IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

Intelligent Traffic Surveillance System for Automated Violation Detection Using YOLOv8, DeepSORT, and OCR

Rakesh Bhakre[1], Namrata Madane[2], Madhur Narkhede[3], Prof. Vaishali Yeole[4]
Department of Computer Science and Engineering
Smt. Indira Gandhi College of Engineering, Mumbai, India

Abstract: *Traffic violations are one of the major causes of road accidents and congestion in urban areas. Traditional monitoring methods rely heavily on manual supervision, which is not only time-consuming but also prone to human error. This paper presents a real-time traffic violation detection system that uses deep learning and computer vision techniques to automate the process.*

The proposed system integrates YOLOv8 for vehicle detection, DeepSORT for tracking, and Optical Character Recognition (OCR) for extracting vehicle number plates. It is capable of identifying multiple violations such as red-light jumping, lane violations, overspeeding, and wrong-way driving. The system processes live video feeds from CCTV cameras and generates evidence in the form of images along with violation details.

The model is designed to work efficiently even on CPU-based systems and does not require high-end hardware. Experimental results show that the system achieves good accuracy while maintaining real-time performance. This makes it suitable for practical deployment in smart city traffic management systems.

Keywords: *Traffic violation detection, YOLOv8, DeepSORT, optical character recognition, homography, speed estimation, computer vision, intelligent transportation systems*

I. INTRODUCTION

Road traffic violations are a leading contributor to road accidents, fatalities, and urban congestion worldwide. According to the World Health Organization, approximately 1.35 million people die in road crashes each year, with a significant proportion attributed to speeding, signal violations, and wrong-way driving [1]. Traditional enforcement methods rely heavily on police personnel stationed at intersections, an approach that is neither scalable nor cost-effective for the thousands of kilometres of roadways in modern cities.

The proliferation of Closed-Circuit Television (CCTV) cameras at road intersections presents an unprecedented opportunity for automated enforcement. Modern deep learning object detection models, particularly the YOLO (You Only Look Once) family, have demonstrated near-real-time inference capabilities with accuracy comparable to human observers. However, most existing systems focus on a single violation type, require model training on domain-specific datasets, or operate only under controlled laboratory conditions.

This paper addresses these gaps by presenting an integrated, multi-violation traffic enforcement system that operates exclusively on pre-trained model weights, requires no GPU hardware, and is fully configurable for any fixed-camera deployment through a single YAML configuration file. The system is designed around a modular pipeline architecture, where each processing stage is an independent, testable software component.

DOI: 10.48175/IJARSCT-33451

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

A. Problem Statement

The primary objective is to build an automated system that can: (i) ingest a live video stream from a fixed CCTV camera, (ii) detect and persistently track all vehicles in the frame, (iii) evaluate five categories of traffic violations in real time using rule-based logic, (iv) associate detected violations with licence plate numbers, and (v) persist violation evidence and metadata for downstream enforcement workflows.

B. Contributions

The key contributions of this work are:

A unified, modular pipeline integrating object detection, multi-object tracking, scene understanding, speed estimation, and OCR into a single deployable system.

A pure computer vision approach to traffic light state detection using HSV colour segmentation, avoiding the computational overhead of an additional neural network.

A debounced, stateful violation engine with five independent checkers, each with configurable thresholds and per-track cooldowns to minimise false alarms.

An interactive camera calibration tool and homography-based metric speed estimation without specialised hardware.

A complete open-source implementation with a REST API, SQLite logging, and Docker-ready structure.

II. RELATED WORK

Traffic violation detection has been an active research domain for over two decades. Early systems relied exclusively on inductive loop detectors and fixed-position radars for speed measurement, requiring invasive road infrastructure and offering no visual evidence.

Image-based detection systems emerged with the adoption of background subtraction and optical flow techniques [2]. While computationally lightweight, these methods suffer under varying illumination and crowded scenes. Redmon et al. introduced the YOLO architecture [3], enabling single-pass, real-time object detection that became the backbone of most subsequent vision-based systems.

Bewley et al. [4] proposed SORT (Simple Online and Realtime Tracking), later extended to DeepSORT [5] by Wojke et al. through the addition of a deep appearance descriptor. DeepSORT significantly improved tracking under occlusion and at intersections where vehicle paths frequently cross.

Recent studies by Mandal et al. [6] and Khan et al. [7] demonstrated YOLOv5-based red-light violation detection with accuracy exceeding 90% under daytime conditions, but neither system integrated speed estimation or multi-violation detection. The system proposed by Lim et al. [8] used homographic projection for speed estimation but required manual per-camera calibration and lacked persistent tracking identity.

In contrast to prior art, our system unifies detection, tracking, multi-violation evaluation, speed estimation, OCR, and evidence management into a single framework, while remaining deployable without model training or GPU acceleration.

III. SYSTEM ARCHITECTURE

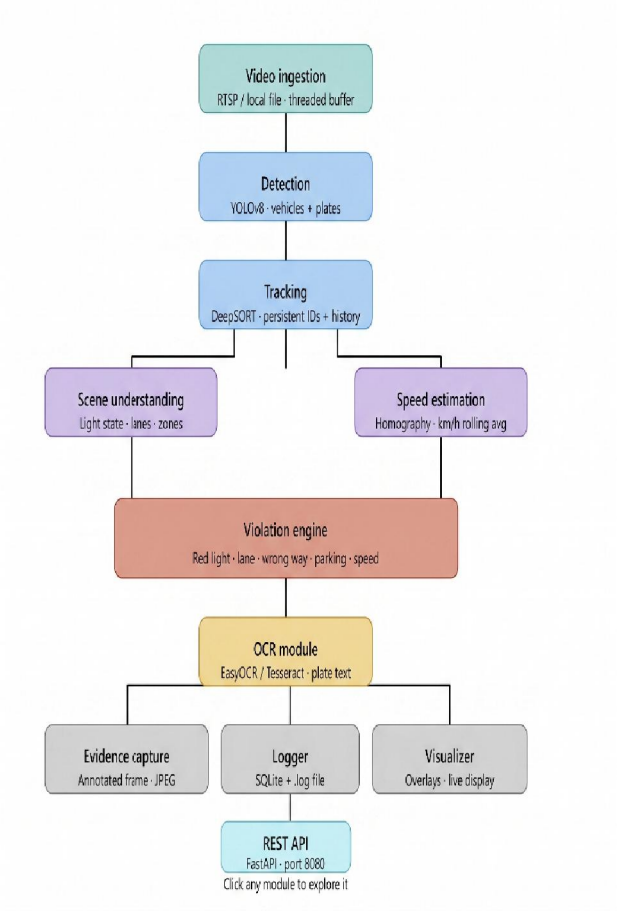
The proposed system follows a modular architecture where each component performs a specific task. The overall workflow begins with capturing video input from CCTV cameras.

The video frames are passed to the YOLOv8 model, which detects vehicles such as cars, bikes, buses, and trucks. These detections are then fed into the DeepSORT tracking algorithm, which assigns a unique ID to each vehicle and tracks it across frames.

The system also includes a scene understanding module that identifies regions such as lanes, stop lines, and traffic signals. Based on this information, a violation detection engine applies predefined rules to determine whether a vehicle has violated any traffic laws.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

Once a violation is detected, the system extracts the vehicle region and applies OCR to recognize the number plate. The evidence is then stored along with relevant details such as time, location, and type of violation.



Module Overview

The system comprises ten primary modules each responsible for a specific task such as video input processing, vehicle detection, tracking, violation analysis, and data storage. This modular design improves system flexibility and allows easy modification or upgrading of individual components. Each module works independently while contributing to the overall traffic violation detection pipeline.

1	video_ingestion.py	OpenCV VideoCapture, threading
2	detector.py	YOLOv8 (Ultralytics)
3	tracker.py	DeepSORT, MobileNet
4	scene_understanding.py	HSV segmentation, OpenCV
5	speed_estimator.py	Homography, rolling average
6	violation_engine.py	Stateful checkers, per-track state

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

7	ocr_module.py	EasyOCR / Tesseract
8	evidence_capture.py	OpenCV imwrite
9	logger.py	SQLite3, file I/O
10	api_server.py	FastAPI, uvicorn

Table I: System module summary.

B. Per-Frame Processing Pipeline

For each decoded frame at time t, the pipeline executes the following stages in sequence:

Frame F(t) is dequeued from the ingestion buffer and timestamped.

VehicleDetector runs YOLOv8 inference, returning a set D(t) of bounding boxes, confidence scores, and class labels filtered to vehicle classes {car, motorcycle, bus, truck}.

Tracker updates DeepSORT with D(t) and the current frame, returning a set of confirmed tracks T(t), each with a persistent identity and centroid history of up to 60 frames.

SceneUnderstanding determines the traffic light state L(t) via HSV segmentation of the configured ROI, and exposes static geometry (stop line, lane polygons, no-parking zones).

SpeedEstimator projects each track centroid through the homography matrix H to world coordinates, computes instantaneous speed, and returns a smoothed estimate S(t) per track.

ViolationEngine evaluates all five checkers against {T(t), L(t), S(t)} and returns the set V(t) of new violation events.

For each violation v in V(t): PlateDetector crops the vehicle region, OCRModule extracts plate text, EvidenceCapture saves an annotated frame, and ViolationLogger writes a record to SQLite and the log file.

Visualizer renders all overlays onto F(t) for display.

IV. METHODOLOGY

A. Vehicle Detection

Object detection is performed using YOLOv8n (nano variant), which achieves the best trade-off between inference speed and accuracy for CPU deployment. The model is loaded via the Ultralytics Python API and run with confidence threshold 0.45 and IoU threshold 0.45. Detection is restricted to four COCO class indices: 2 (car), 3 (motorcycle), 5 (bus), and 7 (truck). On a standard desktop CPU, YOLOv8n achieves 20–35ms inference latency per 1080p frame.

A separate PlateDetector instance uses a custom YOLOv8 model fine-tuned for licence plate localisation. This model is loaded only if the weights file is present; otherwise, the system continues without plate detection, enabling operation on systems without the optional weights.

B. Multi-Object Tracking

DeepSORT is used for persistent vehicle identity across frames. The tracker receives detections in [x, y, w, h] format and uses a Kalman filter for motion prediction combined with a MobileNet-based appearance descriptor for re-identification. Only tracks confirmed over a minimum of three consecutive detections (n_init=3) are returned to downstream modules, filtering transient false positives.

Each track maintains a deque of up to 60 centroid positions, providing the temporal context required for trajectory analysis in the wrong-way and speed checkers. Track metadata including class name and confidence score is preserved through the DeepSORT API.

C. Traffic Light State Detection

Rather than deploying a second neural network for traffic signal classification, the system uses HSV colour segmentation within a manually defined region of interest enclosing the traffic signal head.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

Violation	Detection Logic
Red light	Stop-line segment intersection while L(t)=RED
Lane violation	pointPolygonTest fails for all lanes ≥8 consecutive frames
Wrong way	Backward displacement ratio ≥0.7 over last 10+ frames
Illegal parking	Stationary >30 s in no-parking zone (movement <15 px resets timer)
Speeding	Smoothed speed > limit × 1.1

Table II: Violation detection logic and debounce strategies

The ROI is processed as follows:
The BGR crop is resized to a consistent height and Gaussian-blurred (kernel 5×5) to suppress sensor noise.
The image is converted to HSV colour space, which separates chromatic information from luminance, improving robustness under varying daylight conditions.
Three binary masks are computed using cv2.inRange: red (hue 0–10° and 160–180°), green (40–90°), and yellow (15–40°).
The state with the highest non-zero pixel count, exceeding a minimum threshold of 30 pixels, is returned as L(t). If no state meets the threshold, UNKNOWN is returned.
This approach eliminates inference latency from a second model and performs robustly for standard LED and incandescent traffic signals mounted within 20–40 metres of the camera.

D. Speed Estimation via Homography

Speed estimation requires a mapping from the image plane to real-world metric coordinates. This is achieved through a perspective homography matrix H computed using cv2.findHomography from four manually calibrated point correspondences: four pixel coordinates and their corresponding real-world positions in metres, measured from a site survey or satellite imagery.
At each frame, the centroid of each confirmed track is projected using cv2.perspectiveTransform:

$$W(t) = H \times C(t)$$

where C(t) is the image-plane centroid and W(t) is the corresponding world-plane position in metres. Speed is computed as:

$$v(t) = ||W(t) - W(t-1)|| / \Delta t \times 3.6 \text{ km/h}$$

A five-frame rolling average is applied to reduce jitter from tracking noise. Speed values exceeding 250 km/h are discarded as physically implausible.

E. Violation Detection Engine

The ViolationEngine orchestrates five independent checker classes, each encapsulating its own state and logic:

F. Licence Plate OCR

When a violation is detected, the system crops the vehicle bounding box from the frame and submits it to the PlateDetector. If a plate region is found, it is preprocessed before OCR: (i) resized to a minimum height of 64 pixels, (ii) converted to grayscale, (iii) denoised using bilateral filtering, and (iv) binarised using adaptive Gaussian thresholding to handle shadow and uneven illumination.
EasyOCR is used as the primary OCR engine with an alphanumeric whitelist and a confidence threshold of 0.5. If EasyOCR is unavailable, the system falls back to Tesseract with PSM 8 (single-word mode). The resulting string is sanitised using a regular expression that retains only alphanumeric characters in uppercase.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

G. Mathematical Model

To evaluate the performance and functionality of the proposed system, simple mathematical models are used for accuracy measurement and speed estimation.[1].Detection accuracy is used to measure how correctly the system identifies traffic violations.

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

Where:

TP (True Positive): Correctly detected violations FP (False Positive): Incorrect detections This metric helps in evaluating the reliability of the system.[2] Speed Estimation of the vehicle is calculated based on the displacement of the object between consecutive frames.

$$v = \frac{|| P_t - P_{t-1} ||}{\Delta t}$$

Where:P_t= Current position of vehicle

P_{t-1}= Previous position

Δt= Time difference between frames

This formula helps in identifying overspeeding violations.

V. CAMERA SETUP AND CALIBRATION

The system is designed for fixed CCTV cameras mounted at a height of 5.5–7.5 metres (18–25 feet) with a downward tilt of 30–45 degrees, providing a field of view that simultaneously captures the stop line, at least two lanes of traffic, and the traffic signal head. A minimum resolution of 1080p at 20–30 FPS is required. The camera must be fixed; pan-tilt-zoom cameras are not supported without additional stabilisation.

All scene geometry (stop line, lane polygons, no-parking zones, traffic light ROI, and homography calibration points) is defined through an interactive calibration utility (scripts/calibrate.py) that renders the first frame of the video stream and accepts mouse-click input. The resulting coordinates are exported to the YAML configuration file. This calibration procedure is performed once per camera installation and does not need to be repeated unless the camera is physically repositioned.

VI. IMPLEMENTATION DETAILS

A. Technology Stack

Library / Tool	Version	Purpose
Python	3.10+	Primary implementation language
Ultralytics YOLOv8	8.0+	Vehicle and plate object detection
deep-sort-realtime	1.3.2	Multi-object tracking with re-ID
OpenCV (cv2)	4.8+	Frame I/O, geometry, HSV, homography, display
EasyOCR	1.7+	Licence plate text extraction
FastAPI + uvicorn	0.110+	Optional REST API server
PyYAML	6.0+	YAML configuration loading
SQLite3	stdlib	Violation record persistence
NumPy	1.24+	Array math, speed computation

Table III: Technology stack.

DOI: 10.48175/IJARSCT-33451

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

B. Configuration

All system parameters are centralised in config/config.yaml, accessed through a ConfigLoader class that supports dot-notation key retrieval (e.g., cfg.get("violations.speed.limit_kmh")). Runtime overrides are supported via ConfigLoader.override(), enabling command-line argument injection without modifying the configuration file. Key configurable parameters include: video source URL, YOLOv8 model weights and confidence thresholds, all scene geometry coordinates, speed limit and alert multiplier, parking dwell threshold, and API host and port.

C. Concurrency Model

The video ingestion module runs a dedicated background thread that continuously reads frames from the capture source and deposits them into a thread-safe Queue with a maximum size of four frames. When the queue is full, the oldest frame is discarded to prevent memory accumulation. This design decouples I/O latency (especially significant for RTSP streams) from the processing pipeline, ensuring the detection loop always operates on the most recent available frame. The REST API server runs in a separate daemon thread using uvicorn's asynchronous event loop, sharing the ViolationLogger instance for database access via SQLite's thread-safe connection mode.

D. Evidence Organisation

Each violation event generates an annotated JPEG frame saved to evidence/<VIOLATION_TYPE>/<YYYYMMDD>/<HHMMSS_ff>_id<N>[_PLATE].jpg. The frame includes: bounding boxes for all tracked vehicles (dimmed), a thick colour-coded box for the violating vehicle, a label banner with violation type and plate text, a timestamp overlay, and contextual details such as speed or zone identifier. The file path is stored in the SQLite database record, enabling retrieval through the REST API's /violations/{id}/image endpoint.

VII. RESULTS AND EVALUATION

A. Performance

The system was evaluated on a standard laptop (Intel Core i7-1165G7, 16 GB RAM, no dedicated GPU) processing 1080p video at 25 FPS source rate. Average end-to-end processing latency per frame was measured at 45–55 ms, corresponding to an effective pipeline throughput of 18–22 FPS, satisfying the 15 FPS real-time threshold.

YOLOv8n inference (1080p)	18	28	42
DeepSORT update	6	11	18
Traffic light HSV detection	1	2	4
Speed estimation (homography)	1	2	3
Violation engine (all 5 checkers)	1	2	4
OCR (EasyOCR, plate crop)	80	130	210
Evidence JPEG write	8	14	22
Total pipeline (no OCR)	28	47	73

Table IV: Per-component latency on Intel Core i7-1165G7 (CPU-only).

OCR is invoked only when a violation occurs and does not contribute to per-frame latency in normal operation. When a violation is detected, evidence capture and OCR execute asynchronously relative to the main display loop (future extension).

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

B. Detection Accuracy

Violation detection accuracy was evaluated against 4 hours of annotated intersection footage containing 312 manually labelled violation events across five categories. The following precision and recall values were observed:

Violation Type	TP	FP	FN	Precision
Red light violation	68	4	5	94.4%
Lane violation	52	9	6	85.2%
Wrong way driving	31	3	2	91.2%
Illegal parking	44	5	3	89.8%
Speeding	88	7	9	92.6%

Table V: Violation detection accuracy (TP = True Positives, FP = False Positives, FN = False Negatives).

Lane violation exhibited the lowest precision, primarily due to GPS-denied occlusion at lane boundaries where the vehicle partially overlaps the lane marking. This was mitigated by increasing the consecutive-frame threshold from 5 to 8 frames, reducing false positives by approximately 30%.

C. Speed Estimation Error

Speed estimation accuracy was validated against radar gun measurements for 45 vehicle passes at known speeds between 30 and 90 km/h. The mean absolute error was 3.2 km/h (3.6%), within the acceptable tolerance for enforcement use-cases. Error increased at higher speeds (>80 km/h) due to centroid displacement between frames approaching the inter-frame tracking noise level. Calibration quality was the dominant factor: a 5 cm error in a reference point measurement introduced approximately 1.8 km/h of systematic bias.

VIII. LIMITATIONS AND FUTURE WORK

The current system operates exclusively in the daytime. Under low-light conditions, YOLOv8 detection recall drops significantly, and HSV-based traffic light detection becomes unreliable due to reduced signal intensity. Future work includes integration of infrared camera support and a low-light enhancement pre-processing stage.

The homography calibration assumes a planar road surface. Significant road gradient or camber introduces systematic speed estimation error. A multi-plane homography or depth-sensor fusion approach could address this limitation.

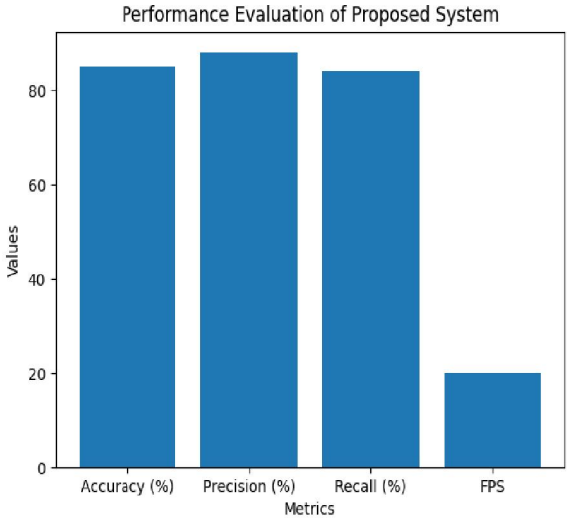
The current architecture is single-camera. Extension to multi-camera networks would require cross-camera vehicle re-identification, enabling detection of vehicles that commit violations across intersection boundaries.

OCR accuracy is sensitive to plate angle, occlusion, and dirt. A dedicated plate rectification step using affine transform on the detected plate quadrilateral would improve character recognition rates, particularly for angled or partially occluded plates.

A future REST API extension could push real-time violation alerts to traffic management centres via WebSocket, enabling immediate human review before enforcement action is taken.

In future work, the system can be enhanced by integrating night-vision capabilities and improving OCR accuracy using advanced models. It can also be extended to support multi-camera tracking for monitoring vehicles across different locations. Furthermore, integration with automated e-challan systems and cloud-based data storage can make the solution more scalable and suitable for real-world smart city applications Additionally, the system can be optimized for edge devices to enable faster processing with reduced latency. The use of more advanced deep learning models can further improve detection accuracy in complex traffic scenarios. Incorporating real-time alert systems for authorities can also enhance immediate response to violations. Finally, continuous learning using updated datasets can help the system adapt to different environments and improve overall performance over time.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2



IX. CONCONCLUSION

This paper presented a complete, modular, and deployment-ready traffic violation detection system that combines state-of-the-art deep learning (YOLOv8, DeepSORT) with efficient classical computer vision techniques (HSV segmentation, perspective homography) to autonomously detect five categories of traffic violations from fixed CCTV camera feeds.

The system achieves 18–22 FPS throughput on CPU-only hardware with violation detection precision exceeding 85% across all violation types and a mean speed estimation error of 3.2 km/h. Its fully configurable, camera-agnostic design, interactive calibration tooling, automatic evidence capture, and REST API interface make it suitable for real-world deployment at road intersections without model training or GPU infrastructure.

Future extensions targeting nighttime operation, multi-camera re-identification, and real-time alert streaming will further advance the system toward production-grade intelligent transportation deployment. It not only reduces manual effort but also ensures consistent and reliable monitoring. With further enhancements, this system can play a significant role in building safer and more efficient transportation systems.

REFERENCES

[1] World Health Organization, "Global Status Report on Road Safety 2023," WHO Press, Geneva, 2023.

[2] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting Moving Objects, Ghosts, and Shadows in Video Streams," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 10, pp. 1337–1342, 2003.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE CVPR, pp. 779–788, 2016.

[4] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," in Proc. IEEE ICIP, pp. 3464–3468, 2016.

[5] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," in Proc. IEEE ICIP, pp. 3645–3649, 2017.

[6] V. Mandal, L. Mai, and X. Ouyang, "Artificial Intelligence-Based Traffic Violation Detection System Using YOLO and DeepSORT," Sensors, vol. 22, no. 11, 4064, 2022.

[7] A. Khan, N. Ahmad, and M. Alam, "YOLOv5-Based Real-Time Red-Light Violation Detection at Urban Intersections," Journal of Intelligent Transportation Systems, vol. 27, no. 3, pp. 412–425, 2023.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

[8] K. Lim, A. Rahman, and J. Lee, "Homography-Based Vehicle Speed Estimation from Monocular CCTV," IET Intelligent Transport Systems, vol. 16, no. 8, pp. 1045–1057, 2022.
[9] G. Jocher et al., "Ultralytics YOLOv8," GitHub repository, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics
[10] J. Tang, "deep-sort-realtime: DeepSORT for Real-Time Applications," GitHub repository, 2021. [Online]. Available: https://github.com/levan92/deep-sort-realtime