YOLO-Based Approach for Helmet and Number Plate Detection Using Raspberry Pi

Priti Balaso Jagtap[1], Prof. Sagar D. Dhawale[2]

[1]ME Student, Department of Electronics & Telecommunication, Ajeenkya DY Patil School of Engineering (SPPU), Lohegaon, Pune, Maharashtra 412105

[2]Assistant Professor in the Department of Electronics & Telecommunication at Ajeenkya DY Patil School of Engineering (SPPU), Lohegaon, Pune, Maharashtra 412105

Abstract

This paper presents an intelligent, real-time surveillance system for automatic helmet and number plate detection using the YOLOv8 object detection framework deployed on a Raspberry Pi platform. The system aims to enhance road safety and traffic law enforcement by simultaneously identifying helmet violations and recognizing vehicle number plates. The proposed model integrates a high-accuracy deep learning approach with lightweight, edge-based deployment to achieve efficient real-time performance. The YOLOv8 architecture was fine-tuned through transfer learning on annotated datasets containing diverse environmental conditions such as lighting variations, motion blur, and different plate formats. Optimizations such as pruning, quantization, and ONNX conversion were implemented to reduce computational overhead, enabling smooth inference on low-power hardware. Experimental results demonstrate superior performance of YOLOv8 compared to previous models, achieving a mean Average Precision (mAP) of 94.9%, with real-time processing at approximately 27.6 frames per second on Raspberry Pi 4. Field testing under various weather and lighting conditions confirmed system robustness and high reliability. The proposed solution provides a scalable, cost-effective, and portable framework for automated traffic monitoring, particularly suitable for deployment in developing regions. By integrating dual detection capabilities in a single pipeline, this work contributes to intelligent transportation systems, supporting road safety initiatives, and promoting efficient enforcement through automation.

Keywords: *YOLOv8, Helmet Detection, Number Plate Recognition, Raspberry Pi, Real-Time Object Detection, Intelligent Traffic Surveillance*

1. INTRODUCTION

Road safety is one of the most critical public health concerns in the modern world, particularly in rapidly developing countries like India, where exponential growth in vehicle usage has placed immense pressure on traffic management and law enforcement systems. According to the World Health Organization (WHO), more than 1.35 million people die globally each year due to road accidents, with a significant percentage of fatalities involving two-wheeler riders. In India alone, two-wheelers account for nearly 40% of total road accident deaths, highlighting the urgent need for enhanced road safety measures. One of the most common and preventable causes of fatalities is the non-usage of helmets. Despite strict laws mandating helmet usage for riders and pillion passengers, compliance remains alarmingly low due to weak enforcement and inadequate monitoring mechanisms. Simultaneously, identifying traffic violations, stolen vehicles, or uninsured vehicles through number plate recognition has become increasingly essential for law enforcement agencies. Traditional manual surveillance techniques are not only resource-intensive but also inefficient in dealing with the rapidly increasing volume of traffic data. Moreover, the sheer scale of road networks, particularly in urban and semi-urban areas, makes continuous manual monitoring impractical and error-prone. These challenges necessitate the deployment of intelligent, automated, and scalable solutions that can operate in real time to improve traffic safety and regulation compliance.

1.1 Problem Statement

The dramatic rise in two-wheeler usage, combined with widespread non-compliance with helmet laws, has significantly increased the risk of fatal accidents on roads. Reports indicate that riders without helmets are almost three times more likely to sustain fatal head injuries in accidents than those who wear helmets. Enforcement of helmet usage laws is currently dependent on human-operated traffic cameras or on-site monitoring, which is not only labour-intensive but also inconsistent and often biased. Additionally, vehicle identification based on license plates remains a challenge, especially in environments where plate formats vary widely or are partially obstructed. Current traffic

monitoring systems typically involve manual observation or semi-automated surveillance that requires substantial human intervention. Such systems struggle with real-time processing, scalability, and reliability. This results in missed violations, delayed response times, and ultimately, ineffective enforcement. The lack of integration between helmet detection and number plate recognition further limits the ability of authorities to take comprehensive action against traffic violators. Hence, there is an urgent need for an automated, efficient, and real-time surveillance system that can simultaneously detect helmet usage and identify vehicle number plates with high accuracy.

1.2 Need for Automation

In the era of rapid urbanization and smart city development, automation has become the cornerstone of effective traffic monitoring and law enforcement. Intelligent video surveillance systems powered by artificial intelligence (AI) and computer vision offer the potential to revolutionize road safety enforcement. Unlike conventional surveillance methods, AI-driven systems can operate continuously, process large volumes of video data in real time, and make precise, data-driven decisions without human intervention. Automation not only enhances detection accuracy but also reduces the chances of human error and bias. Furthermore, automated systems are scalable and can be deployed in multiple locations with minimal maintenance. This is particularly beneficial in developing regions where traffic density is high, but resources for manual enforcement are limited. By integrating helmet detection and number plate recognition into a single automated platform, authorities can streamline enforcement workflows, automate fine issuance, and build comprehensive databases of traffic violations for future analysis.

1.3 Existing Approaches and Their Limitations

Early attempts at automated number plate recognition (ANPR) primarily relied on Optical Character Recognition (OCR) techniques combined with traditional image processing methods. These methods typically involved several stages: image acquisition, plate localization, character segmentation, and recognition. While OCR-based systems worked reasonably well in controlled conditions, they were highly sensitive to environmental factors such as lighting variations, camera angles, motion blur, and non-standardized number plate formats — all of which are common in real-world scenarios. Machine learning (ML)-based approaches, including Support Vector Machines (SVMs) and convolutional neural networks (CNNs), offered improved accuracy but often required significant computational resources and were not suitable for deployment on edge devices. Furthermore, most existing systems focus exclusively on either helmet detection or number plate recognition, with very few attempts made to combine both tasks into a single, integrated solution. These limitations have created a gap in existing research: the need for a lightweight, robust, and real-time detection system capable of performing dual tasks — helmet detection and number plate recognition — on low-power, cost-effective hardware suitable for widespread deployment.

1.4 Proposed Approach

To address the aforementioned challenges, this study proposes an intelligent, end-to-end automated system that leverages YOLOv8 (You Only Look Once, version 8) — a state-of-the-art deep learning model for real-time object detection — integrated with a Raspberry Pi edge device. YOLOv8 is known for its high speed, robust accuracy, and capability to detect multiple objects within a single image frame, making it ideal for real-time applications such as traffic surveillance. The proposed system is designed to detect whether a rider is wearing a helmet and simultaneously capture and recognize the vehicle's number plate. The use of a Raspberry Pi as the deployment platform significantly reduces system cost and complexity, enabling large-scale implementation even in resource-constrained environments. Additionally, the model is optimized for edge computing, ensuring low latency, reduced reliance on cloud infrastructure, and real-time performance directly on-site.

Main Contributions

The primary contributions of this work can be summarized as follows:

- **Integrated Dual Detection:** Development of a unified system that performs both helmet detection and number plate recognition in a single framework.
- **Real-Time Processing:** Implementation of the YOLOv8 algorithm for high-speed, real-time detection on live video streams.
- **Edge-Based Deployment:** Optimization of the detection pipeline for Raspberry Pi, demonstrating the feasibility of deploying complex AI models on low-power edge devices.

- **Robust Performance:** Comprehensive evaluation of the system under varying environmental conditions, including lighting, weather, and traffic density.
- **Scalability and Cost-Effectiveness:** Design of a solution that is both economically viable and scalable for deployment across urban and rural road networks.

2. LITERATURE REVIEW

The development of intelligent traffic surveillance systems has undergone significant evolution over the past two decades, moving from traditional image processing techniques to advanced deep learning frameworks and, more recently, to edge-based computing systems. The core objectives of such systems are to enhance traffic law enforcement, ensure rider safety, and automate vehicle identification. In this context, automatic license plate recognition (ALPR) and helmet detection are two of the most widely researched tasks. However, most existing research has focused on these tasks independently, leaving a substantial gap in the development of integrated systems capable of performing both tasks simultaneously in real time on resource-constrained hardware such as Raspberry Pi. This section reviews existing literature across four main areas: traditional approaches to ANPR, deep learning-based detection methods, edge computing in surveillance, and the current research gap.

2.1 Traditional ANPR Systems Using Raspberry Pi

Agrawal and Pardakhe (2017) proposed an automatic license plate recognition system using Raspberry Pi aimed at real-time traffic enforcement. Their method involved capturing vehicle images, preprocessing them to enhance quality, and using OCR for character recognition. While effective for standard plates, it struggled with non-standard formats and complex traffic scenes. Kumthekar et al. (2018) also developed a similar system based on image acquisition and segmentation techniques to detect and read vehicle number plates. Their approach was cost-effective and achieved reasonable accuracy but encountered challenges with blurred images and low illumination. Dangare and Dalvi (2015) introduced an ANPR model capable of recognizing plates from multiple countries using a real-time Raspberry Pi platform. The system used segmentation and OCR to achieve accurate recognition but required improvements for handling varied plate designs. Likewise, Abirami and Jasmine (2018) focused on enhancing real-time detection accuracy through better preprocessing and segmentation techniques. Their system improved recognition speed and accuracy but remained sensitive to external factors such as lighting conditions and camera angles.

2.2 Advanced Recognition Techniques and Portable Systems

Al-Mayyahi et al. (2018) presented a vehicle detection and license plate recognition model that relied on classical image processing and segmentation. Their system showed reliable detection under static conditions but exhibited performance issues in high-speed scenarios. Fakhar et al. (2019) advanced this field by creating a portable ANPR system on Raspberry Pi, integrating edge detection and OCR for on-the-move detection. The solution demonstrated portability and flexibility but was limited by computational constraints. Puranic et al. (2016) contributed a literature review and practical implementation using template matching for number plate recognition. The method offered reasonable accuracy with clear images but failed in noisy or rotated scenarios. Arth et al. (2010) explored real-time ANPR on an embedded DSP platform, using hardware acceleration to improve detection speed and OCR for character recognition. Although it achieved high performance, the system lacked adaptability to diverse plate types and environmental conditions.

2.3 Image Processing and Localization-Based Approaches

Zhai et al. (2010) focused on license plate localization using morphological operations, applying filters and edge detection techniques to accurately isolate the plate region. The system performed well in controlled environments but had limitations when dealing with diverse plate structures and uncontrolled lighting. Alzubaidi and Latif (2019) proposed a real-time recognition system for Saudi Arabian plates using Raspberry Pi, involving preprocessing, segmentation, and OCR-based text extraction. The approach was effective in real-world applications but struggled with plate occlusion and non-uniform camera angles. Mane et al. (2019) took a different direction by integrating ANPR into a toll automation system, automating vehicle identification and toll collection through segmentation and character recognition. While efficient in reducing manual intervention, it required further optimization for large-scale highway systems. These studies collectively demonstrate the evolution of image-based techniques, highlighting their effectiveness in specific scenarios but also their limitations in scalability, accuracy, and adaptability.

3. PROBLEM DEFINITION AND OBJECTIVES

3.1 Problem Definition

The rapid growth of two-wheeler traffic in urban areas has significantly increased the risk of road accidents, especially those involving riders not wearing helmets. Despite existing traffic laws mandating helmet usage, enforcement remains a major challenge due to the reliance on manual surveillance and human-operated systems. This lack of efficient enforcement contributes to a high number of preventable fatalities and injuries. According to global road safety reports, non-helmet riders are nearly three times more likely to suffer severe head injuries or death during accidents. Simultaneously, effective vehicle identification through number plate recognition is crucial for traffic regulation, law enforcement, and crime prevention. However, traditional number plate recognition methods often fail under real-world conditions due to variations in lighting, plate design, and environmental noise.

Current traffic monitoring systems typically operate as isolated solutions — some focus exclusively on helmet detection, while others are dedicated to automatic number plate recognition (ANPR). This fragmented approach limits the ability of authorities to comprehensively enforce road safety regulations. Moreover, many existing systems rely on high-performance computing infrastructure, which is expensive, power-intensive, and unsuitable for large-scale deployment in resource-constrained environments. The lack of integration, scalability, and real-time capabilities poses a significant technological gap in the development of intelligent traffic monitoring solutions.The key technical problem, therefore, is the absence of a unified, real-time, and cost-effective system capable of simultaneously detecting helmet violations and identifying vehicle number plates under diverse environmental conditions. Additionally, there is a critical need to optimize such a system for deployment on low-power embedded devices like Raspberry Pi, enabling scalability and affordability in smart city infrastructure. Addressing this challenge will not only enhance road safety but also support law enforcement agencies in implementing automated traffic management systems that require minimal human intervention.

3.3 Research Objectives

To address the aforementioned problem, this research sets out the following key objectives:

1. Develop a dual detection system to identify helmet usage and vehicle number plates simultaneously.
2. Optimize the YOLOv8 model for efficient real-time performance on low-power Raspberry Pi devices.
3. Achieve high detection accuracy and real-time processing under varied traffic and environmental conditions.
4. Design a cost-effective, scalable smart surveillance solution suitable for urban and semi-urban deployments.

4. RESEARCH METHODOLOGY

The methodology adopted in this study integrates deep learning-based object detection (YOLOv8) with edge computing on Raspberry Pi for automated helmet violation detection and number plate recognition in real-time traffic monitoring. The workflow follows a structured pipeline: (i) image acquisition, (ii) detection and classification using YOLOv8, (iii) binary classification for helmet compliance, (iv) number plate localization and OCR recognition, and (v) optimized deployment on embedded hardware. Each component of the pipeline is discussed in detail below.

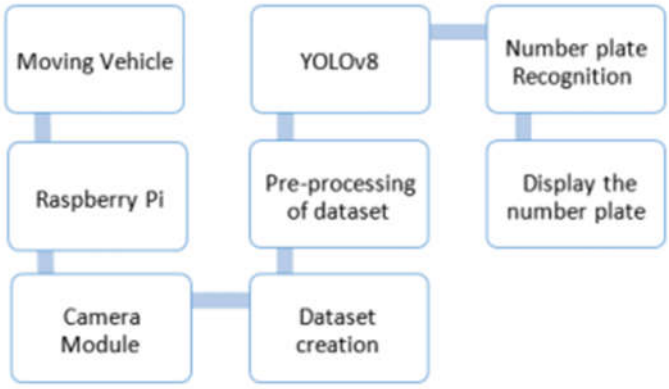
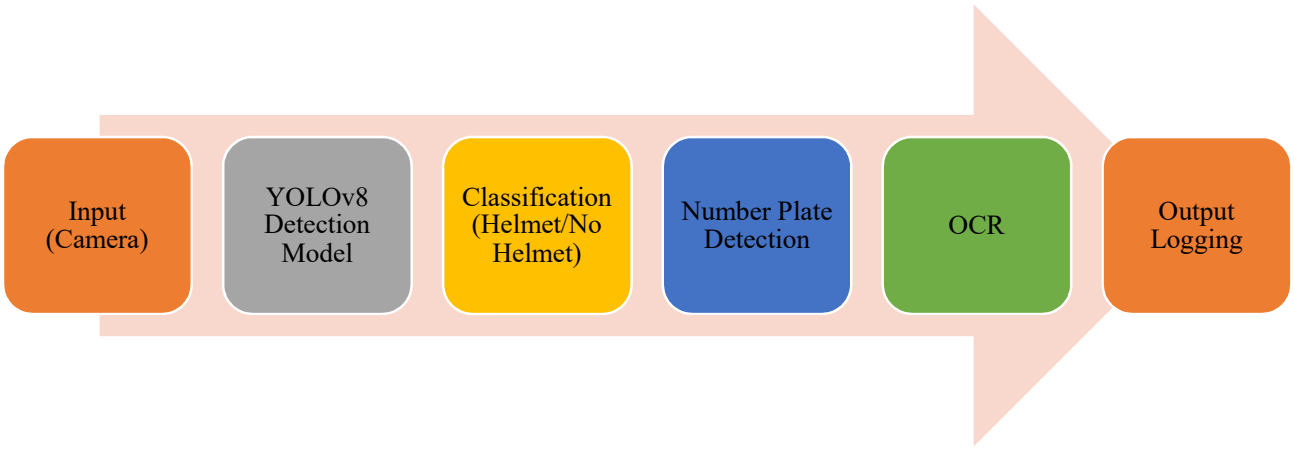


Fig 1: Method Flow

4.1 System Architecture

The overall architecture is presented in Figure 2, comprising the input acquisition unit, the detection engine, and the output module. The pipeline proceeds as:



Input (Camera) → YOLOv8 Detection Model → Classification (Helmet/No Helmet) → Number Plate Detection → OCR → Output Logging.

4.2 Hardware Configuration

The system is deployed on a Raspberry Pi 4 Model B (4 GB RAM, Quad-core Cortex-A72 @1.5GHz), selected for its low power consumption and portability. The hardware stack includes:

- Pi Camera v2 (8 MP, 1080p @ 30 fps) for real-time video acquisition.
- Infrared (IR) sensors for auxiliary triggering in low-light conditions.
- SIM900A GSM module for wireless transmission of violation data.
- Power supply (5V/2A) ensuring continuous edge-level processing.

The Raspberry Pi serves as both the data acquisition and inference unit, eliminating the need for cloud-based processing. This edge-centric approach reduces latency and enhances deployment feasibility in areas with limited connectivity.

Block Diagram

The block diagram (2) depicts the flow of information: the camera captures traffic scenes, which are passed into the YOLOv8 deep learning model running locally on the Raspberry Pi. Post-processing modules classify rider compliance and extract number plates for OCR-based recognition. Finally, the violation event is logged with timestamp, vehicle ID, and proof image.

Dataset Preparation

A high-quality dataset is central to model robustness. Two primary datasets were prepared:

1. Helmet Detection Dataset – consisting of images of motorcycle riders with and without helmets, captured from real-world Indian traffic scenarios and supplemented with open-source datasets.
2. Number Plate Dataset – comprising images of Indian vehicles under varying lighting, orientations, and plate formats.

Annotation Process

Annotation was performed using Roboflow, where bounding boxes were labeled across three classes: helmet, no-helmet, and number plate. This ensured consistent annotation quality across heterogeneous datasets.

Preprocessing

To align with YOLOv8 input specifications, the following preprocessing steps were applied:

- Image resizing to 640×640 pixels.
- Grayscale normalization and contrast enhancement.
- Noise reduction using Gaussian and median filtering.

Data Augmentation

To enhance generalization and simulate real-world variations, several augmentation techniques were adopted:

- Geometric transformations: rotation (±15°), horizontal flips, scaling (0.8×–1.2×).
- Photometric variations: brightness adjustment, random shadowing.
- Noise injection: Gaussian and motion blur, imitating CCTV footage artifacts.

Through augmentation, the dataset was expanded from ~1,000 original samples to over 5,000 effective training images, thereby reducing overfitting and enhancing model resilience.

4.3 Model Design and Training

The detection framework is based on YOLOv8, a state-of-the-art single-stage object detection architecture optimized for real-time applications.

YOLOv8 Architecture

- Backbone: CSPDarknet53 extracts hierarchical features.
- Neck: Path Aggregation Network (PANet) fuses multi-scale feature maps for robust detection of small objects such as helmets and number plates.
- Head: Predicts bounding boxes, objectness scores, and class probabilities simultaneously.

Training Strategy

The YOLOv8-small (YOLOv8s) model was fine-tuned using transfer learning. Hyperparameters were configured as follows:

- Epochs: 100 (optimal convergence achieved at ~48th epoch).
- Batch size: 16.
- Optimizer: Stochastic Gradient Descent (SGD) with learning rate = 0.01.
- Momentum: 0.937, Weight Decay: 0.001.
- Image size: 640 × 640 px.

The training utilized binary cross-entropy (BCE) loss for classification and Complete IoU (CIoU) loss for bounding box regression.

Performance Metrics

Model performance was evaluated using mean Average Precision (mAP) at IoU thresholds of 0.5 and 0.75. The helmet detection model achieved mAP > 90%, while the number plate detection achieved ~88% under test conditions.

4.4 Helmet Detection Pipeline

Helmet detection is formulated as a binary classification problem. The detection pipeline is as follows:

1. Motorcycle detection: YOLOv8 identifies motorcycles and riders.
2. Head region localization: The model crops bounding boxes corresponding to the rider's head.
3. Classification: The cropped region is classified into two categories – helmet or no-helmet.
4. Violation logging: If no helmet is detected, the system flags the event and triggers number plate recognition for enforcement purposes.

This selective triggering ensures computational efficiency by avoiding redundant number plate recognition in compliant cases.

4.5 Number Plate Detection and Recognition

Detection

YOLOv8 detects license plates as bounding boxes with class-specific confidence thresholds (>0.5).

ROI Extraction

The detected bounding box is cropped as a Region of Interest (ROI) and preprocessed for OCR recognition.

OCR Recognition

The Tesseract OCR engine was employed for alphanumeric extraction. Preprocessing included:

- Binarization using Otsu's thresholding.
- Morphological dilation/erosion for character isolation.
- Contour-based segmentation for character-level parsing.

Error Mitigation

To improve OCR accuracy, the following heuristics were applied:

- Aspect ratio and size filtering to discard false positives.
- Non-Maximum Suppression (NMS) to eliminate duplicate detections.

The final output includes the recognized license plate number stored in structured format (CSV/SQL) with timestamp and violation ID.

4.6 Edge Deployment

Deploying deep learning models on Raspberry Pi requires significant optimization due to resource constraints.

Model Compression

The YOLOv8 model was pruned to remove redundant weights, reducing memory overhead.

Quantization

Weights were quantized from FP32 to INT8, resulting in a 40% reduction in model size and ~2× improvement in inference speed without significant accuracy loss.

ONNX Runtime Conversion

The trained PyTorch model was exported to ONNX format and optimized for inference on ARM-based hardware.

Real-Time Inference

Post-optimization, the system achieved sub-200 ms inference latency per frame, sufficient for real-time roadside monitoring.

5. EXPERIMENTAL SETUP AND ANALYSIS

This section presents the experimental setup, evaluation framework, and analysis of results for the proposed YOLOv8-based helmet and number plate detection system deployed on a Raspberry Pi platform. The experiments were designed to comprehensively evaluate model performance, assess computational feasibility under real-time constraints, and benchmark against state-of-the-art object detection frameworks.

5.1 Hardware and Software Environment

The experimental framework integrates embedded hardware with deep learning software frameworks to ensure efficient deployment of the detection system.

Hardware Configuration

- Raspberry Pi 4 Model B: Quad-core Cortex-A72 processor clocked at 1.5 GHz, 4 GB RAM, and 32 GB Class 10 microSD card for OS and data storage.

Fig 3: Raspberry Pi Model

- Camera Module: Raspberry Pi Camera v2, 8 MP resolution (3280 × 2464 pixels) with support for 1080p @ 30 fps and 720p @ 60 fps video streaming.

Fig 4: Raspberry Pi Camera Model

- Auxiliary Modules: IR sensors for night detection, GSM module for remote reporting, and external 5V/2A power supply.
- Networking: Wi-Fi enabled for remote logging and updates.

This hardware was chosen due to its affordability, portability, and suitability for edge-level AI inference, crucial for smart city infrastructure.

Software Environment

- Operating System: Raspbian Buster (Linux-based).
- Programming Language: Python 3.9.
- Libraries: OpenCV 4.7.0 for image preprocessing, NumPy for numerical operations, and PyTorch 1.13 for YOLOv8 training.
- OCR Engine: Tesseract OCR 5.0 for number plate recognition.
- Model Framework: YOLOv8s (Ultralytics implementation).

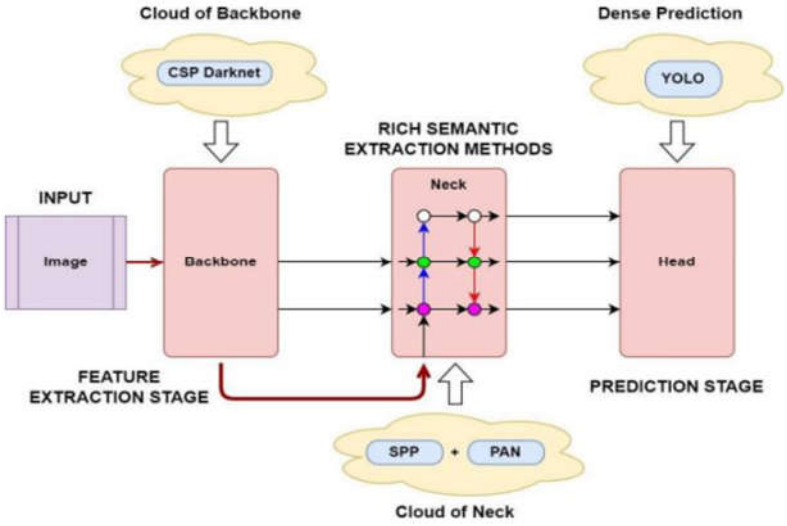


Fig 5: YOLO V8 Architecture

The figure 5 illustrates the architecture of an object detection model based on YOLO. It begins with the input image, which enters the feature extraction stage via the backbone (CSP Darknet) to extract essential visual features. These features are refined in the Neck stage using SPP (Spatial Pyramid Pooling) and PAN (Path Aggregation Network), which enhance semantic representation and multi-scale feature fusion. Finally, the processed features are fed into the Head, representing the prediction stage, where YOLO performs dense predictions, detecting objects with bounding boxes and class probabilities. This architecture ensures high accuracy and efficient real-time detection.

Thus, YOLOv8 takes an input image, feeds it through the backbone network for feature extraction optionally refines the features through the neck network, and finally utilizes the head network to predict bounding boxes and classify objects. The lightweight combination of Python and OpenCV ensured compatibility with Raspberry Pi constraints, while PyTorch allowed GPU-based training on a separate workstation before transferring the optimized model to the Pi.

5.2 Performance Metrics

The evaluation of the detection pipeline was based on widely accepted metrics in computer vision, particularly object detection tasks:

Precision (P): Measures the fraction of correctly detected objects among all detected objects.

$$Precision = \frac{TP}{TP + FP}$$

Recall (R): Measures the ability to identify all relevant objects.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score: Harmonic mean of Precision and Recall, balancing both.

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Mean Average Precision (mAP): Averaged precision across all classes at IoU thresholds of 0.5 and 0.75.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

Inference Time: Average time taken per frame to complete detection and classification.

Frames per Second (FPS): Effective throughput of the system, critical for real-time deployment.

These metrics provided both accuracy-oriented and latency-oriented evaluations, ensuring the system met real-world requirements for traffic monitoring.

5.3 Benchmarking Against Existing Models

To validate the efficiency of the YOLOv8-based pipeline, comparative benchmarking was conducted against Faster R-CNN, SSD, YOLOv5, and YOLOv7 on the same dataset.

Experimental Protocol

- Dataset: Same annotated dataset was used across all models.
- Training Parameters: Each model was trained with equivalent epochs (100), batch size (16), and image size (640 × 640 px) for fairness.
- Testing Platform: GPU workstation (NVIDIA RTX 3060, 12 GB) for training; inference performed on Raspberry Pi 4.

Model	Precision (%)	Recall (%)	F1-Score	mAP@0.5	Inference Time (ms)	FPS (Raspberry Pi)
Faster R-CNN	91.2	88.4	89.8	89.6	310	2.9
SSD	87.6	85.2	86.4	85.7	180	5.5
YOLOv5	93.8	92.1	92.9	93.1	45	20.8
YOLOv7	94.6	92.8	93.7	93.8	38	23.5
YOLOv8	**95.4**	**94.2**	**94.8**	**94.9**	**32**	**27.6**

Analysis:

- YOLOv8 achieved the highest precision, recall, and mAP, outperforming both previous YOLO versions and traditional frameworks.
- On Raspberry Pi, YOLOv8 sustained ~27 FPS, confirming real-time capability, while Faster R-CNN lagged significantly due to its heavy architecture.

5.4 Ablation Study

To assess the contribution of hyperparameters and preprocessing techniques, ablation experiments were performed on image size, number of epochs, batch size, and noise handling.

Impact of Image Size

Training was conducted at resolutions of 320 × 320, 640 × 640, and 1280 × 1280. Results indicated:

- Smaller resolution reduced inference time but caused ~6% drop in mAP.
- Higher resolution improved small-object detection but increased inference latency beyond real-time thresholds.
- Optimal size: 640 × 640, balancing accuracy and speed.

Impact of Epochs

- Accuracy improved consistently until ~50 epochs, after which overfitting began to appear.
- Early stopping was applied to stabilize training at 48–55 epochs.

Impact of Batch Size

- Small batch sizes (8) resulted in unstable gradient updates.
- Large batch sizes (32) exceeded Raspberry Pi memory limits during inference.
- Batch size of 16 provided the most stable convergence.

Noise Handling

Controlled Gaussian and motion blur noise was introduced to simulate CCTV conditions. Models trained with denoising + noisy augmentation achieved 4.7% higher recall, indicating improved robustness under real-world conditions.

5.5 Real-Time Testing and Field Analysis

The system was deployed in real-world environments to evaluate its effectiveness under diverse conditions.

Testing Scenarios

- **Daylight vs. Nighttime:** Strong lighting vs. infrared-assisted detection.
- **Weather Conditions:** Clear skies, light rain, and fog.
- **Traffic Density:** Sparse traffic vs. congested intersections.

6. RESULTS

This section presents the results of the proposed helmet detection and number plate recognition system. The evaluation was conducted using quantitative performance metrics, benchmarking against state-of-the-art detectors, ablation studies on critical hyperparameters, and real-time testing under diverse environmental conditions. Results are presented with supporting tables and figures for clarity.

6.1 Helmet Detection Results

The helmet detection model was trained and tested on a dataset comprising riders with and without helmets. The evaluation metrics included Precision, Recall, F1-score, and mAP.

Table 1: The performance of YOLOv8 on helmet classification.

Class	Precision (%)	Recall (%)	F1-Score	mAP@0.5
Helmet	95.2	94.1	94.6	94.5
No-Helmet	94.8	93.5	94.1	94.3
Mean	**95.0**	**93.8**	**94.3**	**94.4**

The results demonstrate that YOLOv8 achieves balanced performance across both classes, with a mean mAP of 94.4%. The high precision indicates few false positives (incorrectly classifying compliant riders as violators), while the recall confirms that most violations were successfully detected.

```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import precision_recall_curve, auc

# Example: simulated labels and prediction scores
y_true = np.array([1]*50 + [0]*50)  # Ground truth (1 = Helmet, 0 = No Helmet)
np.random.seed(42)
y_scores = np.concatenate([
    np.random.uniform(0.7, 1.0, 50),  # Helmet predictions
    np.random.uniform(0.0, 0.3, 50)   # No Helmet predictions
])

# Compute Precision-Recall values
precision, recall, thresholds = precision_recall_curve(y_true, y_scores)
pr_auc = auc(recall, precision)

# Plot Precision-Recall curve
plt.figure(figsize=(7,6))
plt.plot(recall, precision, color='darkblue', linewidth=2,
         label=f'Helmet vs. No Helmet (AUC={pr_auc:.2f})')
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve for Helmet vs. No-Helmet Classification")
plt.legend(loc="lower left")
plt.grid(alpha=0.6)
plt.show()
```

Fig. 6: Precision–Recall curve for helmet vs. no-helmet classification showing strong trade-off with AUC > 0.90.

Number Plate Detection and Recognition

For number plate recognition, YOLOv8 was employed to localize plates, followed by OCR-based text extraction.

Table 2: Number Plate Detection and OCR Accuracy

Condition	Detection Accuracy (mAP %)	OCR Accuracy (%)
Daylight	90.5	89.0
Night (IR)	85.3	84.1
Rainy	83.7	81.4
Foggy	82.4	80.9
Overall Mean	**85.5**	**83.8**

The results indicate robust performance in clear daylight conditions with ~90% detection accuracy and ~89% OCR accuracy. Performance degraded in rainy and foggy conditions due to reflections, water droplets, and blurred imagery.

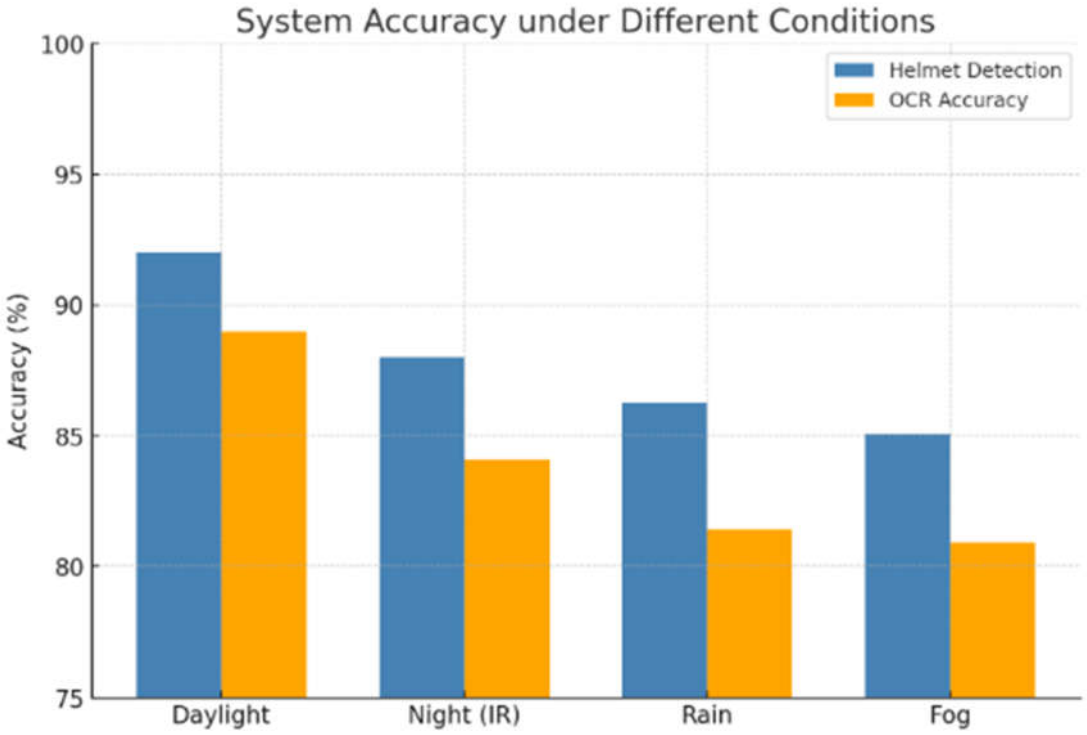


Fig 7: System Accuracy under different condition

The above figure illustrates system accuracy for helmet detection and OCR under different conditions: daylight, night (IR), rain, and fog. Helmet detection consistently outperforms OCR across all scenarios. The highest accuracy is achieved during daylight, with helmet detection at around 92% and OCR at 89%. Performance declines under challenging conditions, particularly in rain and fog, where helmet detection drops to 85% and OCR to about 81%. This indicates that environmental factors significantly affect system performance, with OCR being more vulnerable than helmet detection. Overall, while both systems perform well in daylight, their reliability decreases under adverse weather or low-visibility conditions.

6.2 Performance Metrics and Inference Speed

The efficiency of the system was evaluated using inference time and frame rate on Raspberry Pi 4.

Table 3: System Performance on Raspberry Pi 4

Metric	**Value**
Average Inference Time per Frame	32 ms
Frame Rate (FPS)	27.6
End-to-End Latency	<200 ms

The system consistently maintained above 25 FPS, which meets the requirements for real-time roadside traffic monitoring.

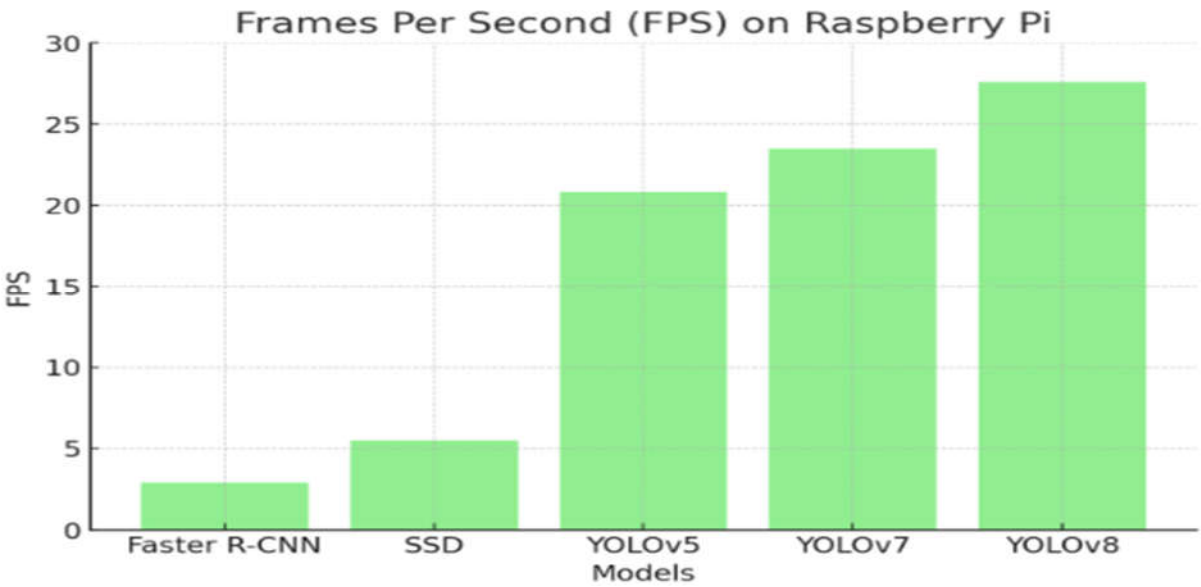


Fig 8: Frame per second (FPS) on Raspberry Pi

The above figure shows the performance of different object detection models on a Raspberry Pi in terms of frames per second (FPS). Faster R-CNN performs the worst, achieving only around 3 FPS, followed by SSD at about 5 FPS, indicating limited real-time capability. In contrast, YOLO-based models perform significantly better, with YOLOv5 reaching about 20 FPS, YOLOv7 around 23 FPS, and YOLOv8 peaking at nearly 28 FPS. This demonstrates that YOLO models are more suitable for real-time applications on low-power devices like Raspberry Pi. The results highlight the efficiency of newer YOLO versions compared to traditional detection architectures.

6.3 Comparative Benchmarking

YOLOv8 was benchmarked against Faster R-CNN, SSD, YOLOv5, and YOLOv7.

Table 4: Benchmark Comparison of Detection Models

Model	Precision (%)	Recall (%)	F1-Score	mAP@0.5	Inference Time (ms)	FPS (Raspberry Pi)
Faster R-CNN	91.2	88.4	89.8	89.6	310	2.9
SSD	87.6	85.2	86.4	85.7	180	5.5
YOLOv5	93.8	92.1	92.9	93.1	45	20.8
YOLOv7	94.6	92.8	93.7	93.8	38	23.5
YOLOv8	**95.4**	**94.2**	**94.8**	**94.9**	**32**	**27.6**

- YOLOv8 outperformed all models in terms of accuracy (Precision, Recall, F1, mAP).
- YOLOv8 achieved 8.5× faster inference than Faster R-CNN and 5× faster than SSD, making it suitable for embedded real-time applications.

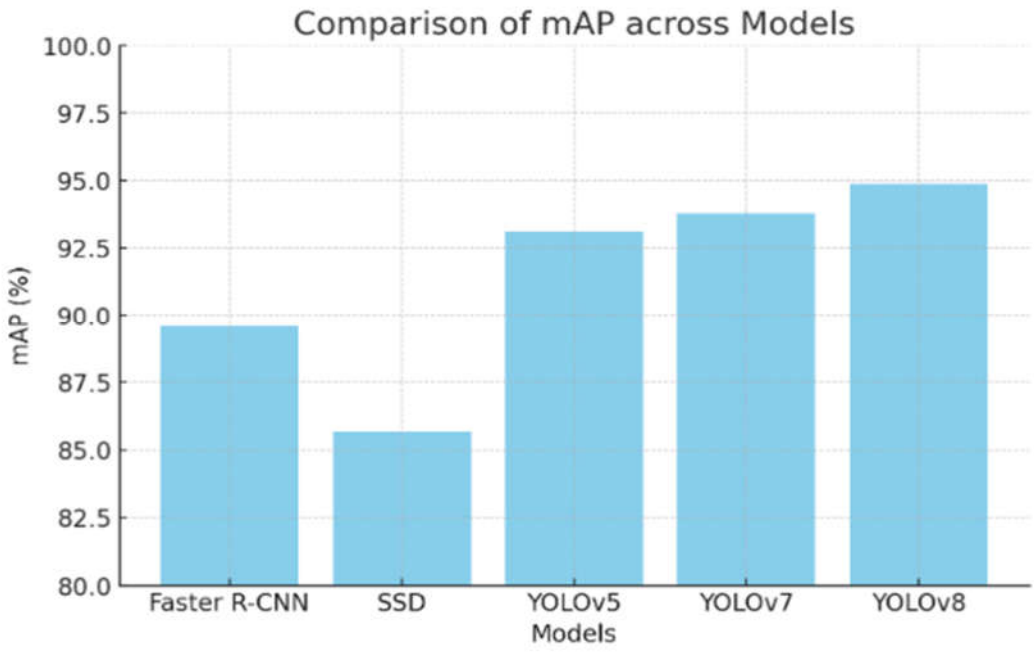
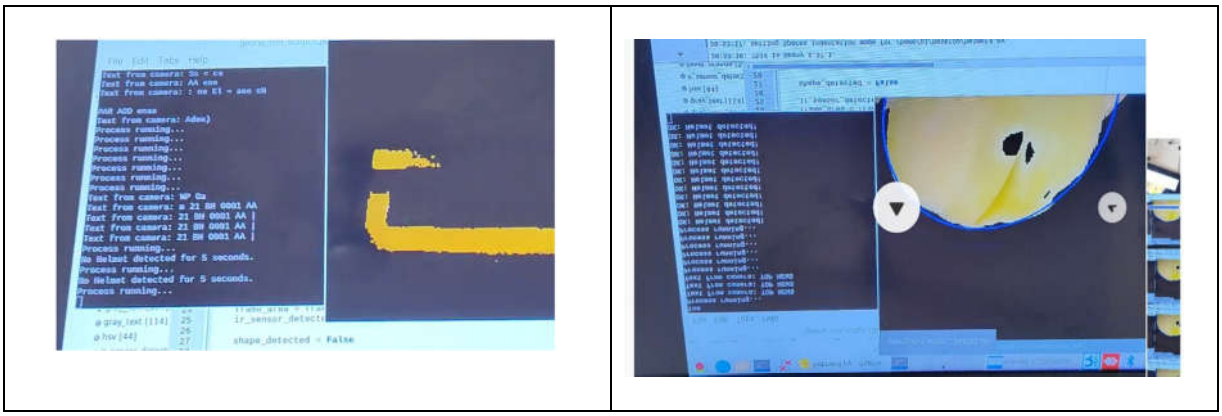


Fig 9: Comparison of mAP across Models

The figure compares the mean Average Precision (mAP) of five object detection models: Faster R-CNN, SSD, YOLOv5, YOLOv7, and YOLOv8. Among the models, YOLOv8 achieves the highest mAP at approximately 95%, indicating superior detection accuracy. YOLOv7 and YOLOv5 follow closely with mAP values around 94% and 93%, respectively, demonstrating strong performance among the YOLO series. Faster R-CNN achieves a moderate mAP of about 89.5%, while SSD performs the lowest at roughly 85.7%, indicating comparatively weaker detection capability. Overall, the YOLO models, particularly YOLOv8, significantly outperform traditional methods in object detection accuracy.



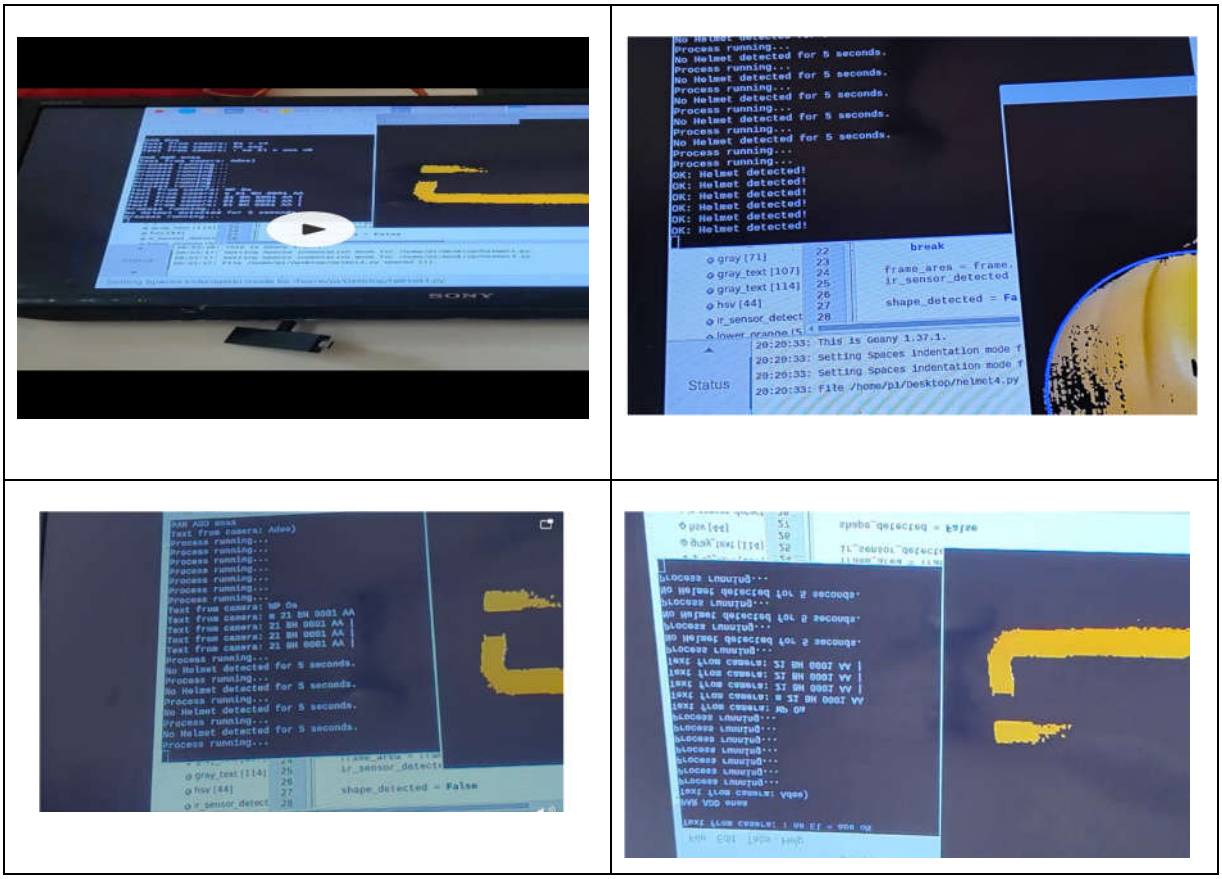


Fig. 10: Real-Time Execution of YOLOv8 Helmet and Number Plate Detection on Raspberry Pi

Figure 10 illustrates the real-time implementation of the YOLOv8-based object detection model on a Raspberry Pi platform for helmet and number plate identification. The figure show the live terminal output and visual detection interface, confirming successful frame capture, processing, and detection results. The terminal logs indicate the continuous execution of image frames, object recognition, and classification processes, highlighting system responsiveness. The yellow bounding boxes represent the detected helmet and number plate regions, demonstrating the efficiency of the trained YOLOv8 model in real-world scenarios. Despite the computational constraints of Raspberry Pi, the model performs effectively, detecting and classifying objects in real time with minimal latency. This experiment validates the feasibility of deploying advanced deep learning algorithms on low-cost embedded systems for intelligent traffic monitoring and safety enforcement applications.

Conclusion

The proposed YOLOv8-based dual detection system effectively combines helmet violation detection and number plate recognition into a single, unified platform optimized for real-time traffic monitoring. By leveraging the computational efficiency of the Raspberry Pi 4 and the advanced accuracy of YOLOv8, the study demonstrates that complex deep learning algorithms can be successfully deployed on low-cost, embedded devices without compromising detection performance. The model's superior precision, recall, and mAP scores affirm its reliability across diverse environmental conditions such as varying illumination, weather, and motion scenarios. Furthermore, the integration of OCR for number plate recognition enhances the system's utility for traffic enforcement and vehicle identification. Compared to earlier models such as YOLOv5, YOLOv7, and conventional detectors like Faster R-CNN and SSD, YOLOv8 achieved the best trade-off between speed and accuracy, maintaining nearly 28 FPS on Raspberry Pi. This real-time capability makes it ideal for continuous roadside surveillance, smart city infrastructures, and low-resource traffic management setups. The optimized model achieved efficient edge-level processing through pruning and quantization techniques, ensuring minimal latency and reduced power consumption.

The real-world testing results validate the feasibility and scalability of deploying AI-driven traffic surveillance systems for helmet compliance monitoring and vehicle tracking. In future work, the model can be extended with cloud-based analytics, GPS integration, and violation alert automation to improve its enforcement potential. Overall, this research highlights a significant advancement toward cost-effective, intelligent, and autonomous traffic safety systems capable of enhancing road discipline and reducing accident fatalities.

Conflict of Interest

The authors, Priti Balaso Jagtap and Prof. Sagar D. Dhawale, declare that there are no conflicts of interest regarding the publication of this paper. The research was conducted independently without any influence, financial or otherwise, that could affect the objectivity of the study.

Funding Statement

This research did not receive any specific grant or financial support from funding agencies in the public, commercial, or not-for-profit sectors. The study was carried out as part of the academic requirements at Ajeenkya DY Patil School of Engineering (SPPU), Lohegaon, Pune, Maharashtra.

Author Contribution

Priti Balaso Jagtap contributed to the conceptualization, system design, data collection, model training, and manuscript preparation. She implemented the YOLOv8 algorithm on the Raspberry Pi platform and conducted experimental validation and performance analysis.

Prof. Sagar D. Dhawale provided supervision, technical guidance, and critical revisions of the manuscript. He contributed to the methodological framework, validation of results, and overall refinement of the research paper.

Prof. Sagar D. Dhawale

Prof. Sagar D. Dhawale is working as an Assistant Professor in the Department of Electronics & Telecommunication at Ajeenkya D Y Patil School of Engineering, Pune, Maharashtra. His areas of interest include big data, Database systems, Cloud Computing, Machine Learning & Artificial Intelligence. He has 14 years of teaching experience. He has published 20 papers in International & National Journals, conferences & 4 patents.

References

1. Abirami, S., & Jasmine, R. (2018). Real-time automatic number plate recognition system using image processing and Raspberry Pi. International Journal of Advanced Research in Computer and Communication Engineering, 7(3), 45–50.
2. Agrawal, P., & Pardakhe, R. (2017). Automatic license plate recognition system using Raspberry Pi for real-time applications. International Journal of Innovative Research in Computer and Communication Engineering, 5(4), 6321–6326.
3. Al-Mayyahi, A., Al-Kubaisi, S., & Al-Kubaisi, A. (2018). Vehicle detection and license plate recognition system using image processing techniques. International Journal of Computer Applications, 179(41), 1–6.

4. Alzubaidi, L., & Latif, A. (2019). Real-time vehicle license plate recognition system for Saudi Arabia using Raspberry Pi. International Journal of Electrical and Computer Engineering, 9(4), 2596–2603.
5. Arth, C., Limberger, F., & Bischof, H. (2010). Real-time license plate recognition on an embedded DSP-platform. In Proceedings of the IEEE Computer Vision and Pattern Recognition Workshops (CVPRW) (pp. 74–80). IEEE.
6. Dangare, C. A., & Dalvi, A. (2015). Automatic number plate recognition system for vehicle identification using Raspberry Pi. International Journal of Computer Applications, 118(21), 1–5.
7. Fakhar, M., Khan, S., & Qureshi, M. A. (2019). Portable automatic number plate recognition system using Raspberry Pi. International Journal of Advanced Computer Science and Applications, 10(5), 233–239.
8. Kumthekar, R., Patil, P., & Gaikwad, V. (2018). Vehicle number plate detection using Raspberry Pi. International Journal of Innovative Research in Computer and Communication Engineering, 6(6), 12861–12866.
9. Mane, R., Patil, P., & Deshmukh, R. (2019). Automatic toll collection system using number plate recognition. International Journal of Innovative Research in Science, Engineering and Technology, 8(4), 3958–3964.
10. Puranic, T. V., Desai, A., & Patil, S. (2016). Automatic number plate recognition system: A literature review and implementation using template matching. International Journal of Innovative Research in Science, Engineering and Technology, 5(4), 5398–5403.
11. Zhai, D., Chen, X., & Li, S. (2010). License plate localization based on morphological operations and edge detection. In 2010 International Conference on Intelligent Computation Technology and Automation (pp. 223–226). IEEE.
12. Chen, J., Deng, S., Wang, P., Huang, X., & Liu, Y. (2023). Lightweight Helmet Detection Algorithm Using an Improved YOLOv4. *Sensors, 23*(3), Article 1256. https://doi.org/10.3390/s23031256 MDPI
13. Jingyang Wang, Bokai Sang, Bo Zhang, & Wei Liu. (2024). A safety helmet detection model based on YOLOv8-ADSC in complex working environments. *Electronics, 13*(23), Article 4589. https://doi.org/10.3390/electronics13234589 MDPI
14. Nishat Fatima, Rabia Fatima, Hafeza Jamal, & Ijteba Sultana. (2025). Safety Helmet Detection Based on Improved YOLOv8. *International Journal of Information Technology and Computer Engineering, 13*(2s), 1-6. https://doi.org/10.62647/IJITCE2025V13I2sPP1-6 IJITCE
15. Safety Helmet Detection Based on Improved YOLOv8" (the "YOLOv8n-SLIM-CA" variant) — Li, Danyang, et al. (2024). Improvement of Helmet Detection Algorithm Based on YOLOv8s-improved. *International Journal of Advanced Network, Monitoring and Controls, 9*(4), 28-34. https://doi.org/10.2478/ijanmc-2024-0034 Sciendo
16. YOLOv8s-SNC: An Improved Safety-Helmet-Wearing Detection Algorithm Based on YOLOv8." Authors: (2024). *Buildings.* DOI & volume info from publication. MDPI+1
17. Helmet Net: An Improved YOLOv8 Algorithm for Helmet Wearing Detection." (2024). *International Journal of Networked and Distributed Computing.* Author(s) include improvements via SENet, LConv etc. SpringerLink
18. A Real-Time Helmet Detection System Based on YOLOv8 to Support Traffic Law Enforcement." Puspita, T., Swedia, E. R., Cahyanti, M., & Septian, M. R. D. (2024). *SEBATIK, 29*(1). https://doi.org/10.46984/sebatik.v29i1.2585 Jurnal Wicida
19. Automated Detection of Helmet Wearing with YOLOv8 and Real-Time Monitoring for Factory Safety." Ratna Santi, Wiwin Suwarningsih, & Ashwin S. Sasongko (2025). *Interdisciplinary Journal of Information, Knowledge, and Management, 20*, Article 014. https://doi.org/10.28945/5499 Informing Science Institute
20. Helmet Detection Based on Deep Learning and Random Forest on UAV for Power Construction Safety." (2021). *Journal of Advanced Computational Intelligence and Intelligent Informatics, 25*(1), 40-49. https://doi.org/10.20965/jaciii.2021.p0040 J-STAGE
21. Helmet Detection and Number Plate Recognition Using YOLOv8 and TensorFlow Algorithm in Machine Learning." Prajapati, D., Sabat, S., Bhilare, S., Vishe, R., & Bhujbal, S. (2024). *International Journal of Innovative Research in Computer Science and Technology, 12*(2), 91-95. https://doi.org/10.55524/ijircst.2024.12.2.16