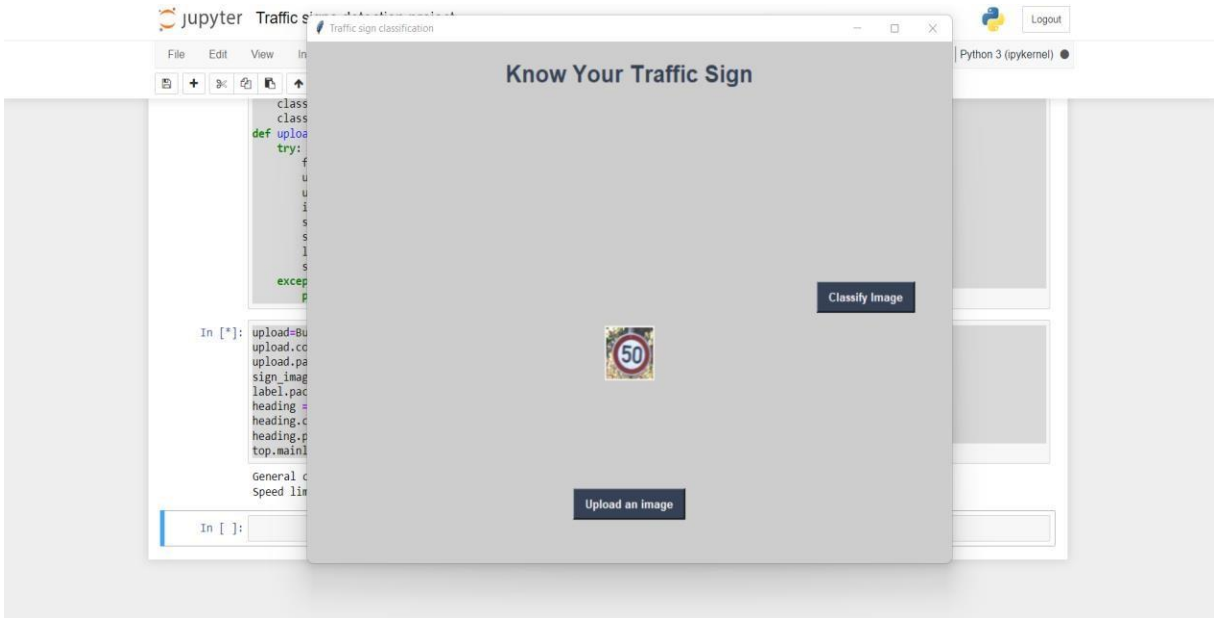
During night time, the environments are very dark and the system is unable to detect any present traffic sign. Detection level shows 0% during night time. To solve this program, the traffic sign is captured after the beam light of the car. The highest and lowest detection in daytime is 94% for “Stop” and 51% for “No entry” sign board respectively. Following Figure shows the detection model Accuracy to detect the traffic sign we get **Accuracy of 94%**

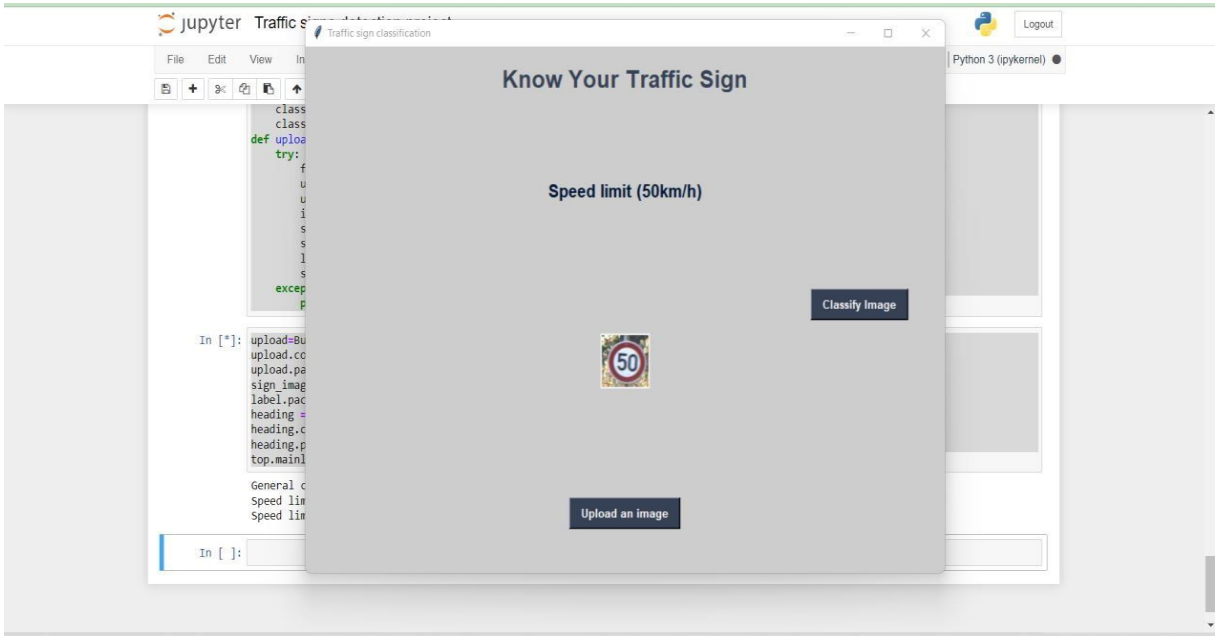
In [8]:
```
#testing accuracy on test dataset
from sklearn.metrics import mean_squared_error
from math import sqrt
import pandas as pd
y_test = pd.read_csv('C:/Users/U Sreekanth/dataset/Test.csv')
labels = y_test["ClassId"].values
imgs = y_test["Path"].values
data=[]
for img in imgs:
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))
X_test=np.array(data)
pred = model.predict(X_test).sum(axis=1)
#Accuracy with the test data

from sklearn.metrics import accuracy_score
print(sqrt(mean_squared_error(labels,pred))*0.05*10)
```

```
94.13501700572444
```

Result:


Traffic sign classification
Know Your Traffic Sign
Classify Image
Upload an image


Traffic sign classification
Know Your Traffic Sign
Speed limit (50km/h)
Classify Image
Upload an image

CHAPTER 9

CONCLUSION

9. CONCLUSION

- The traffic sign recognition is a very helpful driver assistance technique for increasing traffic and driver safety.

- In this project, low computing complexity, adaptive and accurate mechanisms have been applied to extract and recognize the content of each traffic sign.

- Several techniques and methods have been implemented to recognize the traffic sign such as binarization, region of interest (ROI) and pixel's classification has been proven successful.

- The system has been proven to produce high accuracy of detection based on different conditions such as angle of image captured, daytime, night and etc. The results shows that the average recognition can achieve until 35% for the blocked traffic sign. The highest accuracy has been detected at 99.9% during daytime.

Our project has attained, Program Outcomes PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO8, PO9, PO10, PO11, PO12 and Program Specific Outcomes PSO1, PSO2, PSO3.

PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
3	3	2	2	3	2	2	3	3	3	3	3

PSO1	PSO2	PSO3
2	3	2

CHAPTER 10

BIBLIOGRAPHY

10. BIBLIOGRAPHY

[1]. Available online at:
http://en.wikipedia.org/wiki/Advanced_Driver_Assistance_Systems
.

[2]. H. Akatsuka and S. Imai, "Road signposts recognition system", SAE transactions, Vol 96, No. 1, pp. 936-943, Feb. 1987.

[3]. G. Piccioli, E. de Micheli, P. Parodia, and M. Campani, "Robust method for road sign detection and recognition", Image and Vision Computing, Vol 14, No.3, pp. 209-223, Apr. 1996.

[4]. S.-H. Hsu and C.-L. Huang, "Road sign detection and recognition using matching pursuit method", Image and Vision.

[5]. Linda G. Shapiro and George C. Stockman (2001): "Computer Vision", pp 279-325, New Jersey, Prentice-Hall, ISBN 0-13-030796-3

[6]. H. Akatsuka and S.Imai, "Road signposts recognition system", SAE transaction, Vol 96, No. 1,pp. 936-943, Feb.1987.

[7]. Sheldon Waite and ErdalOruklu, "FPGA-Based Traffic Sign Recognition for Advanced Driver Assistance Systems",Journal of Transportation Technologies, background of study on traffic sign, pp.2, 2012.

[8]. Rubén Laguna*, Rubén Barrientos*, L. Felipe Blázquez*, Luis J. Miguel**, "Traffic sign recognition application based on image processing techniques",Preprints of the 19th World Congress the International Federation of Automatic Control Cape Town, South Africa, background of study on traffic sign, pp.104, Aug 24-29, 2014.

[9]. Carlos Filipe Moura Paulo, "Detection and Recognition of Traffic Signs" detection of traffic sign, pp. 4, sep.2007.

[10]. ufuksuataydin, "Traffic Signs Recognition" detection of traffic sign, pp.6, may.2009.

[11]. mehmetbülenthavur, "traffic sign recognition for unmanned vehicle control", color analysis, pp.9-19, Nov 2006.

[12]. Jingwen Feng, "Traffic Sign Detection and Recognition System for Intelligent Vehicles" color analysis, pp.11-14, 2014