International Journal of Linguistics and computational Applications (IJLCA) ISSN 2394-6385(Print)
Volume 13, Issue 2, April – June 2026 ISSN 2394-6393(Online)

Smart Traffic Violation Detection and Reporting System

Boggala Prudhvi Teja[*1], Dr.S.Usharani[2]

[1]*Student, Department of Computer Applications, Viswam Engineering College, Andhra Pradesh, India*
[2]*Professor & Head, Department of Computer Applications, Viswam Engineering College, Andhra Pradesh, India*

Abstract — The rapid increase in urban vehicle density has posed significant challenges to traditional traffic management systems, which rely heavily on manual monitoring and enforcement. These conventional approaches are often inefficient, prone to human error, and incapable of providing continuous surveillance across large-scale road networks. To address these limitations, this paper presents a Smart Traffic Violation Detection and Reporting System that leverages advancements in computer vision, deep learning, and web technologies to automate traffic law enforcement processes.The proposed system utilizes the YOLOv8 object detection model for real-time identification of vehicles and detection of traffic violations such as signal jumping, over-speeding, helmet violations, triple riding, and wrong-way driving. Automatic Number Plate Recognition (ANPR) is implemented using EasyOCR to extract vehicle registration numbers from captured frames. The system further integrates an AI-based event summarization module that generates structured and human-readable violation reports to assist enforcement authorities. A Django-based web application serves as the backend framework, enabling efficient data management, secure user authentication, and real-time monitoring through an interactive dashboard. The system also incorporates analytics features to identify traffic violation trends and hotspots, facilitating data-driven decision-making. The entire architecture is built using open-source tools such as Python, OpenCV, PyTorch, and Chart.js, ensuring scalability and cost-effectiveness. The proposed solution enhances traffic monitoring efficiency, reduces dependency on manual intervention, and contributes to improved road safety by enabling consistent and accurate enforcement of traffic regulations.

Keywords — *Traffic Violation Detection; YOLOv8; Computer Vision; ANPR; Deep Learning; Smart Traffic System.*

1. Introduction

Urban transportation systems are experiencing unprecedented growth due to rapid urbanization and increasing vehicle ownership. This surge has led to traffic congestion, increased accident rates, and frequent violations of traffic regulations. Traditional traffic enforcement methods, which rely primarily on manual observation by traffic personnel, are no longer sufficient to handle the scale and complexity of modern traffic systems. Manual monitoring suffers from several drawbacks, including limited coverage, human fatigue, inconsistency in enforcement, and high operational costs. Moreover, the absence of continuous monitoring allows many violations to go unnoticed, thereby reducing the effectiveness of traffic laws and compromising road safety.

Recent advancements in artificial intelligence (AI) and computer vision have opened new possibilities for intelligent traffic management. Deep learning-based object detection models, particularly those from the YOLO (You Only Look Once) family, have demonstrated remarkable performance in real-time detection tasks. These technologies enable automated monitoring of traffic scenes, accurate identification of vehicles, and detection of violations without human intervention. The Smart Traffic Violation Detection and Reporting System is designed to leverage these advancements by integrating computer vision techniques with a robust web-based platform. The system processes live or recorded video streams from CCTV cameras, detects violations using deep learning models, extracts vehicle registration numbers through ANPR, and generates structured reports for enforcement authorities. The platform also provides a user-friendly dashboard for monitoring violations, reviewing evidence, and analyzing traffic patterns. By automating the detection and reporting process, the system significantly enhances enforcement efficiency and ensures consistent application of traffic rules.

2. Literature Survey

The development of intelligent traffic monitoring systems has gained significant attention in recent years due to the increasing demand for automated enforcement and smart city solutions. Several research works have explored the use of computer vision, machine learning, and IoT technologies to improve traffic management and violation detection. Early traffic monitoring systems relied on sensor-based approaches such as inductive loop detectors and radar-based speed measurement devices. While these systems provided basic functionality, they lacked flexibility and were limited to specific use cases such as speed detection. Moreover, their installation and maintenance costs were relatively high. With the advancement of image processing techniques, vision-based systems emerged as a more scalable solution. Traditional computer vision approaches used background subtraction, edge detection, and feature extraction techniques to identify vehicles and detect violations. However, these methods were highly sensitive to environmental conditions such as lighting, weather, and

DOI: 10.30726/ijlca/v13.i2.2026.132016

occlusions, leading to reduced accuracy. The introduction of deep learning revolutionized the field of traffic monitoring. Convolutional Neural Networks (CNNs) enabled more robust and accurate object detection and classification. Among various models, the YOLO (You Only Look Once) series gained prominence due to its ability to perform real-time object detection with high accuracy. Studies have demonstrated that YOLO-based systems can effectively detect vehicles, pedestrians, and traffic signals in dynamic environments.

Recent research has also focused on integrating Automatic Number Plate Recognition (ANPR) systems with traffic monitoring frameworks. Techniques such as Optical Character Recognition (OCR) using deep learning models have significantly improved the accuracy of license plate detection and recognition under varying conditions. EasyOCR and similar frameworks have been widely adopted for their efficiency and multilingual support. In addition to detection, modern systems incorporate data analytics and visualization tools to provide actionable insights. Web-based dashboards powered by frameworks such as Django and visualization libraries like Chart.js enable real-time monitoring and analysis of traffic patterns. These systems support decision-making by identifying high-risk zones and peak violation periods. Despite these advancements, many existing solutions are either limited in scope, expensive due to proprietary dependencies, or lack integration across multiple functionalities such as detection, reporting, and analytics. The proposed Smart Traffic Violation Detection and Reporting System addresses these gaps by providing a unified, cost-effective, and scalable solution using open-source technologies.

3. System Architecture

3.1 Overall System Architecture

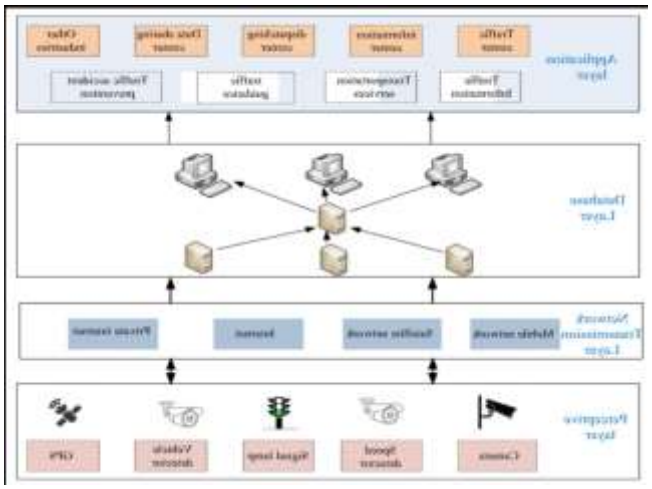

Fig.1: Overall System Architecture

The Smart Traffic Violation Detection and Reporting System is designed using a layered and modular architecture that ensures scalability, flexibility, and efficient processing of real-time traffic data. The architecture is divided into five major layers, each responsible for a specific set of functionalities.

3.1.1 Data Acquisition Layer

This is the foundational layer of the system responsible for capturing video data from surveillance sources. The system connects to CCTV cameras using RTSP streams or processes stored video files. OpenCV's Video Capture functionality is used to retrieve frames continuously.

3.1.2 AI Processing Layer

The AI Processing Layer forms the core intelligence of the system. It utilizes the YOLOv8 deep learning model to perform real-time object detection. Vehicles such as cars, bikes, buses, and trucks are identified with high accuracy.

3.1.3 Event Processing Layer

This layer transforms raw detection outputs into structured violation records. Once a violation is detected:

- The frame is captured as evidence
- AI-generated summaries are created
- Metadata such as time, location, and confidence score is recorded

The processed data is then stored in the database for further review. This layer ensures that all detected events are properly logged and documented.

3.1.4 Application Layer

The Application Layer is implemented using the Django framework. It handles:

- Backend logic
- API routing
- Database interaction
- User authentication

It manages violation records, user access, and system operations. This layer acts as a bridge between the AI system and the user interface.

3.1.5 Presentation Layer

The Presentation Layer provides a user-friendly web interface for traffic authorities. It includes:

- Dashboard for real-time monitoring
- Violation list for review
- Detailed evidence view
- Analytics visualization

DOI: 10.30726/ijlca/v13.i2.2026.132016

The interface is designed with modern UI principles, ensuring ease of use even for non-technical users.

3.2 System Workflow

The system workflow consists of two major processes: automated detection and human validation.

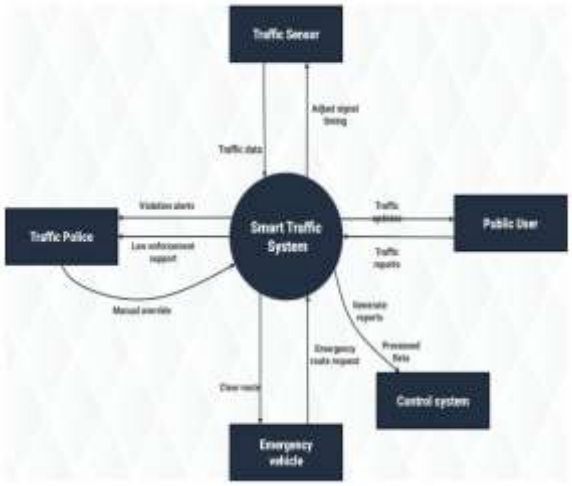

Fig. 2: System Work flow

The Step-by-Step Workflow as,

- Video stream is captured from CCTV cameras
- Frames are extracted using OpenCV
- YOLOv8 detects vehicles and objects
- Violation detection logic is applied
- ANPR extracts vehicle number
- AI generates violation summary
- Evidence image is stored
- Data is saved in the database
- Officer reviews violation via dashboard
- Officer validates or rejects the violation

The Key Features of Architecture is,

- Modular Design → Easy to upgrade individual components
- Scalable System → Supports multiple camera streams
- Real-time Processing → Immediate detection and reporting
- Human-in-the-loop → Ensures accuracy and legal reliability.

3.3 UML Diagrams

3.3.1 Use Case Diagram

The Use Case Diagram of the Smart Traffic Violation Detection and Reporting System involves three major actors: System Administrator, Traffic Official, and AI Detection Engine. The System Administrator is responsible for configuring camera details, managing user accounts, monitoring system health, and reviewing analytical reports. The Traffic Official uses the web dashboard to log in, review pending violations, inspect evidence images, validate or reject violations, search historical records, and view traffic analytics. The AI Detection Engine operates as the automated actor that captures video frames, detects vehicles, classifies violations, performs ANPR, generates incident summaries, and logs violation records into the database. This use case structure clearly shows how the system combines automation with human validation for accurate and reliable traffic enforcement.

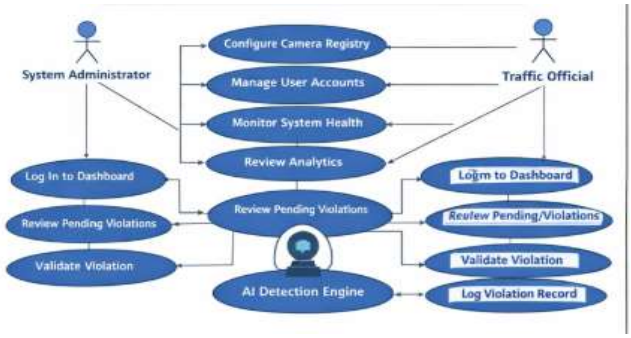

Fig. 3: Use Case Diagram of Smart Traffic Violation Detection and Reporting System

3.3.2 Class Diagram

The class diagram defines the system's core structure. The Camera class stores details like name, location, IP, and status. The Violation class records violation data, including type, vehicle number, timestamp, and evidence. A one-to-many relationship links Camera to Violation. TrafficLog handles analytics data, while TrafficDetector manages detection and processing.

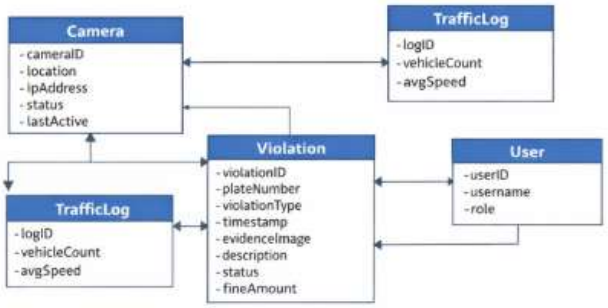

Fig. 4: Class Diagram of Smart Traffic Violation Detection and Reporting System

Main Classes:

- Camera
- Violation
- TrafficLog

DOI: 10.30726/ijlca/v13.i2.2026.132016

- TrafficDetector
- User

Key Relationships:
- One Camera → Many Violations
- One Camera → Many TrafficLogs
- One User → Reviews Many Violations
- TrafficDetector → Detects Violations
- Violation → Stores Evidence and Summary

3.3.3 Sequence Diagram

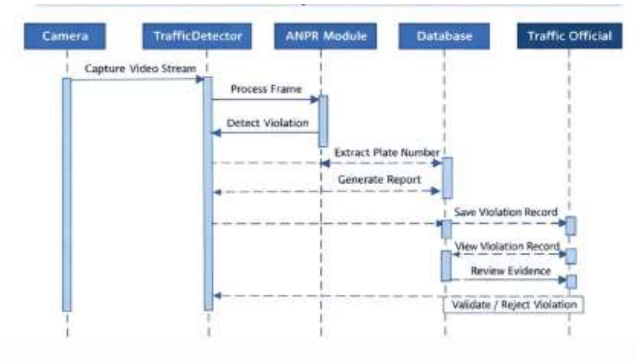

Fig. 5: Sequence Diagram of Smart Traffic Violation Detection and Reporting System

The sequence diagram shows the flow of traffic violation detection. The camera sends video to the Traffic Detector, which uses YOLOv8 to detect violations. ANPR extracts the number plate, and an AI module generates a summary. The data is stored as a violation record, which traffic officials later review and validate or reject.

Sequence Flow:
- Camera sends video stream
- Traffic Detector reads frame
- YOLOv8 detects object/vehicle
- Violation logic checks rule breach
- EasyOCR extracts number plate
- AI Summarizer creates event description
- Database stores violation record
- Traffic Official opens dashboard
- Official reviews evidence
- Official validates or rejects violation

3.3.4 Activity Diagram

The activity diagram shows the workflow from video capture to enforcement. CCTV frames are analyzed to detect vehicles and violations. If none are found, monitoring continues. If detected, the system extracts the number plate, generates a summary, stores evidence, and logs the violation. The officer reviews and validates or rejects the record.

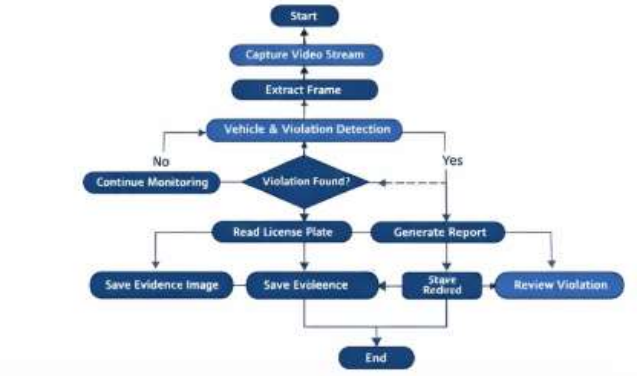

Fig. 6: Activity Diagram of Smart Traffic Violation Detection and Reporting System

4. Implementation

4.1 Modules

The Smart Traffic Violation Detection and Reporting System is organized into six major modules: Camera Integration and Video Ingestion, AI Vehicle Detection, Automatic Number Plate Recognition, AI Event Summarization, Django Web Application, and Traffic Analytics. This modular organization supports independent development, testing, and maintenance, while also ensuring that each functional part of the system can be improved without disturbing the complete workflow. The project document itself defines these six modules and presents them as the core implementation structure of the system.

4.2 Module Description

4.2.1 Camera Integration and Video Ingestion Module

The Camera Integration and Video Ingestion Module is responsible for establishing connections with the available surveillance sources and continuously supplying video frames to the processing engine. This module uses OpenCV's VideoCapture functionality to support RTSP camera streams, local video files, and webcam-based inputs. The system maintains a camera registry through the Camera model, which stores details such as camera name, location, IP address, operational state, and last active timestamp.

4.2.2 AI Vehicle Detection Module

The AI Vehicle Detection Module forms the intelligent core of the proposed system. It is implemented through the TrafficDetector class and uses the Ultralytics YOLOv8 model for real-time object detection. The detector identifies vehicles such as cars, buses, trucks, and motorcycles, and then applies traffic-domain logic to determine whether any violation has occurred. The implementation supports

DOI: 10.30726/ijlca/v13.i2.2026.132016

violation categories including signal jumping, speeding, helmet absence, triple riding, and wrong-way driving.

4.2.3 Automatic Number Plate Recognition Module

The Automatic Number Plate Recognition Module is designed to identify the registration number of vehicles involved in detected violations. Once the vehicle region or plate region is localized, the system performs preprocessing operations such as grayscale conversion, contrast enhancement, and optional alignment correction to improve recognition accuracy.

4.2.4 AI Event Summarization Module

The AI Event Summarization Module converts raw machine detections into readable violation descriptions for traffic officials. The implementation uses the generate_violation_summary function to construct structured incident reports using violation type, vehicle number, location, and confidence score. Each generated summary includes a contextual explanation and may also include a recommendation for review or action.

4.2.5 Django Web Application Module

The Django Web Application Module provides the backend and user-facing control environment of the system. It manages authentication, routing, violation records, review actions, and page rendering. The application follows Django's Model-View-Template architecture, with models handling database structure, views processing requests, and templates generating the final interface. Through this module, officers can log in securely, inspect pending violations, view evidence images, validate or reject records, and monitor the overall status of the system.

4.2.6 Traffic Analytics Module

The Traffic Analytics Module is responsible for transforming recorded traffic data into visual insights. It aggregates violation records across time, location, and category, then presents them through charts and dashboard elements. The implementation uses Django ORM for grouped queries and Chart.js for visualization. Through this module, supervisors can identify peak violation periods, common violation types, and hotspot locations with high traffic rule violations

5. System Requirements

5.1 Hardware Requirements

The Smart Traffic Violation Detection System requires hardware capable of real-time video processing and deep learning inference. A multi-core processor (Intel i5 or AMD Ryzen 5), 16 GB RAM, and at least 256 GB SSD are recommended. A dedicated GPU like NVIDIA GTX 1060 improves detection speed and supports multiple camera streams. Gigabit Ethernet ensures stable video transmission. IP-based CCTV cameras with 1080p resolution and RTSP streaming are required for accurate detection and number plate recognition. A Full HD display supports monitoring. The system can be scaled based on camera count and traffic load.

5.2 Software Requirements

The system uses open-source technologies for flexibility and cost-effectiveness. It is developed in Python 3.10+, with Django for backend operations. OpenCV handles image processing, YOLOv8 enables object detection, EasyOCR performs license plate recognition, and PyTorch supports deep learning. NumPy and Pillow assist processing, while Chart.js provides visualization. It runs on Windows, Ubuntu, or macOS. SQLite is used for development, and PostgreSQL is recommended for scalable production deployment.

6. Testing of the System

Testing is a crucial phase in the development of the Smart Traffic Violation Detection and Reporting System to ensure reliability, accuracy, and performance under real-world conditions. The system undergoes multiple levels of testing including unit testing, integration testing, performance testing, and user acceptance testing.

6.1 Unit Testing

Individual modules are tested independently to ensure correctness. Functions like violation detection, AI summarization, and database operations are validated using various inputs and edge cases with Django's testing tools.

6.2 Integration Testing

This verifies interaction between modules by simulating the full workflow—from video capture to detection, number plate recognition, summary generation, and database storage. The officer review process is also tested.

6.3 Performance Testing

System efficiency is evaluated by measuring YOLOv8 frame processing speed and web response times. GPU-based setups improve real-time detection and support multiple streams, while the web interface is optimized for responsiveness.

DOI: 10.30726/ijlca/v13.i2.2026.132016

6.4 User Acceptance Testing

End users test the system through real scenarios such as reviewing and validating violations. Feedback on usability and performance ensures the system meets practical requirements and supports effective deployment.

7. Conclusion and Future Enhancement

7.1 Conclusion

The Smart Traffic Violation Detection System integrates computer vision and web technologies to automate traffic monitoring and reporting. It reduces manual effort and errors by detecting violations in real time using the YOLOv8 model. Automatic Number Plate Recognition links violations to vehicles, while AI summarization converts outputs into readable reports for easier review.
The Django-based web platform provides a secure interface for viewing, validating, and analyzing violations. Built-in analytics help authorities identify traffic patterns and improve decision-making for road safety.

Overall, the system is scalable, efficient, and cost-effective, improving enforcement accuracy and supporting safer and more organized traffic management.

7.2 Future Enhancement

- Fine-tune YOLOv8 using locally collected traffic data for improved accuracy in different conditions
- Implement automated e-challan generation for end-to-end violation processing
- Integrate SMS and email notifications along with payment gateway support
- Adopt distributed processing architecture for large-scale multi-location deployment
- Use edge computing to reduce latency and improve real-time performance
- Integrate with traffic signal systems for accurate signal violation detection
- Apply predictive analytics to identify high-risk zones and peak violation times
- Develop a mobile application for field officers for real-time monitoring and enforcement

References

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.
[2] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” arXiv preprint arXiv:2004.10934, 2020.
[3] Glenn Jocher et al., “Ultralytics YOLOv8 Documentation,” Ultralytics, 2023.
[4] Adrian Rosebrock, “Deep Learning for Computer Vision with Python,” PyImageSearch, 2019.
[5] Jaided AI, “EasyOCR: Ready-to-use OCR with Deep Learning,” GitHub Repository, 2020.
[6] Gary Bradski, Adrian Kaehler, “Learning OpenCV: Computer Vision with the OpenCV Library,” O’Reilly Media, 2008.
[7] Adam Paszke et al., “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” Advances in Neural Information Processing Systems (NeurIPS), 2019.
[8] Django Software Foundation, “Django Web Framework Documentation,” Version 5.0, 2024.
[9] Chart.js Contributors, “Chart.js Documentation: Simple yet Flexible JavaScript Charting,” 2023.
[10] S. Sivaraman and M. M. Trivedi, “Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis,” IEEE Transactions on Intelligent Transportation Systems, Vol. 14, Issue 4, 2013, pp. 1773–1795.
[11] R. C. Gonzalez and R. E. Woods, “Digital Image Processing,” 4th Edition, Pearson Education, 2018.
[12] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” IEEE CVPR, 2005.
[13] M. A. Hossain et al., “Real-Time Traffic Monitoring System Using Computer Vision and IoT,” International Journal of Advanced Computer Science and Applications, Vol. 12, Issue 6, 2021.
[14] K. He, X. Zhang, S. Ren, J. Sun, “Deep Residual Learning for Image Recognition,” IEEE CVPR, 2016.
[15] A. Krizhevsky, I. Sutskever, G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” NIPS, 2012.

DOI: 10.30726/ijlca/v13.i2.2026.132016