International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Real-Time Suspicious Activity Detection in Surveillance Videos Using Deep Learning and Computer Vision

Prof. Abdulkayyum Shaikh
Assistant Professor
Department of Computer Engineering
SF's Sandip Institute of Technology and Research Center, Nashik-422213, India
kayyum.shaikh22@gmail.com

Shreyas Pagar
Student
Department of Computer Engineering
SF's Sandip Institute of Technology and Research Center, Nashik-422213,India
shreyaspagar2004@gmail.com

Bhushan Khairnar
Student
Department of Computer Engineering
SF's Sandip Institute of Technology and Research Center, Nashik-422213, India
bhushanrajekhairnar@gmail.com

Yash Shirsale
Student
Department of Computer Engineering
SF's Sandip Institute of Technology and Research Center, Nashik-422213, India
shirsaleyash@gmail.com

Shivam Mahajan
Student
Department of Computer Engineering
SF's Sandip Institute of Technology and Research Center, Nashik-422213, India
shivmahajan5588@gmail.com

***Abstract*: Surveillance systems across public and private spaces carry a persistent operational weakness: they depend almost entirely on human operators watching screens, and human attention fades fast. Most conventional CCTV setups do little beyond recording footage passively or replaying stored clips after an incident. That might preserve evidence, sure. It rarely catches a threat as it unfolds, and it almost never pushes a real-time alert to the right person at the right moment. This study runs a broad, systematic review of recent work across Computer Vision (CV), deep learning based object detection, real-time video analytics, and intelligent monitoring frameworks as they relate to building automated surveillance pipelines and threat-aware alerting systems. Then we put forward a conceptual architecture — a Smart Surveillance System Using Machine Learning and Computer Vision. The idea is straightforward. Watch what matters. Detect the threat. Alert the operator before damage is done. How? The framework fuses real-time video acquisition through OpenCV, automated object detection powered by YOLOv8's anchor-free architecture, a microservice pipeline using Node.js and Express.js for detection processing, MongoDB for persistent event logging, and a React.js dashboard delivering WebSocket-driven alerts the moment a security-relevant object enters the frame. We dig into the tools that make it viable: MJPEG streaming for low-latency video delivery, Socket.IO for bidirectional communication, severity-tiered alert classification, and modular service isolation. Drawing insights from state-oftheart sources, this review aims to lay a practical, theoretical, and structural base for the next generation of genuinely intelligent surveillance systems.**

Key Words: Smart Surveillance, Real-Time Object Detection, YOLOv8, OpenCV, Computer Vision, Machine Learning, , React.js Dashboard, MongoDB, Intelligent Monitoring.

I. INTRODUCTION

Surveillance cameras are everywhere now. Airports, banks, shopping malls, highways, residential buildings. [5], [6] The deployment numbers are staggering, honestly. Over a billion cameras installed worldwide, generating petabytes of raw video data every single day. [7] Because the footage never stops.

and most of it goes straight to a hard drive where nobody ever looks at it again. Getting useful security intelligence out of that flood is harder than people admit. It's not just "install a camera." The footage has to be analyzed, the objects in each frame have to be identified, and the system has to know the difference between a person walking normally and something that actually needs attention. Miss any of that and the surveillance setup becomes an expensive recording device, nothing more.

Now deep learning, convolutional neural networks, and singlestage object detectors are changing the conversation. Big time. [1], [14] These models can detect and classify things — not just flag motion. That means you can build automated detection pipelines, generate severity-tiered alerts, and even push real-time notifications that fit the exact context of what the camera just captured. [3] That's the promise, anyway.

But let's be real: the integration hasn't caught up. The literature still doesn't give us a clean end-to-end framework that ties realtime video acquisition to automated object detection and then to instant, severity-classified alerting on a unified dashboard — all as one coherent pipeline. Everyone has pieces. Nobody has the full bridge.

This paper steps into that gap with a systematic review of current deep learning and computer vision based approaches for intelligent surveillance systems. [4] It digs into methods like anchor-free single-stage detection, MJPEG-based low-latency

video streaming, and WebSocket-driven real-time communication that actually pays attention to responsiveness. Then we go a step further and propose an architectural framework: a Smart Surveillance System Using Machine Learning and Computer Vision. The workflow is real-time video capture through OpenCV, frame-level object detection via YOLOv8, severity-based alert classification, persistent logging in MongoDB, and a live React.js dashboard designed to surface and respond to security events the moment they happen.

II. LITERATURE REVIEW

Use of computer vision within the security and surveillance domain has undergone several paradigm shifts over the past two decades, namely the use of handcrafted feature extractors, background subtraction models, region-based convolutional networks, single-stage detectors, and recently anchor-free architectures with transformer attention mechanisms.

A. Traditional Computer Vision and Motion-Based Methods

In early years, surveillance systems relied on classical computer vision techniques including background subtraction, frame differencing, and handcrafted feature descriptors. Stauffer and Grimson [21] proposed Adaptive Background Mixture Models for real-time tracking using Gaussian Mixture Models, achieving reliable foreground detection but with high sensitivity to illumination changes and camera motion. Viola and Jones [19] introduced a boosted cascade of Haar-like features for rapid face and object detection with impressive speed but limited accuracy on complex, cluttered scenes. Dalal and Triggs [20] developed Histograms of Oriented Gradients (HOG) for human detection, reaching competitive pedestrian detection rates but suffering from poor scalability when applied across varying object surveillance. Krizhevsky et al. [22] introduced AlexNet, which dominated the ImageNet challenge and established deep feature learning as the standard. Girshick et al. [13] proposed R-CNN, using region proposals with CNN classification, achieving high accuracy but at prohibitively slow speeds for real-time use. Ren et al. [12] advanced this with Faster R-CNN, introducing Region Proposal Networks to reduce inference time, yet the twostage architecture still fell short for live video. Liu et al. [11] addressed this with SSD, a single-stage detect processing multiple feature maps simultaneously for a better speedaccuracy trade-off. He et al. [14] introduced residual connections in ResNet, enabling deeper networks without degradation - a backbone foundational across modern detectors. Simonyan and Zisserman [29] demonstrated that increasing depth with small 3×3 filters

The YOLO paradigm introduced by Redmon et al. [1] changed the conversation entirely. By framing detection as a single regression problem over an entire image, YOLO achieved realtime performance at 45 FPS with competitive accuracy — a capability that traditional two-stage detectors could not match. Subsequent versions improved steadily: Redmon and Farhadi [23] enhanced multi-scale prediction in YOLOv3, Bochkovskiy et al. [2] introduced mosaic augmentation and self-adversarial training in YOLOv4, and Wang et al. [15] achieved state-of-theart real-time detection with YOLOv7. Jocher et al. [3] released YOLOv8, which shifted to an anchor-free detection head with a decoupled architecture, simplifying deployment and improving accuracy on the COCO benchmark [7]. Terven et al. [4] provided a comprehensive review confirming that this evolution from

TABLE I
COMPREHENSIVE COMPARISON OF INTELLIGENT SURVEILLANCE PARADIGMS

Paradigm Category	Representative Literature	Core Underlying Technology	Diagnostic Assessment Depth	Remediation & Feed back Strategy
Handcrafted Feature Extractors	Viola & Jones [19], Dalal & Triggs [20], Stauffer & Grimson [21]	Haar Cascades, HOG, Background Subtraction	Low (Relies on Pixel-Level Motion and Rigid Templates)	Manual Monitoring, Threshold-Based Motion Triggers
Two-Stage Deep Detectors	Girshick et al. [13], Ren et al. [12]	R-CNN, Faster R-CNN, Cascade R-CNN	Medium (Region Proposals with CNN Classification)	Offline Post-Processing, Batch Frame Analysis
Single-Stage RealTime Detectors	Redmon et al. [1], Liu et al. [11], Jocher et al. [3]	YOLO Family, SSD, Anchor-Free Heads	High (End-to-End Frame-Level Object Classification with Confidence Scores)	Real-Time Bounding Box Overlay, Automated Logging
Lightweight Edge & Transformer Models	Howard et al. [16], Nikouei et al. [9], Dosovitskiy et al. [25]	MobileNets, Lightweight CNNs, Vision Transformers	High (Semantic Understanding with Attention on Resource)	Edge-Triggered Push Alerts, WebSocket Dashboard Streaming

in VGGNet captures richer spatial hierarchies, influencing every classes and viewpoints. These methods operated at a shallow perceptual level — they could flag motion or detect a rigid pattern, sure. But semantic understanding of what the detected region actually represented? That remained out of reach.

B. Deep Learning-Based Object Detection Architectures

The breakthrough of deep convolutional neural networks fundamentally transformed object detection for YOLOv1 to YOLOv8 represents consistent gains in both speed and precision. For edge deployment, Howard et al. [16] introduced MobileNets using depthwise separable convolutions to enable real-time inference on resource-constrained hardware. Nikouei et al. [9] demonstrated the feasibility of running lightweight CNNs on edge devices for surveillance, proving that real-time human detection is achievable even without GPU acceleration.

D. Foundations of Real-Time Integrated Detection The key foundation of an intelligent surveillance pipeline lies in seamless coordination between frame-level object detection, event classification, and real-time notification. Through continuous video stream analysis and severity-tiered comparison, automated systems can assess detected objects through a structured classification matrix. This research reveals that effective surveillance requires end-to-end integration — from video acquisition to persistent event logging and WebSocket-driven dashboard alerting — through modular architecture and feedback loops.

I. GAP ANALYSIS

Despite the rapid advancement of deep learning in video surveillance, a thorough study of the current state-of-the-art architectural designs demonstrates several key gaps, which severely limit their practical deployment effectiveness::

- **Absence of Real-Time Severity-Classified Alerting:** The existing frameworks focus almost exclusively on object detection accuracy and bounding box precision. [1], [4] They fail to classify detected objects into actionable severity tiers immediately upon detection, leaving operators to manually interpret every single alert without prioritization.
- **Failure to Integrate Detection with Persistent Event Logging:** The current detection pipelines operate as isolated inference engines, generating detections frameby-frame but failing to store any structured event history, timestamps, or contextual metadata would enable postincident forensic analysis and trend identification. [6]
- **Absence of Unified Dashboard Visualization:** Although modern object detection models perform flawlessly in terms of identifying objects within video frames, they fail to deliver any centralized operator interface that combines live annotated video, real-time alert feeds, and aggregate detection statistics within a single coherent view. [5]
- **Absence of Modular Microservice Architecture::** Most systems are implemented as monolithic applications that tightly couple video capture, detection, , and display into a single subsequent detection framework. *C. YOLO Family and Lightweight Edge Architectures*

process, offering no ability to scale, replace, or deploy individual components based on operational requirements and hardware constraints. [9]

II. PROPOSED FRAMEWORK

In order to bridge every identified research gap, the current research proposes a Smart Surveillance System Using Machine Learning and Computer Vision. [3], [8] This system functions on a closed-loop processing pipeline that transforms passive video recording into an active, automated detection and alerting process. [4]

The overall architectural flow of the proposed system is visually summarized in Fig. 1. The pipeline operates as a continuous monitoring loop where the camera source first captures live video frames. This triggers the initialization of the detection engine. YOLOv8 then dynamically analyzes each frame to identify and classify objects, and the detections are deeply processed to isolate security-relevant events. Finally, severity-classified alerts are generated and pushed to a real-time dashboard, which subsequently feeds back into the system to foster continuous monitoring and adaptive threat response.

A. Stage 1: Camera Initialization and Video Acquisition

The system initializes the video source using OpenCV's VideoCapture module, supporting webcam, USB camera, IP camera, or pre-recorded video. This establishes a continuous frame capture pipeline, forming the raw data stream feeding the detection architecture. [8] Captured frames are preprocessed through resizing and normalization to match YOLOv8 input requirements (640×640 pixels).

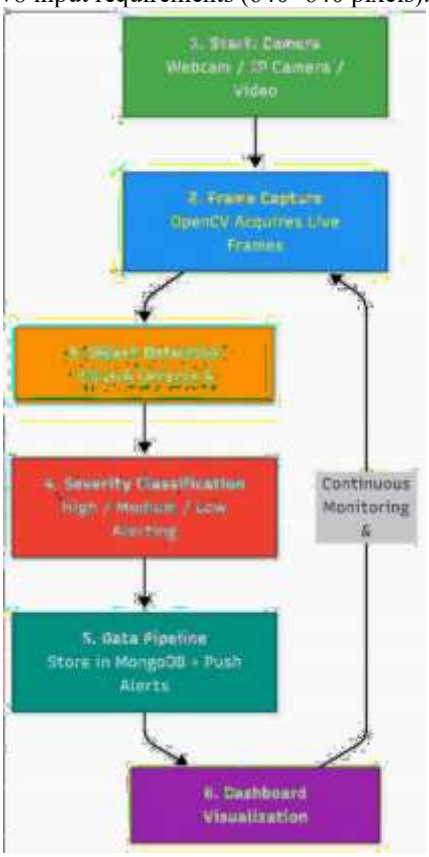


Fig. 1. Adaptive Surveillance System Workflow.

B. Stage 2: Real-Time Object Detection Engine

Instead of depending on manual operator observation, which is prone to fatigue and delayed response, the system runs each preprocessed frame through the YOLOv8 anchorfree detection model. [3] This is done by using the Ultralytics inference engine to perform single-pass detection across the entire frame, identifying multiple objects simultaneously. [1] The model outputs a list of detections, each containing the object class (e.g., person, car, backpack), confidence score (0.0–1.0), bounding box coordinates, and timestamp. The detection engine operates at 15– 30 FPS on standard CPU hardware without requiring GPU acceleration. [9]

C. Stage 3: Severity Classification and Alert Generation Engine

Based on the detected object classes and their confidence scores, the system's severity classification engine assigns each detection an actionable priority tier. [5] This is done by mapping detected objects against a predefined threat matrix: person detections trigger high-severity intrusion alerts, vehicles trigger medium-severity activity alerts, unattended objects like backpacks and suitcases trigger medium-severity suspicious object alerts, and animals or general objects are logged as lowpriority informational events. [6]

D. Stage 4: Persistent Event Logging and Data Management system structures and stores every detection event as a document in MongoDB. In contrast to regular systems On the backend, using detection metadata including class, confidence, bounding box, timestamp, and severity tier, the

which discard detection data after display, our system maintains a persistent, queryable event log that enables post incident forensic analysis, historical trend identification, and aggregate statistical reporting through dedicated REST API endpoints. [34]

E. Stage 6 :WebSocket-Driven Dashboard and Real-Time Visualization

In the dashboard, every detected object displays its severity and confidence score. When a new detection occurs, the Socket.IO engine executes the following process: metrics. [35]

1) **Alert Push:** The backend automatically emits the detection event via WebSocket to all connected dashboard clients in real-time. [17]
2) **Live Annotated Feed:** The MJPEG video stream with bounding boxes and class labels overlaid is rendered directly in the React.js dashboard through a simple img tag pointing to the AI service endpoint. [8]
3) **Statistics Update:** The dashboard statistics panel is automatically refreshed to reflect updated total detection counts, per-class breakdowns, and severity distribution

TABLE II

THEORETICAL WORKFLOW MODULES OF THE PROPOSED FRAMEWORK

System Module	Primary Functional Responsibility	Input Data Stream	Output System Artifact
Video Acquisition	Captures live video frames from camera source	Camera Device / Video File Path	Raw Video Frame Stream
Object Detection Engine	Runs YOLOv8 inference on each frame	Preprocessed 640×640 Frame	Detection List (Class, Confidence, BBox, Timestamp)
Severity Classifier	Maps detected objects to actionable threat tiers	Detection List + Threat Matrix	Severity-Tagged Alert Objects
Data Pipeline	Forwards detections to backend via REST API	Severity-Tagged Detections JSON	HTTP POST to Node.js Backend
Persistent Storage	Stores every detection as a MongoDB document	Incoming Detection JSON	Queryable Event Log with Timestamps
Real-Time Alerting	Pushes new detections to frontend via WebSocket	Stored Detection Event	Socket.IO Emission to Dashboard
Dashboard Visualization	Displays live feed, alerts, and statistics	MJPEG Stream + WebSocket Events	Unified Operator Interface

TABLE III

DETAILED FEATURE MATRIX ACROSS EDUCATIONAL SYSTEMS

System Architecture Capability	Traditional LMS & Classifiers	Swarm / RL Recommenders	Proposed Gen AI Framework
Real-Time Object Detection & Classification	No (Human Eyes Only)	Partial (Motion Blobs, No Class Labels)	Yes (YOLOv8 Anchor-Free, 80 Classes, 15–30 FPS)

Severity-Tiered Alert Generation	No	No	Yes (High / Medium / Low / Info Threat Matrix)
Persistent Searchable Event Logging	No (Raw Recording Only)	No	Yes (MongoDB Timestamped Documents with REST API)
Real-Time Dashboard Alerting	No	No	Yes (WebSocket Push via Socket.IO)
Modular Microservice Architecture	No (Monolithic Hardware)	No (Single Process)	Yes (4 Independent Scalable Services)

III. OPTIMIZATION, GOVERNANCE, AND ETHICAL MECHANICS

For system stability, optimal detection fidelity, and strict adherence to ethical and privacy requirements, the proposed architectural framework will incorporate three levels of cuttingedge governance mechanisms [17]

- **API Security and Access Control Protection:** Communication channels between the AI service, backend, and dashboard will be secured via API key authentication, CORS origin restriction, and MongoDB credential enforcement. [34] Input validation mechanisms guarantee the integrity of incoming detection payloads, preventing injection of fabricated alerts throughout the multi-service architecture.
- **Privacy-Preserved Surveillance and Transparent Logging:** Instead of being an opaque monitoring system that records indiscriminately, the system maintains structured, timestamped event logs with configurable data retention policies through MongoDB TTL indexes, enabling automatic purging of records older than a defined threshold, ensuring compliance with data minimization principles and fostering operator accountability. [17]

IV. COMPARATIVE DISCUSSION

The theoretical architecture proposed in this study has many benefits when systematically compared with current paradigms in surveillance technology:

A. Semantic Detection Proficiency Over Motion-Based Detectors Computer vision algorithms in traditional surveillance (such as background subtraction and Haar cascades explored by Stauffer and Grimson [21] and Viola and Jones [19]) depend upon the pixel-level approach based on motion changes in video frames. The drawback in this methodology is that it does not work when environmental conditions shift and produce false triggers without any semantic understanding of object identity. On the other hand, the framework uses real-time YOLOv8 detection in the form of single-pass inference in order to ensure that alert generation is based on current classification proficiency.

Adaptive Frame Processing Optimization: The detection pipeline is gradually optimized through configurable frame-skipping and resolution scaling based on system load. [3], [9] When the CPU utilization exceeds a threshold during continuous monitoring, the system dynamically reduces processing frequency by analyzing every 2nd or 3rd frame instead of every frame, maintaining real-time responsiveness without sacrificing detection coverage across critical time windows.

B. Integrated Alerting Over Isolated Detection Models

State-of-the-art single-stage and two-stage detection architectures (such as Faster R-CNN by Ren et al. [12]) perform very well in detecting and classifying objects on the basis of benchmark scores and precision metrics. When a detected object appears in a live feed but there is no severity classification or dashboard alerting, there is nothing that can be done by detection models alone.

C. Alignment of Architectural Framework with Contemporary Security Requirement

Through integration of real-time video acquisition, framelevel detection, severity-tiered alert classification, persistent event logging, and immediate WebSocket-driven visualization in a closed-loop microservice structure, the proposed approach provides an absolute paradigm shift from the traditional and passive surveillance frameworks towards active and intelligent monitoring systems. [4]

V. CONCLUSION AND FUTURE DIRECTIONS

The above analysis is based on an extensive systematic review of research studies on deep learning based object detection, intelligent video surveillance and real-time monitoring systems. We have also developed a new theoretical architectural model for a Smart Surveillance System Using Machine Learning and Computer Vision. Our new framework addresses some basic structural flaws that exist in the existing surveillance technology literature by integrating YOLOv8 object detection, severity-tiered alert classification, persistent MongoDB event logging and interactive WebSocket-driven dashboard visualization.

Future research shall concentrate on implementing the theoretical architectural framework empirically through software deployment. The next phase of research shall measure the performance of our prototypical system in multiple deployment environments, with emphasis on detection accuracy, inference latency, alert response time and improvements in security outcomes.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779–788.
[2] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020.
[3] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," Ultralytics, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics
[4] J. Terven, D. M. Córdova-Esparza, and J. A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," Mach. Learn. Knowl. Extr., vol. 5, no. 4, pp. 1680 –1716, 2023.
[5] G. Sreenu and M. A. Saleem Durai, "Intelligent Video Surveillance: A Review through Deep Learning Techniques for Crowd Analysis," J. Big Data, vol. 6, no. 1, pp. 1–27, 2019.
[6] W. Sultani, C. Chen, and M. Shah, "Real-World Anomaly Detection in Surveillance Videos," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018, pp. 6479–6488.
[7] 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 Proc. Eur. Conf. Comput. Vis. (ECCV), 2014, pp. 740–755.
[8] G. Bradski, "The OpenCV Library," Dr. Dobb's J. Softw. Tools, vol. 25, no. 11, pp. 120–125, 2000.
[9] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi, and T. R. Faughnan, "Real-Time Human Detection as an Edge Service Enabled by a Lightweight CNN," in Proc. IEEE Int. Conf. Edge Comput. (EDGE), 2018, pp. 125–129.
[10][10] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2015, pp. 4489–4497.
[11]W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2016, pp. 21–37.
[12]S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[13]R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2014, pp. 580–587.
[14]K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770 –778.
[15]C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bagof-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2023, pp. 7464–7475.
[16]A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, D. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv preprint arXiv:1704.04861, 2017.
[17]V. Pimentel and B. G. Nickerson, "Communicating and Displaying RealTime Data with WebSocket," IEEE Internet Comput., vol. 16, no. 4, pp. 45– 53, Jul. 2012.
[18]J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2009, pp. 248–255.
[19]P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2001, pp. 511–518.
[20]N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2005, pp. 886–893.
[21]C. Stauffer and W. E. L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 1999, pp. 246–252.
[22]A. Krizhevsky, I. Sutskever, and I. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2012, pp. 1097– 1105.
[23]J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.
[24]Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," arXiv preprint arXiv:2107.08430, 2021. [25] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.
[26] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in Proc. Int. Conf. Mach. Learn. (ICML), 2019, pp. 6105–6114.
[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015.
[28] A. Vaswani et al., "Attention Is All You Need," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 5998–6008.
[29] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in Proc. Int. Conf. Learn. Represent. (ICLR), 2015.
[30] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into High Quality Object Detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018, pp. 6154–6162. [31] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as Points," arXiv preprint arXiv:1904.07850, 2019.
[32] P. Sun et al., "Sparse R-CNN: End-to-End Object Detection with Learnable Proposals," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2021, pp. 14454–14463.
[33] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A Forest Fire Detection System Based on Ensemble Learning," Forests, vol. 12, no. 2, p. 217, 2021. [34] MongoDB, Inc., "MongoDB Documentation," 2023. [Online]. Available: https://www.mongodb.com/docs/ [35] Meta Platforms, Inc., "React Documentation," 2023. [Online]. Available: https://react.dev/ [36] OpenJS Foundation, "Node.js Documentation," 2023. [Online]. Available: https://nodejs.org/docs/
[37] O. E. Ojo and A. Adewumi, "YOLO v3: Visual and Real-Time Object Detection Model for Smart Surveillance Systems," in Proc. 5th Int. Conf. Inf. Technol. Educ. Dev. (ITED), IEEE, 2022, pp. 1–8.