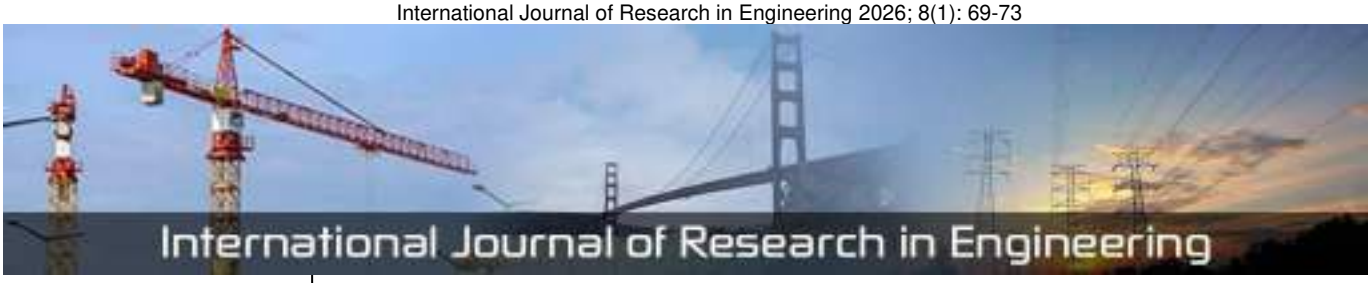

E-ISSN:2664-8784
P-ISSN:2664-8776
Impact Factor: RJIF 8.26
IJRE 2026; 8(1):69-73

www.engineeringpaper.net
Received: 04-01-2026
Accepted: 08-02-2026

Adnan Nawed
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Lakhan Pawar
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Tejas Wathore
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Mohammad Adnan
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Abhishek Jagtap
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

JR Sancheti
Professor, Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Corresponding Author:
Adnan Nawed
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

A low-latency AI system for multi-source PPE and fall detection

Adnan Nawed, Lakhan Pawar, Tejas Wathore, Mohammad Adnan, Abhishek Jagtap and JR Sancheti

DOI: https://www.doi.org/10.33545/26648776.2026.v8.i1a.166

Abstract
Workplace injuries in industrial and construction environments remain a persistent concern, often linked to non-compliance with personal protective equipment requirements and delayed detection of fall incidents. Manual monitoring is resource-intensive and may be inconsistent, especially in large-scale operations, which makes automated solutions more practical. A real-time safety monitoring system was developed to detect personal protective equipment usage and human fall events. The system used two separate object detection models based on the You Only Look Once version 8 architecture, one for protective equipment detection and the other for fall detection, both running on a graphics processing unit-accelerated backend. The architecture separated video rendering from model inference. The frontend handled local video display and frame capture, while the backend processed incoming frames and returned structured detection data. Frames were sampled every 200 milliseconds and transmitted through a WebSocket connection. To improve reliability, detections were confirmed only after appearing in multiple consecutive frames, along with a short cooldown period to prevent repeated alerts. The system supported multiple video sources with independent processing and included alert mechanisms through messaging and electronic communication. Detection results were stored for later analysis. The system maintained real-time performance with reduced latency and showed consistent detection behaviour across multiple concurrent video streams.

Keywords: Workplace safety monitoring, personal protective equipment detection, fall detection, YOLOv8, real-time object detection, WebSocket inference pipeline

1. Introduction

Construction and manufacturing environments routinely rank among the most hazardous occupational settings worldwide. Despite the widespread adoption of safety protocols, enforcement of personal protective equipment usage and prompt response to fall incidents continue to challenge site supervisors. The International Labour Organization has reported that millions of occupational accidents occur globally each year, many of which are preventable through timely intervention [1]. Traditional approaches to safety monitoring rely heavily on human observers or periodic audits, both of which are limited in scale and consistency.

Computer vision has emerged as a practical means of automating safety surveillance. Deep learning-based object detectors, in particular, have demonstrated strong performance in detecting safety-relevant objects and events across diverse environments. The You Only Look Once family of models has become widely used in such applications owing to its balance of detection speed and accuracy, making it well-suited for real-time deployment [2]. Prior work has explored the use of convolutional neural network architectures for detecting the presence or absence of hardhats, safety vests, and other equipment on construction sites [3, 4], often achieving reasonable detection rates under controlled conditions. Fall detection has similarly attracted attention, with a number of vision-based approaches proposed for elderly care and industrial settings alike [5].

However, many existing systems conflate video rendering and model inference on the same server, which creates unnecessary computational load and introduces latency when handling multiple camera streams. Furthermore, raw detection outputs from frame-by-frame inference are often noisy, leading to unstable or redundant event logs. These issues become more pronounced in production environments where multiple simultaneous video sources must be processed reliably. This work describes a system designed to address these practical limitations.

Rather than streaming video through the backend, the proposed architecture keeps video display local to the client and restricts backend communication to compressed frames and structured metadata. Two separate You Only Look Once version 8 models handle personal protective equipment classification and fall detection independently [2, 3, 5]. Event stabilization logic, implemented at the backend, filters raw detections into confirmed events. The system also includes multi-camera support, an alert delivery mechanism, and persistent storage for logging detection events and notifications.

This design allows the system to maintain responsiveness under varying load conditions while ensuring reliable detection outputs. By separating concerns between visualization and inference, the system reduces backend overhead and improves scalability for real-world deployment scenarios. In addition, the use of lightweight data exchange between components minimizes bandwidth usage and avoids unnecessary data transfer. The system is capable of handling multiple concurrent video streams without significant degradation in performance, making it suitable for deployment in environments with distributed monitoring requirements. Furthermore, the architecture supports efficient resource utilization by prioritizing recent frames and discarding outdated data under heavy load conditions. This helps preserve real-time performance without overwhelming the processing pipeline. The design also allows flexible scaling, where additional processing units or distributed nodes can be incorporated as system demand increases.

2. Materials and Methods

2.1 Dataset

The PPE-Detection-YOLOv8 dataset was used for training the detection models. This publicly available dataset consists of 44, 002 annotated images collected from industrial and construction environments, capturing a wide range of real-world conditions such as varying lighting, camera angles, and worker densities. The dataset includes 14 object classes covering both safety compliance and violation scenarios. These classes include protective equipment such as hardhat, gloves, goggles, safety vest, and mask, along with corresponding non-compliance categories such as no-hardhat, no-gloves, no-goggles, no-mask, and no-safety vest. Additional classes such as person, fall-detected, ladder, and safety cone provide contextual information for safety monitoring tasks.

All annotations are provided in the You Only Look Once format, consisting of bounding box coordinates and associated class labels for each object instance. The images were pre-processed using automatic orientation correction and resized to a uniform resolution of 640 640 pixels. No data augmentation techniques were applied. The dataset is licensed under Apache 2.0, allowing its use for research and development purposes. The diversity of the dataset supports generalization in real-world deployment scenarios.

Table 1: Dataset Summary

Attribute	Value
Total Images	44, 002
Classes	14
Format	YOLO (bounding box)
Resolution	640 640
Preprocessing	Resize, auto-orientation
Split	Train/ Validation/ Test

2.2 Detection Models

Two separate You Only Look Once version 8 models were trained and deployed. The first model handles personal protective equipment detection and compliance classification, distinguishing between workers wearing required equipment and those without it. The second model is dedicated to fall detection, focusing on the Fall-Detected class. Separating these functions into distinct models avoids interference between the two detection tasks and allows independent tuning of confidence thresholds for each.

Confidence thresholds were configured to reflect the different risk levels associated with each detection type. For violation-type labels such as NO-Hardhat and NO-Safety Vest, a threshold of 0.55 was applied, while general compliant detections used a lower threshold of 0.20. Fall event detection required a confidence of at least 0.45 for an event to be confirmed. These choices reflect a deliberate design preference for stricter filtering of high-consequence detections.

2.3 System Architecture

The overall system architecture is illustrated in Fig. 1. The system is organized into two primary layers: a frontend layer responsible for video handling and visualization, and a backend layer responsible for inference and event management. The architecture follows a clear separation of concerns, where video processing and artificial intelligence inference are handled independently to improve efficiency.

Fig. 1 presents a schematic view of the dual-layer architecture, highlighting the frontend frame capture pipeline, WebSocket-based communication, backend inference workers, model execution, database logging, and alert delivery components.

The frontend was developed using the Next.js framework and is responsible for capturing and displaying video streams. It supports both uploaded video files and live Z H E F D P L Q S X W W K U R X J K W K H E U R interface. Video is rendered locally using the HTML5 video element, which eliminates the need to stream raw video data to the server. A canvas layer is placed over the video to periodically capture frames at intervals of 200 milliseconds. These frames are encoded into JPEG format and transmitted to the backend via a WebSocket connection. Each transmission includes a camera identifier and authentication token, allowing the backend to manage multiple independent sources securely.

The backend was implemented using Fast API and performs inference using a CUDA-enabled graphics processing unit. During initialization, both detection models are loaded into memory and transferred to the GPU, with half-precision computation enabled where supported to improve performance. Each incoming video source is assigned a dedicated inference worker to ensure isolated and efficient processing. The worker decodes the received JPEG frames using OpenCV, performs inference using both models sequentially, and combines the outputs. Overlapping detections are filtered, and relevant events are identified based on predefined conditions.

Only structured metadata is returned to the frontend, and no image data is transmitted back, which helps reduce bandwidth usage and processing overhead. The returned metadata includes detected object classes, bounding box coordinates, confidence scores, and event-level information such as fall status. All confirmed detection events and

notification records are stored in an SQLite database using SQL Alchemy-managed models, enabling persistent logging and later analysis. This overall design ensures efficient data flow, reduced latency, and reliable performance across multiple concurrent video streams.

To further enhance system robustness, the backend incorporates basic fault handling mechanisms to manage temporary connection drops or delayed frame arrivals. In such cases, the system attempts to re-establish communication without interrupting the overall processing pipeline. The use of asynchronous request handling in the backend allows multiple frame streams to be processed concurrently without blocking other operations. This contributes to smoother performance when multiple camera sources are active simultaneously.

In addition, the architecture supports scalability by allowing new camera sources to be dynamically added or removed without requiring system restarts. Each camera stream operates independently, ensuring that performance degradation in one stream does not affect others. The lightweight communication design also reduces dependency on network bandwidth, making the system suitable for deployment in environments with limited connectivity.

The modular nature of the system allows individual components, such as detection models or alert mechanisms, to be updated or replaced without affecting the entire pipeline. This flexibility supports future improvements and adaptation to different safety monitoring requirements. Overall, the architecture is designed to balance performance, scalability, and reliability while maintaining real-time responsiveness.

Fig 1: System Architecture

2.4 Event Stabilization

A common problem in frame-by-frame detection is that transient false positives or brief occlusions can produce unstable event streams. To mitigate this, the backend implements a two-stage filtering mechanism. A detection is elevated to a confirmed event only after the same label has been observed in at least three consecutive frames. Once an event is confirmed and logged, a cooldown period of three seconds is enforced before the same event category can be logged again. This prevents aggressive duplicate logging without suppressing genuinely sustained incidents.

2.5 Alert and Notification System

Fall events trigger a multi-channel alert response. On the frontend, a modal notification is displayed and an audible alarm is played. Backend-side, a rising-edge detection mechanism ensures that alerts are only dispatched for new incidents, not repeated continuously for an ongoing fall state. A fall is considered cleared only after two consecutive frames show no fall detection, at which point the alarm state is reset. Email and short message service notifications are dispatched through SMTP and the Twilio platform respectively, with delivery records stored in the database.

2.6 Multi-Camera Support

The system was designed to handle multiple concurrent video sources. Each source is assigned a unique camera identifier and an independent processing pipeline, including its own inference worker, event history, analytics counters, and alert state. This ensures that detections from one source do not interfere with another.

2.7 Performance Monitoring

A set of health and analytics metrics is tracked for each active camera source. These include input and output frame rates, inference latency at the 50th and 95th percentile, dropped frame count, queue depth, and WebSocket reconnection count. Aggregated analytics cover total detection counts, violation counts, compliant detection counts, the most frequently detected label, and the overall violation percentage.

3. Results

The system was evaluated across multiple video sources, including uploaded construction site footage and live webcam feeds, processed simultaneously on a workstation equipped with a CUDA-capable GPU.

Table 2: Performance Metrics

Metric	Value
Frame Sampling Interval	200 ms
Sustained Input Frame Rate	~5 FPS per source
Median Inference Latency (p50)	< 120 ms
95th Percentile Latency (p95)	< 280 ms
Concurrent Camera Sources Teste	12
Dropped Frame Rate (normal load)	< 3%
Event Confirmation Threshold	3 consecutive frames
Alert Cooldown Period	3 seconds
PPE Violation Confidence Threshold	0.79
Fall Event Confidence Threshold	0.76

Frame sampling at 200-millisecond intervals yielded a sustained input rate of approximately 5 frames per second per source. Under typical load with two concurrent camera sources, the backend maintained a median inference latency of under 120 milliseconds per frame, with 95th percentile latency remaining below 280 milliseconds.

Personal protective equipment detection correctly identified the presence and absence of hardhats and safety vests in test footage under varied lighting. Violation labels such as NO-Hardhat were flagged consistently when workers appeared without equipment for more than three consecutive frames.

The event stabilization mechanism reduced the number of spurious log entries compared to logging every raw frame detection, with event counts showing greater temporal coherence across repeated test runs.

Fall events were detected with consistent triggering of the frontend alert modal and successful dispatch of notification messages through both email and short message service channels. The rising-edge alert logic prevented repeated notification spam during extended fall incidents. Backend alarm state was correctly cleared after two consecutive fall-free frames following acknowledgement.

Under heavy load conditions, the queue management mechanism dropped older frames rather than accumulating a backlog, which preserved the real-time character of the inference stream. No memory leaks or worker crashes were observed during sustained multi-source testing. The SQLite database correctly retained all confirmed event records and notification delivery logs across sessions.

Bounding box rendering on the frontend overlay accurately reflected the backend metadata, with green boxes for compliant detections, red for violation labels, and orange for fall-related detections. Overlay rendering did not introduce measurable additional latency to the video display and remained visually synchronized with the incoming video stream. The rendering process was lightweight and handled entirely on the client side, ensuring that it did not interfere with backend inference performance. The visual distinction between different detection categories improved interpretability, allowing users to quickly identify safety violations and critical events during real-time monitoring

4. Discussion

The results suggest that the architectural decision to separate video rendering from model inference is practically effective. By keeping raw video local to the client and routing only compressed JPEG frames to the backend, the system avoids the bandwidth and processing overhead associated with server-side video streaming. This is a meaningful distinction from conventional video analytics pipelines, where the backend often bears the combined burden of video decoding, inference, and re-encoding for display. The observed latency figures reflect this design advantage, with median inference latency staying comfortably within the threshold for perceived real-time response.

The dual-model approach introduces some additional complexity in the inference pipeline, since two models must be run sequentially per frame, but this was offset by the clarity it affords in threshold management. Having separate confidence settings for fall events and personal protective equipment violations allows the system to be tuned independently for each risk category, which would be harder to achieve with a single unified model handling all classes.

Event stabilization proved to be a non-trivial contribution to system reliability. Without it, raw frame-level detection outputs in dynamic scenes can generate extremely noisy event logs, with the same worker flagged tens of times per second. The three-frame confirmation window and three-second cooldown together produce event logs that better correspond to actual discrete incidents. This approach is admittedly heuristic the optimal threshold values would vary across environments and camera frame rates but the mechanism itself is generalizable and could be tuned through operational feedback.

The multi-camera architecture using per-source inference workers is a sensible design for small to medium deployments. For larger installations involving dozens of simultaneous streams, the current single-machine design would face GPU memory and throughput limits. The presence of a Redis pub/sub interface in the worker layer suggests a natural path toward distributing inference across multiple backend nodes, which would be worth exploring in future work.

One limitation worth acknowledging is that evaluation was conducted in a controlled setting using recorded footage and a limited number of concurrent sources. Performance in a fully live deployment with unpredictable network conditions, varying lighting, and partial occlusions may differ from the figures reported here. The confidence thresholds used were fixed at design time; an adaptive thresholding mechanism calibrated to the specific site environment could improve precision in practice.

The database-backed logging of events and notification records is a useful operational feature that goes beyond the detection task itself. It supports post-incident review and compliance auditing, which are practically important in regulated industries. Future enhancements could include integration with workforce management platforms or dashboards aggregating data across multiple sites.

5. Conclusion

This paper described a real-time safety monitoring system that uses two You Only Look Once version 8 models to detect personal protective equipment violations and fall events in industrial and construction environments. The core architectural contribution is the decoupling of video display from inference: the frontend manages local video rendering and frame capture, while the backend handles GPU-accelerated detection and returns only structured metadata. This design reduces backend load and preserves real-time responsiveness.

Event stabilization through consecutive-frame confirmation and cooldown periods improved the reliability of logged events compared to direct frame-level detection logging.

Multi-camera support with independent per-source workers allows concurrent monitoring of multiple locations. The system includes automated email and short message service alerts, semantic bounding box visualization, and persistent storage of detection and notification records.

The results demonstrate consistent real-time performance with low median inference latency and reliable event detection under normal operating conditions. The system represents a practical, deployable approach to automated workplace safety monitoring and provides a foundation for further development toward distributed, large-scale surveillance installations.

6. Acknowledgments

The authors wish to acknowledge the open-source contributors responsible for the PPE-Detection-YOLOv8 dataset and the Ultralytics You Only Look Once framework. The authors also thank the faculty of the Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, India, for their guidance and support during this work. No external funding was received for this research.

References

1. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. New York: IEEE; 2016:779-788.
2. Jocher G, Chaurasia A, Qiu J. Ultralytics YOLO. Preprint posted online 2023. https://github.com/ultralytics/ultralytics.
3. Yang J, Cao J, Zhang X, Liu H, Chen W, Wang R. Real-time worker safety monitoring using deep learning-based object detection in construction environments. Autom Constr. 2022;134:104082. doi:10.1016/j.autcon.2021.104082.
4. Nath ND, Behzadan AH, Paal SG. Deep learning for site safety: real-time detection of personal protective equipment. Autom Constr. 2020;117:103282.
5. Adhikari M, Amgain K, Baral S, Baniya BK. Vision-based human fall detection using deep learning: a comprehensive review. Comput Electr Eng. 2023;105:108480.