Journal of Computer Science and Artificial Intelligence
ISSN: 3078-8242 (Print), ISSN: 3078-2309 (Online) | Vol. 9, No. 2, 2026

An Intelligent Traffic Management System Based on Visual Recognition and Cloud Inference

Chenyu Yue *, Qunxiang Wei

School of Electrical and Information Engineering, Ningxia Institute of Technology, ShiZuiShan, 753000, China

* **Corresponding author**: (Email: 13573214438@163.com)

Abstract: Urban intersections, campus perimeters, and industrial parks require continuous monitoring of dense, mixed traffic flows, but manual patrols and full-frame cloud analysis are constrained by coverage, latency, bandwidth, and cost. This paper introduces an event-driven edge-cloud collaborative traffic management system that combines YOLOv8 detection, DeepSORT tracking, pluggable rules, multi-frame confirmation, event-triggered forensics, and tiered cloud review. The edge-side pipeline handles red-light running, wrong-way driving, helmet non-compliance, and illegal parking, and uploads selected keyframes and structured context rather than all video frames. An eight-state event lifecycle, an 18-field evidence record, browser-based monitoring, and 3D visualization are used to support traceability and manual review. In reported 1080p tests on RTX 4060 hardware, TensorRT FP16 reduced single-frame detection latency from about 45 ms to about 15 ms, with the system reported to run at 30 fps. The prototype demonstrates how event-level edge-cloud processing can reduce unnecessary video uploads to the cloud while preserving evidence for review.

Keywords: Intelligent Traffic Management, Edge-Cloud Collaboration, Visual Recognition, Vehicle Tracking, Event-Driven Evidence Capture.

1. Introduction

Urban intersections, school zones, and industrial-park roads bring pedestrians, bicycles, motorcycles, and motor vehicles into dense, changing scenes. These locations often have limited enforcement coverage, while isolated camera snapshots may miss the trajectory and temporal context needed to confirm a violation or reconstruct how it occurred. Manual patrol is labor intensive, and transmitting every video frame to a remote cloud can increase latency, bandwidth use, computing demand, and operating cost. A practical system must pair local response with reviewable evidence.

Recent traffic-monitoring research has examined intelligent infrastructure that combines deep learning with edge computing [1], real-time vehicle tracking in large roadside sensor networks [2], and improved YOLOv8 methods for vehicle detection [3]. YOLOv8 has also been studied for intelligent transportation monitoring [4]. These studies support placing inference near cameras rather than relying solely on the cloud. However, object detection alone identifies what appears in a frame; it does not establish whether movement violated a local rule or preserve an auditable event history.

Traffic-rule enforcement adds temporal and spatial requirements. Computer-vision methods have been used for traffic-violation detection [5], and traffic-light image recognition has been studied for red-light violation detection [6]. Recent work has also proposed improved DeepSORT for multi-target vehicle tracking [7]. In intelligent transportation monitoring, reported YOLOv8 speed and detection quality depend on model size, scene conditions, and image quality [4]. Together, this literature supports a pipeline that accumulates detections along stable tracks, checks behavior over multiple frames, and creates an event only after evidence persists.

Edge-assisted video analytics research shows that resolution, server selection, bandwidth, and compute allocation affect analytical quality and deadline success [8]. Transportation digital-twin studies describe methods for linking operational data with virtual-scene visualization [9,10]. Yet evidence retention, event lifecycles, model review, and human override are often treated separately. System integration must therefore address not only detector inference but also when video is retained, what context is transmitted, how a judgment is recorded, and how a reviewer can reconstruct the decision.

This paper presents an event-driven edge-cloud traffic-management prototype combining YOLOv8n detection, DeepSORT tracking, pluggable violation rules, temporal confirmation, ring-buffer evidence capture, and cloud review. It documents a three-layer architecture, an 18-field evidence record, an event lifecycle, and browser-based visualization. Evaluation uses camera, video-file, and CARLA inputs for four violation categories. In the reported 1080p RTX 4060 test, TensorRT FP16 reduced per-frame inference latency from about 45 ms to 15 ms. The system was also reported to meet a 30-fps real-time requirement. Because no shared labeled test set or measured precision, recall, or mean average precision is provided, detection accuracy and cross-camera generalization are treated as unresolved validation needs rather than established results.

2. System Architecture

The system is designed for locations with limited traffic-monitoring coverage. Its inputs include USB cameras, network cameras, video files, and CARLA simulation streams. It classifies four violation categories: red-light running, wrong-way travel, helmet non-use, and illegal parking. Outputs include event video, keyframes, target trajectories, plate information, rule results, and cloud-review records. Operators can inspect, filter, export, and confirm events through a browser.

The architecture has three layers. Edge perception ingests video, detects and tracks road users, evaluates rules, and

buffers evidence. Cloud review receives three to five representative keyframes and structured event context after a trigger, applies a two-model review workflow, and persists the decision trace. The presentation layer broadcasts events, summarizes system statistics, displays a three-dimensional road scene, and supports human review. Table 1 summarizes the layer responsibilities.

Table 1. System layers and responsibilities.

Layer	Core components	Primary role
Edge perception	YOLOv8; DeepSORT; rule plugins; ring buffer; OCR	Detection, tracking, rule evaluation, event-triggered capture, and structured context generation
Cloud review	Claude Opus 5; GPT-6 Astra; ai_trace v3	Secondary model review, consistency checks, decision explanation, and evidence persistence
Presentation	Flask-SocketIO; WebSocket; Vue 3; Element Plus; ECharts; Three.js	Live event updates, statistics, 3D roadway display, and human-review controls

2.1. Edge Detection and Plate Processing

The edge detector is based on the lightweight YOLOv8n network and is adapted through traffic-scene fine-tuning from COCO-pretrained weights. The detection set covers motor vehicles, non-motor vehicles, pedestrians, traffic lights, lane markings, and selected violation attributes such as helmet use, lane compliance, and signal state. Detection outputs include class, confidence, and bounding-box coordinates; non-maximum suppression removes highly overlapping candidate boxes. Figure 1 records the model-review interface.

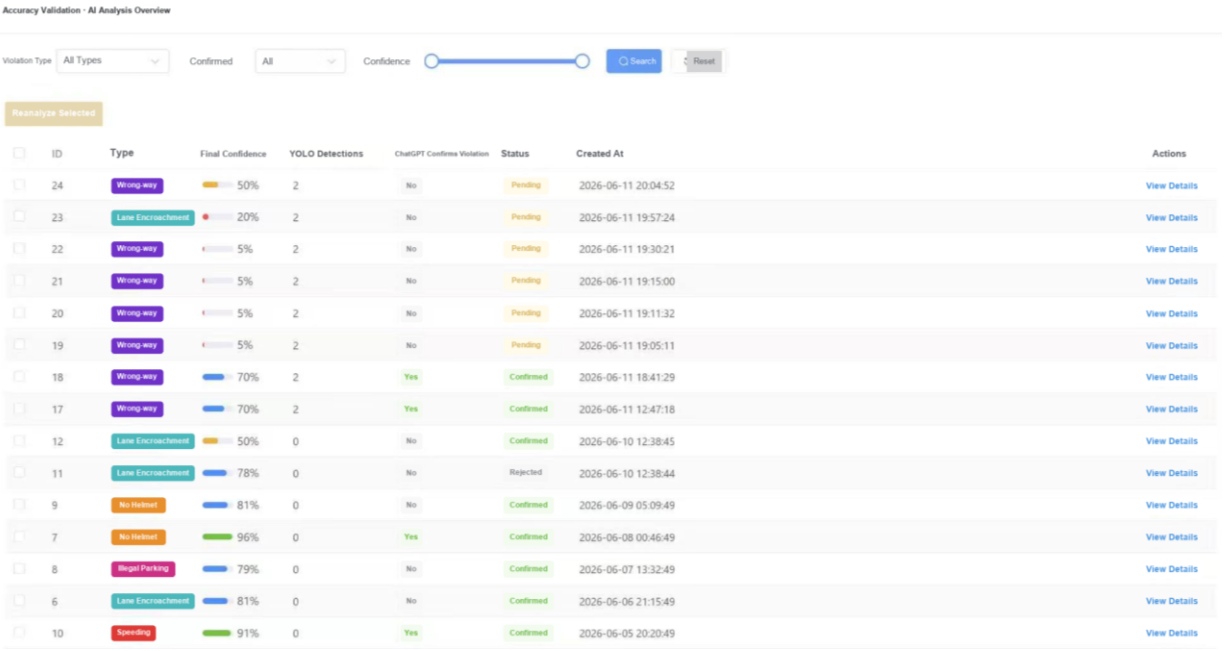

Figure 1. Traffic-violation review interface.

For plate processing, the detected plate region is converted to grayscale, passed through edge extraction, and refined by morphological closing before OCR. PaddleOCR reads the plate characters. The plate crop is also converted to HSV color space, where hue, saturation, and brightness are used to classify plate color. Figures 2-5 preserve the intermediate image sequence and the processed result.



Figure 2. Original plate crop.



Figure 3. Grayscale preprocessing.



Figure 4. Edge extraction.



Figure 5. Morphological closing.

For 1080p video, the YOLOv8n model is exported from PyTorch to ONNX and compiled as a TensorRT FP16 engine for an NVIDIA GeForce RTX 4060 GPU. The reported optimizations include layer fusion, memory reuse, kernel tuning, and asynchronous execution. Under the stated input and hardware conditions, single-frame inference latency decreased from approximately 45 ms to 15 ms, about a threefold speed increase. The reported full-system target is 30 fps; the reported detector timing and system rate are distinct measures.

2.2. Multi-object Tracking and Behavior Analysis

Frame-level detection is extended with DeepSORT to maintain a continuous identity and trajectory for each road user. A Kalman filter predicts motion, a convolutional network extracts appearance features, and cascade matching with the Hungarian algorithm associates new detections with existing tracks. Mahalanobis distance constrains motion consistency, while cosine distance compares appearance. The tracker records a unique track identifier, position, displacement, speed, direction, and recent trajectory history. Figure 6 presents the tracking pipeline.

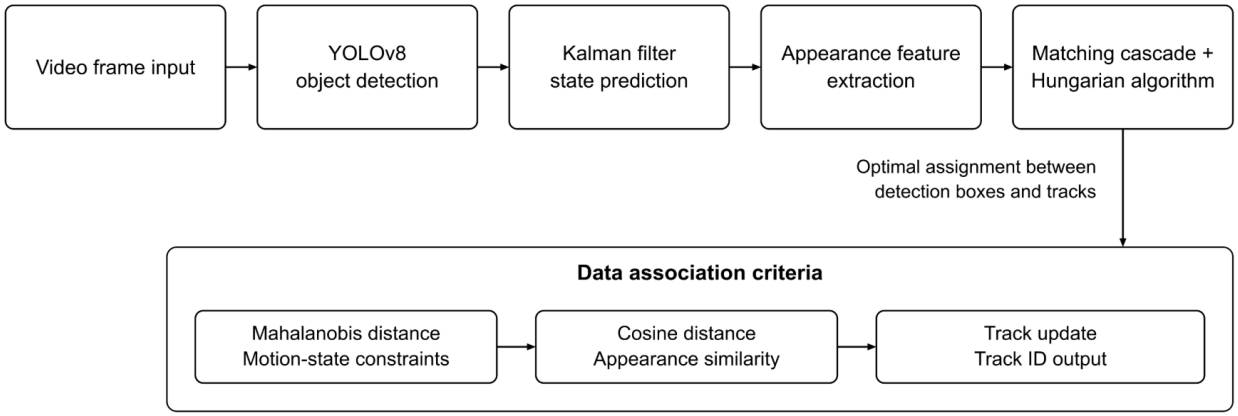

Figure 6. DeepSORT multi-object tracking and data-association pipeline.

The behavior engine derives average and maximum speed, path curvature, coordinate variability, and direction consistency from the track history. Wrong-way travel is assessed from the angle between target motion and the configured road direction. Illegal parking is evaluated from position change and dwell time. Red-light running is assessed jointly from signal state, target position relative to the stop line, and event ordering. Helmet use is determined from helmet detections within the two-wheeler region. These are rule-based tests over a tracked sequence, not independent single-frame labels.

2.3. Multi-frame Confirmation and Event Lifecycle

To reduce false triggers from occlusion, blur, lighting changes, and unstable boxes, the system checks consistency over a sliding time window for each track and violation category. A binary indicator records whether the rule is provisionally satisfied in a given frame. The weighted confirmation score aggregates recent indicators using configurable temporal weights; an event is confirmed only when the score exceeds a configured threshold and the condition persists for the required duration. The manuscript does not specify numeric values for the window, weights, or threshold, so they are not assigned here.

$$b_t^{(m)} \in \{0,1\} \tag{1}$$

$$C_t^{(m)} = \frac{\sum_{i=0}^{K-1} w_i b_{t-i}^{(m)}}{\sum_{i=0}^{K-1} w_i} \tag{2}$$

A confirmed candidate receives a unique event identifier and enters a finite-state lifecycle: DETECTING, TRIGGERED, RECORDING, PENDING_AI, and AI_REVIEWED, followed by CONFIRMED, REJECTED, or MANUAL_REVIEW. A human reviewer resolves the manual-review state as confirmed or rejected. Cooldown and event-merge rules suppress duplicate records for the same track and violation within a short interval. The event buffer retains keyframes from before, during, and after the confirmed behavior.

Each evidence record contains 18 fields: camera identifier, timestamp, location, vehicle identifier and type, plate text and color, violation type, rule confidence, trajectory, signal state, stop-line relation, keyframes, event video, AI trace, final judgment, reviewer, and review time. The system supports dictionary, JSON, and summary outputs for database storage, API transfer, and presentation. Figure 7 shows the event-record interface.

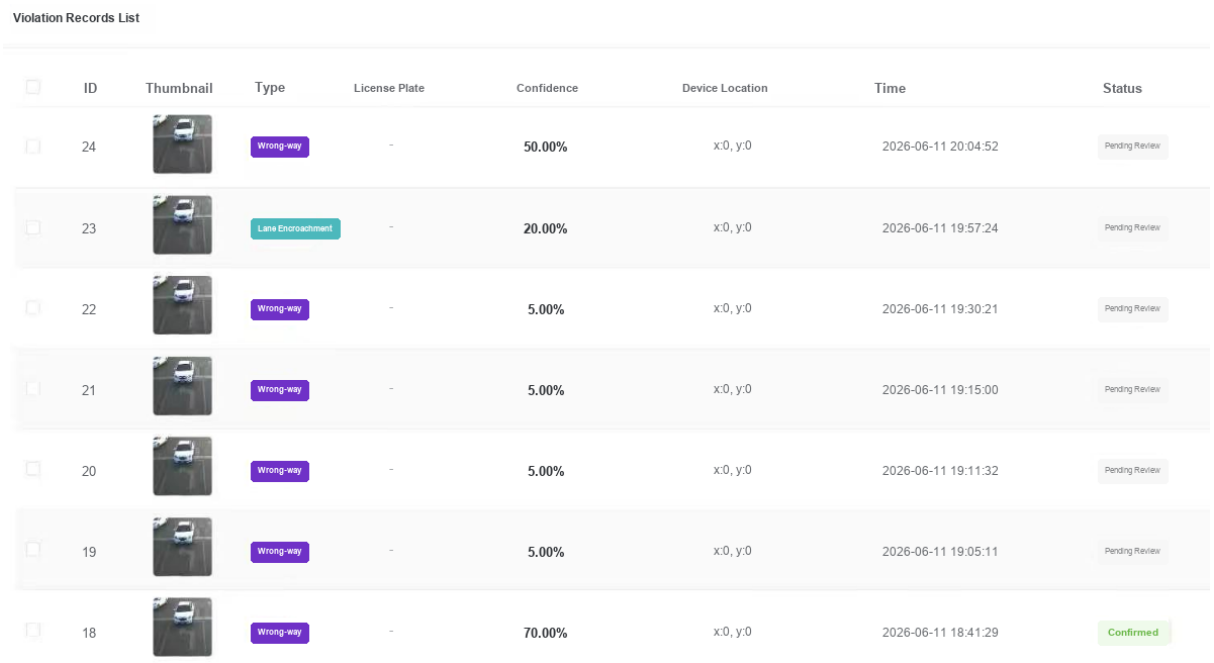

Figure 7. Traffic-violation event record and review list.

2.4. Cloud Review and Visualization

The design routes triggered evidence through a secondary review stage using Claude Opus 5 and GPT-6 Astra, with an AI trace retained alongside the final judgment. This is an architectural description; no comparative accuracy or disagreement-rate measurement is reported. The rule-plugin interface defines initialization, shutdown, analysis, scenario name, and detectable categories, allowing additional traffic, urban-management, or public-security scenarios to be added without rewriting the event engine.

The browser interface includes event center, event details, AI trace, violation list, and system settings. Flask-SocketIO polls system status every 1.5 seconds and sends updates through WebSocket. Three.js places trajectories in a three-dimensional road scene, while CSS2DRenderer overlays plate cards and violation labels; a red directional marker fades after 15 seconds. CARLA 0.9.16 maps from Town01 to Town07 provide simulated road layouts and traffic-density conditions for closed-loop testing.

3. Implementation and Evaluation

A central RealtimeViolationDetector coordinates camera, video-file, and CARLA sources. Each frame passes through detection, trajectory update, behavior analysis, rule plugins, multi-frame validation, and event handling. The evaluation checks four violation categories, the continuity of event records, and whether event video, keyframes, AI traces, and human-review outcomes appear in the Web interface. Tests compare PyTorch inference with TensorRT FP16 using 1080p inputs on an RTX 4060 laptop.

The reported detector latency falls from about 45 ms to 15 ms per frame, a reduction of approximately 66.7%. The system reports 30-fps operation. The evaluation also demonstrates that short rule activations can be organized into reviewable event records and that event-triggered transmission separates frequent local perception from less frequent cloud calls. No unified labeled test set is described, and no measured precision, recall, or mAP values are supplied. The reported evidence therefore supports the implementation and latency claims, but not a quantitative claim about detection accuracy. Table 2 summarizes the stated test conditions and evidence limits.

Table 2. Reported implementation tests and quantitative evidence.

Item	Reported condition or result	Evidence status
Input resolution	1080p video	Test condition stated
Hardware	NVIDIA GeForce RTX 4060 laptop GPU	Test condition stated
Detector inference	About 45 ms to about 15 ms per frame after TensorRT FP16	Reported timing
Latency reduction	About 66.7%; approximately threefold speed increase	Derived from reported timing
System rate	30 fps	Reported; full end-to-end protocol not detailed
Precision / Recall / mAP	No numerical result reported	Requires a shared labeled evaluation set

4. Limitations

Three constraints remain. First, stop lines, lane directions, and no-parking zones require scene configuration, so camera relocation requires recalibration. Second, nighttime scenes, rain, fog, and severe occlusion may reduce detection and tracking stability. Third, cloud-model latency and output format depend on external services. The proposed next steps are low-visibility data augmentation, adaptive thresholds, retry handling, and an offline SQLite queue, followed by field measurement of edge latency, cloud-review time, false positives, and missed events. These are planned validation

items, not completed results.

5. Conclusion

An event-driven edge-cloud traffic-management prototype was designed for camera-based monitoring at intersections, school zones, and other high-density locations. YOLOv8 detection, DeepSORT trajectories, rule plugins, multi-frame confirmation, and ring-buffer evidence capture form the local processing path; selected keyframes and structured context are routed to a secondary cloud review stage. An eight-state lifecycle, an 18-field evidence chain, and browser-based three-dimensional visualization support event traceability and human review.

The reported implementation reduces detector latency from approximately 45 ms to 15 ms with TensorRT FP16 on an RTX 4060 under the stated 1080p test conditions and reports 30-fps system operation. Because dataset-wide precision, recall, and mAP were not measured, accuracy and cross-camera generalization remain open validation questions. The system demonstrates a practical event-level alternative to transmitting every frame, while its field readiness depends on calibrated scenes, adverse-weather robustness, and measured cloud-review performance.

References

[1] Villa, J., García, F., Jover, R., et al. (2024). Intelligent infrastructure for traffic monitoring based on deep learning and edge computing. Journal of Advanced Transportation, 2024, 3679014. https://doi.org/10.1155/2024/3679014

[2] Liu, Y., Gao, B., Lin, P., et al. (2025). A framework for real-time vehicle tracking in large-scale roadside sensor networks. Green Energy and Intelligent Transportation, 4(6), 100362. https://doi.org/10.1016/j.geits.2025.100362

[3] Liu, X., Wang, Y., Yu, D., et al. (2024). YOLOv8-FDD: A real-time vehicle detection method based on improved YOLOv8. IEEE Access, 12, 136280–136296. https://doi.org/10.1109/ACCESS.2024.3453298

[4] Bakirci, M. (2024). Utilizing YOLOv8 for enhanced traffic monitoring in intelligent transportation systems (ITS) applications. Digital Signal Processing, 152, 104594. https://doi.org/10.1016/j.dsp.2024.104594

[5] Ong, C. S., Tee, C., & Goh, M. K. O. (2024). Traffic violation detection using computer vision techniques. JOIV: International Journal on Informatics Visualization, 8(3-2), 1515–1527. https://doi.org/10.62527/joiv.8.3-2.2941

[6] Tomas, J. P. Q., Arcaya, M. A., Perez, M. F., et al. (2024). Traffic light image recognition for detecting violations using YOLOv5 object detection and Faster R-CNN. In Proceedings of the 2024 7th International Conference on Machine Learning and Machine Intelligence (MLMI) (pp. 192–197). ACM. https://doi.org/10.1145/3696271.3696302

[7] Guo, D., Li, Z., & Shuai, H. (2024). Multi-target vehicle tracking algorithm based on improved DeepSORT. Sensors, 24(21), 7014. https://doi.org/10.3390/s24217014

[8] Li, Y., Li, Y., Chen, Y., et al. (2024). Online resolution adaptation and resource allocation for edge-assisted video analytics. Computer Networks, 244, 110342. https://doi.org/10.1016/j.comnet.2024.110342

[9] Ge, C., & Qin, S. (2024). Digital twin intelligent transportation system (DT-ITS)—A systematic review. IET Intelligent Transport Systems, 18(12), 2325–2358. https://doi.org/10.1049/itr2.12539

[10] Aloupogianni, E., Doctor, F., Karyotis, C., et al. (2024). A digital twin architecture for intelligent traffic management based on artificial intelligence. In Intelligent Environments 2024: Combined Proceedings of Workshops and Demos & Videos Session (pp. 143–151). IOS Press. https://doi.org/10.3233/AISE240026