

Automobile Accident Detection and Automated Emergency Alert System Using YOLOv8, ByteTrack, and Deep Learning

Eshwaraju Ajay¹ and Dr. Bitla Prabhakar²

¹ PG Scholar, Department of ECE,

² Professor, Department of ECE

¹² JNTUH University College of Engineering Jagtial, Jagtial, TS, India

¹eshwarajuajaychary@gmail.com, ²bprabhakar2008@gmail.com

Abstract: This paper presents a real-time automobile accident detection and emergency-alert framework for CCTV-based road surveillance. The system uses YOLOv8 to detect vehicles and produce bounding boxes, class labels, and confidence scores. ByteTrack maintains vehicle identities across frames, allowing the computation of center-point trajectories, displacement, velocity, acceleration, relative motion, bounding-box overlap, and time-to-collision. A multi-parameter accident decision module combines proximity, intersection-over-union, relative velocity, sudden motion change, and multi-frame persistence to reduce single-frame false alarms. ConvLSTM-based temporal risk analysis can be incorporated to learn accident-related motion evolution. After a collision is confirmed, the system generates a structured alert containing the timestamp, location, vehicles involved, confidence, severity, and required emergency services. The supplied implementation provides a Flask-based real-time dashboard, local network access, evidence-file management, automatic refresh, incident statistics, and formatted accident alerts. Experimental screenshots demonstrate successful generation of high-severity alerts with 95% confidence, dashboard logging of 14 incidents and 68 evidence recordings, and persistent web-based monitoring. The framework converts passive CCTV infrastructure into an active road-safety system capable of accelerating emergency notification and supporting intelligent transportation and smart-city applications.

Keywords: accident detection, YOLOv8, ByteTrack, ConvLSTM, vehicle tracking, time-to-collision, emergency alert, CCTV surveillance, intelligent transportation system.

I. INTRODUCTION

Road accidents remain a major public-safety concern because the consequences of a collision are frequently intensified by delayed detection and delayed arrival of medical or police assistance. Conventional reporting depends on witnesses, patrol personnel, telephone calls, or operators who manually observe many CCTV feeds. Such approaches are vulnerable to fatigue, limited camera coverage, low visibility, and reporting delay. An automated vision system can continuously inspect traffic video and provide immediate, structured evidence when an accident-like event occurs.

Modern object detectors make camera-based traffic analysis feasible in real time. Ultralytics YOLOv8 provides one-stage object localization and classification and is suitable for video-stream inference [1]. Nevertheless, vehicle detection alone does not establish that an accident occurred. A robust system must maintain object identity across frames, estimate motion, evaluate interactions between vehicles, confirm the abnormal event temporally, and communicate the result to emergency responders.

The uploaded project develops a complete pipeline consisting of CCTV acquisition, frame preprocessing, YOLOv8 vehicle detection, ByteTrack-based tracking, motion and collision analysis, severity estimation, evidence storage, and



automated alert presentation through a web dashboard. This paper restructures that implementation into a formal research-paper flow and analyzes the attached dashboard and alert outputs.

II. RELATED WORK

Real-time road-event analysis has increasingly moved from handcrafted background subtraction and optical flow toward deep detectors and multi-object tracking. YOLO-based models are attractive because they combine localization and classification in a single inference pass. The official Ultralytics framework supports image, video, live-stream prediction, and tracking workflows, making it practical for CCTV deployments [1], [2].

ByteTrack improves tracking continuity by associating not only high-confidence detections but also lower-confidence boxes that may correspond to partially occluded objects [3]. This behavior is useful in crowded traffic scenes, where missed detections and identity switches can corrupt speed and trajectory estimation. Recent vehicle-tracking research has combined improved YOLOv8 variants with ByteTrack to address small targets, occlusion, and complex road backgrounds [4], [5].

Recent accident-detection work has specifically integrated YOLOv8 and ByteTrack for CCTV-based accident monitoring [6]. Other studies have used YOLOv8 for vehicle speed estimation and traffic management, showing that detection and tracking outputs can support higher-level motion measurements [7]. For temporal risk analysis, ConvLSTM is relevant because it performs convolutional state transitions and captures spatio-temporal evolution across video frames [8].

Current intelligent-transportation reviews emphasize environmental variability, night and weather conditions, occlusion, camera geometry, real-time constraints, limited accident datasets, and false-alarm reduction as major open problems [9]. These findings motivate the proposed combination of detection, tracking, interpretable collision equations, multi-frame confirmation, and an operational alert dashboard.

III. METHODOLOGY

The proposed system is designed as a continuous, event-driven video analytics pipeline rather than as a single-frame accident classifier. Its central principle is that an accident must be inferred from the joint evidence of object presence, object identity, motion evolution, geometric interaction, and temporal persistence. The detector therefore performs only the first stage. The final decision is produced after vehicle tracking, trajectory analysis, collision-risk computation, multi-frame verification, severity estimation, evidence management, and alert delivery. This layered methodology follows the uploaded project document and is intended to reduce the false alarms that arise when two vehicles merely appear close because of camera perspective, temporary occlusion, shadow, or bounding-box jitter.

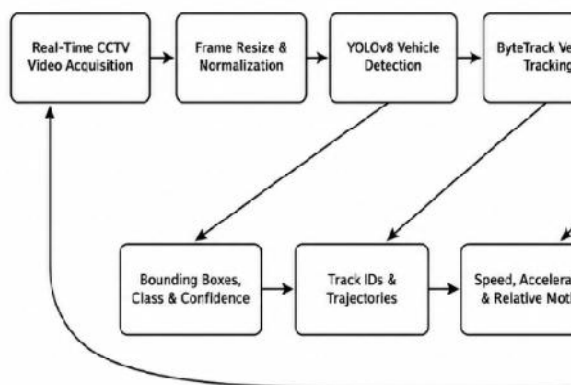


Figure 1. Expanded workflow of the proposed YOLOv8-, ByteTrack-, motion-analysis-, and alert-based accident monitoring system.



3.1 System Architecture and Continuous Processing Cycle

The complete architecture contains six interacting modules: video acquisition and preprocessing, YOLOv8 vehicle detection, multi-object tracking, motion-feature extraction, collision confirmation and severity assessment, and emergency communication. A CCTV camera, stored video, RTSP stream, or local video source provides frames to the processing application. Every accepted detection is associated with a track identity, and the recent history of each track is maintained in memory. Pairwise vehicle interactions are evaluated only for currently active tracks. When the accident conditions are not satisfied, the system immediately returns to the next frame. When the conditions persist, the event is recorded and transmitted to the dashboard and alert service.

This closed-loop architecture separates detection from decision making. Such separation is important because YOLOv8 predicts what objects are present and where they are located, but it does not by itself prove that a collision occurred. The collision-decision stage uses the detector output as measurable geometric evidence and combines it with time-dependent motion features.

Stage	Main input	Processing	Output
Video acquisition	CCTV/video stream	Frame capture and timing	Frame F_t and timestamp
Preprocessing	Raw frame	Resize, normalization, optional enhancement	Model-ready image
YOLOv8 detection	Model-ready image	Inference, thresholding and NMS	Boxes, classes, confidences
ByteTrack	Current detections	Association and ID management	Persistent vehicle tracks
Motion analysis	Track history	Position, speed, acceleration and relative motion	Trajectory features
Collision analysis	Pairwise track features	Distance, IoU, TTC and temporal confirmation	Accident decision
Severity and alert	Confirmed event	Risk scoring, evidence saving and API delivery	Dashboard/API alert

3.2 Real-Time CCTV Video Acquisition

The input video is modeled as an ordered sequence of frames $V = \{F_1, F_2, \dots, F_T\}$. Each frame F_t has an associated capture time t_t . The processing loop reads the stream continuously until the source ends or the operator stops the application. The frame rate is used to convert pixel displacement into a time-dependent motion estimate. For a nominal frame rate FPS, the time interval between consecutive frames is:

$$\Delta t = 1 / \text{FPS}$$

The actual implementation should preferably use timestamp differences rather than assuming a perfectly constant frame rate, particularly for network cameras. When frames arrive late or are dropped, the measured time difference $\Delta t = t_t - t(t-1)$ prevents incorrect speed and acceleration estimates. Each frame is also tagged with the camera identifier and configured location, allowing the alert message to contain meaningful site information instead of placeholders such as TEST_LOCATION.

To support real-time operation, the acquisition and inference stages may be separated into threads or queues. The capture process continuously obtains the newest frame, while the inference process consumes frames at the maximum sustainable rate. A bounded queue avoids unmanageable delay; when processing cannot keep up, stale frames may be dropped so that the system remains close to real time.

3.3 Frame Preprocessing and Scene Stabilization

Each captured frame is resized to the input dimensions required by the selected YOLOv8 model. Pixel values are normalized from the integer range $[0, 255]$ to $[0, 1]$. If F_t is the original frame and $H \times W$ is the detector input size, the preprocessing operation is expressed as:



$$F't = \text{Resize}(F_t, H, W), \quad I_n = I / 255$$

The uploaded methodology identifies illumination variation, shadows, rain, glare, road reflections, and camera noise as important challenges. Therefore, optional preprocessing may include contrast enhancement, gamma correction, mild denoising, or region-of-interest masking. Enhancement must remain conservative because aggressive filtering can alter vehicle boundaries and reduce detector accuracy.

A road-region mask can exclude sky, buildings, sidewalks, and other regions that cannot contain relevant vehicle collisions. This reduces unnecessary detections and pairwise comparisons. For a fixed camera, a homography or perspective calibration can further map image coordinates to an approximate road plane. Although the basic implementation works in pixel coordinates, perspective compensation improves distance and speed interpretation because equal pixel displacement does not represent equal physical displacement at different depths in the image.

3.4 YOLOv8-Based Vehicle Detection

The preprocessed frame is passed to a pretrained or custom-trained YOLOv8 detector [1]. The model produces a set of candidate detections:

$$D_t = \text{YOLOv8}(F't; \theta) = \{(B_i, c_i, s_i)\} \text{ for } i = 1, \dots, N_t$$

$B_i = (x_i, y_i, w_i, h_i)$ denotes a bounding box, c_i is the class label, s_i is the confidence score, and θ represents the trained network parameters. The center point of the box is calculated as:

$$C_i = (x_i + w_i/2, y_i + h_i/2)$$

Only road-object classes relevant to the project, such as car, motorcycle, bus, and truck, are retained. Detections below a selected confidence threshold τ_{conf} are rejected. The confidence can be interpreted as the combination of objectness and conditional class probability:

$$\text{Conf}_i = P(\text{object}) \times P(\text{class}_i | \text{object})$$

Multiple candidate boxes may refer to the same vehicle. Non-Maximum Suppression removes duplicates using intersection-over-union:

$$\text{IoU}(B_i, B_j) = \text{Area}(B_i \text{ intersection } B_j) / \text{Area}(B_i \text{ union } B_j)$$

When the IoU between two boxes of the same class exceeds the NMS threshold, the lower-confidence box is suppressed. The accepted boxes and confidence values are forwarded to ByteTrack. The model size should be selected according to the available hardware: smaller variants reduce latency, whereas larger variants may improve localization under difficult traffic conditions.

3.5 ByteTrack-Based Multi-Object Tracking

Accident detection is inherently temporal; therefore, vehicle identities must be maintained across frames. ByteTrack associates current detections with existing track states and retains useful low-confidence boxes that might otherwise be discarded during short occlusions [3]. Each active vehicle receives a track identifier ID_i . The state of a track may include its latest box, center point, velocity estimate, age, missed-frame count, and a finite history of previous positions. The tracker first predicts where an existing vehicle should appear in the current frame, then performs data association between predicted tracks and detections. High-confidence detections are matched first, followed by eligible lower-confidence detections. Unmatched detections may initialize new tracks, while tracks that remain unmatched beyond a configured age are removed.

The output of tracking is a set of temporally connected trajectories rather than isolated boxes. This continuity is essential for estimating whether two vehicles are approaching, separating, suddenly stopping, or changing direction after contact.

3.6 Vehicle Localization and Trajectory Smoothing

For each tracked vehicle, the bounding-box center provides a compact image-plane position. Raw center points may fluctuate because of detector uncertainty, partial occlusion, vibration, and scale change. An exponential smoothing operation can reduce these fluctuations:



$$Cbar_i^t = \alpha C_i^t + (1 - \alpha) Cbar_i^{(t-1)}, \quad 0 < \alpha \leq 1$$

A larger alpha follows rapid motion more closely, whereas a smaller alpha produces a smoother trajectory. Smoothing should not be so strong that it hides sudden impact motion. The system stores the most recent L positions for every vehicle. The resulting trajectory $T_i = \{Cbar_i^{(t-L+1)}, \dots, Cbar_i^t\}$ is supplied to the motion and collision modules.

For calibrated cameras, the smoothed image point can be transformed to a road-plane coordinate using a homography matrix H . This optional step converts pixel coordinates into approximate metric coordinates and enables more meaningful safe-distance and speed thresholds.

3.7 Motion Feature Estimation

The uploaded methodology explicitly uses displacement, velocity, acceleration, relative motion, and trajectory behavior. For vehicle i , the displacement vector between consecutive frames is:

$$\begin{aligned} \Delta C_i^t &= Cbar_i^t - Cbar_i^{(t-1)} \\ v_i^t &= \Delta C_i^t / \Delta t \\ a_i^t &= (v_i^t - v_i^{(t-1)}) / \Delta t \end{aligned}$$

The speed magnitude is $\|v_i\|$, while the direction is obtained from the displacement vector. Sudden negative acceleration, an abrupt direction change, or an unexpected loss of motion after high-speed travel may indicate an impact. To reduce noise, the system may use a short moving average of velocity and acceleration.

For two vehicles i and j , relative position and relative velocity are computed as:

$$\begin{aligned} r_{ij}^t &= Cbar_j^t - Cbar_i^t \\ v_{rel,ij}^t &= v_j^t - v_i^t \end{aligned}$$

The dot product $r_{ij} \cdot v_{rel,ij}$ indicates whether the vehicles are approaching. A negative value means that the separation is decreasing. Pairwise analysis may be restricted to vehicles whose boxes or centers fall within a broad neighborhood, reducing computation in crowded scenes.

3.8 Geometric Collision Indicators

The collision module combines several indicators rather than trusting a single threshold. The first indicator is the Euclidean center distance:

$$d_{ij}^t = \|Cbar_i^t - Cbar_j^t\|_2$$

A distance below d_{safe} indicates proximity but does not prove collision because perspective can make vehicles in different lanes or depths appear close. The second indicator is bounding-box overlap IoU_{ij} . A high IoU may represent physical contact, but temporary occlusion can also increase overlap. Therefore, distance and IoU are interpreted together with motion.

The third indicator is time-to-collision. A simple approximation uses the closing component of relative velocity along the line joining the vehicles:

$$\begin{aligned} v_{close,ij} &= -(r_{ij} \cdot v_{rel,ij}) / (\|r_{ij}\|_2 + \epsilon) \\ TTC_{ij} &= d_{ij} / \max(v_{close,ij}, \epsilon), \quad \text{when } v_{close,ij} > 0 \end{aligned}$$

A small positive TTC indicates an imminent collision risk. When the vehicles are separating, TTC is treated as infinite and does not contribute to the accident score. Additional indicators may include rapid box-scale change, sudden trajectory intersection, post-contact stopping, and abnormal acceleration.

3.9 ConvLSTM-Based Temporal Risk Prediction

The document also identifies ConvLSTM as an optional deep temporal model. Whereas the rule-based module uses explicit geometric variables, ConvLSTM learns spatial and temporal evolution from a sequence of feature maps or cropped traffic regions [8]. A sequence of K frames or motion maps is supplied to the network. Convolutional gates retain spatial structure while the recurrent state stores motion context.



The ConvLSTM output can be interpreted as a risk probability PDL(accident | $F(t-K+1):F_t$). This probability is not used alone; it is fused with the interpretable geometric score. Such fusion retains explainability while allowing the model to recognize temporal patterns that are difficult to represent with fixed thresholds, such as pre-impact convergence, impact overlap, sudden trajectory discontinuity, and post-impact stopping.

3.10 Multi-Parameter Collision Decision

For every candidate vehicle pair, the system forms a frame-level collision indicator. A transparent logical rule based on the uploaded methodology is:

$$R_{ij}^t = 1[(d_{ij} < \tau_d) \text{ OR } (IoU_{ij} > \tau_{IoU}) \text{ OR } (TTC_{ij} < \tau_{TTC}) \text{ OR } (A_{abnormal} > \tau_a)]$$

A weighted risk formulation can also be used to avoid a hard decision from one noisy feature:

$$S_{ij}^t = w_d f_d(d_{ij}) + w_o IoU_{ij} + w_t f_t(TTC_{ij}) + w_a f_a(a_{i,j}) + w_l PDL$$

The normalized functions f_d and f_t convert distance and TTC to [0,1]. The weights are selected using validation data. The pair is considered collision-positive when S_{ij} exceeds a decision threshold. This design permits threshold tuning for different cameras and road layouts.

3.11 Multi-Frame Confirmation and False-Alarm Suppression

The document emphasizes that a collision must continue for more than one frame before an alert is issued. This multi-frame confirmation is one of the most important reliability stages. Let R_{ijt} be the binary frame-level risk indicator. A collision is confirmed when at least M of the most recent K frames are positive:

$$\text{Confirm}_{ij}^t = 1[\sum_{q=t-K+1}^t R_{ij}^q \geq M]$$

This temporal voting removes short false positives produced by shadows, one-frame detector errors, camera vibration, temporary occlusion, or normal close passing. An event cooldown is also required. Once an alert has been generated for a particular track pair, the application suppresses duplicate messages for a configured interval unless the severity increases or a new event is detected.

Further false-alarm controls include requiring an approaching motion state before using TTC, rejecting overlap between stationary vehicles in parking regions, using lane or road-region information, verifying sudden post-contact deceleration, and excluding pairs with insufficient tracking history.

3.12 Accident Severity Estimation

After confirmation, the event is assigned a severity level. The severity score combines geometric contact, proximity, time-to-collision, relative speed, abnormal acceleration, persistence, and optional ConvLSTM risk:

$$\text{Severity} = \beta_1 IoU + \beta_2 D_{risk} + \beta_3 TTC_{risk} + \beta_4 V_{rel} + \beta_5 A_{risk} + \beta_6 \text{Persistence}$$

All terms are normalized before combination. The score is mapped to LOW, MEDIUM, or HIGH using two thresholds. A high score indicates stronger overlap, high closing speed, very small TTC, large acceleration change, and persistent collision evidence. The result screenshots show HIGH severity, 95% confidence, two involved vehicles, and a recommendation to notify police and ambulance services.

3.13 Evidence Capture and Event Record Creation

When an event is confirmed, the application stores evidence instead of saving only a text message. A ring buffer can retain several seconds of video before the decision, while recording continues for several seconds after the event. The saved evidence may contain the original clip, an annotated clip, the collision frame, cropped vehicle images, and a JSON or database record.

The event record should include a unique event ID, camera ID, configured location, start time, confirmation time, vehicle track IDs, detected classes, detector confidence, collision score, severity, evidence paths, message-delivery status, and operator review status. This structure explains why the dashboard may contain more evidence files than alerts: one incident can generate multiple artifacts.



3.14 Emergency Alert Generation and Messaging API

The confirmed event is transformed into a structured notification. The uploaded document specifies accident status, timestamp, camera ID or location, detected vehicle classes, confidence, frame number, and severity. The alert may be transmitted through SMS, WhatsApp, Telegram, email, a mobile application, or the web dashboard. Recipients can include ambulance services, police stations, hospitals, traffic control rooms, and system administrators.

The alert-delivery module follows a controlled sequence:

1. Create the event record and attach the evidence identifiers.
2. Select recipients according to severity and configured jurisdiction.
3. Format a concise human-readable message with time, location, vehicles, confidence, severity, and required action.
4. Submit the message to the selected messaging API.
5. Store the provider response and delivery status.
6. If delivery fails, retry using exponential delay until the maximum retry count is reached.
7. Escalate persistent failures to the dashboard and an alternative communication channel.

Duplicate alert suppression is performed using the event ID, track pair, camera, and cooldown interval. This prevents repeated messages from being sent every frame while the same accident remains visible.

3.15 Flask Dashboard and Network Access

The implementation includes a Flask-based dashboard that converts the accident-detection process into an operational monitoring system. It displays system status, last-update time, total alerts, current-day alerts, evidence-file count, latest incident time, and detailed alert cards. Manual refresh and ten-second automatic refresh are supported. The application can be accessed on the host through localhost:5000 and from other devices using the host IP address on the same network.

The dashboard reads incident records from the application storage and presents them in reverse chronological order. A production deployment should add authentication, role-based access, HTTPS, event acknowledgment, search and filtering, evidence download controls, and audit logging. The dashboard time should be derived from the same synchronized server clock used by the detection and evidence modules.

3.16 Operational Pseudocode

The end-to-end processing logic can be summarized as follows:

1. Initialize YOLOv8, ByteTrack, optional ConvLSTM, camera configuration, evidence buffer, dashboard storage, and messaging client.
2. Read a frame and timestamp; resize and normalize the frame.
3. Run YOLOv8 inference; filter classes and confidence; apply Non-Maximum Suppression.
4. Update ByteTrack and obtain active vehicle IDs and trajectories.
5. Smooth positions and calculate speed, acceleration, relative motion, distance, IoU, and TTC.
6. Generate frame-level risk scores for candidate vehicle pairs.
7. Update multi-frame confirmation histories.
8. If no pair is confirmed, continue with the next frame.
9. If a pair is confirmed, compute severity and check the event cooldown.
10. Save pre-event and post-event evidence and write the incident record.
11. Update the dashboard and send the emergency alert with retry handling.
12. Continue monitoring for new events.

3.17 Parameter Configuration and Calibration

The accident detector depends on configurable thresholds, and these values must be calibrated for each camera. Important parameters include detector confidence, NMS IoU, maximum track age, track-history length, smoothing



factor, safe center distance, collision IoU threshold, minimum TTC, abnormal acceleration threshold, K-frame window length, M-frame confirmation count, severity boundaries, cooldown period, pre-event recording duration, post-event recording duration, and API retry limit.

A single universal pixel-distance threshold is not recommended because image scale changes with camera height and perspective. Calibration should use representative normal traffic, near-miss, and accident clips. The thresholds are selected to maximize recall while controlling false alerts per hour. Separate profiles may be maintained for highways, intersections, parking areas, tunnels, and night operation.

3.18 Evaluation Protocol

The dashboard screenshots verify functional alert generation but do not fully evaluate accident-recognition accuracy. A complete experimental protocol should divide labeled videos into training, validation, and independent test sets without scene leakage. Vehicle detection is evaluated using precision, recall, and mAP. Tracking is evaluated using IDF1, MOTA, identity switches, and track fragmentation. Accident detection is evaluated using true positives, false positives, false negatives, precision, recall, F1-score, specificity, false alarms per hour, and average detection delay.

Precision = $TP / (TP + FP)$

Recall = $TP / (TP + FN)$

$F1 = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall})$

Alert-system evaluation should additionally measure message delivery success, delivery latency, duplicate-alert rate, evidence completeness, dashboard update delay, and system uptime. Results should be reported separately for daylight, night, rain, fog, occlusion, camera vibration, high-density traffic, and different viewing angles.

IV. RESULTS AND DISCUSSION

The attached outputs demonstrate the web-based alert and monitoring components of the proposed system. The images document multiple test alerts, dashboard statistics, network accessibility, and live system status. Because the supplied screenshots contain test events rather than a labeled benchmark evaluation, the results are interpreted as functional-system evidence. They do not by themselves establish precision, recall, mAP, or false-alarm performance on real accident videos.



Figure 2. Formatted test accident alert generated at 16:38:14, including location, collision type, confidence, vehicles, severity, emergency services, and immediate actions.



ACCIDENT_ALERT

2025-11-29 16:38:36 (227 day(s) ago)

```

|| ACCIDENT ALERT ||
|| Time: 2025-11-29 16:38:36
|| Location: TEST_LOCATION
|| Collision Type: TEST_COLLISION
|| Confidence: 95.0%
|| Vehicles 1 & 2
|| Severity: HIGH
|| Required: Police 🚓 + Ambulance 🚑

```

IMMEDIATE ACTION REQUIRED:

- Dispatch emergency services to location
- Check for injuries
- Secure the area

Figure 3. Second formatted test alert generated at 16:38:36, demonstrating repeatable event creation and logging.

ACCIDENT_ALERT

2025-11-29 17:15:40 (227 day(s) ago)

```

|| ACCIDENT ALERT ||
|| Time: 2025-11-29 17:15:40
|| Location: TEST_LOCATION
|| Collision Type: TEST_COLLISION
|| Confidence: 95.0%
|| Vehicles 1 & 2
|| Severity: HIGH
|| Required: Police 🚓 + Ambulance 🚑

```

IMMEDIATE ACTION REQUIRED:

- Dispatch emergency services to location
- Check for injuries
- Secure the area

Figure 4. Third formatted test alert generated at 17:15:40, confirming persistent operation across separate event times.

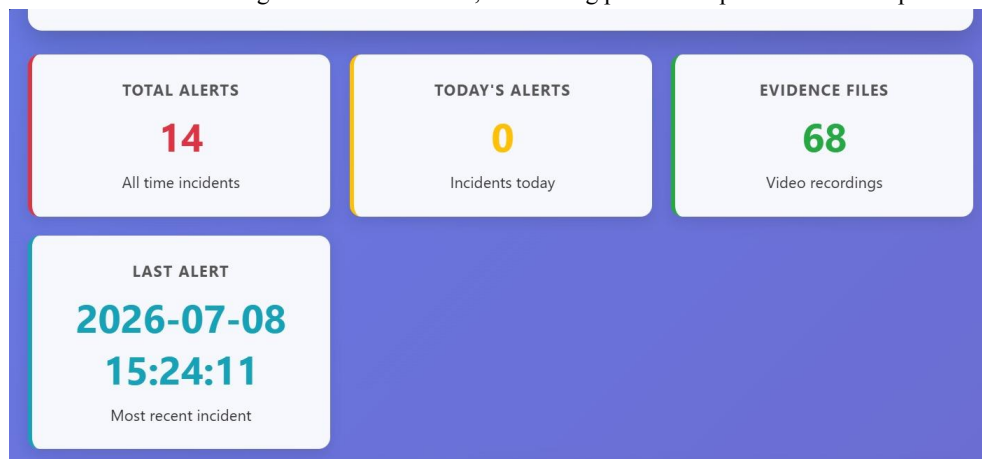
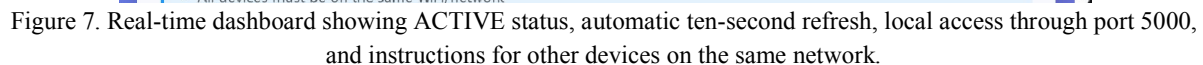
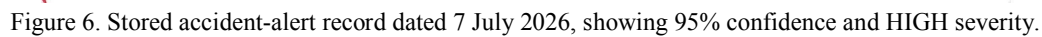


Figure 5. Dashboard summary showing 14 total alerts, zero alerts for the current day, 68 evidence files, and the latest incident time.





The dashboard reports 14 accumulated alerts and 68 evidence files, indicating that the application stores more than one evidence artifact per incident in some cases. The latest-incident card provides a direct time reference, while the separate alert cards preserve detailed event content. Automatic refresh every ten seconds allows the operator to observe new incidents without manually reloading the page.

4.3 Network Operation

The system is served through a Flask endpoint on port 5000. The dashboard displays localhost access for the processing machine and an IP-based address for other devices connected to the same Wi-Fi or LAN. This enables traffic-control staff, administrators, or emergency coordinators to review alerts from another computer or mobile device while the detection process runs on the host system.

4.4 Observed Limitations

The screenshots show TEST_LOCATION and TEST_COLLISION placeholders, so the alert pipeline should next be linked to actual camera metadata and automatically inferred collision categories. The displayed historic alert dates and relative-age labels should also be synchronized with the server clock. Most importantly, test-message success does not verify collision-detection accuracy. A publication-grade evaluation should report a confusion matrix, precision, recall, F1-score, alert accuracy, false alarms per hour, average detection latency, mAP for vehicle detection, tracking IDF1/MOTA, and results under night, rain, occlusion, and camera-perspective changes.

Observed output	Value	Interpretation
Alert confidence	95.0%	Configured test-event confidence
Severity	HIGH	Highest-priority response category
Required services	Police + Ambulance	Actionable emergency routing
Total alerts	14	Cumulative incidents stored by dashboard
Today's alerts	0	No event counted on the screenshot date
Evidence files	68	Saved video/evidence artifacts
Dashboard refresh	10 seconds	Periodic live monitoring
Web service	Port 5000	Local/LAN dashboard access

V. CONCLUSION

This work presented an end-to-end automobile accident detection and emergency-alert system that combines YOLOv8 vehicle detection, ByteTrack multi-object tracking, interpretable motion and collision measures, optional ConvLSTM temporal risk prediction, multi-frame verification, severity estimation, evidence recording, and a Flask-based dashboard. The attached results verify the operation of the alert interface, incident storage, dashboard statistics, network access, and automatic refresh. The system successfully transforms an accident decision into a detailed high-severity alert containing a 95% confidence value and recommended police and ambulance response. Future work should replace test placeholders with camera-derived metadata, integrate a dependable messaging provider, calibrate thresholds using real accident and near-miss videos, evaluate accuracy and latency quantitatively, and add geographic mapping, multi-camera fusion, night enhancement, and edge-device deployment. With these improvements, the framework can become a practical component of smart-city and intelligent-transportation safety infrastructure.

REFERENCES

- [1] Ultralytics, "Explore Ultralytics YOLOv8," Ultralytics Documentation, 2026.
- [2] Ultralytics, "YOLO Multi-Object Tracking in Video," Ultralytics Documentation, 2026.
- [3] Y. Zhang et al., "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," in Proc. European Conf. Computer Vision (ECCV), 2022, pp. 1–21.
- [4] L. You et al., "Multi-Object Vehicle Detection and Tracking Algorithm Based on Improved YOLOv8 and ByteTrack," Electronics, vol. 13, no. 15, Art. no. 3033, 2024, doi: 10.3390/electronics13153033.
- [5] J. Liu et al., "Vehicle Flow Detection and Tracking Based on an Improved YOLOv8n and ByteTrack," World Electric Vehicle Journal, vol. 16, no. 1, Art. no. 13, 2025.
- [6] P. Kalpana et al., "Road Traffic Accident Detection Based on YOLOv8 and ByteTrack," AIP Conference Proceedings, vol. 3204, no. 1, Art. no. 040013, 2025, doi: 10.1063/5.0248651.



- [7] S. M. Shaqib et al., "Vehicle Speed Detection System Utilizing YOLOv8: Enhancing Road Safety and Traffic Management for Metropolitan Areas," arXiv:2406.07710, 2024.
- [8] X. Shi et al., "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Advances in Neural Information Processing Systems*, vol. 28, 2015, pp. 802–810.
- [9] P. Kovačovič et al., "A Comprehensive Analysis of Incident and Object Detection Methods in Intelligent Transportation Systems," *Smart Cities*, vol. 9, no. 3, Art. no. 41, 2026.
- [10] L. Bompani et al., "Multi-Resolution Rescored ByteTrack for Video Object Detection on Ultra-Low-Power Embedded Processors," arXiv:2404.11488, 2024.
- [11] L. Van Ma et al., "Adaptive Confidence Threshold for ByteTrack in Multi-Object Tracking," arXiv:2312.01650, 2023.
- [12] Ultralytics, "Object Detection Datasets Overview and Supported Formats," Ultralytics Documentation, 2026.

