ROAD SIGN DETECTION USING MACHINE LEARNING IN CNN

A PROJECT REPORT

Submitted by

MOHAMED THOUFIK U (811721104069)

SUNDARAMANICKAM N (811721104107)

THAMIJ AHAMED M (811721104113)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

May,2025

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"ROAD SIGN DETECTION USING MACHINE LEARNING IN CNN"** is the bonafide work of **MOHAMED THOUFIK U (811721104069), SUNDARAMANICKAM N (811721104107), THAMIJ AHAMED M (811721104113),** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or anyother candidate.

SIGNATURE

Dr.A.Delphin Carolina Rani M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE
K. Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

SIGNATURE

Mrs.R.Sathya M.E.,(Ph.D).,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

Submitted for the viva-voce examination held on …….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **"ROAD SIGN DETECTION USING MACHINE LERNING IN CNN"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfillment of the requirement of the award of Degree of **BACHELOR OF ENGINEERING**.

Signature

MOHAMED THOUFIK U

SUNDARAMANICKAM N

THAMIJ AHAMED M

Place: Samayapuram

Date :

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtedness to our institution, "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

We extend our sincere acknowledgment an appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering an adequate duration to complete it.

We would liketo thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.,** Principal, who gave the opportunity to frame the project to full satisfaction.

We wholeheartedly thank **Dr. A. DELPHIN CAROLINA RANI M.E., Ph.D.,** Head of the Department of **COMPUTER SCIENCE AND ENGINEERING,** for providing her encouragement in pursuing this project.

We express our deep and sincere gratitude to our project guide, **Mrs. R. SATHYA , M.E., (Ph.D).,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for her incalculable suggestions, creativity, assistance patience, which motivated us to carry out this project.

We render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our department who rendered their help during the period of the work progress.

ABSTRACT

Road sign detection using machine learning in CNN involves the application of advanced algorithms to automatically identify and classify traffic signs from images or video feeds. This task is essential for the development of autonomous driving systems, driver assistance technologies, and intelligent transportation systems. Machine learning techniques, particularly deep learning models like Convolutional Neural Networks (CNNs), are trained on large datasets of labeled road sign images to recognize patterns, colors, and shapes specific to various traffic signs. The detection process typically involves image preprocessing, feature extraction, and the application of classifiers to identify road signs in real-time under diverse environmental conditions. By leveraging machine learning, road sign detection systems can improve safety, reduce human error, and enhance the overall efficiency of road traffic management. The performance of the model is evaluated using standard metrics such as accuracy, precision, recall, and F1 score, demonstrating its effectiveness in handling real-world challenges. The results show that machine learning-based methods can significantly improve road sign detection, contributing to safer driving environments and the advancement of autonomous vehicles. The system is trained on a diverse dataset of road sign images, which include variations in lighting, weather conditions, and angles.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

RR	-	Railroad
PED	-	Pedestrian
BIC	-	Bicycle
TTC	-	Temporary Traffic Control
HOV	-	High Occupancy Vehicle

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Road sign detection using machine learning is crucial for autonomous vehicles, advanced driver-assistance systems (ADAS), and intelligent transportation systems. It enables vehicles to identify and interpret traffic signs in real time, ensuring safe navigation and compliance with traffic regulations. The process typically involves several stages: image preprocessing, feature extraction, object detection, and classification. Initially, raw images or video frames are preprocessed to improve quality by reducing noise, normalizing lighting, and resizing images. Feature extraction methods like edge detection or more advanced techniques like Convolutional Neural Networks (CNNs) are used to automatically identify key features such as shapes and edges.Object detection algorithms like YOLO (You Only Look Once) and Faster R-CNN are employed to locate road signs within an image. These models perform both classification and localization in one step, making them efficient for real-time applications. After detecting the signs, machine learning classifiers, such as CNNs or Support Vector Machines (SVM), categorize them into predefined classes like stop signs, speed limits, or yield signs. Post-processing techniques, like non-maximum suppression, refine the results by eliminating duplicate detections and improving localization accuracy.These models are trained on large datasets like the German Traffic Sign Recognition Benchmark (GTSRB), which helps them generalize across different conditions. Despite challenges like variations in sign designs and environmental factors such as glare or fog, deep learning models have shown significant progress in improving detection accuracy.This technology is vital for autonomous vehicles to obey traffic rules, assist drivers through ADAS, and support traffic monitoring systems. Road sign detection enhances safety, promotes smoother traffic flow, and plays a key role in the advancement of intelligent transportation systems.

1.2 PROBLEM STATEMENT

Road sign detection using machine learning involves accurately identifying and classifying traffic signs from real-time images or video feeds. Autonomous vehicles and advanced driver-assistance systems (ADAS) face challenges in varying weather, lighting, and regional sign designs. Additional difficulties include occlusions, damaged signs, poor visibility, and real-time processing. The goal is to create reliable systems that ensure safety and efficient operation of autonomous vehicles and ADAS across diverse driving conditions.

1.3 OBJECTIVES

The objectives of road sign detection using machine learning focus on developing a system that can accurately detect and classify traffic signs in real-time, ensuring autonomous vehicles and advanced driver-assistance systems (ADAS) can interpret road signs effectively. One key objective is to achieve robustness across diverse conditions, such as varying weather, lighting, and occlusions, while also addressing issues like damaged or worn signs. Real-time processing is crucial, enabling immediate vehicle responses to detected signs for safe navigation. Additionally, the system should generalize well across different regions with varying road sign designs and formats. Another important goal is scalability, allowing the model to adapt to new road sign categories and evolving signage standards. Ultimately, the primary aim is to enhance vehicle safety by providing accurate, reliable road sign detection and classification to ensure the smooth and safe operation of autonomous vehicles and ADAS in diverse driving environments. The model should also have the ability to generalize across different geographical regions, recognizing and interpreting signs with varying shapes, colors, and symbols that differ between countries and regions. Furthermore, the solution should be scalable, easily adapting to new sign categories and evolving standards of road signage. Finally, the overarching goal of road sign detection is to enhance safety by providing accurate, reliable detection that supports the seamless operation of autonomous vehicles and ADAS, reducing the risk of accidents caused by failure to recognize important road signs.

1.4 IMPLICATION

Road sign detection using machine learning enhances the safety and efficiency of autonomous vehicles (AVs) and advanced driver-assistance systems (ADAS). It ensures accurate interpretation of traffic signs, reduces accidents, and supports real-time vehicle responses. The technology also allows AVs to adapt to varying conditions and regional sign designs, improving global mobility.

Efficiency and Accuracy:

Road sign detection ensure reliable, real-time identification and classification of traffic signs. High accuracy prevents misinterpretation, while efficiency enables quick processing for immediate vehicle responses, enhancing safety.

Early Intervention and Support:

The Machine learning-based road sign detection systems enable early intervention by quickly identifying road signs and hazards. This allows autonomous vehicles or driver-assistance systems to adjust driving behavior immediately, preventing accidents. It also helps human drivers by providing alerts, enhancing road safety and promoting timely actions in response to traffic conditions.

Enhanced Security and Privacy:

Road sign detection systems improve vehicle security by ensuring accurate interpretation of traffic signs and preventing violations that could lead to accidents. To enhance privacy, these systems typically process data locally on the vehicle, minimizing the risk of unauthorized surveillance or data breaches, while safeguarding personal information.

Behavioral Changes:

The integration of road sign detection systems encourages safer driving behaviors. Autonomous vehicles can follow traffic rules more consistently, while drivers benefit from reminders and alerts, leading to better adherence to regulations. This technology helps reduce risky driving actions, such as speeding or failing to safety.

CHAPTER 2

LITERATURE SURVEY

2.1 TITLE : A Survey on Road Sign Detection Techniques for Autonomous Driving

AUTHORS : S.Ahamed

YEAR : 2019

This paper compares traditional image processing methods with machine learning techniques for road sign detection, focusing on their application in autonomous driving systems. The authors emphasize the significance of real-time processing and robust feature extraction for accurate detection. Key challenges discussed include environmental factors such as lighting, weather, and regional variations in road sign designs. The paper provides insights into how machine learning techniques can enhance the accuracy of road sign detection and classification, improving the safety and reliability of autonomous vehicles under varied and dynamic driving conditions.in sign designs.

2.2 TITLE : Real-Time Traffic Sign Recognition Using CNNs

AUTHORS : Sai Charan Nutheti, Ganesh Mamidipaka, Harthik Lokinen

YEAR : 2020

This study This study explores the use of Convolutional Neural Networks (CNNs) for real-time traffic sign recognition. CNNs are particularly suited for detecting and classifying road signs due to their ability to automatically learn hierarchical features from images. The authors discuss the effectiveness of CNNs in handling real-world challenges such as partial occlusions, varying lighting, and sign degradation. They also address the importance of dataset diversity and pre-processing techniques in enhancing the accuracy and robustness of traffic sign recognition systems, ensuring reliable performance in various environmental conditions encountered by autonomous vehicles.

2.3 TITLE : Traffic Sign Detection in Autonomous Vehicles

AUTHORS : Abhinav Saxena, Harsh Josh

YEAR : 2021

This review paper provides a comprehensive analysis of traffic sign detection and classification techniques for autonomous vehicles. It compares traditional methods like Support Vector Machines (SVM) with modern approaches such as deep learning models, including CNNs. The authors highlight the strengths and limitations of each method, emphasizing the importance of accuracy, processing speed, and adaptability to various driving environments. The paper also explores the challenges associated with real-world conditions, such as weather, damage to signs, and occlusion, which impact the performance of road sign detection systems in autonomous driving applications.

2.4 TITLE : YOLO-based Road Sign Detection

AUTHORS : Pooja S. Saravade, Pratiksha R. Molak, Bhakti S. Jadhav, Kirti P.

YEAR : 2022

This paper examines the use of YOLO (You Only Look Once) for real-time road sign detection in autonomous vehicles. It highlights YOLO's ability to quickly and accurately identify multiple road signs, ensuring efficient navigation. The authors discuss how YOLO performs under various challenging conditions, including occlusions, changing lighting, and fast-moving objects, which are common in real-world driving scenarios. By showcasing YOLO's effectiveness in these situations, the paper demonstrates how this technology can improve autonomous vehicle navigation, ensuring safer and more reliable driving in diverse environments.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing road sign detection systems primarily rely on traditional image processing techniques or machine learning models such as Support Vector Machines (SVM) and decision trees for sign recognition. These systems often struggle with environmental challenges like poor lighting, weather conditions, or damaged signs. While deep learning models like Convolutional Neural Networks (CNNs) have been applied, they require large, labeled datasets and significant computational resources. Furthermore, many systems lack real-time processing capabilities, affecting their reliability in fast-moving autonomous vehicles or advanced driver-assistance systems (ADAS).

3.2 PROPESED SYSTEM

The proposed system leverages advanced deep learning techniques, particularly Convolutional Neural Networks (CNNs) and real-time object detection frameworks like YOLO (You Only Look Once). These models are capable of accurately detecting and classifying road signs even in complex and dynamic environments, such as varying lighting, occlusions, and weather conditions. By utilizing a more robust and efficient dataset, the system ensures faster and more reliable road sign recognition, which is essential for the safety and performance of autonomous vehicles and ADAS. This system is designed for real-time processing, offering enhanced accuracy and adaptability.

3.3 APPROACH USED

The proposed approach utilizes Convolutional Neural Networks (CNNs) for feature extraction and YOLO (You Only Look Once) for real-time road sign detection. A diverse dataset of road signs is collected and preprocessed to train the models. YOLO's efficiency allows fast, accurate detection even under challenging conditions like varying lighting and occlusions. The system is optimized for real-time processing, enabling autonomous vehicles and ADAS to detect and classify road signs with high .

CHAPTER 4

THEORETICAL CONSIDERATIONS

4.1 IMAGE ACQUISTION AND PREPROCESSING

In road sign detection using machine learning, the Image Acquisition and Preprocessing stage plays a crucial role in ensuring the images are suitable for training models and achieving high detection accuracy. The first step, image acquisition, involves collecting a large and diverse set of images of road signs. These images should vary in terms of sign types (e.g., stop signs, speed limits), environmental conditions (e.g., different lighting, weather), backgrounds (urban, rural, highways), and camera angles. Datasets like the GTSRB (German Traffic Sign Recognition Benchmark) or LISA Traffic Sign Dataset are commonly used for this purpose. These images are typically annotated with bounding boxes to identify the location and class of each road sign within the image, providing the necessary labels for supervised learning.Once the images are collected, the next step is preprocessing, which involves several techniques to enhance the image quality and make it more suitable for machine learning models. First, resizing ensures all images have consistent dimensions, typically to fit the input requirements of neural networks.Normalization is then applied to scale pixel values to a common range, improving the model's training efficiency. Sometimes, images are converted to grayscale if color is not essential for sign identification, reducing computational load.Data augmentation techniques, like rotation, flipping, and scaling, are applied to increase dataset diversity and help the model generalize better. Histogram equalization can enhance image contrast, making road signs more distinguishable in varied lighting conditions. Edge detection methods, such as the Canny edge detector, highlight the contours of road signs, aiding in better localization. Additionally, cropping or selecting a region of interest (ROI) around the road signs focuses the model on the relevant portions of the image, reducing unnecessary complexity.

4.2 CAMERA CALIBRATION AND GEOMETRIC VIEW

Camera calibration and geometric view are essential steps in road sign detection to ensure accurate localization and recognition of road signs in real-world images. Camera calibration refers to the process of determining the intrinsic and extrinsic parameters of the camera. Intrinsic parameters include focal length, principal point, and lens distortion, which affect the image formation. Extrinsic parameters involve the camera's position and orientation in the world, which help relate 2D image coordinates to 3D world coordinates. Calibration is typically achieved by capturing images of a known pattern, such as a checkerboard, and using algorithms (e.g., Zhang's method) to calculate the camera's internal and external parameters. This process is crucial for correcting lens distortions (like radial and tangential distortion) and ensuring that the camera's view accurately represents the scene.Once calibrated, geometric view comes into play, which focuses on the perspective and transformation of the captured images. Geometric transformations such as homography are used to correct for distortion caused by the camera's angle or perspective.

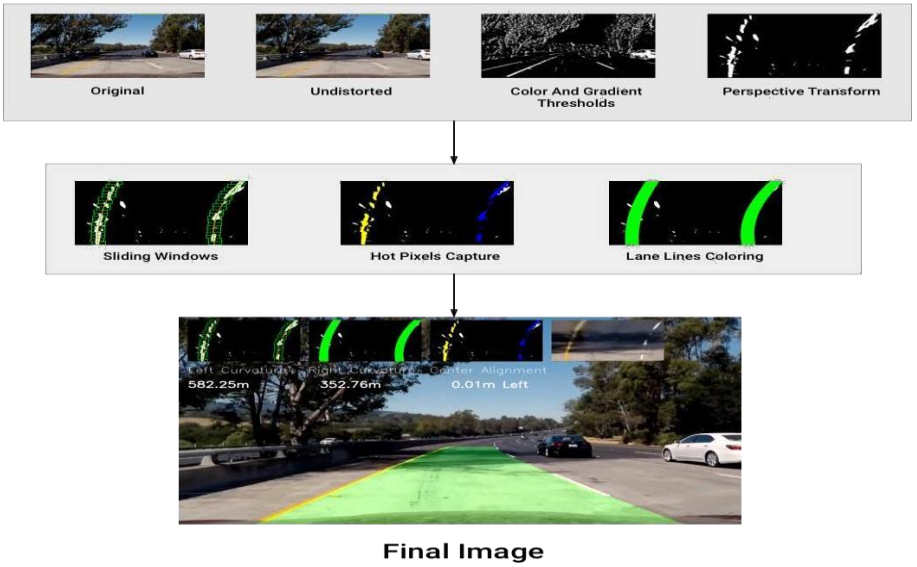


Figure 4.1: GEOMETRIC VIEW

4.3 LIGHTING AND WEATHER VARIABILITY

In the context of road sign detection, lighting and weather variability presents significant challenges. The effects of changing light conditions and varying weather patterns can dramatically alter the appearance of road signs, making it difficult for machine learning models to maintain consistent performance. Below are key subtopics related to this variability:

1. Lighting Variability

Daylight vs. Nighttime Conditions: Road signs can appear very different in daylight versus nighttime, with significant changes in visibility and clarity.

Sunlight and Shadows: Direct sunlight can cause signs to appear overexposed, while shadows cast by trees, poles, or buildings can partially or completely cover the sign, leading to missed detections.

Glare and Reflection: Glare from the sun or other artificial light sources can create bright spots on the surface of signs, washing out key details. Similarly, reflections from the sign's surface (especially on wet or polished signs) can distort its appearance, complicating detection algorithms.

2. Weather Variability

Rain: Rain can cause water droplets to appear on the surface of road signs, potentially blurring their shapes or creating reflections that interfere with the model's ability to detect them. Wet conditions can also result in contrasting lighting conditions due to reflections from the wet surfaces of signs or the road.

Fog and Haze: Fog and haze significantly reduce visibility, often causing signs to appear blurry or indistinct. In such conditions, road signs may also blend into the background, making them harder to distinguish. The reduced contrast in foggy conditions poses a challenge for both color-based and edge-detection methods.

Snow and Ice: Snow and ice can obscure road signs either by covering them completely or partially, making them invisible to detection systems. The reflective surface of snow or ice can also cause glare that impacts the clarity of road signs.

3. Impact on Machine Learning Models

Lighting and weather variability can significantly affect the accuracy of machine learning models that are not trained to handle these conditions. A model trained solely on images taken under clear, sunny conditions may struggle with nighttime or rainy image Overfitting: If a model is trained on images from only one type of lighting or weather condition, it may overfit to those conditions, leading to poor generalization to new, unseen environmental factors. To mitigate these issues, data augmentation strategies such as artificially varying the brightness, adding noise to simulate rain, or using filters to simulate fog or snow are crucial. These techniques allow models to learn to detect road signs across a wide range of conditions, improving robustness.

4. Techniques for Mitigating Lighting and Weather Effects

Histogram Equalization: This method improves image contrast and ensures better visibility of road signs in low-light conditions by enhancing the intensity distribution of the image.Adaptive Contrast Enhancement: This technique adjusts the contrast of an image dynamically, helping to make road signs more prominent, especially in images with poor lighting or fog. Image Smoothing and Filtering: Smoothing techniques, such as Gaussian blur, can reduce noise in the image caused by adverse weather conditions like rain or snow. Depth Information: If available, using depth data from stereo cameras or LiDAR can help the model separate road signs from their backgrounds, making detection easier under poor visibility conditions. Multimodal Sensor Fusion Combining camera images with other sensors (such as infrared or radar) can help mitigate the impact of poor weather and lighting, providing complementary information to enhance detection accuracy. Severe weather conditions, such as thunderstorms or high winds, can cause signs to shake or sway, altering their visual appearance in images. Additionally, wind may blow rain or snow across the sign, creating a shifting visual effect that complicates detection.

4.4 NOISE REDUCTION AND IMAGE ENHANCEMENT

In road sign detection, **noise reduction** and **image enhancement** are critical preprocessing steps that improve the quality of input images and ensure more accurate detection results. Images captured in real-world environments are often subject to various distortions or noise due to factors like sensor limitations, environmental conditions, and poor lighting. Effective noise reduction and enhancement techniques are essential for enhancing the clarity of road signs, reducing computational load, and improving model performance.

1. **Daylight vs. Nighttime Conditions**: Road signs can appear very different in daylight versus nighttime, with significant changes in visibility and clarity. At night, streetlights or headlights may create glare or shadows, which can obscure parts of a sign.

2. **Sunlight and Shadows**: Direct sunlight can cause signs to appear overexposed, while shadows cast by trees, poles, or buildings can partially or completely cover the sign, leading to missed detections. Shadows can also affect color perception, making it harder to distinguish between different types of signs.

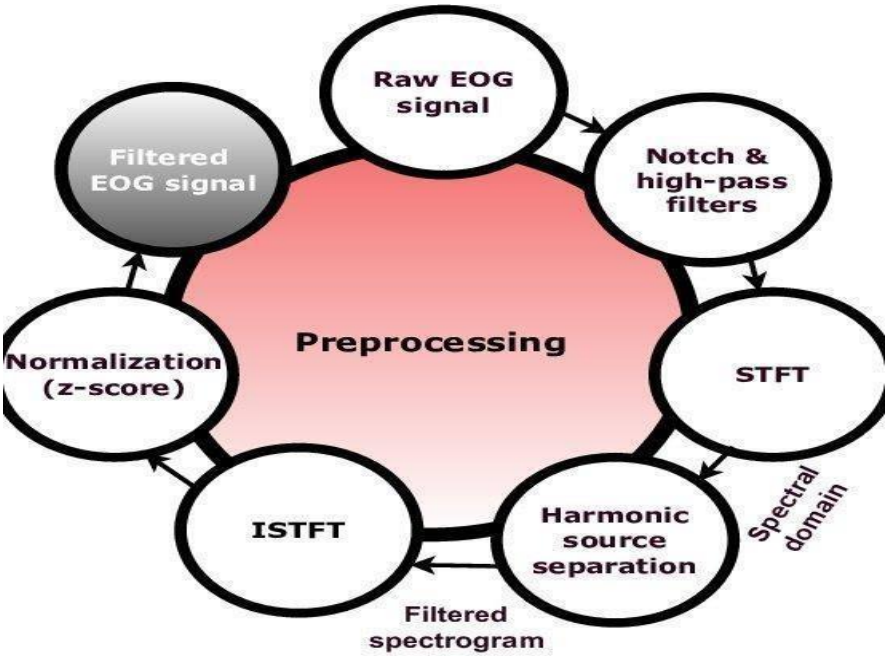


Figure 4.2 : NOISE REDUCTION AND IMAGE ENHANCEMENT

4.5 FEATURE EXTRACTION TECHNIQUES

Feature extraction is a crucial step in the road sign detection process, as it allows the machine learning model to focus on the most relevant and distinguishing characteristics of road signs. The goal of feature extraction is to transform raw image data into a set of informative features that the model can use to classify and detect road signs**.**

1. **Gaussian Noise**: Often caused by sensor imperfections or low light conditions, Gaussian noise introduces random variations in pixel intensity. A Gaussian filter is commonly used to smooth the image and reduce this type of noise by averaging the surrounding pixels.

2. **Salt-and-Pepper Noise**: This noise appears as random white and black pixels scattered throughout an image. Median filtering is an effective technique for removing salt-and-pepper noise. It works by replacing each pixel with the median value of the pixels in its neighbourhood, preserving edges while eliminating outliers.

3. **Speckle Noise:** Often occurring due to poor lighting or image compression, speckle noise results in grainy patterns. A Wiener filter or bilateral filter can be used to reduce speckle noise by applying local statistical methods that take both pixel intensity and local variance into account, smoothing the image while preserving edges.

4. **Wavelet Denoising**: This technique is useful when dealing with multi-resolution images and allows for noise reduction in specific frequency bands while retaining important features of the image. It can be particularly helpful in images with complex textures, such as road signs in busy environments.

4.6 FEATURES:

Handcrafted features are manually designed representations derived from image data that capture specific characteristics of road signs to aid in their detection and classification. Unlike deep learning-based methods, which automatically learn features during training, handcrafted features require domain knowledge and expert understanding to identify relevant image properties. These features are essential for classical machine learning models, where the performance depends heavily on the quality of the feature set. In road sign detection, several handcrafted feature extraction techniques are commonly used to enhance the accuracy and robustness of models.

4.7 DEEP LEARNING ARCHITECTURES:

The cornerstone for many computer vision tasks, including road sign detection, due to their ability to automatically learn and extract relevant features from raw image data. These models can handle the complex variability in images, such as changes in lighting, weather conditions, occlusions, and different perspectives. Below is an overview

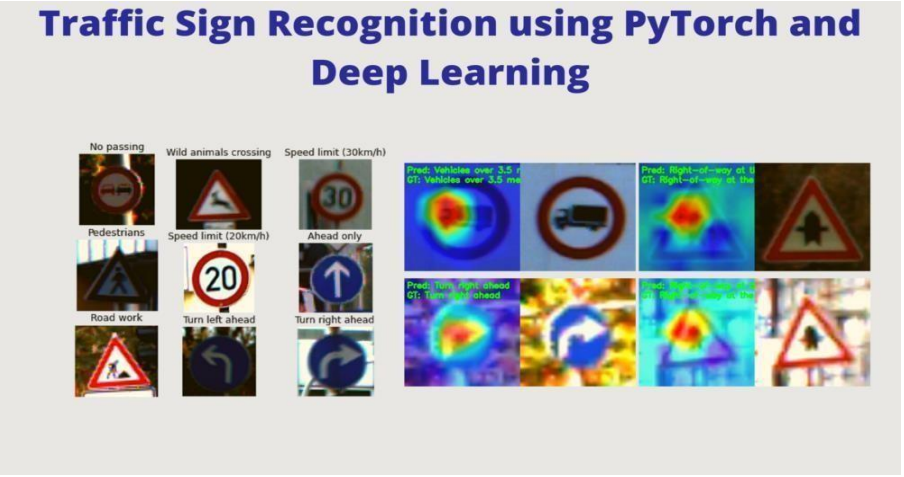


Figure 4.3 : DEEP LEARNING ARCHITECTURE

4.8 OBJECT CLASSIFICATION METHODS:

Object classification is a critical task in road sign detection, where the goal is to identify and label road signs from images based on their visual features. Once the road signs are detected and localized, object classification determines the specific type of sign, such as "stop sign," "yield sign," or "speed limit sign." Several methods can be employed to classify objects within an image, each with varying complexity and performance characteristics. Below is an overview of the most commonly used object classification methods for road sign detection.

1. Traditional Machine Learning Methods: Before the rise of deep learning, traditional machine learning methods were commonly used for object classification. These methods typically involve feature extraction followed by the use of classifiers to assign labels to detected objects. While these methods have been largely replaced by deep learning models, they still have value in certain applications.

Support Vector Machines (SVM): Support Vector Machines are a popular supervised learning algorithm used for classification tasks. In road sign classification, features such as shape, color, and texture are extracted from the image, and SVM is used to learn a decision boundary that separates different classes of road signs. SVM is effective when the feature space is well-defined, and it can be combined with feature extraction techniques like **HOG** (Histogram of Oriented Gradients) to improve classification accuracy.

K-Nearest Neighbors (KNN): KNN is another traditional classification algorithm where each detected object is classified based on the majority class of its nearest neighbors in the feature space. This method works by calculating the distance between the feature vector of a detected object and the feature vectors of labeled training examples. KNN is simple to implement but can be computationally expensive as the size of the dataset increases.

Decision Trees and Random Forests: Decision trees classify objects by creating a series of binary decisions based on feature values, such as color or shape. **Random Forests**, an ensemble of decision trees, can improve accuracy by combining the results of multiple trees to make the final classification decision. raw images.

2. Deep Learning-Based Methods

In modern road sign detection systems, deep learning methods, especially **Convolutional Neural Networks (CNNs)**, have become the dominant approach for object classification due to their ability to learn discriminative features directly from raw image data. These methods often outperform traditional machine learning approaches by automatically identifying patterns in large datasets.

Convolutional Neural Networks (CNNs): CNNs are the most widely used deep learning architecture for image classification tasks, including road sign classification. They consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers, which work together to learn features at various levels of abstraction. CNNs can automatically detect and classify road signs based on their shape, color, and texture, without the need for manual feature extraction.

Feature Learning: In CNNs, the lower layers learn simple features like edges and corners, while deeper layers learn more complex patterns like road sign shapes and colors. The final fully connected layers perform the classification task by associating the learned features with specific road sign labels.

Transfer Learning: Transfer learning is commonly used in road sign detection, where a pre-trained CNN model (such as **VGG16**, **ResNet**, or **Inception**) trained on large datasets like ImageNet is fine-tuned on road sign datasets. This technique allows models to take advantage of pre-learned features from a broader dataset, improving classification accuracy with less training data.

Region-based CNN (R-CNN) and its Variants: In the context of object detection, R-CNNs can be used to first localize road signs and then classify them. The process involves detecting regions of interest (RoIs) that might contain road signs and passing them through a CNN for classification. Variants like **Fast R-CNN** and **Faster R-CNN** improve speed and accuracy by sharing computation across regions or integrating region proposal networks (RPNs) for faster localization and classification.

You Only Look Once (YOLO): YOLO is an end-to-end deep learning architecture for real-time object detection. Unlike traditional object detection approaches that separate localization and classification, YOLO performs both tasks simultaneously.

Single Shot Multibox Detector (SSD): Similar to YOLO, SSD is a fast object detection framework that performs both localization and classification in a single step. It divides the input image into a grid and predicts multiple bounding boxes and class labels for each grid cell. SSD is effective for detecting multiple road signs of different sizes within an image.

Non-Maximum Suppression (NMS): NMS is used to eliminate redundant bounding boxes that may have been predicted by the model for the same road sign. It keeps the box with the highest confidence score and suppresses others that overlap significantly.

Confidence Thresholding: In some cases, the classification model may output a set of predicted probabilities for each class. A confidence threshold can be applied to filter out predictions that do not meet a certain level of certainty, reducing false positives.

Contextual Classification: Post-processing methods may also incorporate context-based rules, such as the relative positioning of road signs within the image or their relationship to road lanes, to improve classification accuracy.

3. Ensemble Methods

Ensemble methods combine multiple classifiers to improve classification accuracy by leveraging the strengths of different models. These methods can be particularly useful for road sign classification when multiple object types are involved, and no single classifier performs optimally across all classes.

Bagging: In **bagging** (Bootstrap Aggregating), multiple instances of the same classifier are trained on different subsets of the data (created by bootstrapping), and their predictions are combined, usually by voting or averaging. **Random Forests** are a specific example of an ensemble method that uses decision trees.

Boosting: **Boosting** algorithms, such as **AdaBoost** or **Gradient Boosting**, sequentially train weak classifiers on the errors made by previous classifiers. This approach focuses on improving performance by correcting misclassifications. Boosting can enhance classification accuracy, especially for challenging road sign categories.

Stacking: In **stacking**, multiple classifiers (e.g., CNNs, decision trees, SVMs) are trained independently, and their outputs are combined using a meta-classifier to make the final decision. This method can be highly effective when different models complement each other's strengths in road sign classification.

Hybrid Models:

In some cases, ensemble methods combine classical machine learning techniques with deep learning models. For instance, a hybrid model could combine the power of a convolutional neural network (CNN) for feature extraction with a traditional machine learning model (like a support vector machine or random forest) for classification. This hybrid approach can leverage both deep feature learning and the strong decision boundaries formed by traditional models

Advantages of Ensemble Methods in Road Sign Detection:

Improved Accuracy: By combining multiple models, ensemble methods can improve the detection performance, especially when individual models have complementary strengths.

Reduced Overfitting: Ensemble methods help reduce overfitting by averaging predictions or focusing on errors from different models, making the system more generalizable.

Better Robustness: The combination of different types of classifiers (e.g., decision trees, CNNs, etc.) ensures that the system is more robust to changes in road sign appearance, lighting conditions, and partial occlusions.

4.9 OBJECT DETECTION ALGORITHMS:

Object detection is a crucial task in computer vision, particularly for road sign detection, where the goal is not only to classify the object (e.g., stop sign, yield sign, etc.) but also to localize it within an image by drawing a bounding box around it. Over the years, a variety of object detection algorithms have been developed, each with its own strengths and weaknesses. These algorithms can be categorized into two main types: traditional methods and deep learning-based methods. Below is an overview of the most commonly used object detection algorithms, with a focus on their application in road sign detection. Object detection algorithms play a critical role in road sign detection, with the goal of accurately localizing and classifying road signs in diverse environments. Traditional algorithms like Haar cascades and HOG + SVM have been largely replaced by deep learning-based methods, which offer significant improvements in accuracy and efficiency.

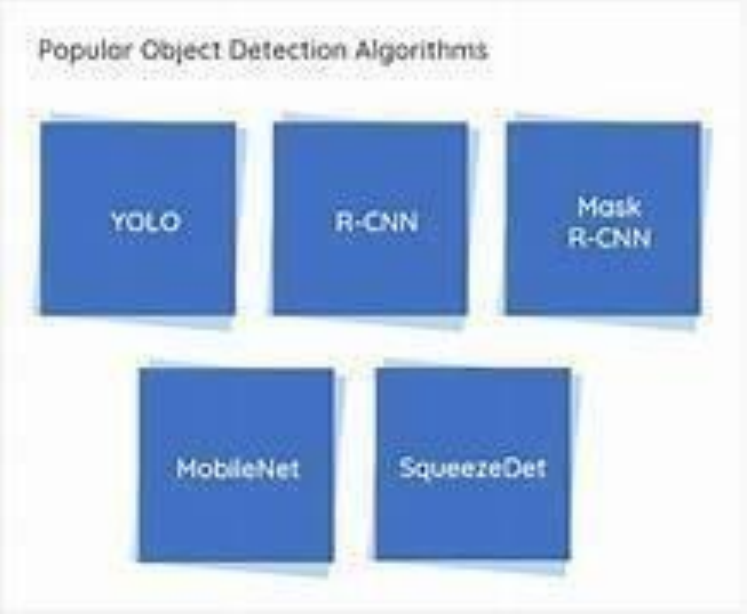


Figure 4.4: OBJECT DETECTION ALGORITHMS

4.10 END-TO-END MODEL TRAINING:

End-to-end model training refers to the process of training a deep learning model where all components of the system — from image input to object detection and classification — are learned simultaneously and optimized together. This type of training is crucial in road sign detection systems, as it allows for seamless integration of various stages of the pipeline, including feature extraction, object localization, and classification. By training an end-to-end model, the system learns to automatically extract the most relevant features and make predictions directly from raw input data, eliminating the need for manual feature engineering.

4.11 HANDLING VARIATIONS IN SIGN APPEARANCE:

Data Augmentation: One of the most effective ways to handle lighting variations is through data augmentation. This involves artificially modifying the training data to simulate different lighting conditions. Techniques like random brightness adjustment, contrast variation, and gamma correction are commonly used to ensure that the model sees a wide range of lighting conditions during training.

Dealing with Different Lighting Conditions: Lighting plays a critical role in image quality and the visibility of road signs. Variations in natural lighting, such as sunlight, shadow, or low-light conditions during night-time, can drastically affect the appearance of road signs, making it harder for a detection system to accurately identify them.

Histogram Equalization: This technique enhances the contrast of an image by redistributing the intensity levels. It helps in improving visibility in low-light or high- contrast situations, making it easier for the model to detect road signs.

Lighting-Invariant Features: Some advanced models are designed to focus on specific features of road signs that are less sensitive to lighting changes. For example, edge features (using methods like the Sobel operator) can help detect road signs even when their colors are altered due to lighting conditions.

Robust Feature Extraction: Deep learning models can be trained to focus on the most relevant features of road signs that are least likely to be occluded. For example, signs with distinct shapes (e.g., circular, triangular, or rectangular) or symbols (e.g., stop signs or yield signs) can be detected even when some portions of the sign are hidden.

4.12 ADDRESSING OCCLUSIONS AND CLUTTER:

In road sign detection, **occlusions** and **clutter** are common challenges that can significantly affect the accuracy of detection systems. **Occlusions** occur when parts of a road sign are blocked by other objects, such as vehicles, trees, or other road signs. **Clutter** refers to the presence of irrelevant objects or background elements in an image, which may confuse the model and lead to false detections. Effectively addressing occlusions and clutter is crucial to improving the robustness and reliability of road sign detection systems, especially in complex, real-world driving environments.

1. Handling Occlusions

Occlusions can occur in many situations, such as when a road sign is partially blocked by a passing vehicle or obstructed by foliage. These occlusions make it difficult for traditional detection systems to detect and classify the sign. Below are strategies to address this issue:

Robust Feature Extraction: Deep learning models, particularly Convolutional Neural Networks (CNNs), are capable of learning hierarchical features that are less sensitive to occlusions. By learning both global and local features of road signs, the model can still identify and classify partially occluded signs. For instance, even when part of a road sign is hidden, its shape, color, or distinctive features may still be visible enough for the model to make an accurate prediction.

Region Proposal Networks (RPNs): Techniques such as **Region Proposal Networks (RPNs)** used in Faster R-CNN allow the model to generate multiple candidate bounding boxes for potential objects. This approach increases the chances of detecting road signs even when they are partially occluded. RPNs can focus on different regions of interest in the image, making it more likely to find a sign despite partial visibility.

Multi-Scale Detection: When signs are occluded, they may appear at different sizes depending on how much of the sign is visible. Multi-scale detection methods, such as **YOLO** and **Single Shot Multibox Detector (SSD)**, use multiple feature maps at different resolutions to identify signs of varying sizes. This helps the model detect road signs even when they are partially hidden or appear smaller than usual.

Synthetic Data and Occlusion Simulation: One effective way to improve the model's robustness to occlusions is by simulating occlusions during the training phase. This can be achieved by artificially occluding parts of the signs in the training data using techniques .

Occlusion-Aware Networks: Some advanced architectures are specifically designed to handle occlusions by explicitly accounting for the visibility of road signs. These models use attention mechanisms or context-based reasoning to infer information about occluded parts of the sign and make more accurate predictions.

2. Handling Clutter

Clutter in road sign detection arises when irrelevant objects or background noise obscure or distract from the road sign itself. For example, a busy street with multiple vehicles, pedestrians, or other objects can create a cluttered environment that makes it harder for the model to identify road signs correctly.

Background Subtraction and Segmentation: One approach to reduce clutter is by segmenting the relevant areas of the image that are likely to contain road signs. **Background subtraction** techniques can help isolate objects of interest by removing irrelevant background features. Using methods such as **GrabCut** or **U-Net** for semantic segmentation, the system can focus on regions that are more likely to contain road signs.

Selective Search and Region Proposal: **Selective search** and **Region Proposal Networks (RPNs)** can help identify promising regions in an image for object detection. Instead of scanning the entire image, these techniques focus on generating potential regions where road signs are likely to appear. This helps avoid false positives caused by clutter and narrows down the search area for further analysis.

Contextual and Spatial Information: Incorporating spatial and contextual information is another strategy for reducing the effects of clutter. For example, road signs often appear in specific contexts, such as near intersections or along roadways. By leveraging the spatial relationships between road signs and their surroundings, the model can prioritize these regions and ignore irrelevant objects that are less likely to be signs.

Class-Aware Detection: In a cluttered environment, it's important for the model to differentiate between road signs and other similar-looking objects (e.g., advertising boards, traffic lights, or vehicle license plates). Training the model on specific classes of road signs allows it to focus on distinguishing road signs from other visual elements in the image. By using **class-aware detection** algorithms, the model can filter out objects that are unlikely.

Attention Mechanisms: Attention-based models can help address clutter by enabling the network to focus on the most relevant parts of the image. These models learn to "attend" to

the regions where road signs are most likely to appear, ignoring irrelevant features in the image. Attention mechanisms have been shown to significantly improve performance in cluttered scenes by guiding the model's focus.

3. Combining Occlusion and Clutter Handling Techniques

In many real-world scenarios, occlusions and clutter occur together, making road sign detection even more challenging. Combining the strategies mentioned above can help improve the model's ability to detect signs in such environments:

Hybrid Approaches: Combining multiple techniques, such as combining **RPNs** with **segmentation** methods and **attention mechanisms**, can provide a more comprehensive solution to handling occlusions and clutter. By using segmentation to isolate regions of interest and applying attention to focus on the relevant areas, the model can more effectively detect road signs even in highly cluttered and occluded environments.

Multi-Task Learning: Multi-task learning allows the model to simultaneously perform multiple tasks, such as object detection, semantic segmentation, and occlusion handling. This enables the model to learn more complex relationships between road signs and their surroundings, improving its ability to handle both occlusions and clutter. For example, a model trained to detect road signs along with road lanes or vehicles could better infer the location of a sign that is partially blocked or surrounded by other objects.

4. Real-World Applications and Challenges

In real-world applications, addressing occlusions and clutter is particularly important for autonomous vehicles or driver assistance systems. Road signs are often placed in areas with high traffic density, where they may be partially blocked by other vehicles or obstructed by environmental factors such as trees or construction zones. In these situations, the detection system must be able to operate in real-time and provide accurate sign detection despite these challenges.

4.13 REAL-TIME PROCESSING AND EFFICENCY:

In road sign detection, **real-time processing** and **efficiency** are critical factors for deploying systems in real-world applications, particularly in autonomous vehicles and advanced driver assistance systems (ADAS). These systems must process images and make decisions swiftly to ensure the safety and reliability of navigation. Real-time processing refers to the ability of the system to process data and provide outputs in a time frame that meets the requirements of real-world scenarios, typically in fractions of a second. Efficiency, on the other hand, is concerned with how well the system utilizes computational resources, balancing the trade-off between accuracy and the speed of operation. To achieve these objectives, several strategies and technologies are employed.

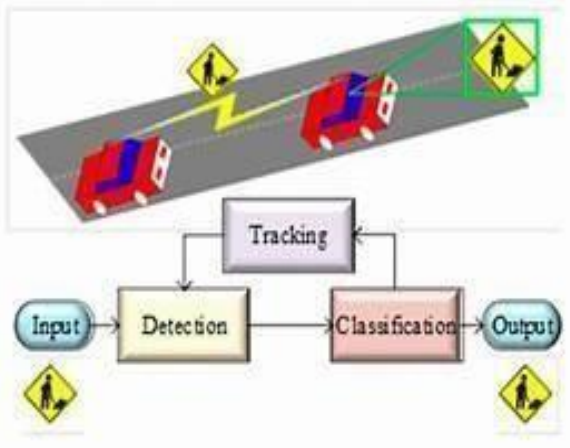


Figure 4.5: REAL-TIME PROCESSING AND EFFICENCY

CHAPTER 5

MODULE DESCRIPTION

1.1 IMAGE ACQUISITION AND PREPROCESSING:

In road sign detection systems, **image acquisition** and **preprocessing** are crucial steps that significantly impact the accuracy and efficiency of detection models. These processes are essential for converting raw visual data into a form that can be used effectively by machine learning algorithms, ensuring that the model can accurately recognize and classify road signs in various environments.

1.2 FEATURE EXTRACTION AND REPRESENTATION:

The **Feature extraction** involves identifying and isolating important information from the input images that can be used to distinguish between different road signs. In the context of road sign detection, this step aims to extract various attributes, such as geometric shapes (e.g., circular, triangular), color (e.g., red, yellow), and texture patterns (e.g., reflective surfaces or surface irregularities),

1.3 CLASSIFICATION AND DETECTION MODELS:

They are fundamental tasks that allow a system to identify and categorize road signs based on visual input. Classification refers to the task of assigning a label to an object (in this case, a road sign), while detection involves locating the object within an image or video stream. Together, classification and detection enable road sign detection systems to not only identify the presence of a road sign but also determine its exact position and type.

1.4 POST-PROCESSING AND VALIDATION TECHNIQUES:

After the initial detection and classification of road signs, **post-processing and validation techniques** are applied to refine and validate the results, ensuring that the road sign detection system is both accurate and reliable. These techniques help filter out false positives, enhance the precision of detections, and ensure that the system performs well under varying conditions. Post-processing also involves interpreting the raw output from the detection models, such as bounding boxes or classification labels, into a final decision.

1.5 Real-Time Detection and Performance:

- The expense report module provides users with detailed insights into their spending habits and pat
- Computer Vision: The primary technology for real-time road sign detection is computer vision, which uses cameras and sensors to capture the road scene and process visual data in real time.
- Image Preprocessing: Techniques such as filtering, edge detection (e.g., Canny), and color space transformations are applied to enhance road sign detection accuracy.
- Object Detection Algorithms: These are the core algorithms used to identify and classify road signs. Common algorithms include:
- Haar Cascades: A machine learning-based object detection method.
- Convolutional Neural Networks (CNNs): Deep learning models used for both detection and classification.
- Region-based CNNs (R-CNN): Used to propose regions for detecting specific objects, including road signs.
- YOLO (You Only Look Once): A real-time object detection system that processes the entire image in one go, providing fast and accurate predictions.
- Sensor Fusion: Real-time detection often integrates data from various sensors such as cameras, LiDAR, and radar. This combination helps improve detection reliability, especially in challenging conditions (e.g., fog, night driving).

CHAPTER 6

SYSTEM REQUIREMENTS

6.1 HARDWARE REQUIREMENTS

- Processor – Intel i3 or Higher.
- RAM – 6GB or Higher.
- Storage – 150GB or Higher.
- Camera – AI Thinking ESP 32

6.2 SOFTWARE REQUIREMENTS

- Operating System – Windows 11 or Higher.
- Languages Used – PYTHON,MACHINE LEARNING.

CHAPTER 7

METHODOLOGY

7.1 INTRODUCTION

The development of an income and expense tracker using PHP and MySQL involves a systematic approach to designing and implementing the software solution. This methodology encompasses several stages, including requirements gathering, system design, database design, implementation, testing, and deployment. The following is an overview of the methodology involved in each of these stages:

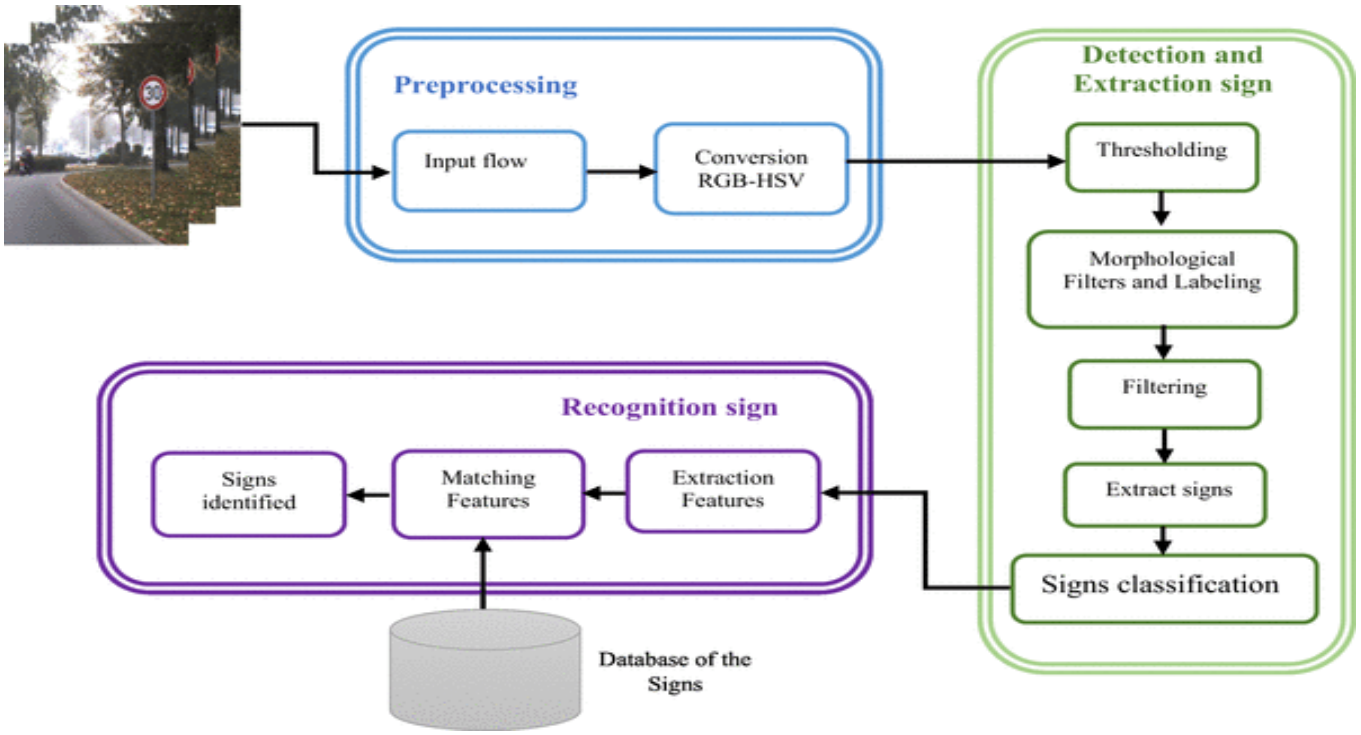


Figure 7.1: Workflow of the implementation.

7.2 SYSTEM ARCHITECTURE

The system architecture of the income and expense tracker consists of several components that work together to manage user data, record transactions, analyze financial information, and provide a user-friendly interface. The architecture follows a typical three-tier model, with presentation, application logic, and data management layers. The system architecture of the income and expense tracker using PHP and MySQL follows a three-tier model, with distinct layers for presentation, application logic, and data management. By adopting best practices in security, scalability, and performance, the architecture ensures the reliability, security, and efficiency of the application. This system architecture provides a robust foundation for building an income and expense tracker using PHP and MySQL, ensuring the effective management of financial data and user interactions.

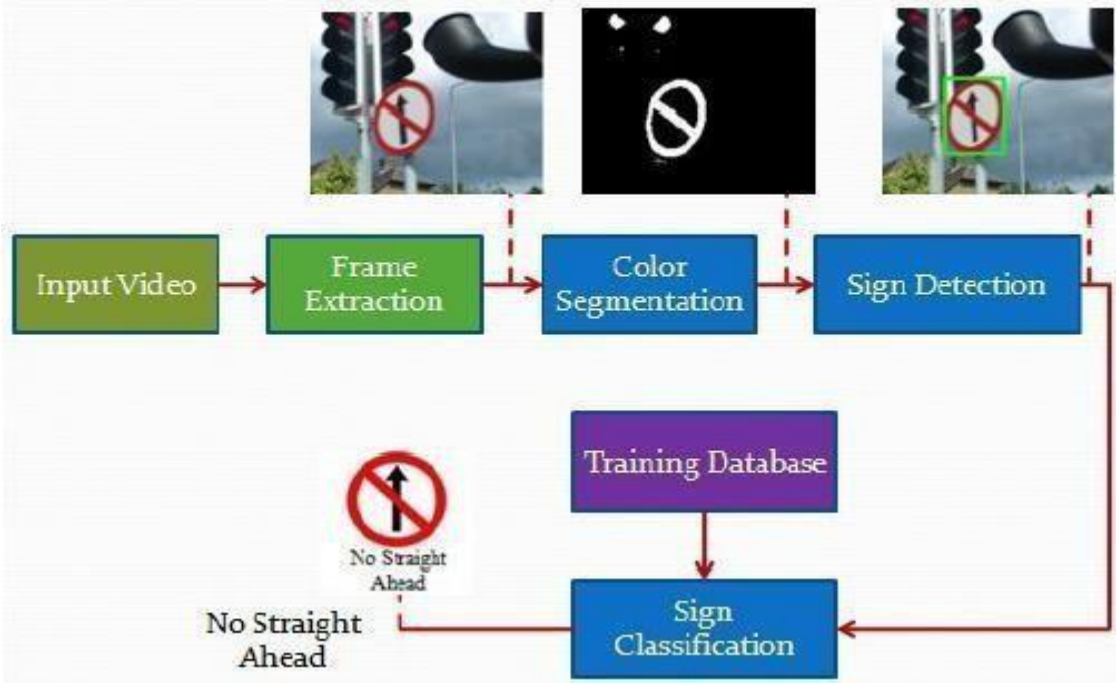


Figure 7.2: Architecture & Mechanism of the system.

7.3 IMPLEMENTATION DETAILS

To create an income and expense tracker using PHP and SQL, first, set up a MySQL database with tables for income and expenses using the SQL commands provided. Create a PHP configuration file (config.php) to connect to the database. Implement an add_record.php script to handle form input for adding income and expense records, and a view_records.php script to display these records in a tabular format. Use a form in add_record.php to capture and submit data such as type (income/expense), date, category, amount, and description, and insert this data into the respective database table. Finally, display the stored records in view_records.php by fetching and listing them from the database. Ensure to replace placeholders in the configuration file with actual database credentials.

7.4 SMART CONTRACT

- Implementation is the process of building the web according to its design.
- A web implementor uses hypertext markup language (HTML), Cascading Style Sheets (CSS) to develop structure and design of web.
- PHP has been used as server-side scripting language, were as sql is used to communicate with the data base.
- These make it possible for a web to be dynamic so that it could interact with the user.
- The implementation process resembles web development because it involves using a specific syntax for encoding web structures or a programming language in a formal language in computer files.
- Implementing a income and expense tracker using PHP and MySQL involves several steps, from data collection and preprocessing to model training and evaluation.
- Below is a simplified example using PHP and MySQL for a binary classificationtask based on historical user performance data.
- Note that this is a basic example, and in a real-world scenario, you would likely need a more complex model and feature engineering.

Reviewing the code written in the Python Code:

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import numpy
#load the trained model to classify sign
from keras.models import load_model
model = load_model('traffic_classifier.h5')
#dictionary to label all traffic signs class.
classes = { 1:'Speed limit (20km/h)',
            2:'Speed limit (30km/h)',
            3:'Speed limit (50km/h)',
            4:'Speed limit (60km/h)',
            5:'Speed limit (70km/h)',
            6:'Speed limit (80km/h)',
            7:'End of speed limit (80km/h)',
            8:'Speed limit (100km/h)',
            9:'Speed limit (120km/h)',
            10:'No passing',
            11:'No passing veh over 3.5 tons',
            12:'Right-of-way at intersection',
            13:'Priority road',
            14:'Yield',
            15:'Stop',
            16:'No vehicles',
            17:'Veh > 3.5 tons prohibited',
            18:'No entry',
            19:'General caution',
            20:'Dangerous curve left',
            21:'Dangerous curve right',
            22:'Double curve',
            23:'Bumpy road',
            24:'Slippery road',
            25:'Road narrows on the right',
            26:'Road work',
            27:'Traffic signals',
            28:'Pedestrians',
            29:'Children crossing',
```

```
            30:'Bicycles crossing',
            31:'Beware of ice/snow',
            32:'Wild animals crossing',
            33:'End speed + passing limits',
            34:'Turn right ahead',
            35:'Turn left ahead',
            36:'Ahead only',
            37:'Go straight or right',
            38:'Go straight or left',
            39:'Keep right',
            40:'Keep left',
            41:'Roundabout mandatory',
            42:'End of no passing',
            43:'End no passing vehicle with a weight greater than 3.5 tons' }
#initialise GUI
top=tk.Tk()
top.geometry('800x600')
top.title('Traffic sign classification')
top.configure(background='#CDCDCD')
label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = Label(top)
def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((30,30))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    pred = model.predict([image])[0]
    pred_class_index = numpy.argmax(pred)
    sign = classes[pred_class_index+1]
    print(sign)
    label.configure(foreground='#011638', text=sign)
def show_classify_button(file_path):
    print(file_path)
    classify_b=Button(top,text="Classify Image",command=lambda: classify(file_path),padx=10,pady=5)
    classify_b.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
```

```
    classify_b.place(relx=0.79,rely=0.46)
def upload_image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)
        uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)))
        im=ImageTk.PhotoImage(uploaded)
        sign_image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass
upload=Button(top,text="Upload an image",command=upload_image,padx=10,pady=5)
upload.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True)
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="check traffic sign",pady=20, font=('arial',20,'bold'))
heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

Explanation about each function in the smart contract:

It seems like you might be referring to the "home" function or element on a web page. In the context of a website, the "home" function typically refers to a link or button that, when clicked, takes the user back to the main or starting page of the website. Here's a more detailed explanation:

Navigation Convenience:

The home function is designed to provide users with a convenient way to return to the main or central page of a website. This is particularly useful when users have navigated deep into the site's structure and want to quickly go back to the starting point.

Consistency and User Expectations:

Including a home function is a common practice in web design. Users often expect to find a way to return to the homepage easily. It adds a level of consistency across different websites and helps users understand the navigation structure.

Website Branding:

The home page is often considered the face of a website, serving as a re presentation of the brand or organization. The home function reinforces this by allowing users to easily access the main content and branding of the site.

User Interface Design:

The home function is usually implemented as a link or button, often represent ed by a house icon or the word "Home." Its placement is typically in a prominent location, such as the navigation bar, making it easily noticeable and accessible.

Browser Default Behavior:

In web browsers, clicking on the website logo in the top-left corner often serves the same purpose as clicking on a home button. Browsers also provide keyboard shortcuts (e.g., pressing the Home key) to navigate to the homepage.

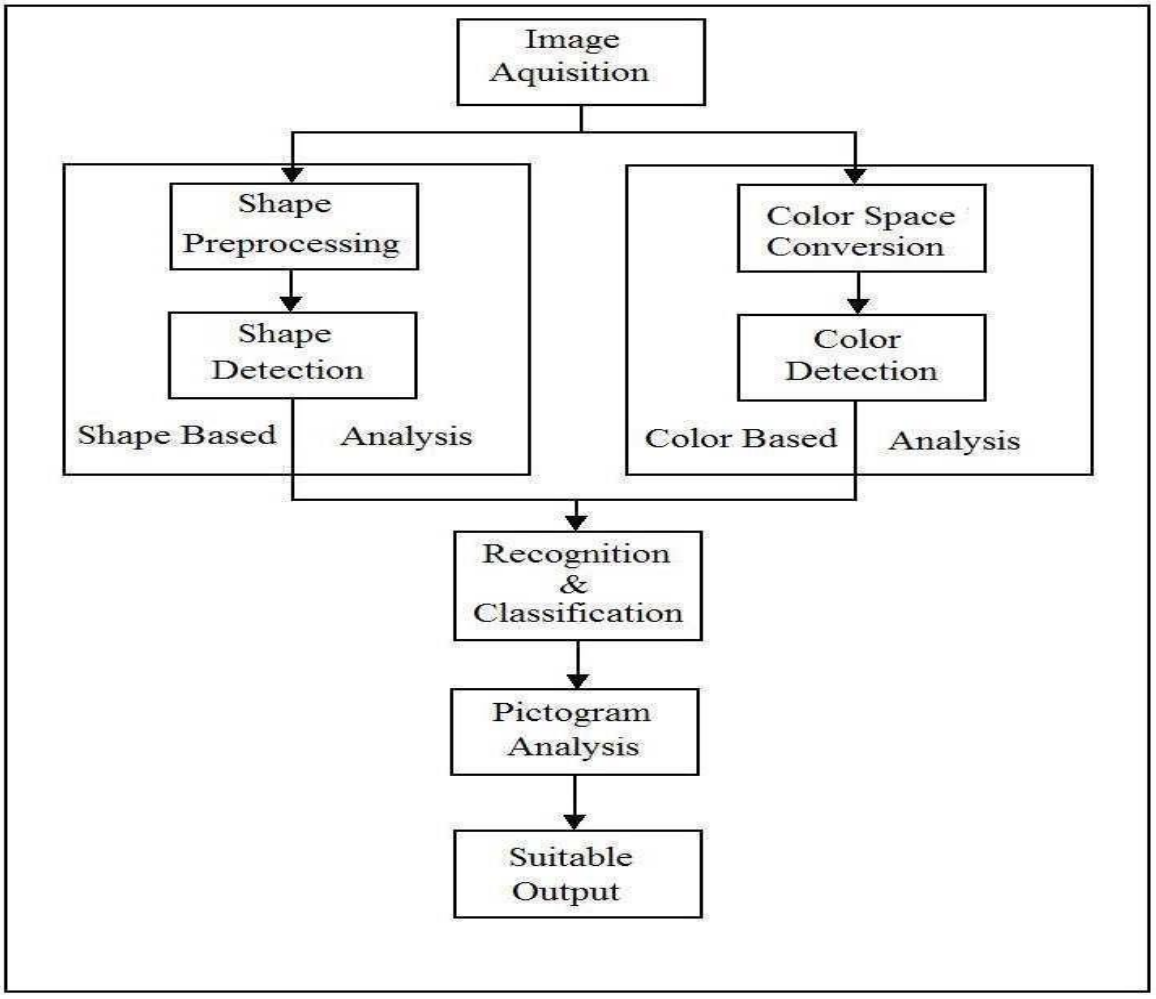


Figure 7.3: Flowchart of Home page in the system.

User Authentication:

The primary purpose of the login function is to authenticate users. By entering their credentials (usually a username or email and a password), users prove their identity to the website or application.

Access Control:

The login function is closely tied to access control. After authentication, the webpage can determine what level of access or permissions the user has. This allows websites to tailor the content and features based on the user's role or authorization level.

Security:

The login function is crucial for securing user accounts and protecting sensitive information. It ensures that only authorized individuals can access certain parts of the website or perform specific actions. Proper security measures, such as encryption and secure password storage, are essential components of a robust login system.

Session Management:

When a user logs in, a session is typically created to keep track of their activities during their visit. This session allows users to stay authenticated as they navigate different pages within the site without having to log in again for each interaction.

Forgot Password Functionality:

Web pages often include a "Forgot Password" link or function as part of the login process. This allows users to reset their passwords if they forget them, typically by sending a password reset link to their registered email address.

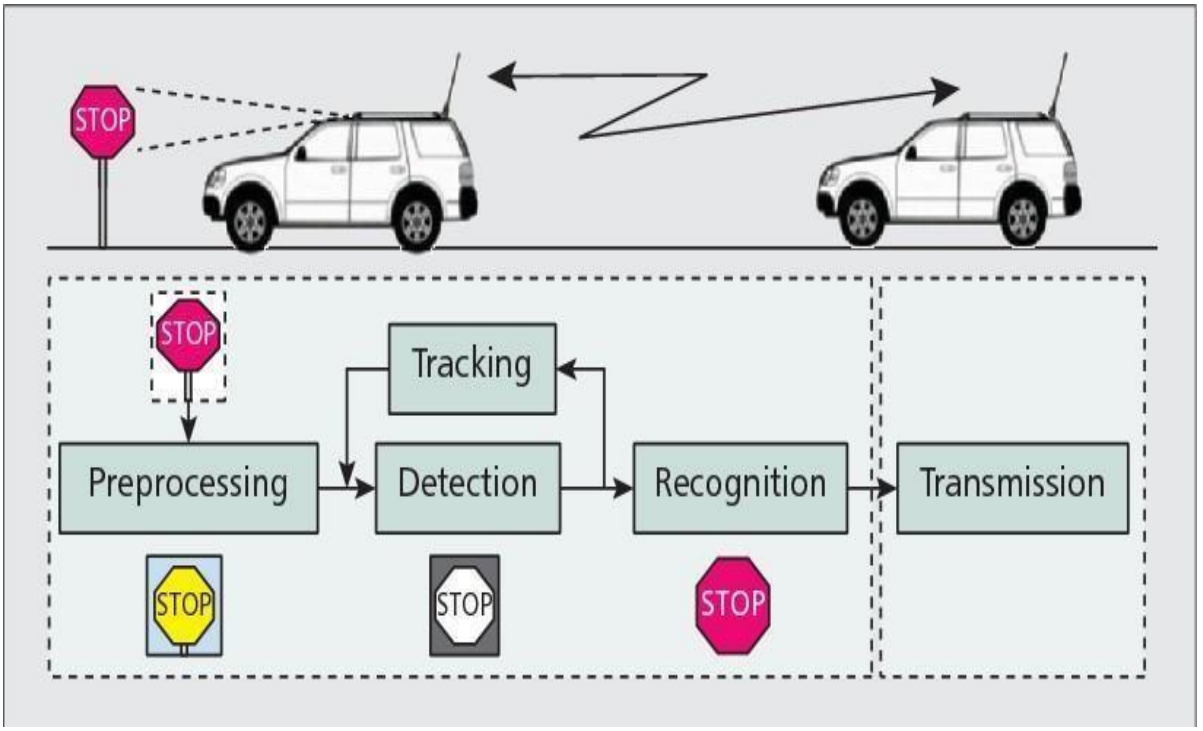


Figure 7.4: Traffic Sign Detection, recognition, and transmission system stages

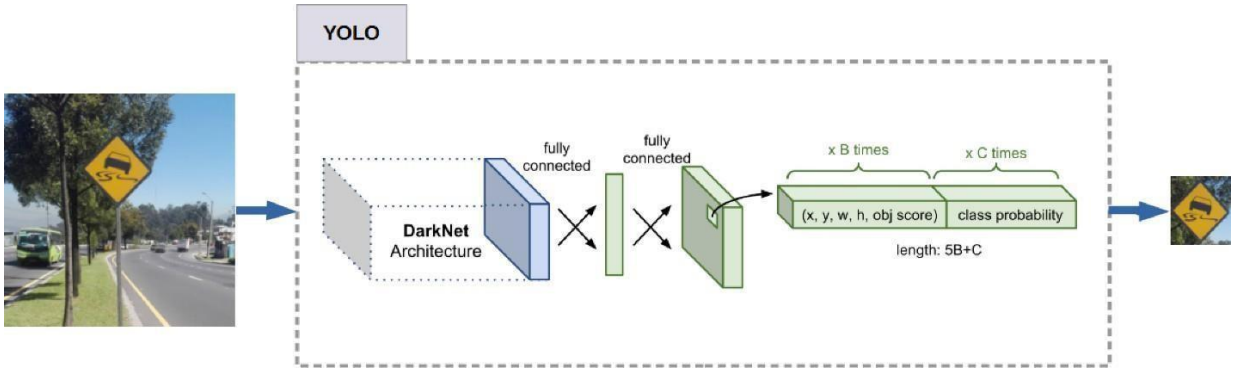


Figure 7.5: YOLO Object Detection Algorithm

The logout function on a web page allows users to terminate their current session, effectively ending their authenticated state and revoking access to any personalized or restricted content. Here's an explanation of the logout function:

Session Termination:

When a user logs out, the web page or application terminates the user's session. This action invalidates the session token or cookie associated with the user, making it no longer valid for authentication.

Security:

The logout function is crucial for security. It ensures that if a user forgets to log out or if they share a computer with others, their session is not left open, reducing the risk of unauthorized access to their account.

Clearing Session Data:

Logging out often involves clearing any stored session data on the client side, such as cookies or local storage. This further ensures that the user is fully the authenticated state.

Redirect or Confirmation:

After a successful logout, it's common to redirect users to a designated "logged out" or "home" page. Additionally, some websites provide a confirmation message to inform users that they have been successfully logged out.

Preventing Back Button Use:

To enhance security, some websites implement measures to prevent users from using the browser's "back" button to navigate back into secured areas after logging out. This is often achieved by disabling the caching of sensitive pages or using other techniques. When a user logs in, a session is typically created to keep track of their activitiesduring their visit. This session allows users to stay authenticated as they navigate different pages within the site without having to log in again for each interaction.

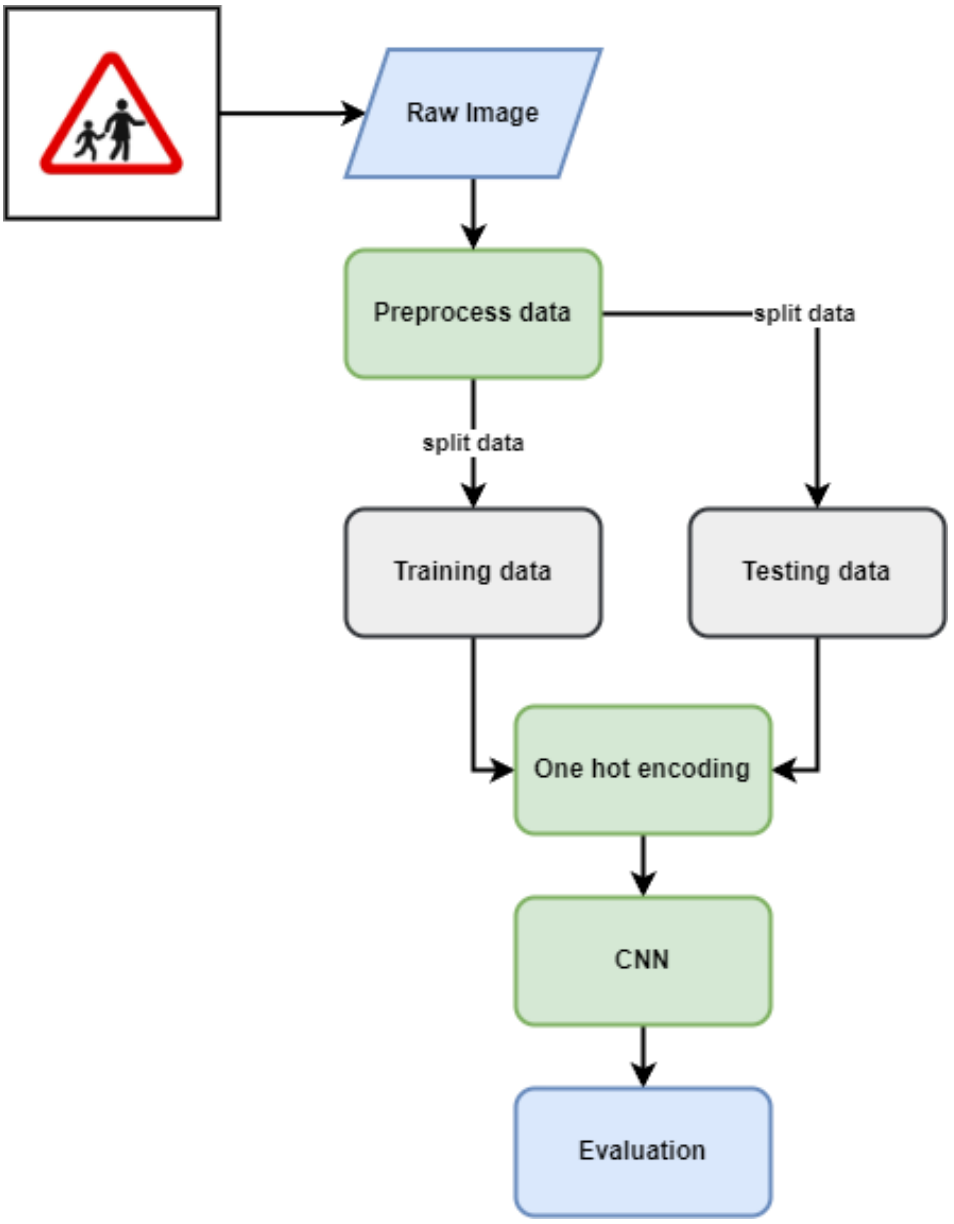


Figure 7.6: Flowchart of Advanced Traffic

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION:

In conclusion, Python's versatility and rich ecosystem of libraries and frameworks have made it a go-to language for a wide array of applications, ranging from data analysis and machine learning to web development and automation. Its simplicity, combined with powerful libraries like Pandas, TensorFlow, Flask, and OpenCV, enables developers to tackle complex problems efficiently. Python's evolution has empowered individuals and organizations to build robust systems, analyze large datasets, and create innovative solutions across various domains. Whether for scientific research, financial modeling, or real-time applications, Python continues to evolve, solidifying its position as one of the most widely used programming languages in the world. Its ease of use, scalability, and broad community support ensure that Python will remain a dominant tool in the development of future technologies and solutions.

FUTURE ENHANCEMENT:

Enhancing a As technology continues to advance, the future of Python-based systems and applications holds exciting possibilities. Some key areas for future enhancement in Python-related applications include:

Integration with Emerging Technologies: Python is likely to see greater integration with emerging technologies like **Quantum Computing**, **Blockchain**, and **5G Networks**. These integrations will enable the development of more advanced solutions for data security, decentralized applications, and high-speed data processing.

Enhanced Performance and Efficiency: Although Python is known for its ease of use, it can sometimes be slower compared to compiled languages like C++. Future enhancements could focus on improving Python's performance through optimizations in interpreters like **PyPy** or through better integration with low-level languages to speed up critical sections of code.

APPENDIX A

Sample code

app.py

```
import tkinter as tk
from tkinter import filedialog
from tkinter import
from PIL import ImageTk, Image
import numpy
#load the trained model to classify sign
from keras.models import load_model
model = load_model('traffic_classifier.h5')
#dictionary to label all traffic signs class.
classes = { 1:'Speed limit (20km/h)',
2:'Speed limit (30km/h)',
3:'Speed limit (50km/h)',
4:'Speed limit (60km/h)',
5:'Speed limit (70km/h)',
6:'Speed limit (80km/h)',
7:'End of speed limit (80km/h)',
8:'Speed limit (100km/h)',
9:'Speed limit (120km/h)',
10:'No passing',
11:'No passing veh over 3.5 tons',
12:'Right-of-way at intersection',
13:'Priority road',
14:'Yield',
15:'Stop',
16:'No vehicles',
17:'Veh > 3.5 tons prohibited',
18:'No entry',
19:'General caution',
20:'Dangerous curve left',
21:'Dangerous curve right',
22:'Double curve',
23:'Bumpy road',
24:'Slippery road',
25:'Road narrows on the right',
26:'Road work',
27:'Traffic signals',
28:'Pedestrians',
29:'Children crossing',
30:'Bicycles crossing',
31:'Beware of ice/snow',
32:'Wild animals crossing',
```

```
33:'End speed + passing limits',
34:'Turn right ahead',
35:'Turn left ahead',
36:'Ahead only',
37:'Go straight or right',
38:'Go straight or left',
39:'Keep right',
40:'Keep left',
41:'Roundabout mandatory',
42:'End of no passing',
43:'End no passing vehicle with a weight greater than 3.5 tons' }
#initialise GUI
top=tk.Tk()
top.geometry('800x600')
top.title('Traffic sign classification')
top.configure(background='#CDCDCD')
label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = Label(top)
def classify(file_path):
global label_packed
image = Image.open(file_path)
image = image.resize((30,30))
image = numpy.expand_dims(image, axis=0)
image = numpy.array(image)
pred = model.predict([image])[0]
pred_class_index = numpy.argmax(pred)
sign = classes[pred_class_index+1]
print(sign)
label.configure(foreground='#011638', text=sign)
def show_classify_button(file_path):
print(file_path)
classify_b=Button(top,text="Classify Image",command=lambda:
classify(file_path),padx=10,pady=5)
classify_b.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
classify_b.place(relx=0.79,rely=0.46)
def upload_image():
try:
file_path=filedialog.askopenfilename()
uploaded=Image.open(file_path)
uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)))
im=ImageTk.PhotoImage(uploaded)
sign_image.configure(image=im)
sign_image.image=im
label.configure(text='')
show_classify_button(file_path)
except:
pass
upload=Button(top,text="Upload an image",command=upload_image,padx=10,pady=5)
```

```
upload.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True)
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="check traffic sign",pady=20, font=('arial',20,'bold'))
heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

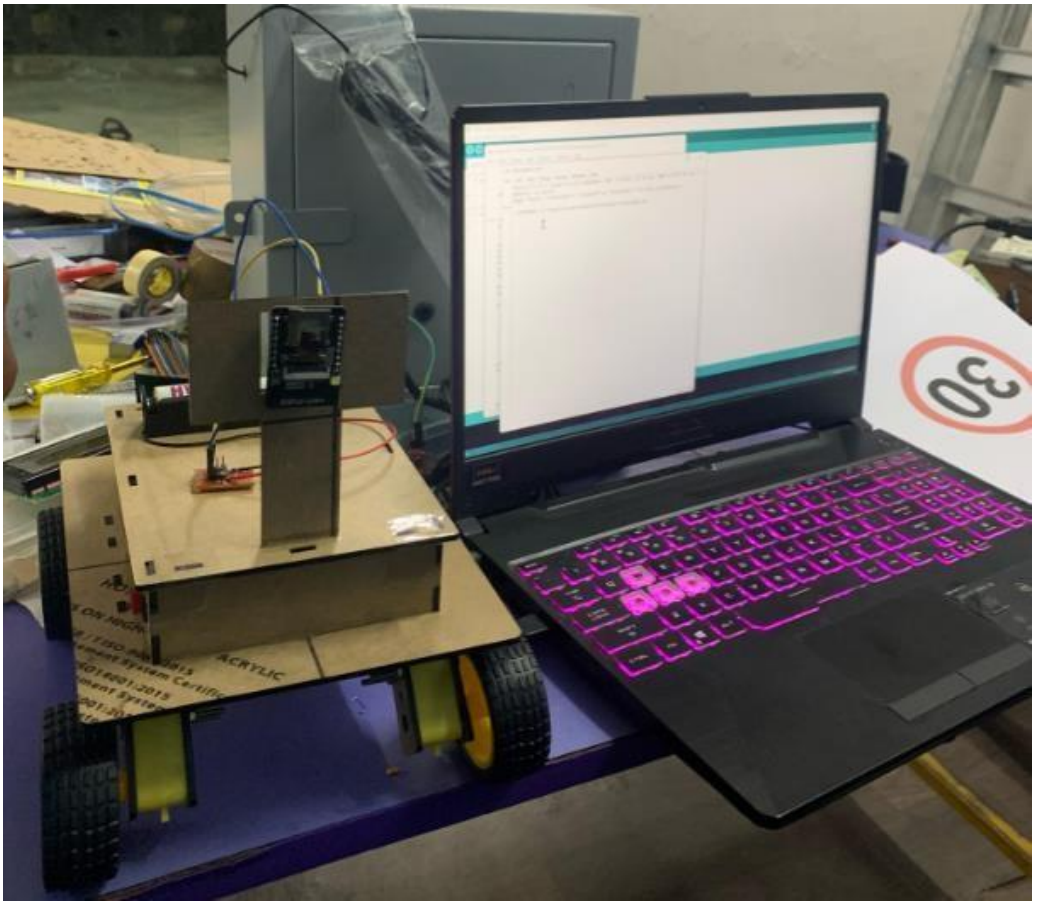
model.py

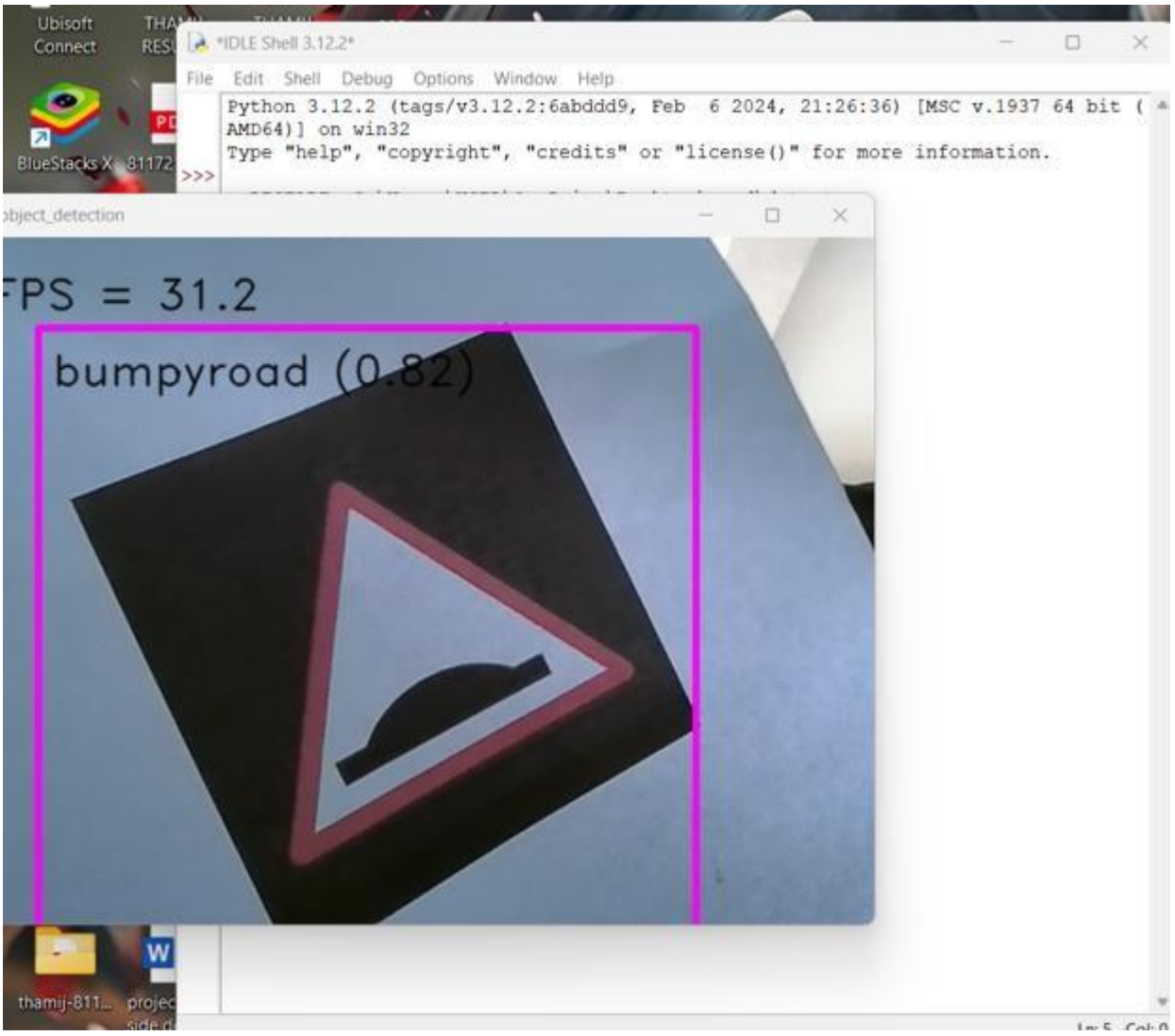
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
data = []
labels = []
classes = 43
cur_path = os.getcwd()
for i in range(classes):
path = os.path.join(cur_path, 'archive', 'Train', str(i))
images = os.listdir(path)
for a in images:
try:
image = Image.open(os.path.join(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)
print(data.shape, labels.shape)
X_t1, X_t2, y_t1, y_t2 = train_test_split(data, labels, test_size=0.2, random_state=42)
print(X_t1.shape, X_t2.shape, y_t1.shape, y_t2.shape)
y_t1 = to_categorical(y_t1, 43)
y_t2 = to_categorical(y_t2, 43)
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5, 5), activation='relu', input_shape=X_t1.shape[1:]))
```

```
model.add(Conv2D(filters=32, kernel_size=(5, 5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
eps = 15
history = model.fit(X_t1, y_t1, batch_size=32, epochs=eps, validation_data=(X_t2, y_t2))
model.save("my_model.h5")
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()
y_test = pd.read_csv('archive/Test.csv')
labels = y_test["ClassId"].values
imgs = y_test["Path"].values
data = []
print(imgs)
for img in imgs:
image = Image.open(os.path.join('archive', img))
image = image.resize((30, 30))
data.append(np.array(image))
X_test = np.array(data)
pred = np.argmax(model.predict(X_test), axis=-1)
from sklearn.metrics import accuracy_score
print(accuracy_score(labels, pred))
model.save('traffic_classifier
```

APPENDIX B

SCREENSHOTS




Ubisoft Connect
BlueStacks X
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
bject_detection
PS = 31.2
bumpyroad (0.82)
thamij-811...

REFERENCES:

1. Tamia Ruvimbo Masendu, Aanajey Mani Tripath, May 2022, Daily Expense T Tracker, International Journal of Research in Engineering, Science and Management,5(5), pp.90-92

2. Abishek Hagawane, Roshan Gopalghare, Prathmesh Isawe, Mrunal Aware, 07 June 2022, Daily Expense Tracker, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 8(3), pp. 265-268.

3. Velmurugan.R, Mrs. P. Usha, March 2021, Expense Tracker Application, international journal of innovative research in technology,7(10), pp.191-194.

4. Shivam Mehra, Prabhat Parashar 2021 Daily Expense Tracker, International Journal of Research in Engineering and Science (IJRES), Volume 09(12), pp. 70-73.

5. N.ZahiraJahan, K. I.Vinodhini, 2016 Personalized Expense Managing Assistant Using Android, International Journal of Computer Technique, Volume 3(2) pp. 60-67.

6. Darsh Shah,Sanay Shah,Ritik Savani, Dr. Bhavesh Patel,Ashwini Deshmukh, 2021, Application for Predictive Recommendation and Visualization of Personal Expenses, Journal of Emerging Technologies and Innovative Research (JETIR), 8(3) pp.2196-2201.

7. S. Chandini1, T. Poojitha, D. Ranjith, V.J. Mohammed Akram, M.S Vani, V.Rajyalakshmi, Mar 2019, Online Income and Expense Tracker,International Research Journal of Engineering and Technology (IRJET), 6(3), pp.4119-4124.

8. P. Thanapal , Mohammed Yaseen Patel, T.P. Lokesh Raj and J. Satheesh Kumar, 2015. Income and Expense Tracker, Indian Journal of Science and Technology (IJST), 8(S2), pp 118–122.

9. Ram Kumar Sharma, Shwetank Pandey, Vaibhav Tripathi, Sonam Gupta, Neha kumari, 2021. Expense Tracker, International Journal of All Research Education and Scientific Methods (IJARESM), 9(S6), pp. 3166-3170.

10. Prof Miriam Thomas, Lakshmi P, and Dr. Mahalekshmi T,2020 sept, Expense Tracker, International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), 9(4), pp.5-9

11. Atiya kazi1, Praphulla s. kherade, Raj s. vilankar, Parag m.sawant, May 2021, expense tracker, iconic research and engineering journals, 4(11), pp.19-21.

12. Aman Garg, Mukul Goel, Sagar Mittal, Mr. Shekhar Singh, Apr 2021, Expense Tracker, International Journal for Research in Applied Science & Engineering Technology (IJRASET),9(IV), pp.1067-1070.

13. Uday Pratap Singh, Aakash Kumar Gupta, Dr Balamurugan,2021 April, Spending Tracker, Turkish Journal of Computer and Mathematics Education, 12(6), pp.5095-5103.

14. Nidhi Jitendra Jadhav, Rutuja Vijay Chakor, Trupti Mahesh Gunjal, Damayanti. D. Pawar, April-2022, expense tracker, International Research Journal of Modernization in Engineering Technology and Science(IRJMETS),4(4), pp.1676