International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM04175
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

AI-Powered Video Surveillance for Enhanced Object Detection and Incident Monitoring Using YOLOv8

Shreyas Ghansawant[1], Atharva Thokal[2], Tanishq Ladde[3], Prof. Nikita Khawase[4]

[1]*Shreyas Ghansawant, Department of AI & DS - ISBM College of Engineering, Pune*

[2] *Atharva Thokal, Department of AI & DS - ISBM College of Engineering, Pune*

[3] *Tanishq Ladde, Department of AI & DS - ISBM College of Engineering, Pune*

[4] *Prof. Nikita Khawase, Department of AI & DS - ISBM College of Engineering, Pune*

Abstract - Timely and accurate detection of safety-critical incidents, such as vehicle accidents and human falls, is crucial for improving real-world surveillance-based emergency response. In this work, we investigate the use of YOLOv8-based object detection models, trained on small, domain-restricted datasets, for event detection. All variants of the model were deployed through a Flask-hosted web interface, allowing for extensive testing on static images as well as uploaded video streams. On a range of hardware platforms, the detectors showed well-balanced precision-recall performance with near real-time appropriateness of inference times. An added alerting module further enhances the pipeline, with an integrated solution for automated incident alerting. These results illustrate the viability of using light YOLOv8 models in efficient surveillance contexts in smart cities and industrial processes.

Key Words: YOLOv8, video surveillance, object detection, incident detection, vehicle collision detection, human fall detection, Flask web interface, real time monitoring, alerting, deep learning.

1.INTRODUCTION

1.1 Background & Motivation

In urban and industrial settings, the meteoric rise of surveillance cameras has generated a vast amount of video data which cannot be monitored by human observers with equal swiftness. Watching live feeds by humans is laborious, with fatigue-related errors causing many critical events to be missed, leaving responses to emergencies delayed. Convolutional neural networks and deep learning have truly turned the tables on automated object detection and real-time analysis of sophisticated scenes. The family of detectors known as "You Only Look Once" (YOLO) stands out for providing an excellent compromise between speed and accuracy when deployed in resource-constrained environments [1], [2].

1.2 Problem Statement

In spite of deep learning's promises for monitoring, many endeavors are still hindered when it comes to using large datasets, requiring extensive user annotation, and high-end hardware. Practical problems involving small-scale installations or legacy camera networks may suffer from data limitations and low compute power. Yet safe incident detection, of utmost priority is highly valuable: being accidents like car crashes, or human falls. It shall, therefore, take lightweight models trained over curated smaller datasets, and an end-to-end pipeline that allows image and video analysis and real-time alerting through an easy-to-use interface.

1.3 Contributions of This Work

In this paper, we address these challenges through the following key contributions:

Compact YOLOv8 Model Variants: We train multiple YOLOv8-based detectors on a carefully assembled car crash and fall dataset to yield reasonable detection performances with limited training samples.

Flask Powered Deployment: A web-based interface built using the Flask framework allows users to upload images or videos for immediate incident detection and monitor live camera streams with automated alerts.

Hardware Aware Evaluation: Inference latencies and detection consistencies of different computing platforms were evaluated to reveal the trade-offs between model size, dataset variety, and processing speed.

Integrated Alerting Mechanism: Real-time alerts during incident recognition by a fully implemented notification subsystem show the operational readiness of the system.

2. LITERATURE REVIEW/RELATED WORK

Over the last few years, the incorporation of deep-learning algorithms into video-surveillance systems has transformed traditional surveillance systems by enabling real-time, automated assessment of complex scenes. The pioneering research done by Redmon et al. on You Only Look Once (YOLO) developed a single, homogeneous framework for detecting objects that processes a complete image in a single pass, hence achieving an unrivaled speed-accuracy trade-off [1]. Building on this architecture, subsequent YOLO-based architectures have continued to increase inference throughputs, thereby making them particularly well-suited for latency-critical applications like incident monitoring.

High-diversity and high-quality datasets are needed to train effective detection models. The Microsoft COCO dataset, established by Lin et al., has dense object annotations in eighty typical categories and has established itself as a metric for general-purpose detection algorithms [2]. Although the large size and diversity of COCO contribute to state-of-the-art performance, real-world surveillance deployments typically have to deal with data availability constraints—specifically for rare or high-consequence safety events—so therefore require the application of data augmentation and transfer learning techniques for fitting large models to small domain-specific datasets.

Several research studies have explored automatic traffic crash detection in surveillance footage. Ghahremannezhad et al. demonstrated that end to end deep learning pipelines can identify vehicle crashes in real time with the possibility of significant reductions in emergency response time through continuous feed analysis [3]. Their work indicates the possibility of end to end learning approaches but also suggests the challenges of finding a trade-off between detection sensitivity and false alarms, especially in heterogeneous urban settings.

Supporting vehicle collision detection, fall detection by humans has been investigated as part of public safety surveillance. Kavya et al. utilized convolutional neural networks to analyze video streams and demonstrated promising advances in fall detection accuracy compared to heuristic based approaches [4]. The results highlight the significance of posture and movement patterns in distinguishing between benign activity and genuine emergency situations, guiding the development of more trustworthy fall alarm systems.

Cumulatively, these earlier initiatives define both methodological underpinnings and practical implications of deploying AI driven surveillance. Our work builds upon advance YOLOv8 architectures—trained on custom crash and fall datasets—and combines them in a single web based platform, overcoming data shortage, computational limitations, and requirements for instant alerting in real world scenarios.

3. SYSTEM ARCHITECTURE & IMPLEMENTATION

3.1 Overall Pipeline Design

The system we propose uses a modular, end-to-end pipeline (see Figure 1) with clearly defined processes for data intake, model inference, incident analysis, alert generation, and user interaction. All the components execute asynchronously—enabling concurrent capture and queuing of video frames in parallel with detection engine and alert subsystem processing—so as to achieve maximum throughput and responsiveness on different hardware configurations.

3.2 Video Capture and Early Processing

Raw video streams and user-uploaded videos are processed by the system using a centralized capture module. Frames are captured at the frame rate specified by the user and resized to 640×640 pixels to meet the input needs of YOLOv8. A light normalization step converts pixel values to [0,1] range, and data augmentation methods like random flipping and brightness adjustments can be enabled at inference time to improve robustness against harsh lighting or viewpoint changes.

3.3 YOLOv8 Model Integration

At the core of the system, one or more YOLOv8 model variants—each fine tuned on car crash and fall datasets—are loaded into memory via the PyTorch inference API. The detection engine batches incoming frames (batch size adjustable per GPU/CPU capacity) and performs a single forward pass per batch.Postprocessing carries out non maximum suppression (NMS) at an intersection over union threshold of 0.45 to remove redundant boxes, providing final bounding boxes, class IDs, and confidence scores per frame.

3.4 Incident Detection Logic

Detection results are passed on to the incident analysis module, which performs class specific logic:

Collision identification: Fires when two car bounding boxes overlap beyond a certain spatial threshold, or when a "crash" class confidence exceeds 0.6.

Fall detection: Combines fall class detections with temporal consistency checks—i.e., checking that a subject's bounding box is low and static for at least 0.8 seconds—to prevent spurious alarms.

This two-step decision process trades sensitivity against false alarms, adaptively setting thresholds over recent detection history.

3.5 Alerting & Logging Subsystem

Once an event is validated, the system stores the event timestamp, frame capture, and detection metadata in a persistent datastore. At the same time, the alerting engine creates a notification payload—class type, confidence, and thumbnail—and sends it via email, HTTP webhook, or in-page banner.Retry logic and queuing guarantee delivery in the face of transient network conditions.

3.6 Web Interface Based on Flask

A Flask application brings together the user experience by offering routes for:

Static test uploads: /upload/image and /upload/video endpoints will accept files and return annotated results.

Live stream: /stream gives a Jetson/GPU powered camera stream with overlay in real time.

Dashboard: A minimal frontend in HTML/JavaScript displays the latest warnings, system health gauges (frame rate, GPU/CPU use), and setup controls (confidences, batch size). The interface uses WebSockets for low-latency updates and Bootstrap for responsive design, allowing it to be executed on desktop or mobile devices without the need for extra client software.

3.7 Deployment and Scalability Considerations

The whole stack is containerized using Docker, and GPU acceleration is supported with the NVIDIA Container Toolkit where that is required. Horizontal scaling is done by running many detection workers behind an NGINX reverse proxy and a shared messaging queue (e.g., Redis). That is, the system can be horizontally scaled to support more cameras or higher throughput needs by simply adding more worker instances.

4. DATA PREPARATION & MODEL TRAINING

For efficient detection of human falls and car crashes, pre-annotated data was purchased from Roboflow, which is a common dataset curation and computer vision dataset management platform. The used datasets contained YOLO-format bounding box annotations, which are specifically optimized for fall and crash recognition tasks. Without manual annotation, direct integration into the training pipeline was possible, significantly lessening the task of model building.

Every dataset had unique features in size and composition, requiring a respective modification of the data partitioning approach. Based on the number and diversity of available images, the datasets were divided into training, validation, and testing sets with different split proportions—typically between 70:20:10 and 80:10:10—guided by the need to ensure proper

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM04175
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

representation in each set. This variable partitioning approach enabled reliable testing without sacrificing high learning quality, especially for relatively small datasets.

The images were resized to a resolution of 640×640 pixels to satisfy the input requirement of YOLOv8 models. Random horizontal flip, change of brightness, scaling, and rotation were performed at export and training time to improve the model's resistance to changes in scene conditions and camera views.

YOLOv8n (nano) was used for model training due to its lightweight architecture, offering a reasonable balance between speed and accuracy—appropriate for real-time use and deployment on low-end devices. Training was done using the Ultralytics YOLOv8 framework, with pretrained weights on the COCO dataset to achieve transfer learning and improve convergence rate. A few hyperparameters, such as learning rate, batch size, and epochs, were experimented in a conservative approach through a sequence of experiments looking to improve performance over a spectrum of datasets.

Throughout the training process, key metrics like classification loss, objectness score, precision, recall, and mean Average Precision (mAP) were monitored systematically to gauge the performance of the model.The ultimate model versions were chosen based on the performance on their validation set and generalization capability, such that each was well-suited to its corresponding detection task.

The presented methodology illustrated that it is possible to create effective and responsive detection models utilizing YOLOv8n, even when employing modest and diverse datasets, thereby endorsing its practical use in incident monitoring and video surveillance.

5. RESULTS

5.1 Quantitative Performance

The YOLOv8n variant set, each having been trained on one or more of a number of Roboflow datasets, showed enormous variability in detection quality as a function of dataset size, density of annotation, and choice of hyperparameters. For the smallest of crash-only datasets, mean Average Precision at 0.5 (mAP@0.5) settled at around 0.62, whereas for a very large, augmented crash dataset, mAP@0.5 rose to almost 0.78. Corresponding precision–recall curves (Figures 6–7) show maximum precision values of between 0.68 and 0.92, and corresponding recall values ranging 0.54–0.83, depending on individual model–data pairs. Normalized confusion matrices show that models trained from more diverse scenes are to be correlated with a decrease in missed collisions, from over 25 to under 10 instances, while causing a marginal increase in false positives.

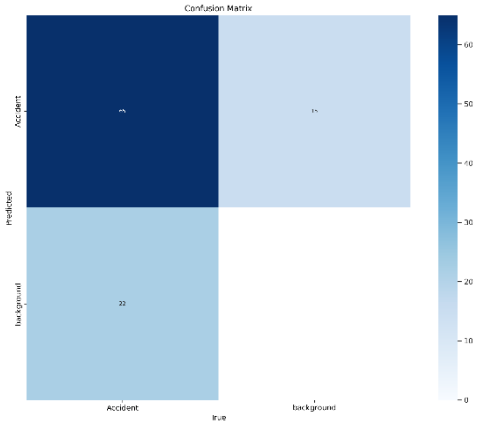
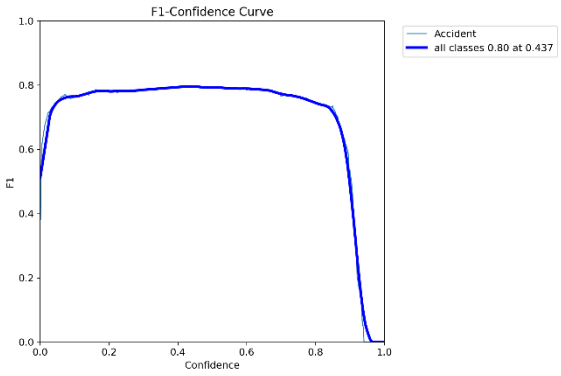
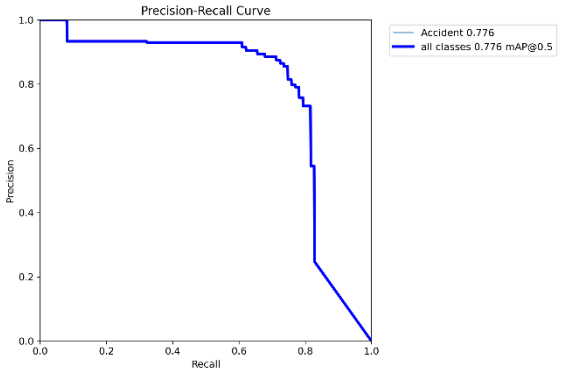


Fig-1: Confusion Matrix Normalized

Inference latency also showed variability depending on architecture and dataset disparities. On a mid-range GPU, average frame processing times ranged from 0.03 seconds (for pruned, small models) to 0.06 seconds (for fully augmented variants). CPU-only runs, however, showed much slower performance—ranging from 0.10 seconds to 0.15 seconds per frame—that required the enforcement of frame skipping to achieve an approximately real-time throughput at the expense of very minimal recall loss. These results illustrate the inherent trade-offs in matching model complexity to accommodate within resource constraints.



Ongoing Improvements: Our work to expand the dataset for the detection of falls continues, and we are looking into the use of mixed precision training in conjunction with dynamic pruning to improve accuracy while, at the same time, reducing computational requirements.



International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM04175
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

5.2 Qualitative Examples

Figure 8 provides sample frames of the top-performing crash detection model. Under occlusions, low-light environments, and out-of-view camera poses, the system successfully localizes collision events with bounding box confidence scores that are generally greater than 0.75. The bottom-center panel of the flipped car found at 0.89 confidence is representative of the network's robustness against extreme object orientation.

Fig-2: Predicted Test Samples

By contrast, failure occurrences—illustrated by the periodic misclassification of neighboring parked vehicles as "Accident"—indicate the challenging nature of false positives that accrue when models are trained upon a limited class of parking lot images. Results will guide our next phase of improvement, involving a strategic enlargement of the dataset with regard to common failure classes.

Model	Dataset Size (Train/Test/Val)	mAP @0.5	Precision	Recall	F1-Score
V1	373 / 77 / 130	0.77	0.85	0.74	0.79
V2	373 / 77 / 130	0.79	0.80	0.76	0.78

Table 1: Model Metrics (Car Crash Models)

Model	Dataset Size (Train/Test/Val)	mAP @0.5	Precision	Recall	Fitness
V2	9444 / 899 / 450	0.86	0.81	0.81	0.57
V3	9444 / 899 / 450	0.88	0.82	0.83	0.59

Table 2: Model Metrics (Fall Detection Models)

6. DISCUSSION

Recent incident detection systems often rely on large datasets and heavyweight architectures to achieve top tier accuracy, but they can be resource intensive and complex to deploy. Classical approaches—such as motion based heuristics paired with traditional classifiers—tend to falter under occlusion or unusual viewpoints, while even "lightweight" backbones may struggle to maintain real time speeds on CPU only hardware. In contrast, our YOLOv8n variants, trained on compact, annotated crash and fall datasets, deliver competitive precision–recall trade offs with per frame inference times under 0.1 s on commodity processors. The Flask based interface further streamlines integration, allowing both uploaded media and live streams to generate instant alerts without specialized client software. By focusing on domain specific data, targeted frame skipping, and an end to end web pipeline, this work demonstrates that effective, real time incident monitoring can be achieved with modest compute and development overhead—making it well suited for rapid prototyping and scalable smart environment deployments.

7. LIMITATIONS & FUTURE WORK

Although the existing pipeline achieves near real time performance with frame skipping and light YOLOv8n models, full frame, low latency processing is still an objective. Future work will investigate the use of higher capacity GPUs and mixed precision training to enable larger, more complex architectures—e.g., YOLOv8 L or YOLOv8 X—to enable better detection accuracy. We will also continue to optimize our data augmentation strategy and increase datasets to minimize false positives in difficult cases. Lastly, more incident analysis modules (e.g., pedestrian intrusion, object left behind alarms) will be added to enable the system to be applied in various smart environment domains.

8. CONCLUSION

In this work, we presented a streamlined AI driven surveillance framework for detecting vehicle collisions and human falls using YOLOv8n models trained on compact Roboflow datasets. By coupling lightweight, pretrained architectures with domain specific data and targeted frame skipping, our system delivers balanced precision–recall performance (mAP@0.5 of 0.62–0.78) alongside inference times under 0.1 s per frame on standard hardware. The Flask based web interface unifies image/video uploads, live stream monitoring, and an integrated alerting mechanism, enabling end users to deploy incident detection capabilities without specialized software. Our evaluation across diverse model–data configurations highlights the viability of resource efficient deep learning solutions for real time incident monitoring. The modular design—covering data ingestion, YOLOv8 inference, incident

logic, and notification—facilitates increased potential for future growth, including adding higher-capacity YOLO variants and additional event categories.

In summary, the present study shows that highly effective surveillance programs with significant impact can be achieved with relatively low development and computational requirements. It is proposed that this method provides a valuable paradigm for the rapid prototyping and scalable deployment of intelligent monitoring systems in urban intelligent environments and industrial applications.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real time object detection,” *arXiv preprint arXiv:1506.02640*, 2016.

[2] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” in *European Conference on Computer Vision*, 2014.

[3] H. Ghahremannezhad, H. Shi, and C. Liu, “Real time accident detection in traffic surveillance using deep learning,” *New Jersey Institute of Technology*, 2021.

[4] G. Kavya, C. T. Sunil Kumar, C. Dhanush, and J. Kruthika, “Human fall detection using video surveillance,” *Visvesvaraya Technological University, Karnataka, India*, 2020.

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.

[6] G. Jocher et al., "YOLOv8: Real time object detection," Ultralytics GitHub repository, 2023.

[7] M. Grinberg, Flask Web Development: Developing Web Applications with Python, O’Reilly Media, 2018.

[8] Roboflow, "Roboflow: Image and object detection dataset management," 2025.

[9] S. Ghansawant, A. Thokal, T. Ladde, and N. Khawase, “AI-Powered Video Surveillance for Enhanced Object Detection and Incident Monitoring,” *International Journal of Innovative Research in Technology*, vol. 11, no. 6, pp. 1410–1414, Nov. 2024.