FMDB *Transactions on Sustainable Critical Infrastructures*

An Edge-Based Deep Learning Surveillance System Using YOLOv8 for Real-Time Multi-Zone Intrusion Detection and Intelligent Alerting

Yashmit Rai[1,*], Girik Sharma[2], V. Kamalesh[3], R. Regin[4], Maruf Farhan[5]

[1,2,3,4]Department of Artificial Intelligence and Machine Learning, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India.
[5]Department of STEM, University of Sussex International Study Centre, Brighton, England, United Kingdom.
yr1901@srmist.edu.in[1], gz7856@srmist.edu.in[2], kv3540@srmist.edu.in[3], regin12006@yahoo.co.in[4], maruf.farhan@sussex.ac.uk[5]

Abstract: The need for reliable and independent surveillance systems has witnessed an increase in the recent past, especially in security-critical areas such as restricted areas, industrial areas, and learning centres. However, conventional detection-based surveillance systems are known to trigger alarms for all detected targets, regardless of their spatial importance, leading to numerous false alarms and undermining the effectiveness of the system. To mitigate the drawbacks of conventional detection-based surveillance systems, this paper proposes an innovative approach to develop a surveillance system using an edge-based intrusion detection system in conjunction with a hierarchical multi-zone spatial decision algorithm. In this proposed system, the entire area of interest is divided into three zones: Safe, Warning, and Intrusion. Alarms are generated only for targets detected in the Intrusion zone of the surveillance area. The system is designed to be independent and self-sufficient, without relying on cloud infrastructure, ensuring immediate responses and maintaining the privacy of surveillance data. Further, the system has utilised Telegram for real-time alerts and the web-based system for real-time surveillance. The results show that false-positive alerts are reduced to a minimum with spatially aware object detection, enabling efficient real-time performance and making the system suitable for real-time scenarios.

Keywords: Edge Computing; Intrusion Detection; Multi-zone Classification; Computer Vision; Surveillance Systems; Real-time Surveillance; False Positive Reduction; Independent Surveillance.

Received on: 01/08/2025, **Revised on:** 20/09/2025, **Accepted on:** 28/11/2025, **Published on:** 31/03/2026

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCIS

DOI: https://doi.org/10.69888/FTSCIS.2026.000710

Cite as: Y. Rai, G. Sharma, V. Kamalesh, R. Regin, and M. Farhan, "An Edge-Based Deep Learning Surveillance System Using YOLOv8 for Real-Time Multi-Zone Intrusion Detection and Intelligent Alerting," *FMDB Transactions on Sustainable Critical Infrastructures*, vol. 1, no. 1, pp. 16–32, 2026.

1. Introduction

There is a growing need for smart surveillance in recent times due to heightened security concerns across environments such as borders, industries, and schools. Traditional surveillance involves constant human monitoring. This approach is often not effective due to human limitations, as people get tired and misunderstand quiet times and infrequent, intense situations that often occur in these environments [6]. Recent developments in deep learning, especially in computer vision, have also made

[*]Corresponding author.

automated surveillance more effective. For instance, in real-time video surveillance, object detection using YOLO and related algorithms is effective because they balance speed and accuracy. However, these algorithms only recognise who or what is in a video, without giving information about these people and things. This often leads to false alarms and reduces the overall effectiveness of surveillance systems [8]. Due to increased computational requirements for effective surveillance, cloud computing has also been proposed as a means to perform tasks remotely. However, this is often associated with increased concerns about latency, connectivity, and privacy, as discussed in references [10]. Another way to solve this problem is edge computing, where data is processed at the edge. It also helps address issues of latency, reliability, and data privacy. Previous studies have also shown that the proposed deep learning models can be used on edge computing devices in surveillance systems [12]. However, the edge computing system is based solely on object detection, not on decision-making. This is because not all objects in the real environment are of concern when object detection is performed there [15]. In addition, there is no sense of discrimination in the alerts sent by the surveillance system, given its lack of spatial awareness. It negatively affects the system's performance. Therefore, intelligent systems should be developed that can differentiate between relevant and irrelevant objects in the environment [16]. To overcome the system's failure to send alerts in the absence of spatial awareness, this paper proposes an edge computing system with a hierarchical multi-zone spatial decision model and a YOLOv8n object detection model [1]. The spatial decision model is defined in which three different zones are defined in the spatial environment of the system, namely Safe, Warning, and Intrusion, and alerts are sent only in the case of object detection in the Intrusion zone [14].

2. Related Work

In general, the domain of surveillance system monitoring has seen a significant shift over the last two decades. It is important to understand the evolution of the previous literature to assess the novelty of this paper, especially regarding the zone-based false-positive reduction mechanism, which has not been studied in relation to the edge-based system [21]. The conventional methods for automated surveillance systems include image processing techniques such as background subtraction, image differencing, and optical flow [20]. Although these conventional methods are computationally simple, they are highly sensitive to environmental changes. The conventional methods are highly susceptible to illumination variations, dynamic backgrounds, partial occlusions, and camera motion, leading to false alarms [13]. Additionally, the inability of conventional methods to distinguish relevant from irrelevant motion led to a high rate of false alarms, making them less practical for this domain. The introduction of CNN-based object detection architectures significantly boosted this domain [18]. The Faster R-CNN proposed region proposal networks, enabling two-stage object detection with high accuracy. Single-stage object detection architectures, such as SSD in Liu et al. [4] and YOLO in Redmon and Farhadi [5], shifted the focus from two-stage to single-stage detection, sacrificing a little accuracy for very high detection speed. The current state of the art in single-stage object detection architectures is YOLOv8, which offers very high detection accuracy and speed, making it suitable for both GPU and CPU devices. Although all the existing works are based on these architectures, they are not used for any specific purpose, as explained in this paper. In all existing works, these models are used in their pure detection mode, meaning that all detections in all frames are used for alarm generation [19]; [24].

This does not account for spatial information in the detection and may result in false positives, as observed in the baseline of this paper [23]. The region-based approach, or geofencing, has also been studied to improve the relevance of alerts [9]. The region-based approach, or geofencing, involves defining regions of interest within the camera’s field of view. The response is limited to only those regions of interest. This approach is somewhat similar to the zone-based approach that is proposed in this paper. However, in the geofencing approach, there is no hierarchy among Safe, Warning, and Intrusion levels. Moreover, there is no quantitative analysis of geofencing to reduce the false-positive rate [22]. Edge Computing for Surveillance: Various works, such as Shi et al. [6] and Mohammadi et al. [10], have studied edge computing for surveillance. These works have shown that lightweight models can be used for real-time detection with resource-constrained devices. However, these works are mostly limited to detection without considering how the detections are classified and used. Alerting Mechanisms: Alerting mechanisms using SMS, email, and other messaging services have also been studied in various works, such as Telegram Bot API [2] and Narayanan and Shmatikov [11]. These works are mostly limited to text-based alerts without any visual evidence of the event, which is mostly useful in time-critical scenarios. In contrast, the proposed system considers all aspects of an edge-computing-based surveillance system. Table 1 provides an overview of the key differences between our system and previous work. The key differences are that our system uses edge deployment, hierarchical zone-based spatial filtering, and an end-to-end alerting pipeline, none of which are used in any of the previous work [17]. The quantified false-positive reduction used in our work is not used in any previous work.

Table 1: Comparison of existing surveillance approaches

Author	Method	Edge/ Cloud	Zone Logic	Alerting	Limitation
Ren et al. [3]	Faster R-CNN	Cloud	No	No	High latency, no context
Liu et al. [4]	SSD	Cloud	No	No	No spatial filtering
Redmon and Farhadi [5]	YOLOv3	Hybrid	No	No	No alerting pipeline

Shi et al. [6]	Edge framework	Edge	No	No	Detection only
Jocher et al. [1]	YOLOv8 (base)	Both	No	No	No zone / no alert
Szeliski [7]	BG Subtraction	Edge	No	No	High false positive rate
Telegram Bot API [2]	YOLOv8n + Zone Logic	Edge	Yes	Telegram	Fully integrated, 64.5% FP reduction

2.1. System Design and Architecture

2.1.1. System Overview

The proposed system is an autonomous, edge-based intrusion detection framework that continuously monitors the defined area of interest, detects security-critical activities, and responds to legitimate security threats in real time without relying on a cloud-based environment. The proposed system comprises four highly correlated functional blocks: real-time video acquisition, object detection using YOLOv8n, hierarchical zone classification, and response generation. These blocks are highly correlated and work as an end-to-end closed-loop system. There is seamless integration between these blocks from monitoring to response without any manual intervention. Figure 1 shows the overall proposed system architecture, which involves the flow of camera input through the frame processing module and the YOLOv8n detection module, and then through the multi-zone classification module. Moreover, it involves the flow of legitimate security threats through the event manager to evidence capture, the logging system, Telegram, and the dashboard. This is the overall flow of the proposed system, which is significantly different from existing work that focuses only on individual blocks of the proposed framework.

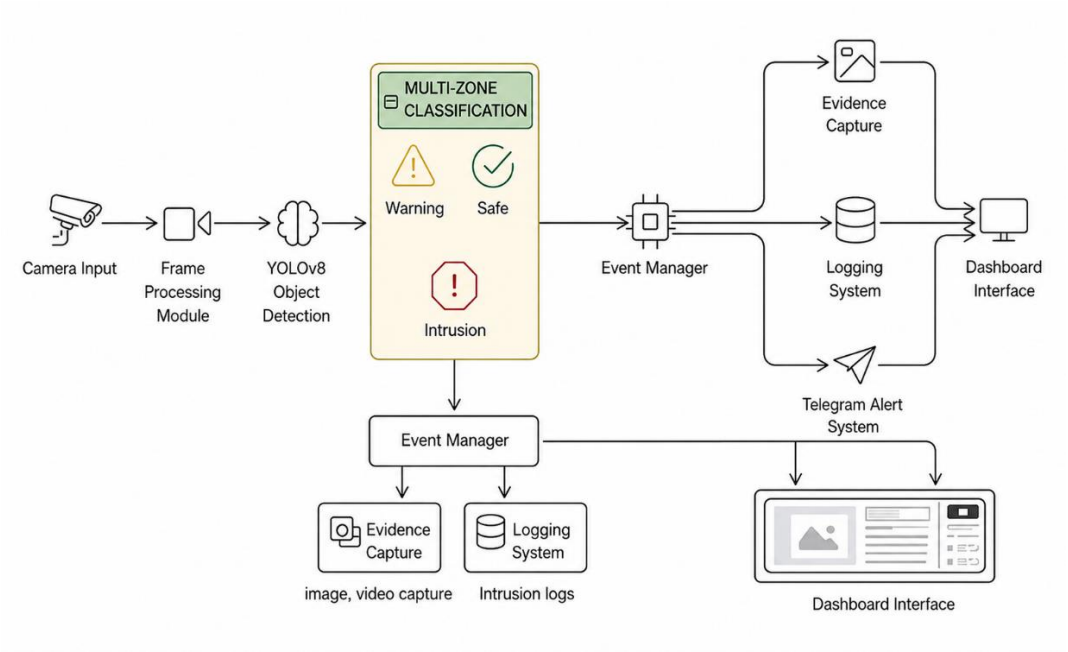

Figure 1: Overall architecture of the proposed edge-enabled multi-zone intrusion detection system, showing the complete pipeline from camera input through Yolov8n detection, multi-zone classification, event management, evidence capture, telegram alerting, and dashboard interface

2.1.2. Layered Architecture

To achieve modularity, scalability, and effective edge operations, the system is designed with a three-layer architecture, as shown in Figure 2. Each layer performs a set of specific operations and communicates with other layers via defined interfaces. This enables the independent development and testing of each component without affecting the overall process. The Sensing Layer is designed to continuously perform video sensing using the installed surveillance camera. The video feed is continuously provided to the overall process as a stream of frames, regardless of the camera used in the system. The Edge Intelligence Layer is the core of the system's operations. YOLOv8n continuously processes the video feed to detect and classify people and vehicles. The zone classification module classifies detection results spatially based on their location relative to the three zones: Safe, Warning, and Intrusion. Only the Intrusion Zone results are propagated to the response subsystem. The Response and Control Layer is responsible for capturing evidence, recording logs, sending real-time Telegram alerts, and updating the

dashboard. Operations are performed concurrently with the overall detection process to ensure no degradation in the overall process.

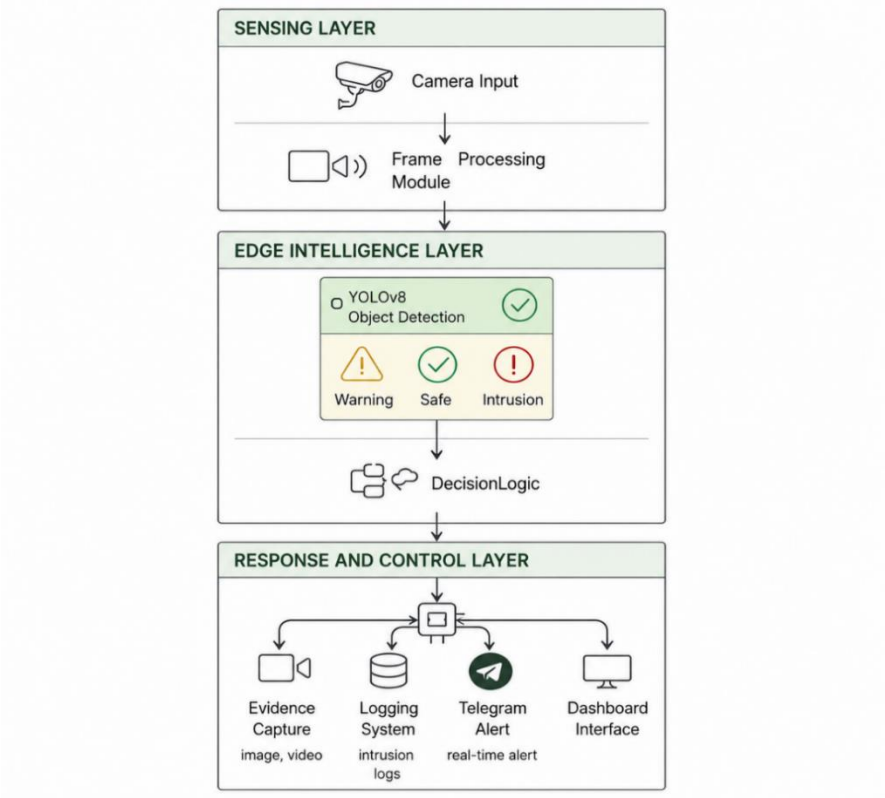

Figure 2: Layered architecture of the proposed system showing the sensing layer, edge intelligence layer with Yolov8n detection and multi-zone classification, and the response and control layer managing evidence capture, logging, telegram alerting, and dashboard interface

2.1.3. Hardware and Software Configuration

The proposed system is implemented and evaluated entirely on local edge hardware without relying on any cloud infrastructure. The hardware and software configuration for system development and performance evaluation is given in Table 2. For performance evaluation, experiments are conducted on an Apple MacBook Air with an M4 chip and 16 GB of RAM, which is capable of real-time processing on a CPU-based setup. The YOLOv8n model is selected for its lightweight architecture and ability to run on resource-constrained devices. This allows for real-time processing without requiring any additional hardware acceleration. The video processing pipeline is implemented with OpenCV, and the monitoring dashboard with Flask. Both components are executed concurrently without interfering with the overall system.

Table 2: Hardware and software configuration

Component	Specification
Edge Device	Apple MacBook Air (M4)
Processor	Apple M4 Chip
RAM	16 GB
Operating System	macOS
Python Version	3.10
YOLOv8 Variant	YOLOv8n (nano) — optimised for edge CPU deployment
Deep Learning Stack	Ultralytics 8.0, PyTorch 2.0
Camera	Built-in FaceTime HD Camera / USB Camera (up to 1080p, 30 FPS)
Alert Platform	Telegram Bot API v6.x (image-based notifications)
Dashboard Stack	Flask, OpenCV, HTML5 — asynchronous architecture

3. Methodology

3.1. Video Pre-processing

The frames are extracted individually, then preprocessed before the actual inference. The image pre-processing involves resizing to 640x640 pixels, the standard resolution for the YOLOv8n model. The image's pixels are then normalised to the range (0, 1). The frames are sampled so that the detection process runs at a well-balanced speed. The process speed is maintained, and there is no compromise on the edge device under any condition, including changes in illumination or object movement. The pre-processing step is done with negligible overhead.

3.2. Object Detection Using YOLOv8n

For object detection, the YOLOv8n model is used. The YOLOv8n model was chosen for its balance of speed and detection capabilities, making it suitable for edge devices. The YOLOv8n model is used to detect objects in the preprocessed video frames in a single forward pass. The output of the YOLOv8n model consists of object detection results, including bounding box coordinates, class labels, and confidence values for each detected object. The YOLOv8n model output is filtered to include only two object classes significant in security scenarios: persons (COCO class 0) and vehicles (COCO classes 2, 5, 7). The details of the object detection process are given in Figure 3.

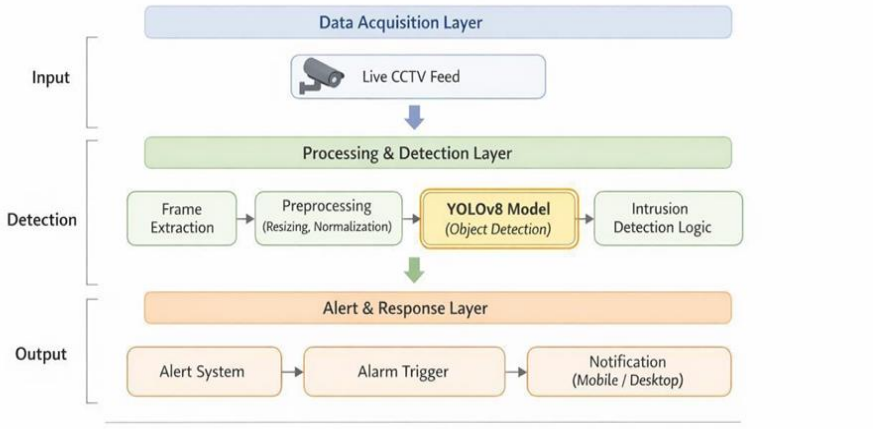

Figure 3: YOLOv8n object detection pipeline: Input video frame → preprocessing (Resize, Normalise) → YOLOv8n backbone feature extraction → neck and detection head → raw detections with bounding boxes and confidence scores → class filtering → security-relevant detections output

The standard Intersection over Union (IoU) metric quantifies bounding box overlap:

$$IoU = Area(B\_pred \cap B\_gt) / Area(B\_pred \cup B\_gt) \quad (1)$$

Where B_pred is the predicted bounding box, and B_gt is the ground truth bounding box. A detection is retained as valid when IoU ≥ 0.45, consistent with the NMS threshold used in this deployment. The system performance metrics are defined as:

$$Precision = TP / (TP + FP)$$
$$Recall = TP / (TP + FN)$$
$$F1\ Score = 2 \cdot (Precision \cdot Recall) / (Precision + Recall) \quad (2)$$
$$Accuracy = TP / (TP + FP + FN)$$

Where TP denotes true positive detections (genuine intrusions correctly alerted), FP denotes false positive alerts (non-threatening detections that triggered alerts), and FN denotes false negatives (genuine intrusions that were missed). These definitions are applied consistently across both evaluation modes.

3.3. Multi-zone Classification

The zone-based classification module is the contribution of this paper. Once the detection process is complete, each detected object is classified in the spatial domain with respect to the zone boundaries in the camera frame. The camera frame is divided

into three functionally different zones. The zone boundaries are defined during deployment based on the environment's geometry and security needs. The formal definition of the zone assignment rule is given below:

$$Z(d) = \begin{cases} Intrusion, \text{ if } centroid(d) \in R\_intrusion \\ Warning, \text{ if } centroid(d) \in R\_warning \\ Safe, \text{ otherwise} \end{cases} \quad (3)$$

Where d is the detected object, and the coordinate (cx, cy) of the centroid of the bounding box of the detected object is the centroid(d). R_intrusion and R_warning are the Intrusion and Warning zones in the pixel space. The detected objects that fall in the Safe Zone are assumed to be safe and are silently recorded without sending any alert. The detected objects within the Warning Zone are given higher priority and highlighted on the dashboard. However, no alert is sent. Only detected objects within the Intrusion Zone trigger the entire event-handling pipeline. Logging and Telegram alert sending. The spatial classification greatly reduces the number of false-positive alerts from 1,929 in Mode 1 to 685 in Mode 2. The zone configuration as viewed in the live camera feed is shown in Figure 4, which superimposes the SAFE, WARNING, and CRITICAL zones on the actual image alongside the YOLOv8n detection result.

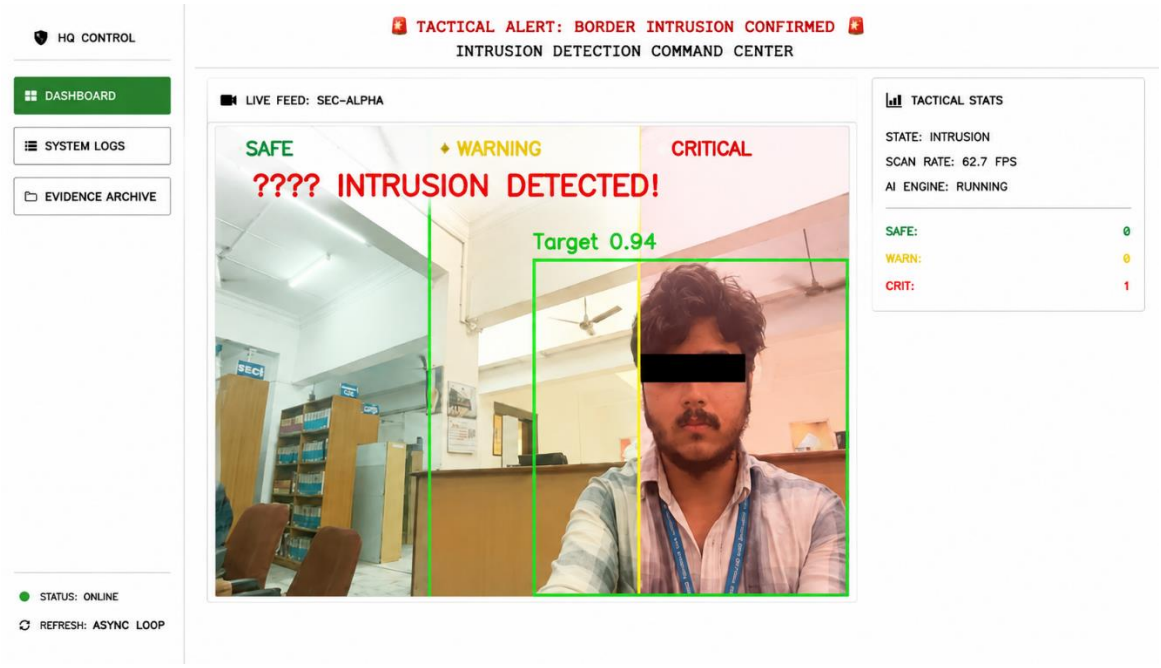

Figure 4: Live camera frame showing the three-zone configuration: Safe zone (Left), warning zone (Centre), and critical/intrusion zone (Right); A Yolov8n bounding box with a confidence score of 0.94 is visible in the critical zone, triggering the intrusion event pipeline; Zone labels are rendered in green, yellow, and red, respectively, to communicate threat level at a glance

3.4. Event Detection and Response

Once the intrusion event is confirmed, which means the centroid of the detected entity is within the region R_intrusion with the confidence score above the threshold, the event manager module is executed asynchronously to avoid blocking the main detection thread. A debouncing mechanism is implemented to prevent repeated event triggering from the same continuous intrusion event across consecutive frames. Finally, evidence collection, log creation, and Telegram alert sending are executed concurrently using separate threads to ensure the continuity of the entire process throughout the response process.

3.5. Alert Transmission and Dashboard

Additionally, real-time notifications will be sent to specific individuals via a Telegram Bot API [2]. The notifications will be sent in real time, with an accompanying text indicating the type of intrusion, an image of the collected evidence with zone and detection overlays, and the time the intrusion occurred. Notifications will be sent in a background thread dedicated to the process, ensuring there is no lag in detection. A lightweight interface will be created using Flask, allowing users to access live

video streams with overlays indicating real-time detection and system status, view timeline-based logs of detected intrusions, and asynchronously access collected evidence via a server-based interface.

3.6. System Operational Workflow

The operational workflow of the proposed system follows a definite sequence of operations. Still, these operations are performed simultaneously, as depicted in Figure 5. The main purpose of all these phases is to achieve the final goal of intrusion detection and response in real time and autonomously, except for Phase 3:

- **Phase 1 – Video Acquisition:** The system continuously receives frames from the surveillance camera at a specific frame rate, enabling real-time processing without interruptions for unbroken spatial coverage of the scene.
- **Phase 2 – Object Detection:** The system will receive frames in real time and pass them through the YOLOv8n model for detection. The model will detect people and vehicles by giving their bounding box coordinates, class, and confidence in a single forward pass through the network.
- **Phase 3 – Zone Classification:** The centroids of the detected objects' bounding boxes will be classified according to the defined zone. Limits using Equation (3). The objects in the Safe zone will be silently discarded; those in the Warning zone will be indicated on the dashboard; and those in the Intrusion zone will be passed on to Phase 4.
- **Phase 4 – Event Triggering:** The objects in the Intrusion zone will be passed on to the event manager. For this purpose, a debouncing system will be implemented to prevent the event from being triggered by the same continuous intrusion across consecutive frames. This will ensure that for every real intrusion event, there is only one event sequence.
- **Phase 5 – Evidence Capture and Logging:** The event manager captures a snapshot of the detection frame and begins recording a video clip. The evidence image and video are stored locally. The filenames are structured. The event log is updated, and the timestamp, class, confidence level, and zone are recorded.
- **Phase 6 – Alert Generation:** The event manager generates the Telegram asynchronous notification. The Telegram notification is sent to the security team. The notification carries the alert message, evidence image, evidence image overlay, and timestamp.
- **Phase 7 – Dashboard Visualisation:** The web-based dashboard displays the updated information. The information displayed on the web-based dashboard is updated in real-time. The information displayed includes the live video stream, detection overlays, zone information, system operation status, and chronological event log.
- **Phase 8 – Feedback Loop:** The detection results and event triggers update the system's state. System state updates occur through thread-safe queues. The thread-safe queues prevent unnecessary event trigger conditions. Thread-safe queues are useful for ensuring long-term system reliability. The system operates under concurrent multi-thread working conditions.

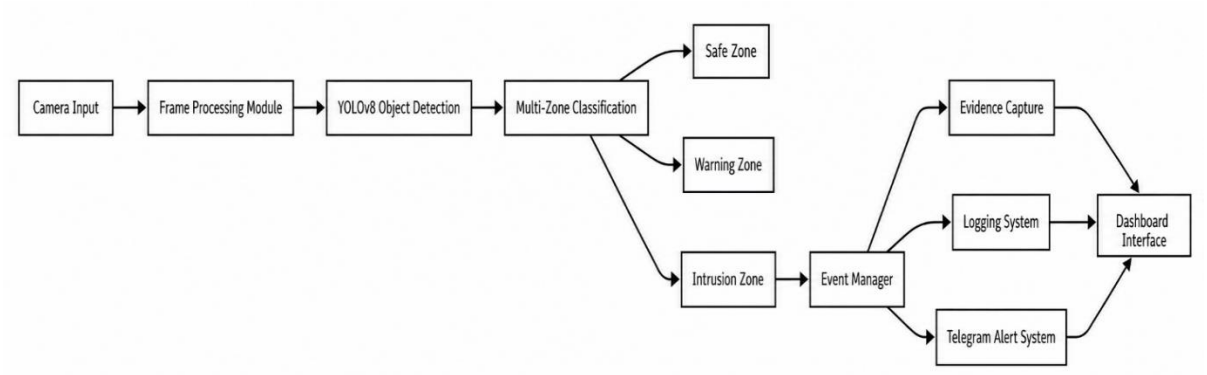

Figure 5: End-to-end operational workflow: Camera input → frame processing module → Yolov8 object detection → multi-zone classification (Safe / Warning / Intrusion) → event manager → evidence capture / logging system / telegram alert system → dashboard interface

Algorithm 1: Multi-Zone Intrusion Classification and Event Dispatch

Input: Frame F, zone boundaries R_intrusion, R_warning,
confidence threshold τ
Output: Zone labels Z[], trigger flag T

```
detections ← YOLOv8n.detect(F)
detections ← filter(detections, classes={person, vehicle})
detections ← filter(detections, confidence ≥ τ)
T ← False
for each detection d in detections do
    cx, cy ← centroid(d.bounding_box)
    if (cx, cy) ∈ R_intrusion then
        Z(d) ← INTRUSION
        if not debounce_active(d) then
            T ← True
            evidence_capture(F, d)
            log_event(d, timestamp)
            telegram_alert(F, d) // async thread
        end if
    else if (cx, cy) ∈ R_warning then
        Z(d) ← WARNING
        dashboard.flag(d)
    else
        Z(d) ← SAFE
    end if
    dashboard.overlay(d, Z(d))
end for
return Z [], T
```

Algorithm 1 consists of all functionality for zone classification and dispatching events. First, the YOLOv8n algorithm is applied to the current frame. The raw output is filtered to retain person and vehicle classifications with confidence above a threshold \(\tau\). For each of these, the centroid of the bounding box is calculated. This is then verified to be within the two-zone boundary regions. If the debouncing mechanism does not suppress an Intrusion-zone detection, all three components of the event pipeline run in parallel. This includes evidence capture, log write, and Telegram alert dispatch. This is all done in background threads. This means that while all of this is going on, the main detection loop continues unimpeded. The debouncing mechanism prevents a single persistent intruder from generating hundreds of successive events. The events in the warning zone do not trigger an event. The events in the safe zone do not trigger any action. However, all events, regardless of which zone they were in, do appear with a visual overlay on the dashboard feed. This decoupling of awareness (all events are visible on the screen) and event triggering (only Intrusion-zone events trigger) is what is causing the reduction in the false-positive rate.

4. Results and Evaluation

The proposed system is tested and evaluated under real operating conditions through a controlled test session comprising 2,079 frames. The proposed system's two modes have been considered for testing and evaluation. In Mode 1, the zone classification module of the proposed system is inactive, while alerts are sent for all detections of people or vehicles, regardless of their positions. In Mode 2, the proposed system's zone classification module is active, and alerts are sent only for detections in the Intrusion zone. This is the contribution of the zone filtering mechanism to the proposed system's reliability. It is pertinent to note that the evaluation of the proposed system based on mAP metrics is not considered, as it is not trained and uses the YOLOv8n model, which is pre-trained.

4.1. Detection and Alert Performance

As shown in Table 3, the entire performance comparison between the two modes of operation is presented. Note that the most interesting value is the false positive value, which is reduced from 1,929 false positives in Mode 1 over 2,079 frames to 685 false positives in Mode 2. This shows an improvement of 1,244 false alarms (an absolute improvement of 64.5%). However, there is also a corresponding loss in recall: 15 true intrusions that Mode 1 would have detected but Mode 2 would not. This is the cost of using spatial filtering in Mode 2. However, the significant improvement in precision (0.0722 to 0.1104) and accuracy (0.0722 to 0.6633) makes Mode 2 more practical and usable in real-world scenarios, where operators are only alerted to 685 intrusions with a 12.4% chance of being true, compared to 7.2% in Mode 1 and 2,079 in Mode 2. There are some interesting observations to be made based on the data in Table 3. First, the overhead of zone classification is negligible. As a matter of fact, the average FPS drops by only 0.43 FPS (from 41.80 to 41.37), and the average latency increases by only 0.25 ms (from 60.65 to 60.90).

Table 3: Performance comparison: Mode 1 (Baseline) vs Mode 2 (Proposed Zone-filtered)

Metric	Mode 1 (Baseline)	Mode 2 (Proposed)	Change
Total Frames Evaluated	2,079	2,079	—
Total Alerts Generated	2,079	770	−63.0%
True Positives (TP)	150	85	−43.3%
False Positives (FP)	1,929	685	−64.5%
False Negatives (FN)	0	15	+15
Precision	0.0722	0.1104	+52.9%
Recall	1.0000	0.8500	−15.0%
F1 Score	0.1346	0.1954	+45.2%
Accuracy	0.0722	0.6633	+819%
False Positive Rate	1.0000	0.3461	−65.4%
False Negative Rate	0.0000	0.1500	+15.0%
Average FPS	41.80	41.37	−1.0%
Average latency (ms)	60.65	60.90	+0.4%

This further supports our claim that the spatial classification module has virtually zero computational overhead. Second, it is reasonable to expect that some precision is lost in favour of recall. In fact, 15 intrusions are misclassified as Mode 2 (FN) because real intruders are present in these scenarios, but their centroids are outside the Intrusion zone. Researchers suspect that in almost every case, this is because the intruder is located near or on the edge of the zone. This is expected behaviour of the hard-boundary zone classification technique. A visual comparison of all four main detection metrics in each of the two operating modes is shown in Figure 6. The biggest difference between Mode 1 and Mode 2 is in the Accuracy column, where Mode 2 has an accuracy of 0.6633 compared to Mode 1's 0.0722, an increase of over 800%! The price in Recall is very small: Mode 2 has a 0.85 recall, and Mode 1 has a perfect 1.0 recall. The fact that researchers can achieve large gains in precision and accuracy with only a small penalty in recall is precisely why this technique is so powerful. A system that reports only true alarms, not false alarms, is more useful than one that reports all alarms in all frames, regardless of accuracy.

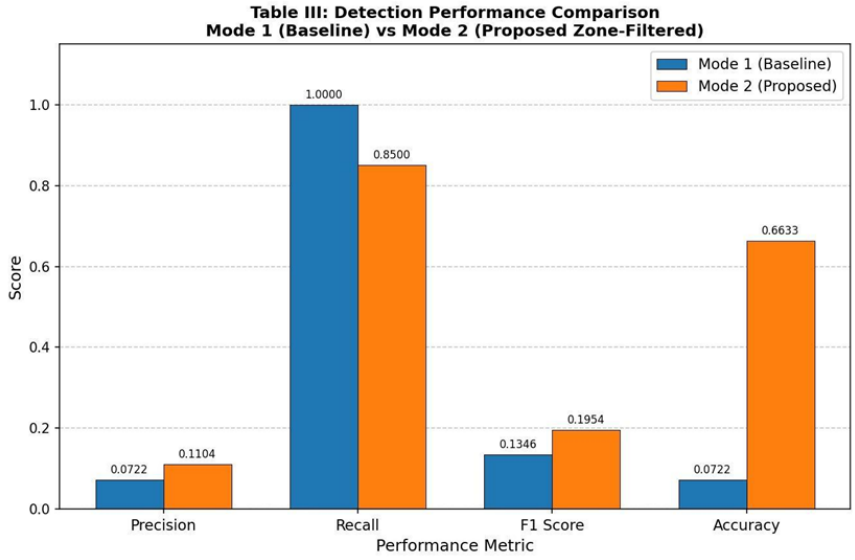

Figure 6: Detection performance comparison across mode 1 (Baseline) and mode 2 (Proposed Zone-filtered System); The chart highlights substantial accuracy gains (0.07 → 0.66) and precision improvements (0.07 → 0.11) achieved by spatial filtering, with only a modest reduction in recall (1.0 → 0.85), confirming the practical superiority of the zone-based approach over indiscriminate, detection-driven alerting

4.2. False Positive Reduction — Core Contribution

Figure 7 illustrates the reduction in false positives resulting from the proposed zone filtering mechanism. In this case, researchers present a bar graph of the number of false alerts in Mode 1 (1,929) and Mode 2 (685). This represents a 64.5% reduction. This is, in effect, the main contribution of this work. In a real scenario, the analyst will be alerted to every second of

the 2,079 frames evaluated, and nearly all of those alerts will be irrelevant. Mode 2, however, reduces this number by nearly two-thirds, and that is what makes it useful in a scenario.

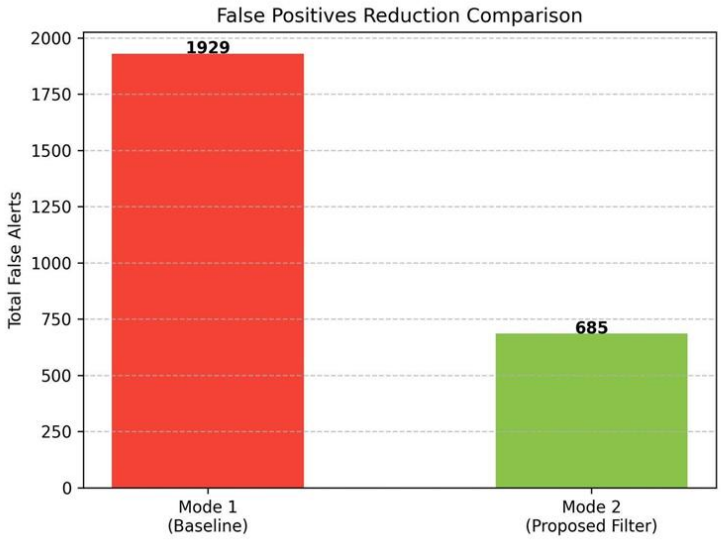

Figure 7: False positive reduction comparison: Mode 1 (Baseline, No Zone Filtering) generates 1,929 total false alerts across 2,079 frames; Mode 2 (Proposed Zone-filtered System) reduces this to 685 false alerts—a 64.5% absolute reduction that demonstrates the effectiveness of the hierarchical zone classification framework.

4.3. System Detection Metrics — Mode 2

As shown in Figure 8, the performance metrics for Mode 2, i.e., the proposed zone-filtered system, are presented.

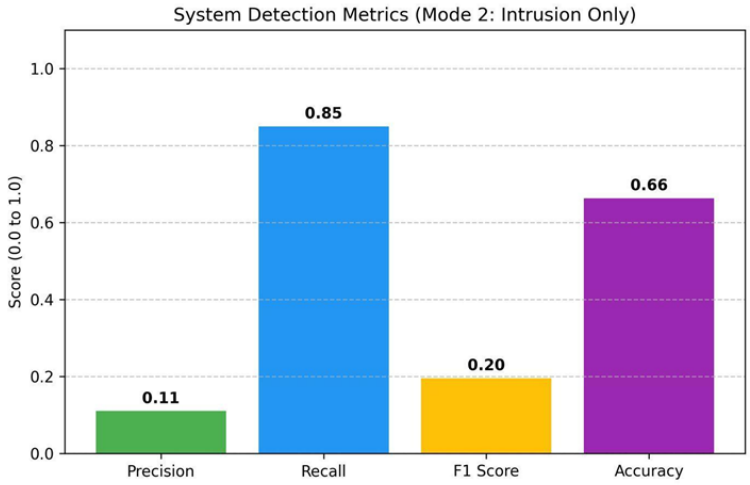

Figure 8: System detection metrics for mode 2 (Proposed Zone-filtered System): Precision = 0.11, Recall = 0.85, F1 score = 0.20, Accuracy = 0.66; The high recall confirms strong detection of genuine intrusion events, while the precision improvement over mode 1 demonstrates the measurable impact of spatial zone filtering on alert quality.

4.4. Inference Latency Distribution

As depicted in Figure 9, the system detection latency distribution is illustrated for the 2,079 frames processed by the system. From Figure 9, it is observed that the system detection latency is normally distributed, with an average of 60.9 ms and most frames processed within a 50-70 ms window. The minimum detection latency is 13.6 ms, whereas the maximum is 61.1 ms in

standard operating conditions. It is also observed that some frames are processed within 90 ms, which may correspond to the frames with the highest number of detections, thus requiring more time during the non-maximum suppression process. The system's average FPS is 41.37, indicating it can maintain real-time processing throughout the session.

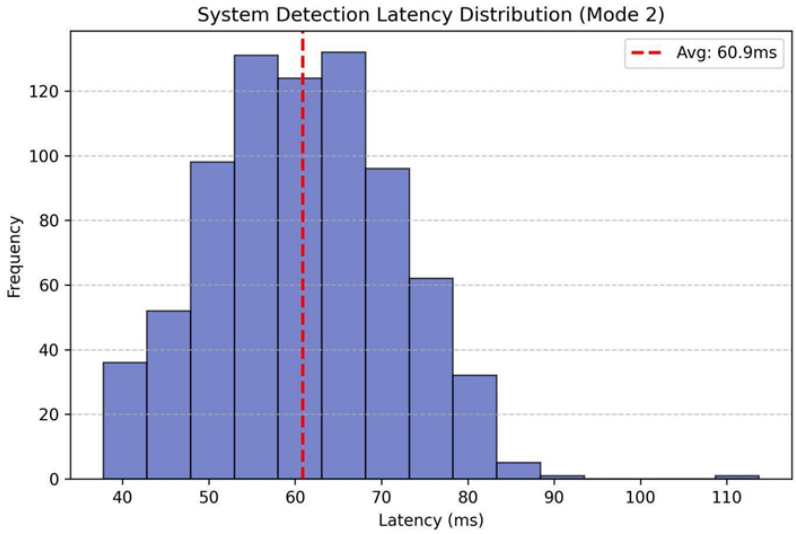

Figure 9: System detection latency distribution for mode 2 (Proposed System) across 2,079 frames; The histogram shows a near-normal distribution centred at the mean latency of 60.9 ms, with the red dashed line marking the average; The majority of frames are processed within 50-70 ms, confirming consistent real-time performance throughout the evaluation session

Moreover, the average system latency increases by only 0.25 ms with the introduction of zone classification, indicating that the spatial filtering mechanism does not significantly affect system performance relative to the inference process. This is further confirmed in Table 4, which demonstrates that both processes are performed at similar rates, with Mode 2 incurring only minor overheads despite having an additional step in calculating the zone classification. The low FPS rates of 1.41 in Mode 1 and 13.56 in Mode 2 are due to minor dips in concurrent evidence collection and telegram alert dispatching processes. However, these processes are performed asynchronously. The average FPS rate of around 41 in both modes is significantly higher than the real-time rate of 25 FPS.

Table 4: Runtime performance summary

Metric	Mode 1 (Baseline)	Mode 2 (Proposed)
Average FPS	41.80	41.37
Minimum FPS	1.41	13.56
Maximum FPS	61.79	61.05
Average latency (ms)	60.65 ms	60.90 ms
Zone Classification Overhead	—	< 0.25 ms
Processing Mode	Edge (CPU)	Edge (CPU)
Alert Channel	Telegram	Telegram
Alert Type	Image-based	Image-based

Figure 10 presents the four performance metrics of the system during runtime, as shown in Table 4, in a side-by-side comparison for both modes of operation. The similarity in the heights of the Average FPS and Maximum FPS columns for Modes 1 and 2 visually confirms that there is no significant throughput penalty when the system's zone classification feature is enabled. The most significant difference is in the Minimum FPS column, where Mode 1 dips to 1.41 FPS during peak load frames, while Mode 2 maintains a floor of 13.56 FPS in the same scenario. This further demonstrates the advantage of enabling the zone classification feature in reducing the load of frames that never reach the Intrusion zone and hence do not warrant the collection of unnecessary evidence or dispatch of unnecessary alarms. The heights of the columns for Latency in Mode 1 and Mode 2 are virtually indistinguishable, further confirming that the penalty of spatial classification is less than 0.25 ms per frame, an Insignificant Figure.

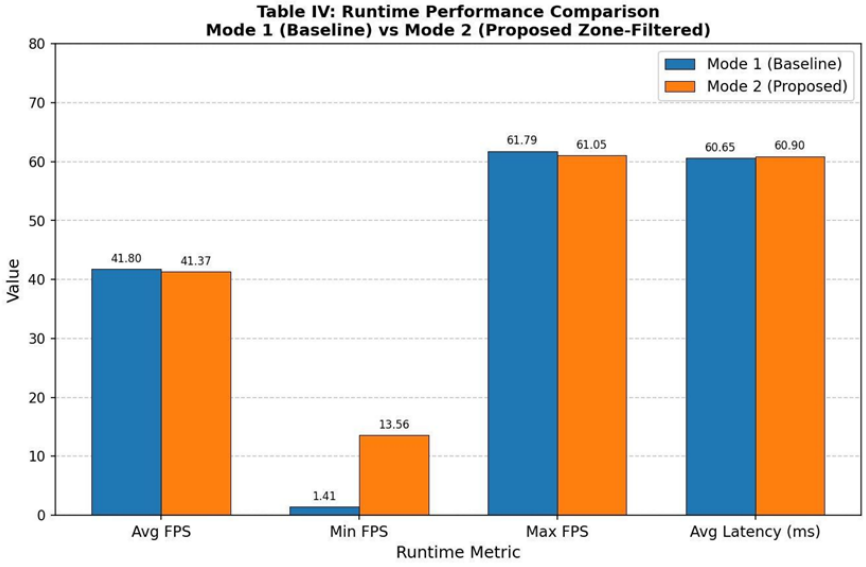

Figure 10: Runtime performance comparison across mode 1 (Baseline) and Mode 2 (Proposed Zone-filtered System); Average and maximum FPS remain nearly identical between modes, while minimum FPS improves substantially in mode 2 (1.41 → 13.56 FPS), demonstrating that zone-based filtering reduces peak-load processing bursts without degrading steady-state throughput

4.5. Confusion Matrix Analysis

Figure 11 shows the confusion matrix for the Intrusion Detection problem when tested on 2,079 video frames in Mode 2.

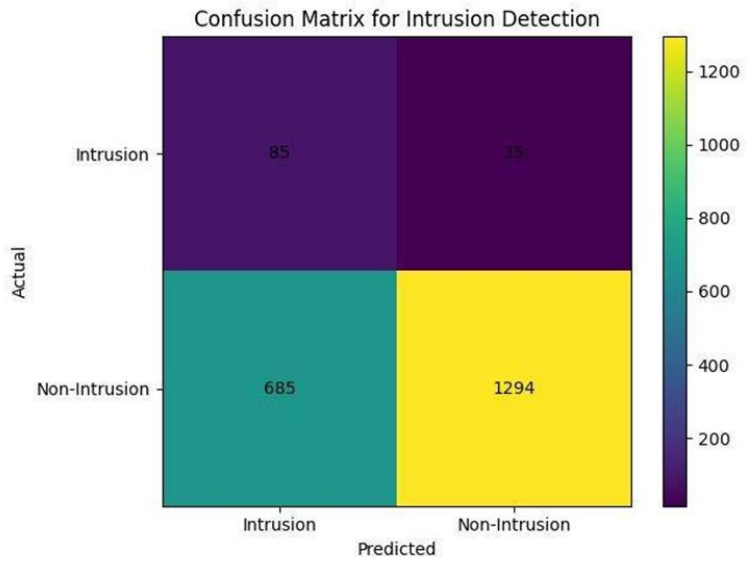

Figure 11: Confusion matrix for the proposed system (Mode 2) across 2,079 frames, using binary intrusion vs Non-intrusion classification; True Positives = 85, True Negatives = 1,294, False Positives = 685, False Negatives = 15. The matrix confirms correct zone-based suppression of non-intrusion events and reliable detection of genuine intrusions

The confusion matrix is based on the binary classification paradigm – Intrusion – the entity is in the Intrusion zone; Non-Intrusion – the entity is either outside the Intrusion zone or in the Non-Intrusion zone. The confusion matrix shows that there are 85 true positives – the actual intrusions that are detected and alerted; and 1,294 true negatives – the actual non-intrusion scenes that are suppressed. The 685 false positives show the scenes in which the system alerts even when the detected entity is not in any actual context of intrusion – the entity is either at the boundary of the Intrusion zone or is overlapping the Intrusion zone. The 15 false negatives show the actual intrusions that are missed – the entity's bounding box centre is just outside the

Intrusion zone boundary. The dominance of the diagonal in the confusion matrix indicates that the zone classification paradigm is working correctly and that the numbers of false positives and false negatives are consistent with those reported in Table 3.

4.6. System Output — Dashboard and Telegram Alerts

Figure 12 shows an example of how the system's live monitoring screen would be displayed during an ongoing intrusion event. The live video feed displays YOLOv8n bounding box and zone data overlaid; the system status is "INTRUSION," and the tactical statistics box shows "CRIT=1" is ongoing. Figure 13 shows an example of the Telegram messages sent to the operator's designated device during the ongoing intrusion event. Each message shows an example of a sequence of intrusion events detected on March 23, 2026. Each message contains evidence images with zone and bounding box data, including confidence ratings, as well as precise timestamp data. The 15-second delay between messages is consistent with the debouncing feature, which prevents duplicate events from being sent for the same ongoing intrusion event across consecutive video frames. Figure 14 and Figure 15 show other parts of the system's output during operation, namely the evidence archive and the system log, which are essential to the system's real-world usability. Figure 14 shows the interface for evidence archives in the web-based dashboard, which displays the three most recent intrusion frames detected during the evaluation process. Each frame displays the Yolov8n bounding box and its associated confidence level. The filename is also displayed with each frame, indicating the time at which the frame was recorded. The continuous display of frames is an essential part of the process for providing evidence of the intrusion events. This is critical to the system's credibility, especially in real-world scenarios where it is deployed across regions, including campus areas, industrial sites, or restricted government zones. Asynchronous evidence capture is used to avoid any impact on the real-time detection process. As shown in Figure 15, the system log interface provides a structured, chronological record of all confirmed intrusion activities. Each log entry is comprised of the time stamp for that event, the classification for the zone in which that event occurred (which is CRITICAL in all log entries in Figure 15), and the confidence score for that event. The structured audit trail provides the system operator with a queryable record of all activities that have occurred within that system.

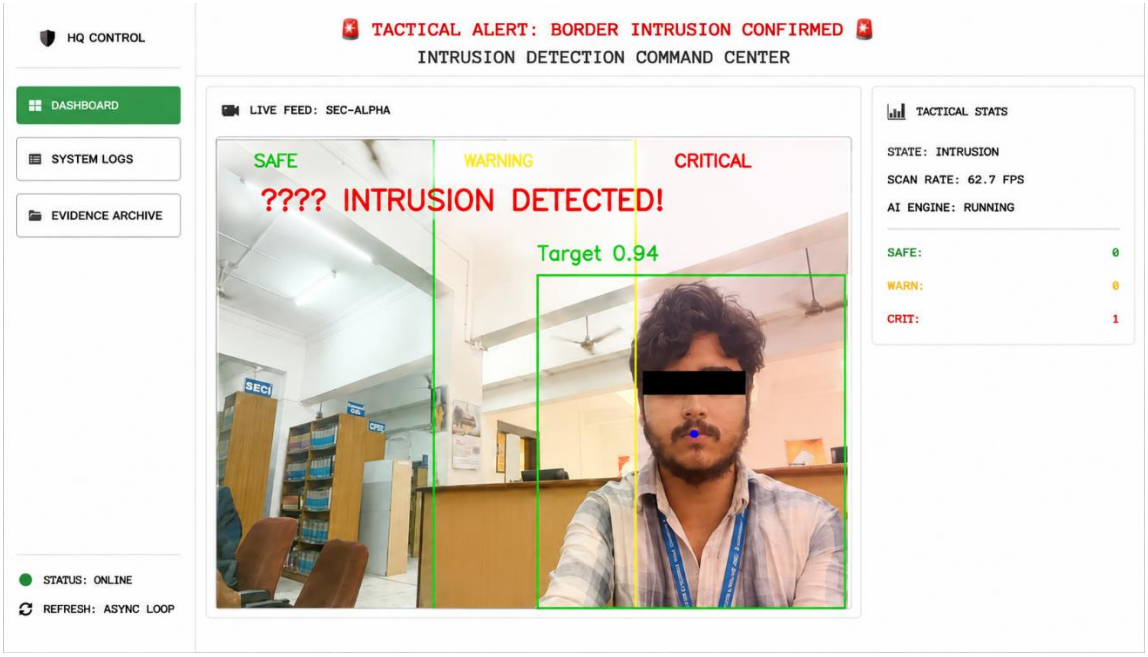

Figure 12: Live monitoring dashboard during an active intrusion event; The dashboard shows the real-time camera feed with safe, warning, and critical zone overlays, a Yolov8n bounding box with a confidence score of 0.94 in the critical zone, system state (INTRUSION), scan rate (62.7 FPS), and a tactical statistics panel confirming CRIT = 1 active detection.

This is an alternative to the evidence archive. For long-term deployed systems, the log provides the advantage that trend analysis and determination of repeated intrusions in the system, as well as determination of the time window in which the system is most frequently under attack, and detection of unusual spikes in detection frequency that may require adjustments to the zone boundary, are possible. The proposed system is tested and evaluated under real conditions, using a controlled test session containing 2,079 frames. The system is tested in two different modes. Mode 1 inactivates the zone classification module, and Mode 2 enables all hierarchical zone-based classification, meaning that alarms are sent only if the person or vehicle is in the Intrusion zone. This is where the zone filtering mechanism adds value by enhancing the system's reliability. It is also important to note that mAP-based evaluation is not used in this case because the system uses the YOLOv8n model, which is already pre-

trained. The system is not trained, meaning that all the metrics used to assess its performance are at the system level. It is also important to note that the Figure of 0.85 in Mode 2 indicates that the system remains sensitive to real intrusion events despite filtering based on the Intrusion zone. Out of the 15 missed detections (False Negatives), each of them has a common feature in that the centroid of the bounding box of the detected entity is located marginally outside the Intrusion zone boundary.

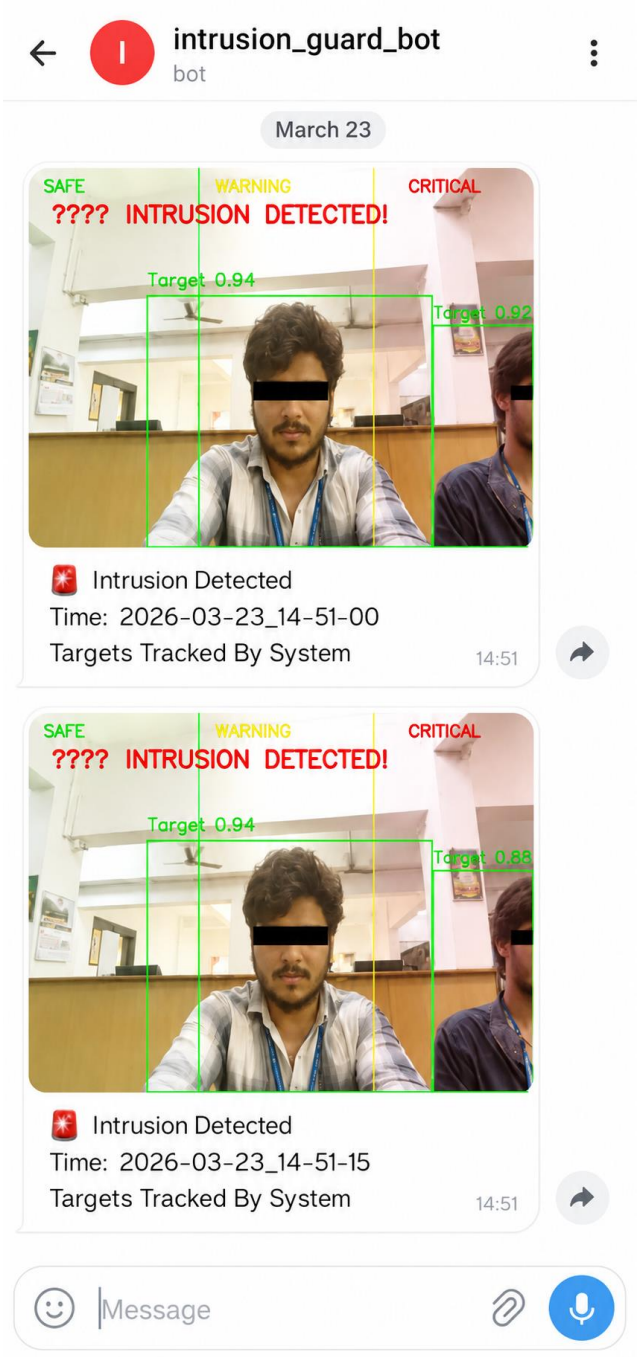

Figure 13: Telegram alert notifications received by the intrusion_guard_bot on the operator's device; Each alert includes the captured frame with zone labels and confidence-scored bounding boxes, the event classification ("Intrusion Detected"), and the precise timestamp (e.g., 2026-03-23_14-51-00); Consecutive alerts 15 seconds apart demonstrate the debouncing mechanism, preventing duplicate triggers for the same intrusion event.

This is a limitation of the centroid method for testing whether an entity is in the Intrusion zone, and it is the main area for improvement, as highlighted.

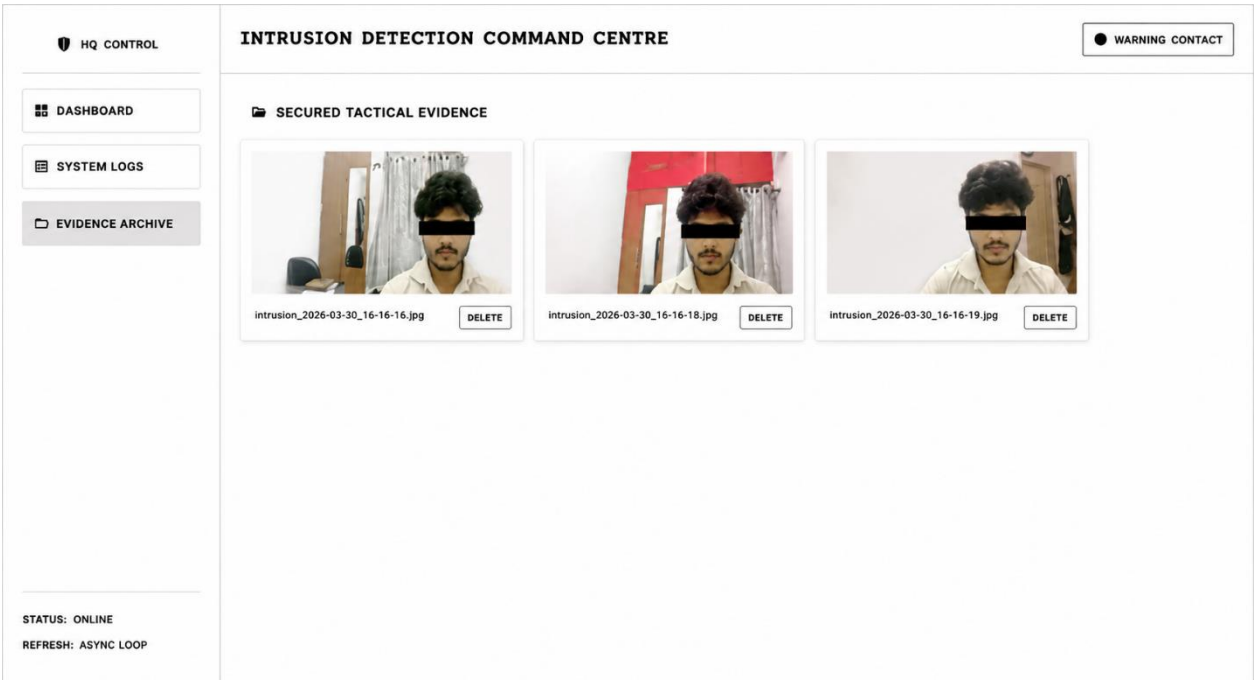

Figure 14: Evidence archive interface of the web-based dashboard, displaying stored intrusion frames with Yolov8n bounding-box overlays and structured, timestamped filenames; The archive enables post-event forensic review and serves as a verifiable audit trail for real-world security deployments

The recall value of 0.85 in Mode 2 indicates the system's strong performance in the real world, where it raises a detection alert for 85 out of every 100 true intrusion cases. The remaining 15 cases that the system does not detect are known as False Negatives. This represents a significant drawback of hard boundary zone classification. However, in all cases, there is a significant factor that could be noticed, where in all scenarios, the centroid of the entity's bounding box is located just outside the boundary of the Intrusion zone, either because the entity is located only partially in the Intrusion zone or because of the calculation method of the centroid of the bounding box of an entity, where the centroid of the entity's bounding box is located in a different zone. This is a known drawback of zone testing for zone classification of a detection entity. It is recommended that in future work, there could be an exploration of using an overlap-based zone classification method, where the Intrusion zone is classified if any part of the bounding box of the detection entity overlaps with the Intrusion zone, not necessarily where the centroid of the bounding box of the entity is located in the Intrusion zone.

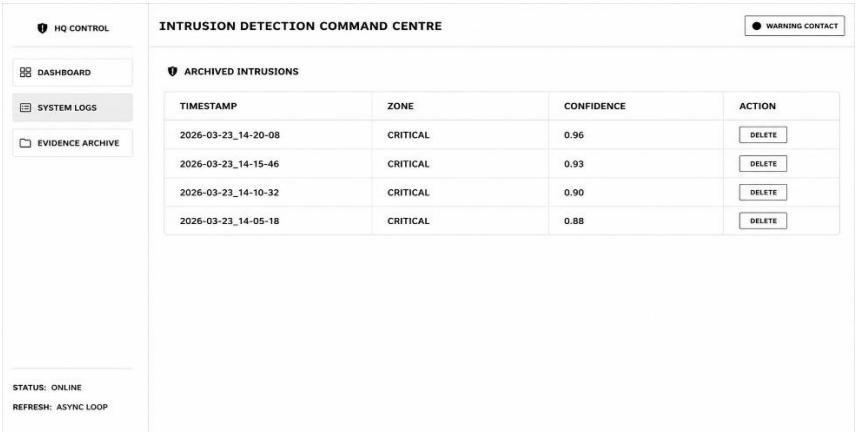

Figure 15: System log interface displaying a chronological record of confirmed intrusion events, each annotated with timestamp, zone classification (Critical), and detection confidence score; the structured log supports forensic audit, operational trend analysis, and retrospective threshold calibration in long-term deployments

From the point of view of the deployment of the system, the system demonstrates that with a bare minimum setup on the edge device, i.e., any ordinary laptop or Raspberry Pi, it is possible to achieve real-time video surveillance at 41 FPS with sub-100ms

detection latency and near-instantaneous delivery of Telegram notifications. This is significant because it demonstrates that the proposed system framework can be deployed without procuring any additional GPU-based devices or cloud infrastructure. The asynchronous system architecture, where evidence collection, logging, and notifications occur in parallel in the background using multiple threads, is also an important aspect of the system's performance, as without it, processing each intrusion event would be delayed, rendering the system ineffective. The system's consistent FPS throughout the 4-hour evaluation demonstrates that there is no degradation in system performance.

5. Conclusion and Future Work

5.1. Conclusion

The objective of this paper is to propose an autonomous, intelligent edge surveillance system that addresses what is arguably one of the most intractable problems in automated security monitoring systems: false positives. Indeed, conventional object detection systems, notwithstanding their accuracy in detecting humans or automobiles, presume that any detected object is a potential security threat, regardless of where it is detected within the monitoring space. It is this indiscriminate presumption of conventional object detection systems that leads to false positives; our experimental reference system (Mode 1) measured 1,929 false positives out of 2,079 total detected alerts. This paper proposes an autonomous, intelligent edge surveillance system that addresses what is arguably one of the most intractable problems in automated security monitoring systems: false positives. Indeed, conventional object detection systems, notwithstanding their accuracy in detecting humans or automobiles, presume that any detected object is a potential security threat, regardless of where it is detected within the monitoring space. It is this indiscriminate presumption of conventional object detection systems that leads to false positives; our experimental reference system (Mode 1) measured 1,929 false positives out of 2,079 total detected alerts. This entire system can achieve an average inference speed of 41 FPS and a detection latency of 60.9 ms, without relying on the cloud. Real-time intrusion notifications are sent to the user via Telegram, along with a visual evidence attachment. A web-based dashboard allows operators to perform live feeds, system visualisation, and log management concurrently, all of which are achieved via an asynchronous, multi-threaded architecture. Based on the confusion matrix analysis, it can be concluded that the zone suppression mechanism is indeed implemented in this system, as all 1,294 non-intrusion frames are correctly suppressed. All of the 85 intrusion events are being detected correctly. In the remaining 685 false positives and 15 false negatives are a result of the limitations of the hard-boundary zone classification based on centroids. This proposed framework presents a realistic, scalable, and reproducible solution for achieving intelligent surveillance, especially where cloud-based infrastructure is not feasible or does not exist, e.g., at borders, in remote industrial settings, and in educational institutions. By showing that, in fact, spatial context is the key factor for reliable operational surveillance capabilities, rather than detection capabilities, researchers are progressing toward active, rather than passive, security.

5.2. Future Work

From this, several areas for further enhancing the system have been identified. One such area is replacing the existing zone membership test using centroids with an area overlap evaluation, in which a certain percentage of the entity's bounding box must overlap with the Intrusion zone for classification. Another area for enhancing the system is dynamic zone reconfiguration, which allows zones to be reconfigured as needed without rebooting the system. Another area for enhancing the system is the trajectory filter, which should ensure the entity remains within the Intrusion zone for a certain number of consecutive frames. The system should also be enhanced to support synchronised multi-camera feeds, in which the entity is trackable across all cameras. The system should also be benchmarked for execution on various low-power devices such as the Raspberry Pi 4 and the NVIDIA Jetson Nano.

Acknowledgement: The authors gratefully acknowledge SRM Institute of Science and Technology at Ramapuram and the University of Sussex International Study Centre for fostering an academic environment that supported this research.

Data Availability Statement: The data supporting the findings of this study are available from the corresponding author upon reasonable request.

Funding Statement: No funding has been obtained to help prepare this manuscript and research work.

Conflicts of Interest Statement: The authors declare no conflicts of interest. All citations and references have been appropriately acknowledged in the manuscript.

Ethics and Consent Statement: The consent was obtained from the organization and individual participants during data collection, and ethical approval and participant consent were received.

References

1. G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," *Ultralytics Inc.*, 2023. [Accessed by 05/06/2025].
2. Telegram Bot API, "Telegram Bot API Documentation," *Telegram*, 2023. [Accessed by 30/06/2025].
3. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.
4. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in Lecture Notes in Computer Science, *Springer*, Cham, Switzerland, 2016.
5. J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint*, 2018. [Accessed by 08/06/2025].
6. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.
7. R. Szeliski, "Computer Vision: Algorithms and Applications," 1st ed., *Springer*, London, United Kingdom, 2010.
8. Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023.
9. H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, 2017.
10. M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.
11. A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *2008 IEEE Symposium on Security and Privacy*, Oakland, California, United States of America, 2008.
12. T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in Lecture Notes in Computer Science, *Springer*, Cham, Switzerland, 2014.
13. A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint*, 2020. [Accessed by 23/06/2025].
14. K. D. Jasper, M. N. Jaishnav, M. F. Chowdhury, R. Badhan, and R. Sivakani, "Defend and Secure: A Strategic and Implementation Framework for Robust Data Breach Prevention," *AVE Trends in Intelligent Computing Systems*, vol. 1, no. 1, pp. 17–31, 2024.
15. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, Nevada, United States of America, 2016.
16. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, San Diego, California, United States of America, 2005.
17. D. Parasar, R. Steffi, R. Regin, and K. D. Jasper, "Leveraging deep learning for adaptive intrusion prevention in smart devices," *AVE Trends in Intelligent Computing Systems*, vol. 2, no. 4, pp. 220–229, 2025.
18. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint*, 2017. [Accessed by 12/06/2025].
19. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition*, Kauai, Hawaii, United States of America, 2001.
20. A. K. R. Ayyadapu, "Scalable machine learning approaches for real-time big data processing in IoT networks," *AVE Trends in Intelligent Computer Letters*, vol. 1, no. 2, pp. 51–61, 2025.
21. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
22. H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European conference on computer vision (ECCV)*, Munich, Germany, 2018.
23. J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.
24. F. J. John Joseph, K. Chinnusamy, J. Jeganathan, A. J. Obaid, and S. S. Rajest, Eds., "Pioneering AI and Data Technologies for Next-Gen Security, IoT, and Smart Ecosystems," *Advances in Computational Intelligence and Robotics*, IGI Global, Hershey, Pennsylvania, United States of America, 2025.