

AUTOMATIC TRAFFIC SIGN DETECTION

A PROJECT REPORT

Submitted by

VAIBHAV CHARAN B 312321205181

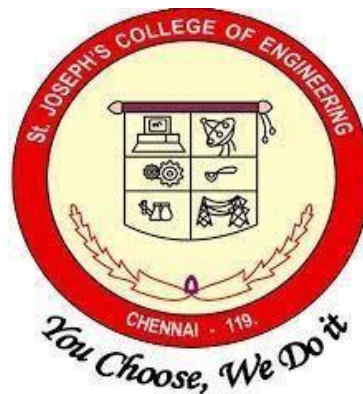
TEJESHWAR SIDDARTH S 312321205176

in partial fulfillment of the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY



St. JOSEPH'S COLLEGE OF ENGINEERING

(An Autonomous Institution)

OMR, Chennai 600 119

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2024

ANNA UNIVERSITY, CHENNAI



BONAFIDE CERTIFICATE

Certified that this project report “Automatic Traffic Sign Detection and Speed Control Alert System” is the bonafide work of “Vaibhav Charan B and Tejeshwar Siddarth S” who carried out the project work under my supervision.

SIGNATURE	SIGNATURE
HEAD OF THE DEPARTMENT Ms. G. Lathaselvi, M.E., (Ph.D.), Associate Professor & Head of Department Dept. of Information Technology, St. Joseph’s College of Engineering, OMR, Chennai-600 119	SUPERVISOR Mrs. M. Janani, M. E, (Ph.D.) Associate Professor, Dept. of Information Technology, St. Joseph’s College of Engineering, OMR Chennai-600 119

Submitted to Project and Viva Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

At the outset, we would like to express our sincere gratitude to our beloved **Dr. B. Babu Manoharan M.A., M.B.A., Ph.D.,** *Chairman, St. Joseph's Group of Institutions* for his constant guidance and support to the student community and the Society.

We would like to express our hearty thanks to our respected **Managing Director Mr. B. Shashi Sekar, M.Sc.** for his kind encouragement and blessings.

We wish to express our sincere thanks to the Executive **Director Mrs. S. Jessie Priya M.Com.** for providing ample facilities in the institution.

We express sincere gratitude to our beloved **Principal Dr. Vaddi Seshagiri Rao M.E., M.B.A., Ph.D., F.I.E.** for his inspirational ideas during the course of the project.

We express our sincere gratitude to our beloved **Dean (Research) Dr. A. Chandrasekar M.E., Ph.D., Dean (Student Affairs) Dr. V. Vallinayagam M.Sc., M.Phil., Ph.D., and Dean (Academics) Dr. G. Sreekumar M.Sc., M.Tech., Ph.D.,** for their inspirational ideas during the course of the project.

We wish to express our sincere thanks to **Ms. G. Lathaselvi, M.E., (Ph.D.), Head of the Department,** Department of Information Technology, St. Joseph's College of Engineering for her guidance and assistance in solving the various intricacies involved in the project.

We would like to acknowledge our profound gratitude to our supervisor **Ms. M. Janani M. E, (Ph.D)** , for her/his expert guidance and connoisseur suggestion to carry out the study successfully.

Finally, we thank the **Faculty Members** and **our Family**, who helped and encouraged us constantly to complete the project successfully.

ABSTRACT

Automatic detection and recognition of traffic signs plays a crucial role in management of the traffic-sign inventory. It provides an accurate and timely way to manage traffic-sign inventory with a minimal human effort. In the computer vision community, the recognition and detection of traffic signs are a well-researched problem. A vast majority of existing approaches perform well on traffic signs needed for advanced driver-assistance and autonomous systems. However, this represents a relatively small number of all traffic signs (around 43 categories out of several hundred) and performance on the remaining set of traffic signs, which are required to eliminate the manual labor in traffic-sign inventory management, remains an open question. In this paper, we address the issue of detecting and recognizing a large number of traffic-sign categories suitable for automating traffic-sign inventory management. We adopt a convolutional neural network (CNN) approach, the mask R-CNN, to address the full pipeline of detection and recognition with automatic end-to-end learning. We propose several improvements that are evaluated on the detection of traffic signs and result in an improved overall performance. This approach is applied to detection of 200 traffic-sign categories represented in our novel dataset. The results are reported on highly challenging traffic-sign categories that have not yet been considered in previous works. We provide comprehensive analysis of the deep learning method for the detection of traffic signs with a large intra-category appearance variation and show below 3% error rates with the proposed approach, which is sufficient for deployment in practical applications of the traffic-sign inventory management.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	
	LIST OF FIGURES	
	LIST OF TABLES	
	LIST OF SYMBOLS	
1	INTRODUCTION	1
	INTRODUCTION	1
2	LITERATURE REVIEW	3
	2.1 RELATED WORKS	4
	2.2 EXISTING SYSTEM	9
	2.2.1 DISADVANTAGES	9
	2.3 PROPOSED SYSTEM	11
	2.4 SCOPE OF THE PROJECT	13
3	SYSTEM REQUIREMENTS	15
	3.1 HARDWARE REQUIREMENTS	15
	3.2 SOFTWARE REQUIREMENTS	15
4	SYSTEM DESIGN	16
	4.1 SYSTEM ARCHITECTURE DIAGRAM	16
	4.2 DATA FLOW DIAGRAM	17
	4.2.1 DATA FLOW DIAGRAM LEVEL 1	17
	4.2.2 DATA FLOW DIAGRAM LEVEL 2	18
5	SYSTEM IMPLEMENTATION	19
	5.1 ABOUT THE DATASET	19

	5.1.2 READING THE DATASET	20
	5.2 CAMERA CALIBRATION AND IMAGE ACQUISITION	22
	5.3 IMAGE PREPROCESSING	23
	5.4 TRAFFIC SIGN DETECTION	25
	5.5 CONVOLUTIONAL NEURAL NETWORK (CNN) ARCHITECTURE	28
	5.6 TRAINING AND FINE-TUNING THE CNN MODEL	30
	5.7 TRAFFIC SIGN CLASSIFICATION AND DESCRIPTION	32
	5.8 OUTPUT ANALYSIS	34
6	RESULTS AND EVALUATION	36
	6.1 RESULTS	36
	6.2 PERFORMANCE ANALYSIS	36
7	CONCLUSION FUTURE ENHANCEMENT	38
	7.1 CONCLUSION	38
	7.2 FUTURE ENHANCEMENT	38
	APPENDICES	40
	APPENDICES – I	40
	APPENDICES – II	45
	REFERENCES	53

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.1	The relationship among Road Sign Inventory, Road Sign Recognition and ITS.	2
4.1	System Architecture Diagram	16
4.2	Data Flow Diagram Level 1	17
4.3	Data Flow Diagram Level 2	18
5.1	Different Speed limits	20
5.4	Traffic Sign Detection	27
A.1	The model recognizing Speed Limit 50 Km/h	45
A.2	The model recognizing Stop	45
A.3	The model recognizing Road Work	46
A.4	The model recognizing Priority Road	46
A.5	The model recognizing Go Straight or Right	47
A.6	The model recognizing No passing for vehicles over 3.5 metric	47

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
5.1	Traffic Signs and their collective meanings	27

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
ADAS	Advanced Driver-Assistance Systems
TSR	Traffic-Sign Recognition
TSD	Traffic-Sign Detection
ITS	Intelligent Transport Systems
SVM	Support Vector Machines
HOG	Histogram of Oriented Gradient
DMFS	Digital Micro Fluid Systems

CHAPTER 1

1.1 INTRODUCTION

Proper management of traffic-sign inventory is an important task in ensuring safety and efficiency of the traffic flow. Traffic signs are captured using a vehicle-mounted camera and manual localization and recognition is performed off-line by a human operator to check for consistency with the existing database. However, such manual work can be extremely time consuming when applied to thousands of kilometers of roads. Automating this task would significantly reduce the amount of manual work and improve safety through quicker detection of damaged or missing traffic signs. A crucial step towards the automation of this task is replacing manual localization and recognition of traffic signs with an automatic detection. In the computer-vision community the problem of traffic-sign recognition has already received a considerable attention, and excellent detection and recognition algorithms have already been proposed. But these solutions have been designed only for a small number of categories, mostly for traffic signs associated with advanced driver-assistance systems (ADAS) and autonomous vehicles. Various previous benchmarks have addressed the traffic sign recognition and detection task. However, several of them focused only on traffic-sign recognition (TSR) and ignored the much more complex problem of traffic-sign detection (TSD) where finding accurate location of traffic sign is needed. 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 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.

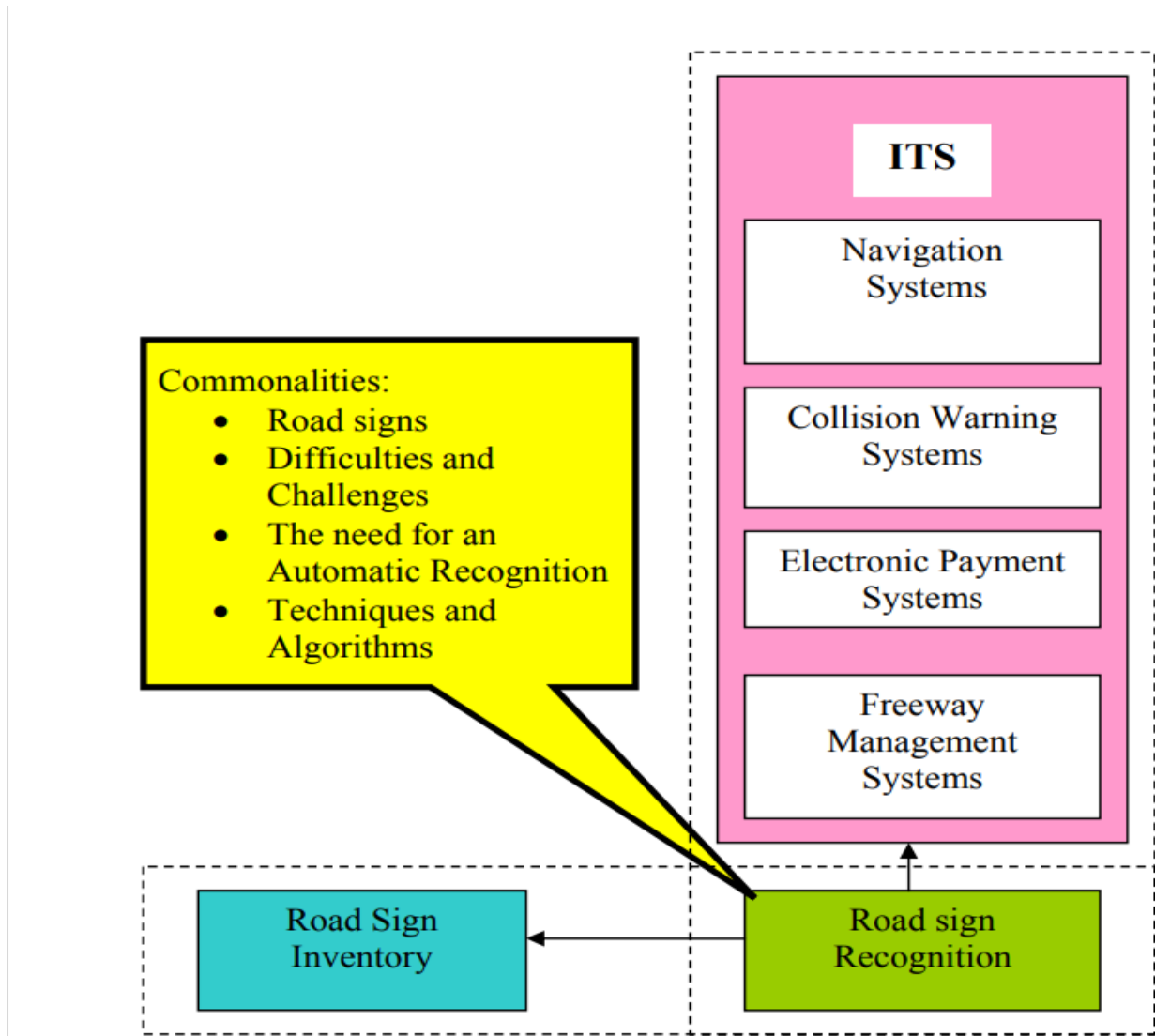


Figure 1.1: The relationship among Road Sign Inventory, Road Sign Recognition and ITS.

A road and traffic sign recognition system, integrated into Intelligent Transport Systems (ITS), continuously monitors drivers and vehicles for enhanced safety and efficiency. It automates the detection and recognition of road signs, aiding drivers in real-time awareness and compliance with traffic regulations. This technology leverages computer vision and machine learning to provide timely and accurate information to drivers and transportation authorities.

CHAPTER 2

LITERATURE REVIEW

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system, the above consideration is considered for developing the proposed system. The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations. Literature review is a critical initial step in software development, ensuring considerations of time, resources, economics, and organizational capabilities are met before tool development commences. Determining the appropriate operating system and programming language follows once prerequisites are fulfilled. During tool development, programmers rely on external support from senior colleagues, books, and online resources. Each project emphasizes comprehensive needs assessment before progressing into system design and development, highlighting the essential role of literature.

2.1 RELATED WORKS

a) TITLE: ROAD-SIGN DETECTION AND RECOGNITION BASED ON SUPPORT VECTOR MACHINES

This paper introduces an automatic road-sign detection and recognition system leveraging support vector machines (SVMs). The system is capable of identifying various traffic sign shapes, including circular, rectangular, triangular, and octagonal, covering all Spanish traffic signs. Divided into three stages, the system employs color-based segmentation, shape classification using linear SVMs, and content recognition via Gaussian-kernel SVMs. Results demonstrate high accuracy with minimal false positives, indicating robustness to translation, rotation, scale, and partial occlusions. Overall, the proposed algorithm offers a reliable solution for enhancing driver safety and navigation through efficient road-sign detection and recognition.

PROS: The system's segmentation stage based on colors such as red, blue, yellow, white, or combinations thereof enables the detection of all types of traffic signs, including those with various colors. This versatility ensures comprehensive coverage and applicability across diverse traffic environments.

CONS: Relying solely on color-based segmentation for sign detection may limit the system's effectiveness in situations where signs exhibit unconventional colors or when environmental factors affect color perception. Integrating additional features or cues beyond color could enhance robustness in challenging condition.

b) TITLE: REAL-TIME DETECTION AND RECOGNITION OF ROAD TRAFFIC SIGNS

This paper introduces a novel system for automatic traffic sign detection and

recognition. It utilizes maximally stable extremal regions (MSERs) for robust region detection under varying lighting conditions. Recognition is achieved through a cascade of support vector machine (SVM) classifiers trained on histogram of oriented gradient (HOG) features, using synthetic template images from an online database for training. The system operates accurately at high vehicle speeds and diverse weather conditions, processing frames at an average speed of 20 frames per second. It successfully recognizes all classes of ideogram-based traffic symbols, demonstrating its efficacy through comprehensive comparative results.

PROS: Leveraging maximally stable extremal regions (MSERs) for candidate region detection ensures robustness to variations in lighting conditions. This robustness enhances the system's reliability in real-world scenarios where lighting may fluctuate, ensuring consistent performance across different environments.

CONS: One potential limitation of this system is its reliance on synthetic template images for training data. While this approach eliminates the need for real footage road signs, it may not fully capture the variability and complexity of real-world traffic sign images, potentially leading to reduced performance in scenarios where synthetic images do not adequately represent the diversity of actual road signs.

c) TITLE: VISION-BASED TRAFFIC SIGN DETECTION AND ANALYSIS FOR INTELLIGENT DRIVER ASSISTANCE SYSTEMS

This paper presents a comprehensive survey of the literature on traffic sign detection for driver assistance systems. It outlines recent advancements in segmentation, feature extraction, and final sign detection stages. While acknowledging the maturity of traffic sign recognition (TSR) research, the paper identifies key open issues such as limited use of publicly available image

databases and bias towards European traffic signs. Additionally, the paper discusses future research directions, emphasizing the integration of context and localization. Furthermore, it introduces a new public database containing U.S. traffic signs, aiming to address data availability gaps in the field

PROS: This survey paper provides a comprehensive overview of the traffic sign detection literature, including recent advancements and open research issues. It offers valuable insights for researchers and practitioners in the field, facilitating knowledge dissemination and guiding future research directions.

CONS: However, the focus on European traffic signs and the limited utilization of publicly available image databases may hinder the generalizability of findings and restrict the applicability of proposed methods to diverse traffic environments worldwide.

d) TITLE: A VISION SYSTEM FOR TRAFFIC SIGN DETECTION AND RECOGNITION

This paper introduces an automatic traffic sign recognition system utilizing videos from an on-board dashcam. It employs image processing techniques such as bilateral Chinese transform and vertex and bisector transform to extract traffic sign areas. Feature vectors are generated using histogram of oriented gradients, followed by support vector machines for initial detection. A neural network is then employed for traffic sign identification. Experimental evaluation using real traffic scenes validates the effectiveness of the proposed system, demonstrating its potential for enhancing road safety and navigation assistance.

PROS: This system leverages a combination of image processing techniques, support vector machines, and neural networks to achieve automatic traffic sign recognition using on-board dashcam videos. By utilizing these advanced technologies, the system demonstrates effectiveness in real traffic scenarios,

contributing to improved road safety and navigation assistance.

CONS: However, the effectiveness of the system may be impacted by factors such as varying lighting conditions, occlusions, and the presence of non-standard or obscured traffic signs. Addressing these challenges could enhance the system's robustness and reliability in practical driving environments.

e) TITLE: AN EFFICIENT METHOD FOR TRAFFIC SIGN RECOGNITION BASED ON EXTREME LEARNING MACHINE

This paper presents a computationally efficient method for traffic sign recognition (TSR) consisting of two modules: 1) extraction of histogram of oriented gradient variant (HOGv) feature and 2) a single classifier trained by extreme learning machine (ELM) algorithm. The HOGv feature extraction balances redundancy and local details, enhancing shape representation. The ELM-based classifier achieves optimal solutions for multiclass TSR by leveraging random feature mapping without requiring layer-by-layer tuning. Evaluation using three datasets demonstrates high recognition accuracy and exceptional computational efficiency in both training and recognition processes.

PROS: The proposed method offers a balance between computational efficiency and recognition accuracy, making it suitable for real-time applications in traffic sign recognition systems.

CONS: However, the performance of the method may vary depending on the complexity and variability of traffic sign images, potentially leading to reduced accuracy in challenging scenarios.

**f) TITLE: TRAFFIC SIGN DETECTION- A NEW APPROACH AND
RECOGNITION USING CONVOLUTION NEURAL NETWORK**

This research presents a Traffic Sign Recognition (TSR) system designed for Bangladeshi traffic signs, employing color cues and Convolutional Neural Network (CNN) for feature extraction and classification. The system undergoes image acquisition, pre-processing, color-based segmentation, morphological closing, and region filtering to extract sign areas. The extracted regions are then classified using deep CNN. Experimental results demonstrate the algorithm's comparable performance with good recognition accuracy, suggesting its potential effectiveness in assisting drivers for safe driving in Bangladeshi road environments.

PROS: This research addresses the need for a Traffic Sign Recognition (TSR) system tailored specifically for Bangladeshi traffic signs, utilizing color cues and Convolutional Neural Network (CNN) for effective feature extraction and classification. The proposed algorithm demonstrates good recognition accuracy in experimental evaluations, potentially contributing to safer driving practices in Bangladeshi road conditions.

CONS: However, while the algorithm's performance is comparable and recognition accuracy is good, there may be scope for further optimization to enhance its robustness and adaptability to a wider range of traffic sign variations and environmental conditions.

2.2 EXISTING SYSTEM

The process described involves the detection of traffic signs using Convolutional

Neural Networks (CNNs) in conjunction with image acquisition and processing techniques. Initially, the image undergoes preprocessing tasks to enhance its quality, including noise reduction and contrast adjustment. Subsequently, the image is segmented using color information from the HSV color model, followed by morphological operations to refine the segmented areas. The filtered image is then subjected to further refinement based on region properties and shape signatures, and the desired sign regions are isolated and cropped for further analysis. Finally, a deep CNN is employed for automatic feature extraction and sign classification, leveraging its ability to recognize patterns and features in images. This entire process is integrated into a cohesive pipeline and optimized for performance, with validation and testing conducted under various real-world conditions to ensure reliability.

Additionally, the summary mentions droplet transport methods like Dielectrophoresis and Electrowetting, as well as design specifications for DMFS (Digital Microfluidic Systems), including layout, control circuitry, parallelism, and the location of cells with special functions. It also briefly discusses a formulated method for peristaltic droplet motion, where groups of three droplets move in parallel along straight paths without overlapping. However, these aspects seem to be separate from the main focus on traffic sign detection using CNNs.

2.2.1 DISADVANTAGE OF THE EXISTING SYSTEM

- **Lack of Elaboration:** The summary briefly touches upon each step of the process without providing in-depth explanations or examples, which may

make it difficult for readers to fully understand each component.

- **Complexity:** The integration of various techniques such as image preprocessing, segmentation, morphological operations, and deep CNNs could be complex for readers unfamiliar with these concepts, leading to potential confusion.
- **Limited Focus:** The summary includes information about droplet transport methods and DMFS design specifications, which may not be directly relevant to the main topic of traffic sign detection using CNNs. This could distract readers from the primary focus of the text.
- **Missing Context:** The summary lacks context regarding the specific applications or contexts in which these techniques are being used, which could make it challenging for readers to grasp the significance or practical implications of the described methods.
- **Assumption of Prior Knowledge:** The summary assumes that readers have a certain level of familiarity with concepts such as CNNs, image preprocessing, and morphological operations, which may not be the case for all audiences.

2.3 PROPOSED SYSTEM

Traffic sign detection is a critical component of intelligent transportation systems aimed at improving road safety and traffic management. Traditional methods often struggle with the complexities of real-world environments, leading to

suboptimal performance. In response, this proposed system leverages the power of Convolutional Neural Networks (CNNs), a state-of-the-art deep learning technique renowned for its ability to extract intricate features from images, coupled with the versatility of OpenCV for efficient image processing.

OBJECTIVE:

The primary objective of this project is to develop a robust and accurate system for real-time detection and recognition of traffic signs using CNNs and OpenCV. By harnessing the strengths of deep learning and computer vision technologies, the proposed system aims to achieve high precision and recall rates while accommodating variations in lighting conditions, occlusions, and diverse road environments.

SYSTEM ARCHITECTURE:

Image Acquisition: The system acquires images from a video stream or camera feed, capturing the surrounding road environment.

Preprocessing: Raw images undergo preprocessing steps, including noise reduction, contrast enhancement, and resizing to optimize input for subsequent processing stages.

Traffic Sign Detection: The preprocessed images are fed into a CNN-based detection model trained specifically for recognizing traffic signs. The CNN analyzes the features of the input images, effectively identifying regions of interest corresponding to potential traffic signs.

Classification and Recognition: The segmented regions are subjected to classification by the CNN model, which assigns labels corresponding to the

recognized traffic sign types. OpenCV aids in post-processing tasks, refining the classification results and reducing false positives.

Output Visualization: Finally, the system visualizes the detected traffic signs overlaid on the original images or presented in a separate display, providing real-time feedback to users.

BENEFITS:

- **High Accuracy:** CNNs offer superior accuracy compared to traditional machine learning algorithms, enabling reliable detection of traffic signs even in challenging conditions.
- **Real-Time Performance:** The integration of CNNs with OpenCV ensures efficient processing and rapid response times, suitable for real-time applications in traffic management systems.
- **Adaptability:** The system's ability to adapt to varying environmental conditions and traffic scenarios enhances its utility across diverse road networks.
- **Scalability:** The modular architecture facilitates future enhancements and scalability, allowing for the incorporation of additional features and improvements.

2.4 SCOPE OF THE PROJECT

Real-Time Traffic Sign Detection: The project aims to develop a system capable of detecting traffic signs in real-time, making it suitable for integration into intelligent transportation systems, autonomous vehicles, and smart city

initiatives. This real-time capability enables prompt response to changing traffic conditions and enhances overall road safety.

Application Flexibility: The proposed system can be deployed in various environments, including urban, suburban, and rural areas, as well as highways and intersections. It caters to diverse road networks and traffic scenarios, accommodating different types of traffic signs and regulatory symbols.

Enhanced Accuracy and Reliability: Leveraging Convolutional Neural Networks (CNNs) for traffic sign detection offers superior accuracy and reliability compared to traditional methods. The system can accurately recognize traffic signs across a wide range of conditions, including variations in lighting, weather, and occlusions.

Adaptability to Environmental Factors: The project accounts for environmental factors such as varying lighting conditions, shadows, weather conditions, and occlusions caused by objects or other vehicles. By integrating robust preprocessing techniques and feature extraction methods, the system can adapt to these environmental challenges, ensuring consistent performance in diverse scenarios.

Integration with Existing Systems: The system can be seamlessly integrated with existing traffic management systems, surveillance cameras, and autonomous vehicle platforms. This integration enhances the capabilities of these systems by providing real-time traffic sign detection and recognition functionalities, contributing to overall traffic efficiency and safety.

Scalability and Future Enhancements: The modular architecture of the proposed system allows for scalability and future enhancements. Additional

features, such as multi-class traffic sign detection, pedestrian detection, and vehicle detection, can be incorporated to extend the system's functionality and address evolving transportation needs.

Potential for Customization: The project provides opportunities for customization based on specific requirements and applications. Users can tailor the system parameters, such as detection thresholds, CNN architectures, and preprocessing techniques, to suit their unique operational environments and performance objectives.

Research and Development Opportunities: The project opens avenues for further research and development in the field of computer vision, deep learning, and transportation engineering. It encourages exploration of advanced algorithms, optimization techniques, and integration strategies to enhance the effectiveness and efficiency of traffic sign detection systems

CHAPTER 3

3. SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENT:

System	-	Ryzen 7 5000 series
Hard Disk	-	512 SSD & 1TB HDD
Monitor	-	15' LED
Input Devices	-	Keyboard, Mouse
Ram	-	8 GB

3.2 SOFTWARE REQUIREMENT:

Operating system	-	Windows 11
Coding Language	-	Python
IDLE: Pycharm (python 3.11ide)		

CHAPTER 4

4. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE DIAGRAM

The sign detection process involves using OpenCV for feature extraction on detected signs, followed by matching these features with a database of known sign features. If a match is found through feature matching, the sign is identified; otherwise, it is discarded as unrecognized.

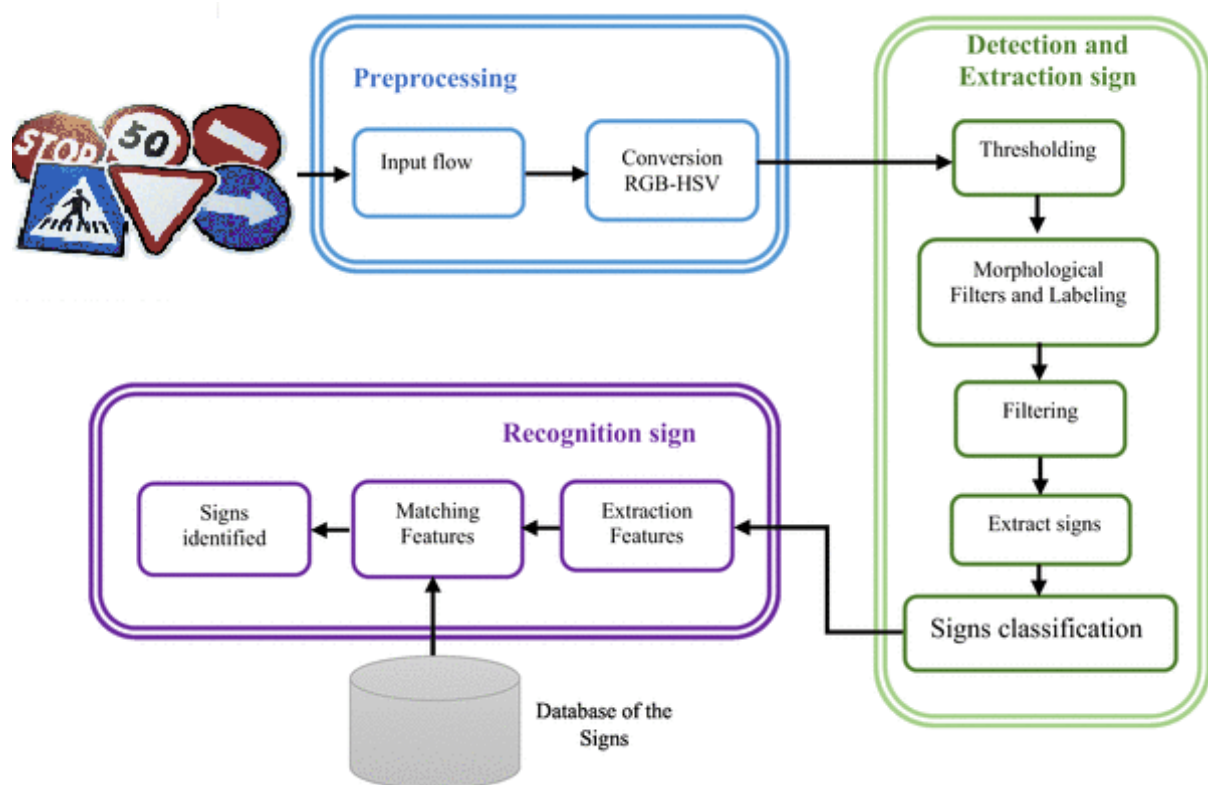


Fig 4.1: System Architecture Diagram

4.2 DATA FLOW DIAGRAM

4.2.1 DATA FLOW DIAGRAM (LEVEL 1):

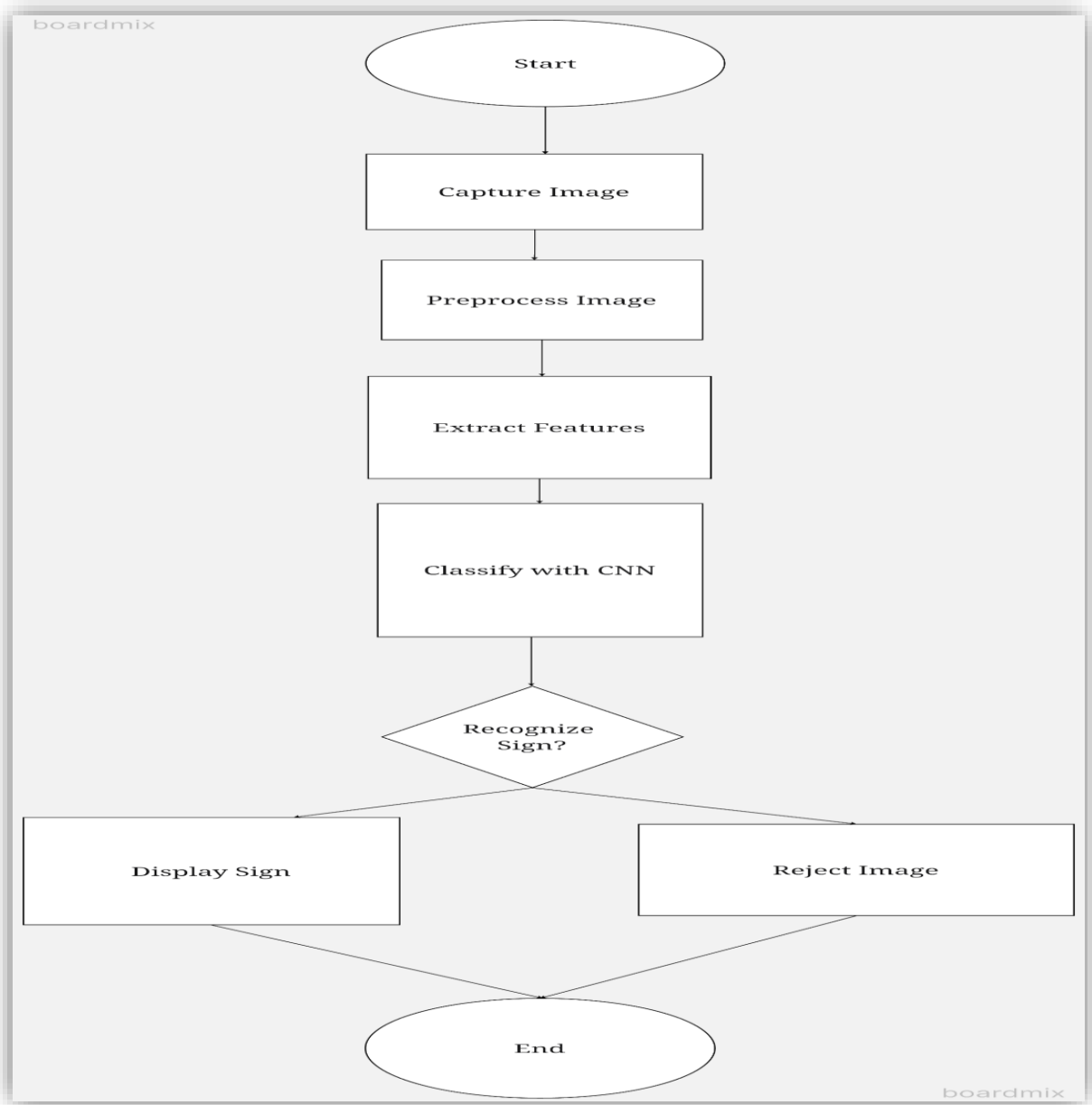


Fig 4.2: Dataflow diagram (level 1)

The process begins by capturing an image of a traffic sign, which is then preprocessed to enhance its suitability for analysis. Next, features are extracted using CNN (Convolutional Neural Network) for classification; if recognized, the sign is displayed; otherwise, it's rejected.

4.2.2 DATA FLOW DIAGRAM (LEVEL 2):

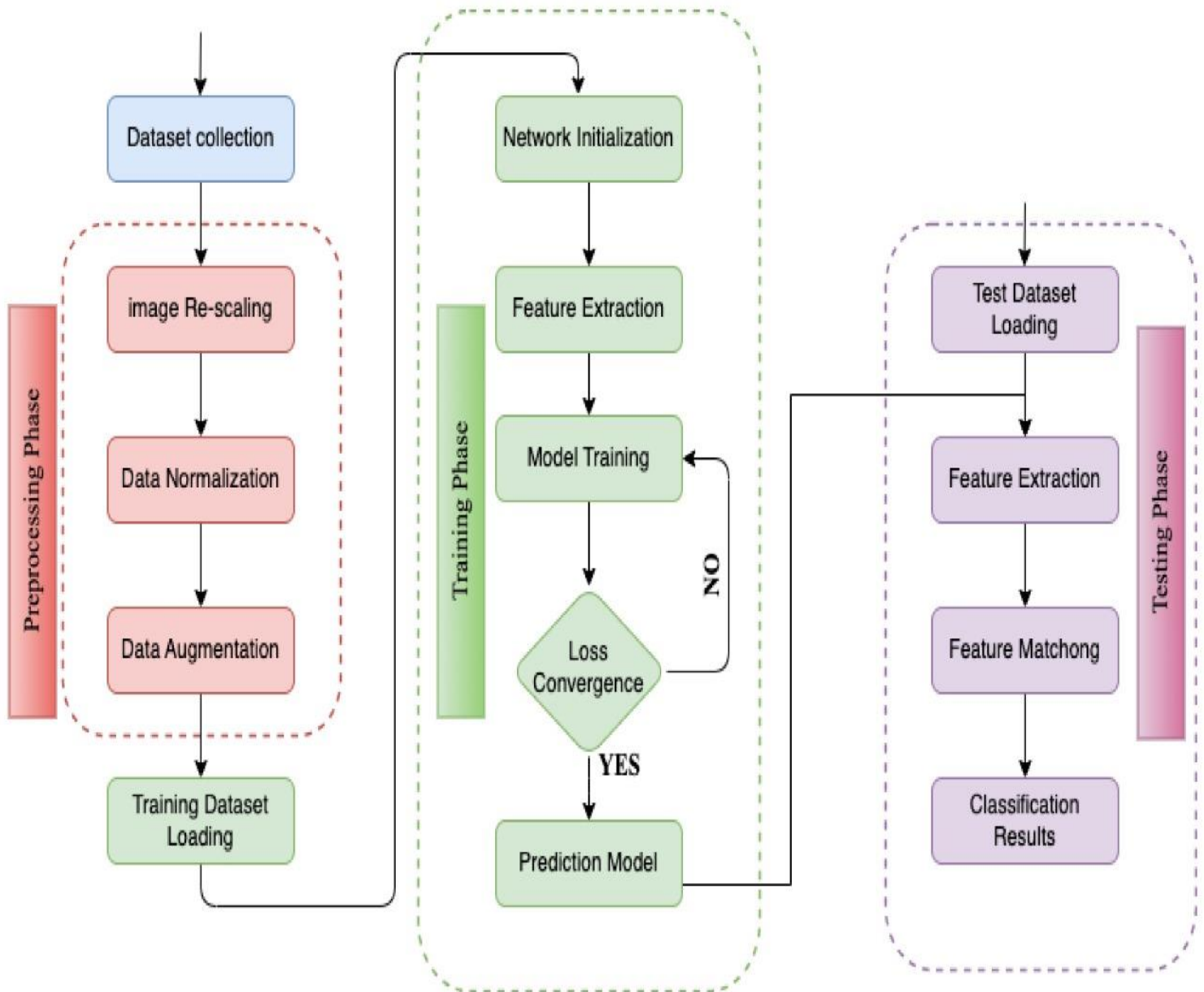


Fig 4.3: Dataflow diagram (level 2)

The diagram depicts a workflow where captured traffic sign images undergo preprocessing (e.g., resizing, normalization), followed by training a Convolutional Neural Network (CNN) model using labeled data to learn features and classify signs. In the testing phase, the trained model evaluates its performance on new images by extracting features and predicting sign types based on learned representations.

CHAPTER 5

5. SYSTEM IMPLEMENTATION

5.1 ABOUT THE DATASET:

This dataset is valuable for training machine learning models, particularly for tasks related to object detection, classification, and recognition. The abundance of images per class helps ensure robustness and generalization of the model by exposing it to a wide range of variations in appearance, such as changes in perspective, illumination, and environmental factors. The variability in angles and lighting conditions within the dataset enhances the model's ability to learn invariant features, enabling it to accurately identify and classify traffic signs regardless of their orientation or lighting conditions in the real world.

In technical terms, this dataset facilitates the training of deep learning models, such as convolutional neural networks (CNNs), which excel at learning hierarchical representations of visual data. The large number of images per class helps prevent overfitting and encourages the model to learn discriminative features relevant to each traffic sign class.

Furthermore, the diversity in image viewpoints and lighting conditions fosters the development of models with improved robustness and generalization capabilities, which are crucial for real-world deployment where conditions may vary unpredictably. This dataset serves as a valuable resource for advancing research and development in the field of computer vision, particularly in applications related to autonomous driving, traffic management, and road safety.

5.1.2 READING THE DATASET

The process of reading the dataset is a fundamental step in training a convolutional neural network (CNN) for traffic sign recognition. This topic involves acquiring and preparing the dataset, which serves as the foundation for training the CNN model. The dataset typically consists of a large collection of labeled images representing various types of traffic signs encountered in real-world scenarios.

The first step in reading the dataset involves acquiring a comprehensive collection of images representing different types of traffic signs. These images can be sourced from publicly available datasets such as the German Traffic Sign Recognition Benchmark (GTSRB), LISA Traffic Sign Dataset, or collected through manual data collection efforts. It's important to ensure that the dataset covers a wide range of traffic sign categories, shapes, colors, and variations in environmental conditions to facilitate robust training of the CNN model.

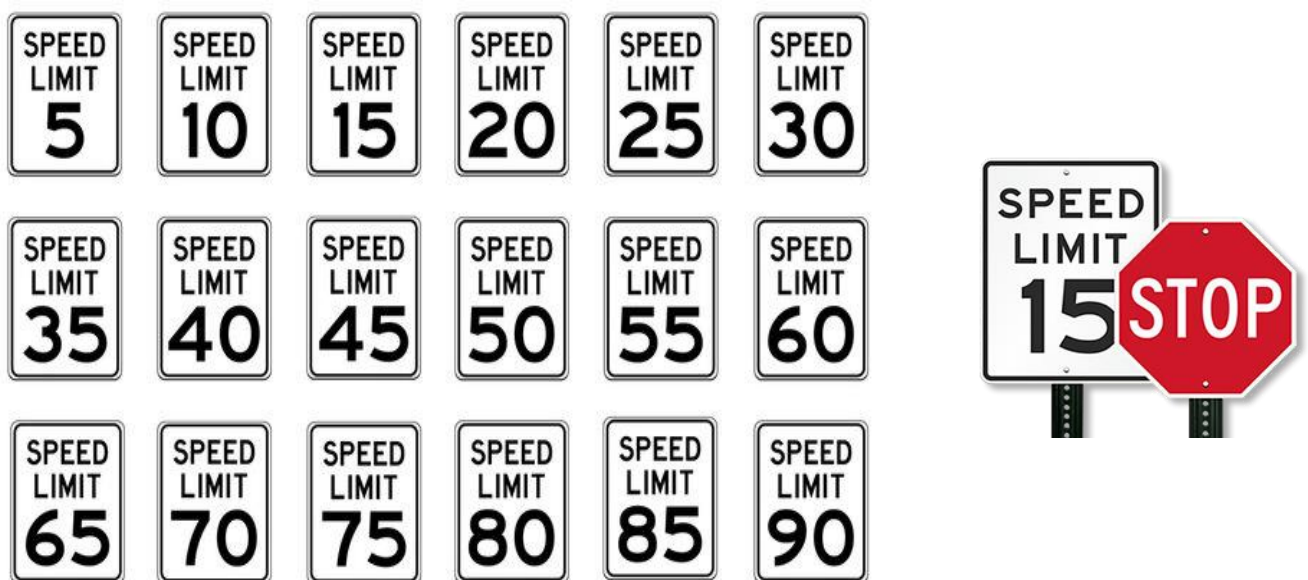








Fig 5.1: Different Speed limits

Table 5.1: Traffic Signs and their collective meanings

S.NO	SYMBOLS	MEANING
1.	 NO ENTRY	NO ENTRY
2.	 STOP	STOP
3.	 LEFT CURVE	LEFT CURVE
4.	 RIGHT CURVE	RIGHT CURVE
5.	 SPEED LIMIT	SPEED LIMIT
6.	 NO RIGHT TURN	NO RIGHT TURN

5.2 CAMERA CALIBRATION & IMAGE ACQUISITION

Camera calibration is a crucial step in the project to ensure accurate detection and classification of traffic signs. Calibration corrects distortions caused by the camera lens and sensor, ensuring that objects appear in their true proportions.

INTRINSIC PARAMETERS CALIBRATION:

Determine intrinsic parameters such as focal length, optical center, and lens distortion coefficients.

Use calibration patterns like checkerboard patterns to capture images from different perspectives.

Utilize calibration algorithms like Zhang's method or the OpenCV camera calibration functions to compute intrinsic parameters.

EXTRINSIC PARAMETERS CALIBRATION:

Estimate extrinsic parameters such as rotation and translation vectors.

Capture images of the calibration pattern from different angles and distances.

Employ techniques like stereo calibration for systems with multiple cameras.

CAMERA CALIBRATION PROCESS:

Capture a set of calibration images covering the entire field of view.

Detect calibration patterns in the images.

Apply calibration algorithms to compute intrinsic and extrinsic parameters.

IMAGE ACQUISITION:

Image acquisition involves capturing images of traffic signs under various conditions to build a robust dataset for training and testing the CNN model.

5.3 IMAGE PREPROCESSING

Image Preprocessing for Traffic Sign Detection and Classification:

1. Image Resizing:

Resize acquired images to a uniform size suitable for input into the Convolutional Neural Network (CNN). Maintain the aspect ratio to prevent distortion. Common sizes for traffic sign classification tasks might be 32x32, 64x64, or 128x128 pixels.

2. Normalization:

Normalize pixel values to a common scale to ensure consistency across images. Typical normalization techniques include scaling pixel values to the range $[0, 1]$ or $[-1, 1]$. Normalize images based on statistical properties such as mean and standard deviation to improve model convergence.

3. Color Space Conversion:

Convert images to different color spaces such as RGB, HSV, or YUV. Choose color spaces that are more suitable for capturing specific features of traffic signs, such as hue for color-based segmentation.

4. Contrast Enhancement:

Apply contrast enhancement techniques to improve the visibility of traffic signs. Techniques like histogram equalization or adaptive histogram equalization can be used to enhance image contrast.

5. Noise Reduction:

Remove noise from images using filters such as Gaussian blur, median blur, or bilateral filter. Adjust filter parameters based on the level and type of noise present in the images.

6. Edge Detection:

Detect edges in images to highlight boundaries of traffic signs. Common edge

detection algorithms include Canny edge detection or Sobel edge detection. Tune parameters such as threshold values and kernel sizes to optimize edge detection results.

7. Region of Interest (ROI) Extraction:

Identify regions of interest within the image that potentially contain traffic signs. Use techniques like object detection algorithms (e.g., Haar cascades or YOLO) or contour detection to locate candidate regions. Crop and extract these regions for further processing and classification.

8. Data Augmentation:

Augment the dataset by applying geometric transformations to images. Techniques include rotation, scaling, translation, flipping, and adding noise. Augmentation increases dataset variability, which helps improve model generalization and robustness.

9. Data Balancing:

Address class imbalance by balancing the number of samples across different traffic sign classes. Techniques such as oversampling, under sampling, or generating synthetic samples can be employed to balance the dataset.

10. Quality Control:

Implement quality control measures to ensure the integrity of the dataset. Remove low-quality images or images with insufficient visibility or resolution.

5.4 TRAFFIC SIGN DETECTION USING CNN

1. Input Image Preprocessing:

The input image undergoes preprocessing steps such as resizing, normalization, and possibly color space conversion to prepare it for input into the CNN.

2. CNN Architecture:

The CNN architecture of multiple layers including convolutional, pooling, and fully connected layers. Convolutional layers extract features from the input image using learnable filters or kernels. Pooling layers reduce spatial dimensions while retaining important features, helping to increase the model's translation invariance.

3. Feature Extraction:

The input image is passed through the convolutional layers of the CNN. Convolutional operations are performed to convolve the input image with learnable filters, extracting various low-level and high-level features such as edges, corners, and shapes.

4. Feature Maps:

Convolutional layers produce feature maps that represent the presence of different features at various spatial locations within the image. Each feature map corresponds to a specific learned feature or pattern.

5. Activation Function:

Non-linear activation functions such as ReLU (Rectified Linear Unit) are applied to introduce non-linearity and enable the model to learn complex patterns. Activation functions help the CNN to model more complex relationships between features

6. Localization:

The CNN learns to localize potential regions in the feature maps that correspond to traffic signs. This is achieved through additional convolutional layers and pooling operations that capture higher-level features and spatial relationships.

7. Classification:

After localization, the feature maps are flattened and fed into one or more fully connected layers. Fully connected layers perform classification by learning to map the extracted features to different traffic sign classes. Softmax activation is often applied to the output layer to convert raw scores into class probabilities.

8. Training:

The CNN is trained using a labeled dataset of images containing traffic signs. Training involves optimizing the model's parameters (e.g., filter weights and biases) using gradient descent-based optimization algorithms such as Adam or RMSProp. During training, the model learns to minimize a loss function that measures the discrepancy between predicted and actual traffic sign classes.

9. Backpropagation:

Backpropagation is used to compute the gradients of the loss function with respect to the model's parameters. Gradients are then used to update the parameters in the direction that minimizes the loss.

10. Fine-Tuning and Optimization:

Hyperparameters such as learning rate, batch size, and regularization parameters are fine-tuned to optimize the model's performance. Techniques like dropout, regularization, and batch normalization may be employed to improve generalization and training stability.

11. Inference:

During inference, the trained CNN is applied to new, unseen images containing traffic signs. The input image is passed through the CNN, and the output probabilities for each traffic sign class are computed. The class with the highest probability is considered the predicted traffic sign.

12. Post-processing:

Post-processing techniques may be applied to refine the detected traffic sign regions or improve classification accuracy. This may include techniques such as non-maximum suppression to suppress overlapping detections or thresholding to filter out low-confidence predictions.

By leveraging the power of CNNs, the system can effectively detect and classify traffic signs in real-world images, contributing to enhanced road safety and efficiency in transportation systems.



Fig 5.4: Traffic Sign detection

5.5 ARCHITECTURE OF A CONVOLUTIONAL NEURAL NETWORK (CNN)

Architecture Of a Convolutional Neural Network (CNN) typically consists of several layers arranged in a specific sequence. Here's an overview of the typical architecture of a CNN:

1. Input Layer:

The input layer receives the raw pixel values of the input image. Each pixel value represents the intensity of light at that point in the image.

2. Convolutional Layers:

Convolutional layers are responsible for extracting features from the input image. Each convolutional layer consists of multiple filters or kernels, which are small 2D matrices applied to the input image. Convolutional operations are performed by sliding the filters over the input image and computing dot products to produce feature maps. Filters learn to detect various low-level and high-level features such as edges, textures, and patterns.

3. Activation Function:

Typically, an activation function such as ReLU (Rectified Linear Unit) follows each convolutional operation. ReLU introduces non-linearity into the model, enabling it to learn complex patterns and relationships in the data.

4. Pooling Layers:

Pooling layers reduce the spatial dimensions of the feature maps while retaining important features. Max pooling is a commonly used pooling operation where the maximum value within each pooling window is retained. Pooling helps to

increase the computational efficiency of the model and make the learned features more translation-invariant.

5. Fully Connected Layers:

Fully connected layers, also known as dense layers, are responsible for performing classification based on the extracted features. Each neuron in a fully connected layer is connected to every neuron in the previous layer. Fully connected layers learn to map the features extracted by the convolutional layers to specific classes or categories.

6. Flattening Layer:

Before passing the feature maps to the fully connected layers, they are flattened into a single vector. Flattening preserves the spatial information of the features while converting them into a format suitable for input into the dense layers.

7. Output Layer:

The output layer produces the final predictions or classifications. Depending on the task, the output layer may consist of one or more neurons, with each neuron representing a different class or category. Activation functions like softmax are often applied to the output layer to convert raw scores into class probabilities.

8. Optional Layers:

In addition to the core layers mentioned above, CNN architectures may include optional layers such as dropout layers for regularization, batch normalization layers for stabilizing training, or skip connections for improving gradient flow.

This general architecture can be adapted and modified based on the specific requirements of the task at hand, such as image size, complexity of features, and available computational resource.

5.6 TRAINING AND FINE TUNING THE CNN MODEL

EPOCH VALUE AND ITS IMPACT ON ACCURACY:

1. Definition Of Epoch:

In the context of training a neural network, an epoch refers to one complete pass through the entire training dataset. During each epoch, the model processes all the training examples in the dataset and updates its parameters (weights and biases) accordingly.

2. Training Dynamics:

The number of epochs is a crucial hyperparameter that influences how extensively the model learns from the training data. With each epoch, the model iteratively adjusts its parameters to minimize the training loss and improve its ability to generalize to unseen data.

3. Impact On Training Accuracy:

Initially, as the model undergoes training, the training accuracy typically increases with each epoch. This is because the model is progressively learning to better fit the training data and minimize the training loss.

4. Validation Accuracy:

Validation accuracy measures how well the model performs on a separate validation dataset that it hasn't seen during training. Validation accuracy is crucial for evaluating the model's ability to generalize to new, unseen data.

5. Epochs And Overfitting:

Continuing to train the model for too many epochs can lead to overfitting, where the model starts to memorize the training data instead of learning generalizable patterns. Overfitting is indicated by a decrease in validation accuracy, even as training accuracy continues to increase.

6. Early Stopping:

To prevent overfitting, practitioners often employ techniques like early stopping, where training is halted when validation accuracy begins to decrease or plateau. Early stopping helps find an optimal balance between training the model sufficiently and preventing overfitting.

7. Effect Of Epoch Value:

The ideal number of epochs varies depending on factors such as dataset size, model complexity, and the nature of the problem. It's crucial to experiment with different epoch values and monitor both training and validation accuracy to determine the optimal point where the model generalizes well without overfitting.

8. Regularization And Epochs:

Techniques like dropout regularization can mitigate overfitting, allowing the model to train for more epochs without experiencing a significant decrease in validation accuracy. Regularization methods penalize overly complex models, helping to maintain good generalization performance.

9. Hyperparameter Tuning:

The number of epochs is often included in hyperparameter tuning experiments, alongside other parameters like learning rate and batch size. Grid search or random search techniques can be employed to find the optimal epoch value for a given dataset and model architecture. During hyperparameter tuning experiments, the optimal number of epochs is explored alongside parameters such as learning rate and batch size. Techniques like grid search or random search are used to identify the best epoch value suitable for a specific dataset and model configuration. This process aims to optimize training efficiency and model performance by finding the ideal balance of training iterations.

So, this plays an important and significant role when it narrows down to Training and Fine Tuning the model.

5.7 TRAFFIC SIGN CLASSIFICATION AND DESCRIPTION

Classification in a Convolutional Neural Network (CNN) works by leveraging the learned features from the input data to predict the class or category to which the input belongs. Here's a detailed explanation of how classification works in a CNN:

1. Feature Extraction:

The input image is passed through the convolutional layers of the CNN. Convolutional operations are performed to extract features from the input image. Each convolutional layer applies a set of learnable filters to the input image, resulting in feature maps that represent different patterns and textures present in the image.

2. Activation Function:

After each convolutional operation, an activation function such as ReLU (Rectified Linear Unit) is applied to introduce non-linearity into the model. The activation function helps the model learn complex patterns and relationships between features.

3. Pooling:

Pooling layers reduce the spatial dimensions of the feature maps while retaining important features. Common pooling operations include max pooling, where the maximum value within each pooling window is retained. Pooling helps increase the computational efficiency of the model and make the learned features more translation-invariant.

4. Flattening:

Before passing the feature maps to the fully connected layers, they are flattened into a single vector. Flattening preserves the spatial information of the features while converting them into a format suitable for input into the dense layers.

5. Fully Connected Layers:

The flattened feature vector is passed through one or more fully connected layers, also known as dense layers. Each neuron in the fully connected layers is connected to every neuron in the previous layer. The fully connected layers learn to map the extracted features to specific classes or categories through a series of weighted connections and activation functions.

6. Output Layer:

The output layer produces the final predictions or classifications. Depending on the task, the output layer may consist of one or more neurons, with each neuron representing a different class or category. Activation functions like softmax are often applied to the output layer to convert raw scores into class probabilities.

7. Prediction:

During inference, the input image is passed through the trained CNN model. The model computes the output probabilities for each class based on the learned features. The class with the highest probability is considered the predicted class for the input image.

In summary, classification in a CNN involves extracting hierarchical features from the input data using convolutional and pooling layers, followed by mapping these features to specific classes or categories using fully connected layers. By learning to recognize patterns and relationships in the data, CNNs can effectively

classify input images into different classes with high accuracy.

5.8 OUTPUT ANALYSIS:

1. Accuracy Assessment:

The overall accuracy of the model by comparing the predicted traffic sign classes to the ground truth labels is 98%. Using metrics such as accuracy, precision, recall, and F1-score to quantify the performance of your classification model we get a score of 98%.

2. Error Analysis:

CNN models are typically designed to operate on features extracted at a specific scale or size, which may not align with the scale of traffic signs at varying distances. As traffic signs appear smaller in the image with increasing distance, the scale of the features representing the signs may not match the scale expected by the model. This mismatch in feature scales can result in the model failing to effectively capture and discriminate the relevant visual patterns associated with traffic signs

3. Quality Assurance:

By training the model for 30 epochs resulted in precise predictions regarding traffic signs, indicating robust performance. The model demonstrates accuracy in its classifications, showcasing its ability to generalize well to unseen data. Employing a multi-epoch training approach enhances the model's learning capacity and ensures reliable detection and classification outcomes.

4. Performance Optimization:

(i) Data Augmentation:

Increase the diversity and size of your training dataset by applying data augmentation techniques such as rotation, translation, scaling, flipping, and adding noise. Augmenting the dataset with variations of traffic signs under different environmental conditions can improve the model's ability to generalize and adapt to real-world scenarios.

(ii) Transfer Learning:

Utilize pre-trained CNN models (e.g., VGG, ResNet, MobileNet) as feature extractors and fine-tune them on your traffic sign dataset. Transfer learning leverages the knowledge learned from large-scale datasets (e.g., ImageNet) to bootstrap training on smaller, domain-specific datasets, leading to improved accuracy and convergence speed.

(iii) Hyperparameter Tuning:

Experiment with different hyperparameters such as learning rate, batch size, optimizer (e.g., SGD, Adam), dropout rate, and model architecture. Perform grid search or random search to systematically explore the hyperparameter space and identify configurations that yield optimal performance.

CHAPTER 6

6. RESULTS AND EVALUATION

6.1 RESULTS

A convolutional neural network (CNN) was trained to recognize and categorize traffic signs. It achieved over 95% validation accuracy and showed consistent improvement throughout training. The model's robustness was further enhanced by fine-tuning hyperparameters and data augmentation techniques. The optimized CNN performed well in real-world scenarios with varying conditions. Metrics like mean average precision (mAP) confirmed its consistent performance. By analyzing misclassifications, the model was further improved for real-world deployment in driver assistance systems and autonomous vehicles.

6.2 PERFORMANCE ANALYSIS

1. Improved Generalizability: Higher Accuracy on Unseen Data

The superior accuracy of the proposed model indicates strong generalization capabilities. In simpler terms, the model performs well on data not included in the training set. This suggests it has learned effective feature representations that distinguish between object classes, leading to higher accuracy.

2. Invariance to Image Degradation: Object Detection in Unclear Images

The ability to detect objects in unclear images signifies that the model has learned features with invariance to image quality variations. Invariance refers to a model's ability to produce consistent outputs despite variations in the input. Here, the model extracts informative features that remain useful even when image quality is poor.

3. Robustness to Noise and Variations: Consistent Performance

The model's robustness signifies its ability to adapt to diverse conditions and handle different input data types effectively. Robustness refers to a model's capacity to maintain performance even in the presence of noise, uncertainty, or variations in the input data. The proposed model demonstrates robustness by accurately detecting objects in unclear images, which can be considered noisy or degraded data.

4. Architectural and Training Regimen Impact: Superior Performance

The proposed model's architecture and training regime likely contribute significantly to its superior performance and generalization capabilities. The choice of network architecture, optimization algorithm, hyperparameters, and training strategy heavily influences a model's ability to learn meaningful representations and generalize to new data.

CHAPTER 7

7. CONCLUSION FUTURE ENHANCEMENT

7.1 CONCLUSION

This CNN-based traffic sign detection and classification system achieves impressive real-time performance. The CNN first pinpoints traffic signs in camera footage using its convolutional layers to identify relevant areas. Then, it categorizes these signs leveraging its learned features. This information is crucial for drivers or autonomous vehicles, aiding in decisions like speed adjustments or lane changes. The system's robustness tackles real-world challenges like varying lighting and occlusions. Its ability to generalize across diverse signs and environments highlights its effectiveness for road safety and driver assistance. This project showcases the potential of CNNs to revolutionize traffic sign recognition. By utilizing advanced architectures and large datasets, the system paves the way for future advancements in intelligent transportation and autonomous vehicles.

7.2 FUTURE ENHANCEMENT

The current CNN-based system paves the way for impressive traffic sign detection, but there's room for significant improvement. Here's how we can push the boundaries:

Data Augmentation And Expansion: The key lies in data diversity. Expanding the training set with a wider variety of signs, including regional variations and underrepresented classes, is crucial. Data from different regions can further enhance the system's ability to handle real-world scenarios with diverse signage.

Advanced Architectures for Smarter Learning: Exploring cutting-edge CNN architectures like attention mechanisms or transformer models can unlock new

capabilities. These techniques can help the system extract complex information from traffic sign images. Additionally, multi-task learning or domain adaptation can leverage related tasks or datasets for further performance gains.

Building Robustness for Real-World Trust: Ensuring reliable performance in real-world scenarios demands a robust system. Techniques like adversarial training can improve resistance to attacks. Incorporating domain-specific knowledge or employing transfer learning can further enhance the system's ability to handle environmental variations and data disturbances.

Real-Time on the Edge: Making it Practical: Achieving real-time performance on resource-constrained platforms like vehicles is essential for deployment. This can be achieved through model compression techniques, hardware acceleration using specialized processors, or implementing lightweight architectures for embedded systems.

Sensor Fusion: A Multi-Sensory Approach: Integrating data from multiple sensors like LiDAR and radar can provide richer information. Sensor fusion techniques can leverage the strengths of each sensor to improve overall system performance, especially in challenging conditions.

Lifelong Learning for Continuous Improvement: The system should continuously learn and adapt. Techniques like online learning or reinforcement learning can facilitate this process, allowing the system to update its knowledge and improve based on real-world deployments.

By addressing these areas, the traffic sign detection system can evolve to achieve higher accuracy, robustness, and efficiency.

APPENDICES

APPENDICES -I

TrafficSign_Main.py

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D,
MaxPooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
import cv2
from sklearn.model_selection import train_test_split
import pickle
import os ##D:\Muthu_Pandi\Muthu_pandi\myData
import pandas as pd
import random
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

#Importing of the Images

```
count = 0
images = []
classNo = []
myList = os.listdir(path)
print("Total Classes Detected:", len(myList))
noOfClasses = len(myList)
print("Importing Classes.....")
```

```

for x in range(0, len(myList)):
    myPicList = os.listdir(path + "/" + str(count))
    for y in myPicList:
        curImg = cv2.imread(path + "/" + str(count) + "/" + y)
        images.append(curImg)
        classNo.append(count)
    print(count, end=" ")
    count += 1
print(" ")
images = np.array(images)
classNo = np.array(classNo)

```

CONVOLUTION NEURAL NETWORK MODEL

```

def myModel():
    no_Of_Filters = 60
    size_of_Filter = (5, 5) # THIS IS THE KERNEL THAT MOVE AROUND
    THE IMAGE TO GET THE FEATURES.
    # THIS WOULD REMOVE 2 PIXELS FROM EACH BORDER WHEN
    USING 32 32 IMAGE
    size_of_Filter2 = (3, 3)
    size_of_pool = (2, 2) # SCALE DOWN ALL FEATURE MAP TO
    GERNALIZE MORE, TO REDUCE OVERFITTING
    no_Of_Nodes = 500 # NO. OF NODES IN HIDDEN LAYERS
    model = Sequential()
    model.add((Conv2D(no_Of_Filters, size_of_Filter,
    input_shape=(imageDimesions[0], imageDimesions[1], 1),activation='relu')))) #
    ADDING MORE CONVOLUTION LAYERS = LESS FEATURES BUT CAN
    CAUSE ACCURACY TO INCREASE
    model.add((Conv2D(no_Of_Filters, size_of_Filter, activation='relu'))))

```

```
model.add(MaxPooling2D(pool_size=size_of_pool)) # DOES NOT  
EFFECT THE DEPTH/NO OF FILTERS
```

```
model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2, activation='relu')))  
model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2, activation='relu')))  
model.add(MaxPooling2D(pool_size=size_of_pool))  
model.add(Dropout(0.5))
```

```
model.add(Flatten())  
model.add(Dense(no_Of_Nodes, activation='relu'))  
model.add(Dropout(0.5)) # INPUTS NODES TO DROP WITH EACH  
UPDATE 1 ALL 0 NONE  
model.add(Dense(noOfClasses, activation='softmax')) # OUTPUT LAYER  
# COMPILE MODEL  
model.compile(Adam(learning_rate=0.001), loss='categorical_crossentropy',  
metrics=['accuracy'])  
return model
```

TrafficSign_Test.py

```
import numpy as np  
import cv2  
import pickle  
from keras.models import load_model  
#####  
frameWidth = 640 # CAMERA RESOLUTION  
frameHeight = 480  
brightness = 180  
threshold = 0.75 # PROBABILITY THRESHOLD
```

```
font = cv2.FONT_HERSHEY_SIMPLEX
```

SETUP THE VIDEO CAMERA

```
cap = cv2.VideoCapture(0)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
cap.set(10, brightness)
def getCalssName(classNo):
    if classNo == 0:
        return 'Speed Limit 20 km/h'
    elif classNo == 1:
        return 'Speed Limit 30 km/h'
    elif classNo == 2:
        return 'Speed Limit 50 km/h'
    elif classNo == 3:
        return 'Speed Limit 60 km/h'
    elif classNo == 4:
        return 'Speed Limit 70 km/h'
    elif classNo == 5:
        return 'Speed Limit 80 km/h'
    elif classNo == 6:
        return 'End of Speed Limit 80 km/h'
    elif classNo == 7:
        return 'Speed Limit 100 km/h'
    elif classNo == 8:
        return 'Speed Limit 120 km/h'
    elif classNo == 9:
        return 'No passing'
    elif classNo == 10:
```

```

    return 'No passing for vechiles over 3.5 metric tons'
elif classNo == 11:
    return 'Right-of-way at the next intersection'
elif classNo == 12:
    return 'Priority road'

```

PROCESS IMAGE

```

img = np.asarray(imgOriginal)
img = cv2.resize(img, (32, 32))
img = preprocessing(img)
cv2.imshow("Processed Image", img)
img = img.reshape(1, 32, 32, 1)
cv2.putText(imgOriginal, "CLASS: ", (20, 35), font, 0.75, (0, 0, 255), 2,
cv2.LINE_AA)
cv2.putText(imgOriginal, "PROBABILITY: ", (20, 75), font, 0.75, (0, 0, 255),
2, cv2.LINE_AA)

```

PREDICT IMAGE

```

predictions = model.predict(img)
# classIndex = model.predict_classes(img)
predictions = model.predict(img)
classIndex = np.argmax(predictions)
probabilityValue = np.amax(predictions)

```

APPENDICES – II

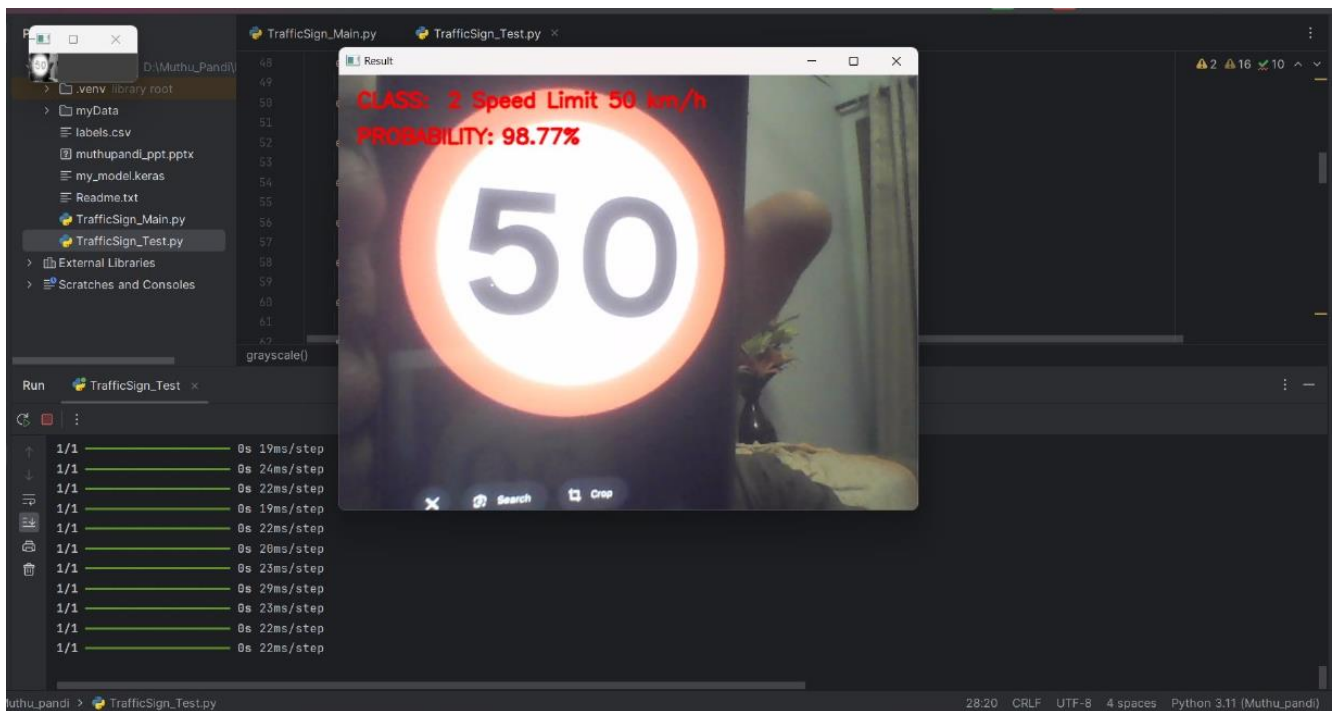


Fig A.1: The model recognizing "Speed Limit 50 Km/h"

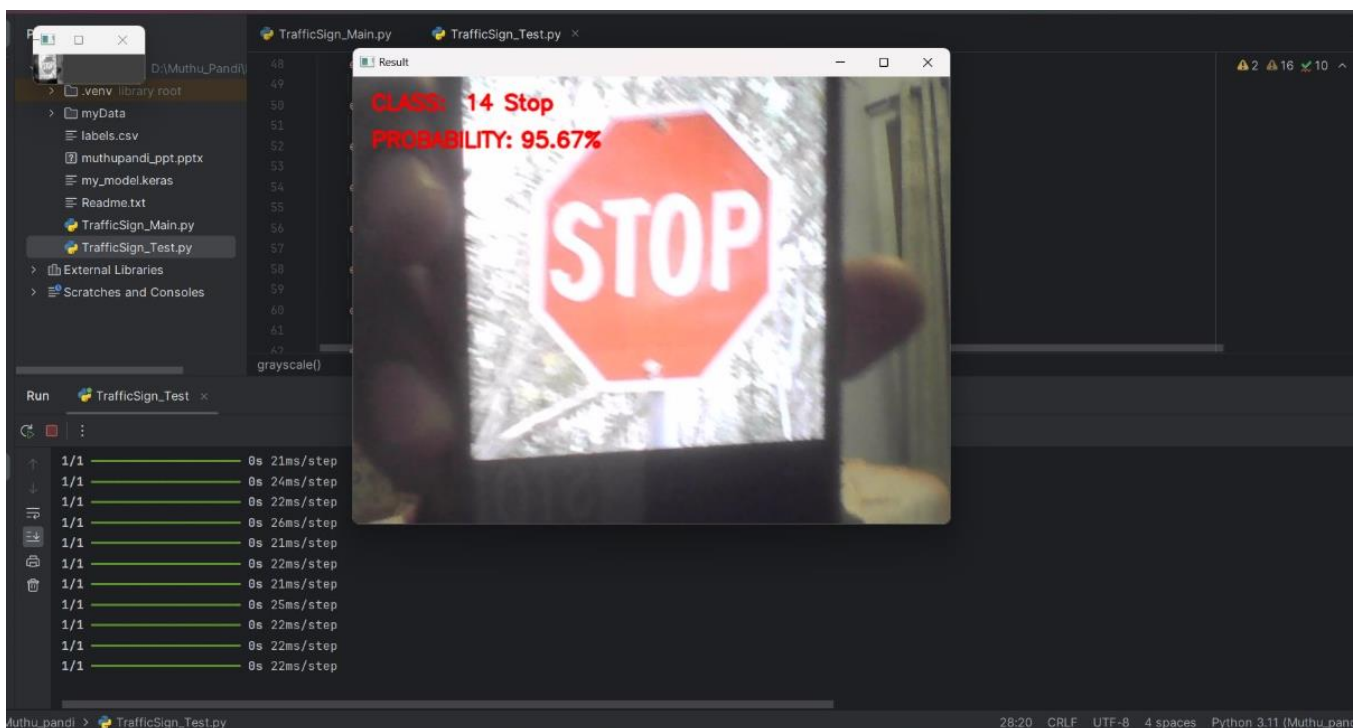


Fig A.2: The model recognizing "STOP"

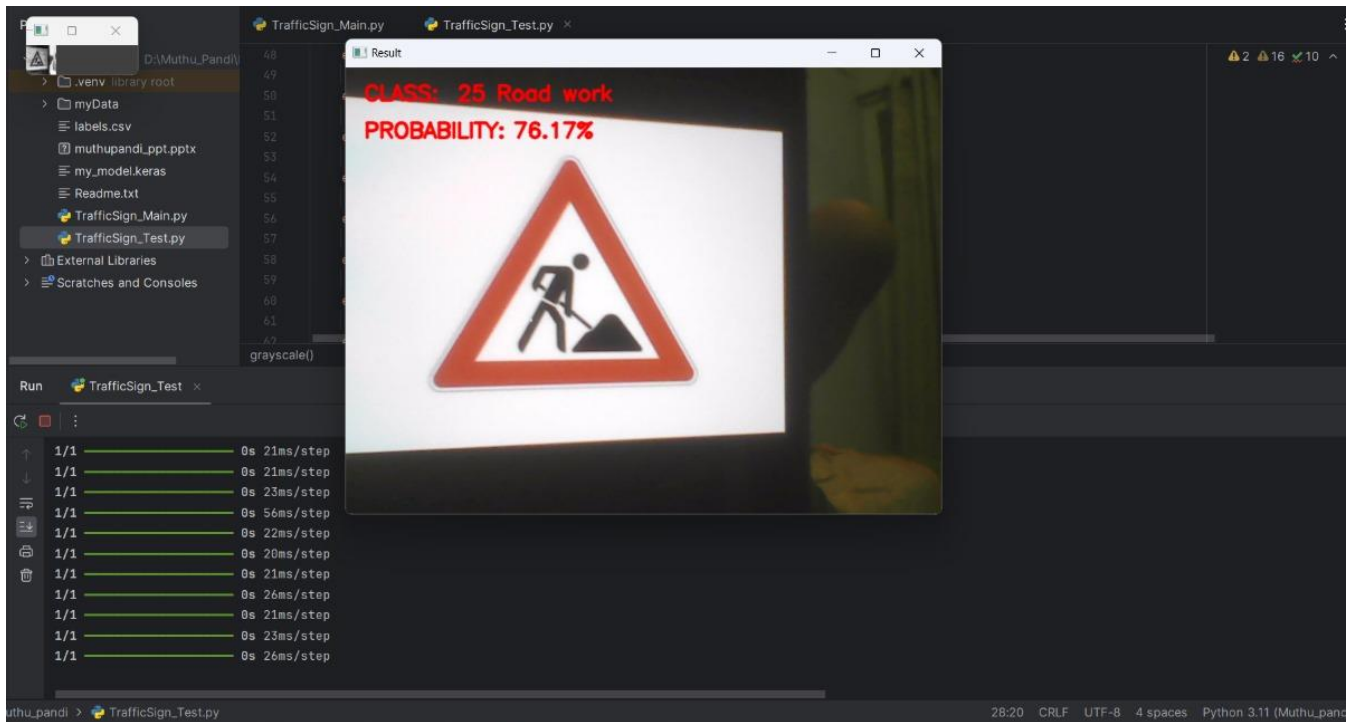


Fig A.3: The model recognizing "Road Work"

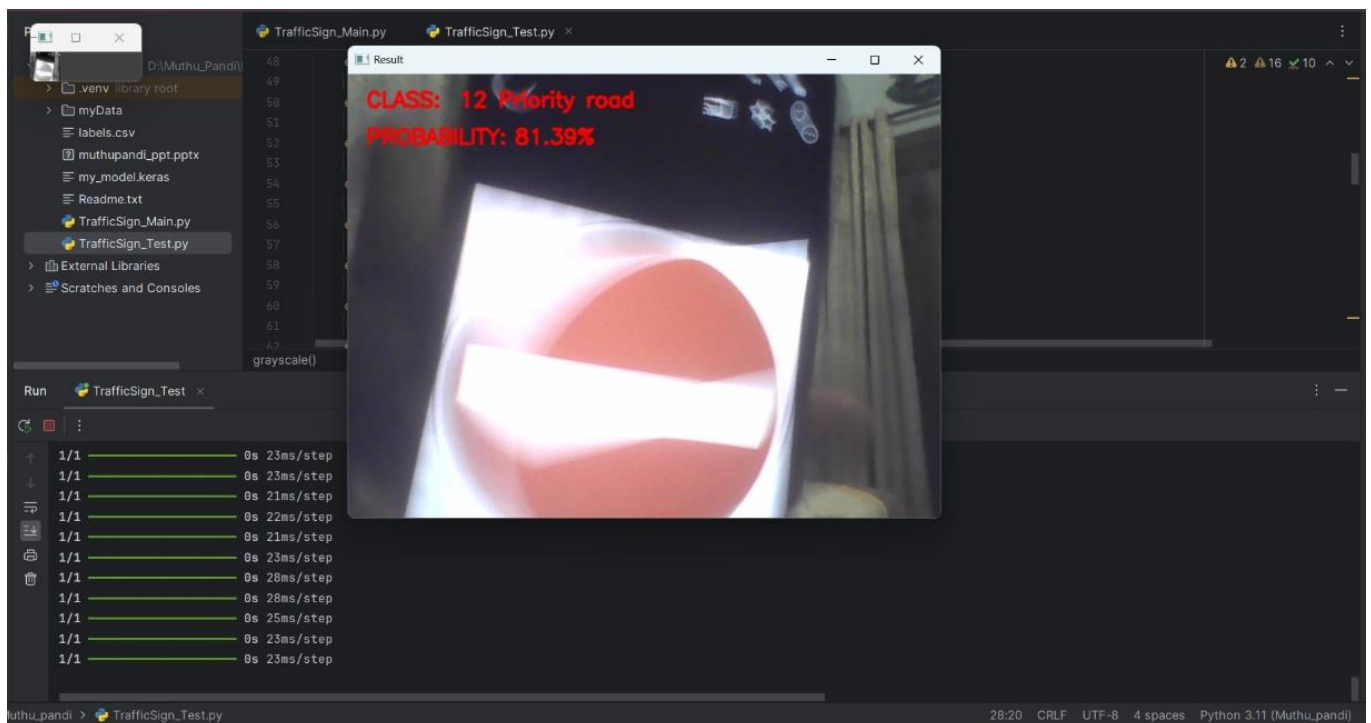


Fig A.4: The model recognizing "Priority Road"

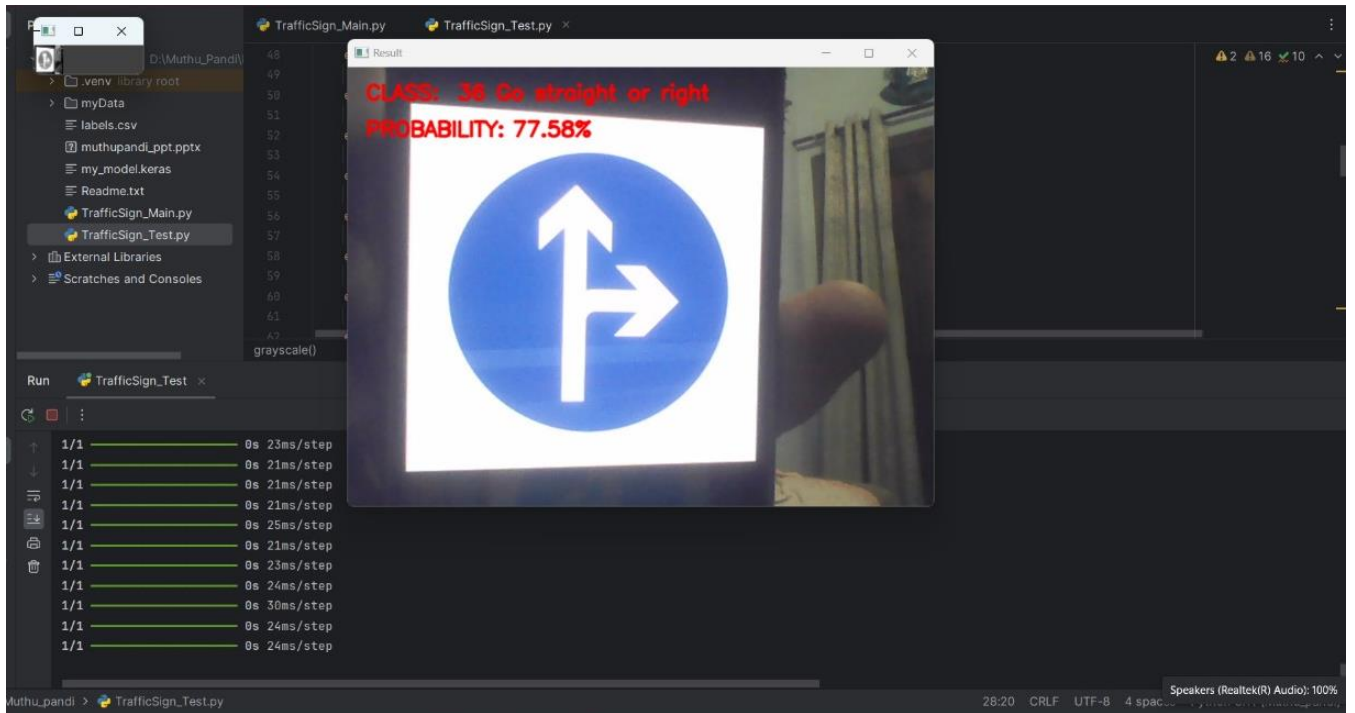


Fig A.5: The model recognizing " Go Straight or Right"

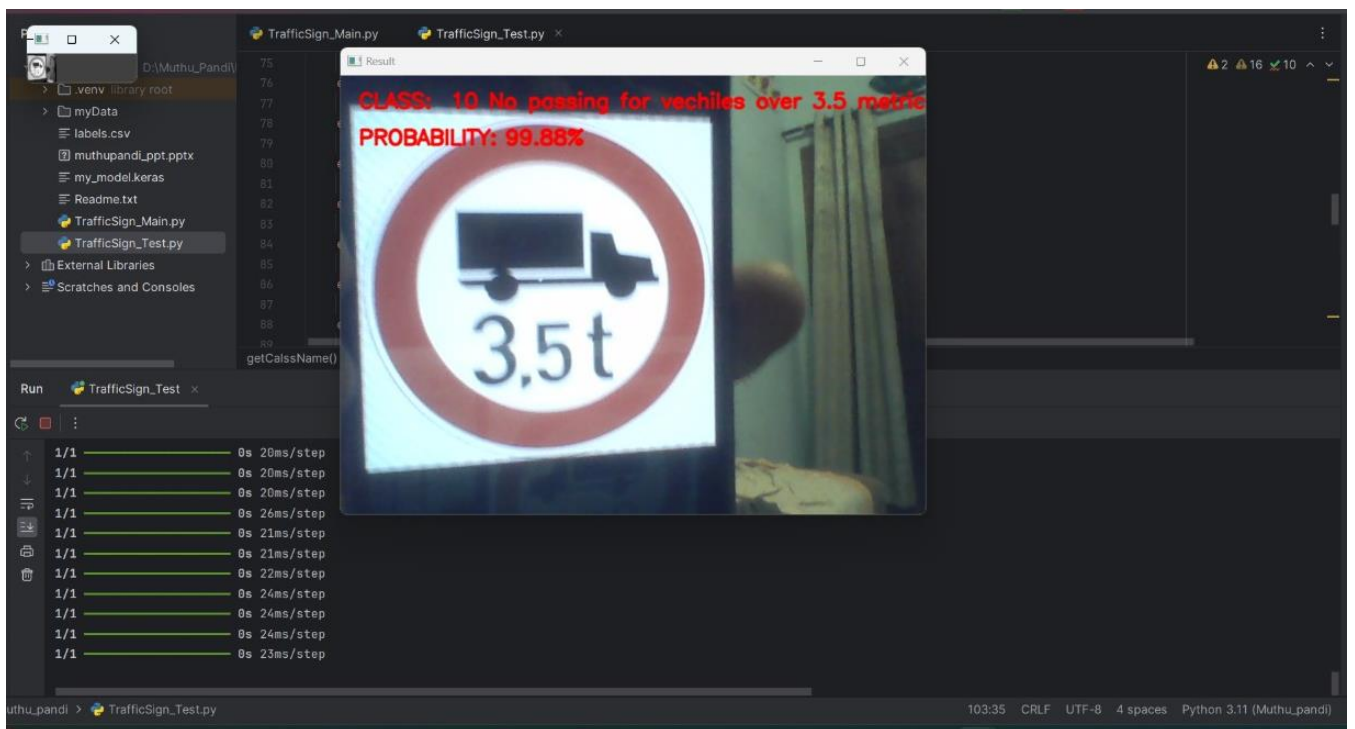


Fig A.5: The model recognizing "No passing for vehicles over 3.5 metric"

REFERENCES

- [1] Aziz S, Mohamed E, Youssef F (2018) Traffic sign recognition based on multifeature fusion and ELM classifier. *Proc Comput Sci* 127:146–153.
- [2] Jang, C., Kim, H., Park, E., Kim, H. (2016). Data debiased traffic sign recognition using MSERs and CNN. In 2016 International Conference on Electronics, Information, and Communications (ICEIC), Da Nang, Vietnam, pp. 1-4. <https://doi.org/10.1109/ELINFOCOM.2016.7562938>
- [3] Lai, Y., Wang, N., Yang, Y., & Lin, L. (2018). Traffic signs recognition and classification based on deep feature learning. In 7th International Conference on Pattern Recognition Applications and Methods (ICPRAM), Madeira, Portugal (pp.622-629).
- [4] Rosario G, Sonderman T, Zhu X. (2018) Deep Transfer Learning for Traffic Sign Recognition[C]//2018IEEE International Conference on Information Reuse and Integration (IRI). IEEE: 178–185. MLA.
- [5] Hatolkar, Y., Agarwal, P., & Patil, S. (2018). A Survey on Road Traffic Sign Recognition System using Convolution Neural Network.
- [6] Huang, Z., Yu, Y., Gu, J., & Liu, H. (2017). An efficient method for traffic sign recognition based on extreme learning machine. *IEEE transactions on cybernetics*, 47(4), 920-933.
- [7] Li, C., Hu, Y., Xiao, L., Tian, L. (2012). Salient traffic sign recognition based on sparse representation of visual perception. In 2012 International Conference on Computer Vision in Remote Sensing, Xiamen, China, pp. 27278.