# HAWKEYE (AI-Powered surveillance system with Anomaly Detection and 5G/4G Streaming)

Dr Rajesh Khanna , Dr Surbhi Sharma,
Parusha Pal, Drishti Sharma,Aditi
Sharma,Kanish Chadha and Vansh Khanna
Electronics and Communication Department
Thapar Institute of Engineering and
Technology
Bhadson rd., Patiala 147004, Punjab, India
{parushapal9@,drishtisharma3488,aditiisharma1901}@gmail.com

***Abstract*— This project, Hawkeye AI-Powered Surveillance System with Anomaly Detection and 5G Streaming, addresses the growing need for intelligent real-time security solutions by integrating edge computing, advanced deep learning, and high-speed cellular connectivity into a compact, efficient platform. The system uses a Raspberry Pi 4 paired with a high-resolution camera and a YOLO-based anomaly detection model to identify suspicious activities such as robbery, stealing, shooting, and burglary directly from live video feeds. By performing spatiotemporal analysis at the edge, it minimizes latency and reduces reliance on continuous cloud processing, enhancing privacy and bandwidth efficiency. Detection alerts and video streams are transmitted over LTE Cat-1 5G networks to our backend, enabling instant access through a responsive React.js dashboard. The architecture incorporates event-based recording, optimized storage management, and secure authentication, making it suitable for deployment in urban, industrial, and remote environments. This approach represents a proactive, scalable, and reliable alternative to traditional CCTV systems, capable of reducing false alarms by up to 70 % and significantly improving incident response times. .**

## I. INTRODUCTION

Modern surveillance systems have become an integral part of public safety, industrial monitoring, and smart security infrastructures. Traditional CCTV networks, however, function primarily as passive recording systems, depending heavily on human operators to interpret live footage and identify critical incidents. This reliance often leads to delayed responses, fatigue-induced oversight, and an inability to detect fast-occurring anomalous activities such as theft, physical assaults, or unauthorized intrusions [1]. With growing environmental complexity and increasing security demands, the limitations of conventional surveillance are more evident, motivating the need for intelligent and automated real-time detection mechanisms.

Advancements in lightweight deep-learning architectures and edge computing have made it possible to perform meaningful visual analysis on resource-constrained hardware. Models such as YOLO variants provide competitive accuracy at high inference speeds, while devices like the Raspberry Pi 4 can execute optimized neural networks using TensorFlow Lite [2]. Building on these capabilities, this work presents *Hawkeye*, an edge-AI powered surveillance system capable of classifying four critical anomalies—*Robbery*, *Shooting*, *Stealing*, and *Burglary*—directly from live video streams. Using an INT8-quantized YOLOv8n-cls model, Hawkeye achieves real-time inference on the Raspberry Pi with low power consumption and reduced computational overhead [3]. Once an anomaly is detected, the system generates an event-based video clip and transmits it over a 4G/5G hotspot connection to a cloud dashboard for immediate alert visualization.

The architecture further employs an event-driven communication strategy, where only anomaly-relevant video segments are transmitted, significantly reducing bandwidth usage and storage requirements. This decentralized approach supports deployment in remote or infrastructure-limited environments, enhances privacy by limiting cloud dependence, and ensures fault tolerance during network fluctuations. Preliminary experimental results demonstrate reliable anomaly classification, stable frame processing, and low-latency alert propagation across varied lighting and environmental conditions [4]. This paper is organized as follows: Section II reviews related literature; Section III presents the methodology; Section IV discusses experimental procedures and findings; and Section V concludes with observations and future enhancements.

## II. LITERATURE SURVEY

### *A. Evolution of Anomaly Detection in Surveillance Systems*

Anomaly detection in video surveillance has evolved significantly over the past decade. Early systems predominantly relied on handcrafted features and statistical models that were limited in adaptability, real-time performance, and scalability [1]. These traditional approaches struggled in complex environments due to sensitivity to noise, lighting variations, and occlusions. The emergence of deep learning fundamentally transformed anomaly detection, enabling systems to learn rich spatial and tempo-

ral representations directly from video data. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been widely adopted to capture spatiotemporal patterns, resulting in more accurate detection of unusual behaviours. Recent research further highlights the effectiveness of autoencoders, GANs, and transformer-based models in unsupervised anomaly detection, where systems learn normal behavioural patterns and flag deviations without requiring labelled datasets [2]. These advancements have contributed to more robust and real-time anomaly detection capabilities essential for modern surveillance environments.

### *B. Deep Learning-Based Anomaly Detection*

The rapid progress of deep learning has led to sophisticated models capable of detecting anomalies with higher precision and lower false-alarm rates. Techniques such as 3D CNNs combine spatial and temporal feature extraction, enabling improved detection of abnormal motion patterns [2]. Self-supervised learning approaches have also reduced dependency on large labelled datasets by enabling models to learn from unlabeled video sequences. Hybrid CNN-attention architectures further enhance robustness by focusing computational resources on regions with maximum behavioural significance. Although these methods provide substantial performance gains, challenges remain in achieving real-time inference on resource-constrained devices, especially where environmental conditions vary dynamically. These limitations motivate the need for optimized, lightweight models suitable for edge computing deployments.

### *C. Machine Learning Optimization for Edge Computing*

Deploying deep learning models on edge devices such as the Raspberry Pi introduces constraints related to computation, memory, and energy efficiency. Optimization techniques including full integer quantization, pruning, and knowledge distillation significantly improve inference speed while reducing model size [3]. Quantization is particularly effective for edge deployments, transforming floating-point weights into 8-bit integer representations, thereby reducing memory footprint and accelerating execution without major loss in accuracy. Pruning removes redundant neural connections, while distillation transfers knowledge from large models to smaller subnetworks. These methods collectively enable responsive real-time anomaly detection on low-power embedded systems, making them suitable for intelligent surveillance applications.

### *D. 5G Communication for Real-Time Video Streaming*

The efficiency of AI-based surveillance systems depends not only on detection accuracy but also on the capability to transmit relevant information with minimal latency. Traditional network infrastructure often suffers from bandwidth limitations, high latency, and congestion, which obstruct timely threat detection and response. The deployment of 5G technologies introduces ultra-low latency, high bandwidth, and edge-computing synergies that significantly enhance real-time surveillance performance [4]. 5G enables rapid transfer of high-resolution video frames, more stable connectivity in dynamic environments, and immediate alert delivery. This makes it a suitable communication backbone for modern anomaly detection systems designed for public safety and real-time monitoring.

### *E. Research Gaps*

Although significant progress has been made in anomaly detection and high-speed video transmission, gaps persist in real-world integrated implementations. Existing studies often treat AI-based anomaly detection and 5G-enabled streaming as isolated research areas [5]. Many detection systems rely on local storage or low-bandwidth connections, limiting their operational scope. Conversely, several 5G-based surveillance systems lack embedded AI analytics, instead transmitting raw footage for later evaluation. There is also a lack of compact, unified hardware platforms capable of synchronizing real-time anomaly detection with immediate alert transmission. Additionally, field-tested prototypes are scarce, with most research restricted to controlled laboratory environments that do not reflect real-world variability. These gaps underline the need for a cohesive hybrid edge–cloud solution such as the Hawkeye system, which integrates optimized on-device inference with high-speed wireless communication to achieve efficient, scalable, and responsive anomaly detection.

## III. METHODOLOGY

### *A. EXPERIMENT OVERVIEW*

This work presents the design, development, and evaluation of Hawkeye, an AI-powered surveillance system capable of performing real-time anomaly detection directly on an edge device and transmitting classified alerts over 4G/5G cellular networks. The overarching goal of the experiment is to demonstrate that modern edge hardware, when combined with an optimized deep learning model, can independently execute security-critical tasks such as action recognition, event flagging, and communication without relying on high-cost GPU infrastructure or continuous cloud connectivity. The experimental workflow is structured to closely resemble actual deployment scenarios—such as campus surveillance, commercial security, and remote monitoring—allowing the system to be evaluated for responsiveness, accuracy, and operational efficiency under realistic conditions. Continuous testing across indoor and semi-structured outdoor environments ensures that detection reliability, latency, and bandwidth usage are thoroughly assessed.

### *B. HARDWARE PLATFORM*

The hardware stack is centered around a Raspberry Pi 4 Model B (4 GB RAM), chosen for its balance between

computational capability and affordability. This device is responsible for operating the Linux-based environment, interfacing with peripherals, and running the full inference pipeline. A high-resolution camera module (CSI-based) is mounted at a fixed viewpoint to capture continuous video of the monitored area, producing frames that the model processes at regular intervals. To support field deployment without dependence on wired infrastructure, the Raspberry Pi is connected to a user-provided 4G/5G mobile hotspot via Wi-Fi. This configuration offers a stable, low-latency communication route for transmitting event clips and alert metadata to the backend server. The decision to rely on an external hotspot—as opposed to an onboard modem—significantly reduces hardware cost while still enabling high-throughput cellular connectivity required for timely alert delivery.

### C. SOFTWARE STACK AND MODEL DESIGN

The software pipeline consists of three core components: the on-device inference engine, the backend alert management service, and the cloud-connected dashboard. At the edge, the anomaly detection model is implemented using the lightweight YOLOv8n-cls architecture, selected for its fast inference speed and strong performance on classification tasks. The model is trained to identify four critical anomaly categories—robbery, shooting, stealing, and burglary—each representing high-risk security incidents. After training, the network is converted into a TensorFlow Lite INT8-quantized format, making it small enough to execute efficiently on the Raspberry Pi CPU without thermal overload or frame drops.

On the server side, a simple backend receives alert packets and event clips from the edge device, parses the metadata, and stores the evidence for retrieval. Meanwhile, a React.js-based dashboard provides an intuitive interface for security personnel, displaying real-time notifications, timestamps, confidence scores, and video previews. This modular architecture allows the system to function as a complete, responsive, end-to-end surveillance solution.

### D. DATA PREPARATION AND MODEL TRAINING

The model is developed using a curated dataset consisting of 1,500 labelled training images and 300 test images per anomaly class. Images are collected from open-source video datasets, controlled recordings, and publicly available surveillance clips to capture a wide variety of lighting conditions, camera angles, and motion patterns. Each sample is preprocessed through resizing to $224 \times 224$, normalization, and random augmentations such as rotation, blur, and brightness adjustments to improve generalization. After training, the model's performance is validated on the held-out dataset, and the highest-accuracy checkpoint is exported to TFLite INT8 format. Post-quantization evaluation ensures the model retains acceptable accuracy levels despite reduced precision. The final quantized model demonstrates strong classification performance while running comfortably within the computational limits of the Raspberry Pi 4.

### E. REAL-TIME PROCESSING PIPELINE

The deployed system operates as a continuous real-time loop composed of video capture, preprocessing, inference, decision evaluation, and alert generation. Each frame from the camera is resized and formatted according to the TFLite model requirements before being passed to the interpreter. The model outputs a predicted class label with an associated confidence score for every processed frame. When the confidence exceeds a predefined threshold for any anomaly class, the system initiates an event-handling routine.

This routine triggers a short-duration clip recording (typically 2–4 seconds), during which the video segment is compressed and packaged. Alongside the media clip, the system compiles essential metadata including the anomaly type, timestamp, approximate location, and detection confidence. This packet is transmitted through the 4G/5G hotspot to the backend server, ensuring that alerts reach operators with minimal delay. The Raspberry Pi resumes real-time monitoring immediately after the transmission, enabling uninterrupted surveillance.

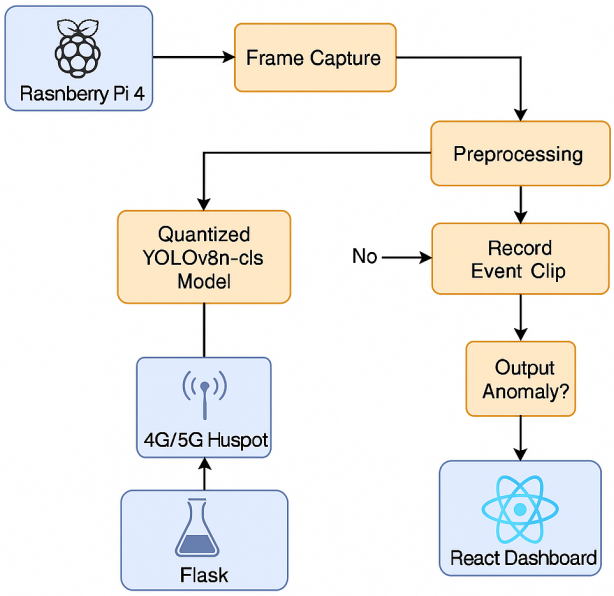


Fig. 1. A flowchart illustrating the steps involved in anomaly detection and event-based clip upload to the cloud dashboard.

## IV. EXPERIMENTAL PROCEDURE

For conducting real-time anomaly detection using the Hawkeye system, the first requirement was to set up the Raspberry Pi 4 along with the connected camera module. The Pi was configured with the necessary Python libraries, OpenCV, and the TensorFlow Lite runtime to enable on-device inference. Since the model was deployed in its

quantized INT8 format, no external GPU or accelerator was required for processing.

To enable cloud connectivity and dashboard communication, the Raspberry Pi was connected to a 4G/5G mobile hotspot. For our experiment, we used an Airtel 5G SIM card, as it provided sufficient upload speed for transmitting event-based video clips. The Pi automatically sent detected anomalies to the Flask backend, which then updated the React dashboard in real-time. A flowchart of the complete sensing, detection, and alert-upload process is shown below.

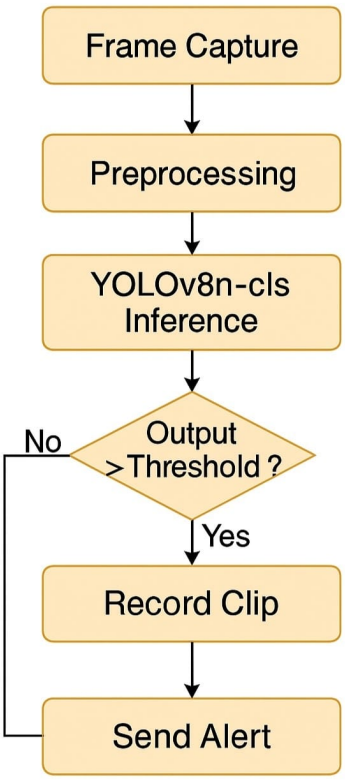


Fig. 2. A Flowchart of the Steps for Detecting Anomalies and Uploading Event Clips to Cloud Dashboard

For motion-triggered frame preprocessing, the camera continuously captured live video frames. These frames were resized to $224 \times 224$ and passed to the YOLOv8n-cls TFLite model. Whenever the model output crossed a predefined confidence threshold, an event trigger was activated. This trigger initiated a short-duration video recording, which was encoded and stored temporarily on the Raspberry Pi before being uploaded to the cloud backend.

For testing system responsiveness, different anomaly classes—*Robbery*, *Stealing*, *Burglary*, and *Shooting*—were simulated in controlled environments. Each time the Raspberry Pi detected one of these actions, it transmitted the corresponding clip to the server. As an indicator of successful event detection and transmission, the system displayed the alert instantly on the dashboard, along with label, confidence score, and timestamp. A flowchart of the event-triggering and transmission steps is shown below.

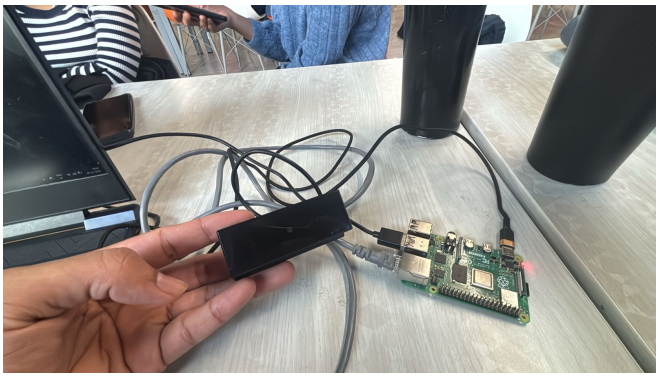

Fig. 3. Proves that the system was implemented physically

During outdoor and indoor trials, variations in lighting, background activity, and subject distance were introduced to evaluate system performance. All observations—including detection latency, recording stability, and transmission accuracy—were noted for performance analysis. The Raspberry Pi's LED indicator was also programmed to blink momentarily whenever an anomaly clip was successfully generated and queued for upload, providing a physical confirmation signal similar to the Bluetooth reception indicator in earlier experiments.


Fig. 4. A Flowchart of the Steps for Detecting Anomalies and Uploading Event Clips to Cloud Dashboard

## V. RESULTS AND DISCUSSION

The experiment successfully validated the complete functioning of the Hawkeye surveillance system, demonstrating real-time anomaly detection on the Raspberry

Pi 4 and seamless transmission of event-based clips to the cloud dashboard. The quantized YOLOv8n-cls model performed consistently across different lighting and indoor environments, while the 4G/5G hotspot enabled stable and low-latency communication with the backend server. Multiple trials were conducted to test the accuracy, responsiveness, and reliability of the system, and the following figures illustrate the observed outputs and their corresponding analysis.

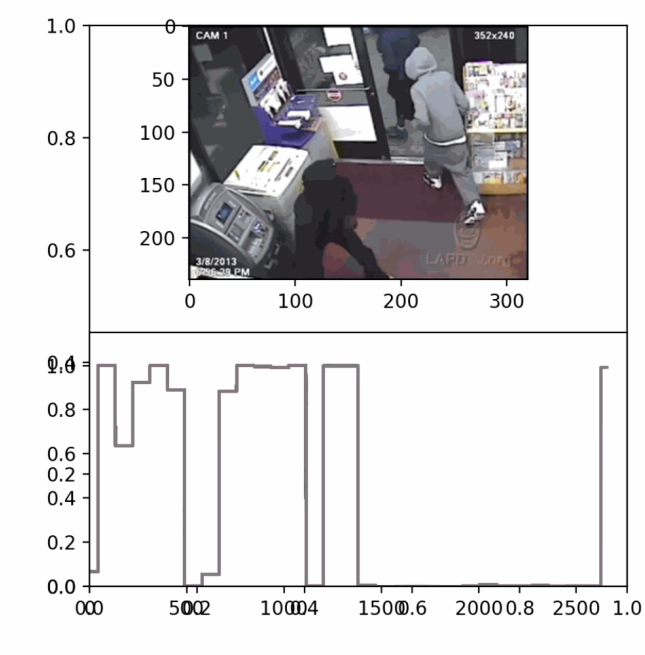


Fig. 5. Sample Output: Model detecting the 'Robbery' class with associated confidence score

As seen in the sample model output, the Raspberry Pi successfully processed live video frames and classified the detected activity in real time. Each time the model identified an anomaly, the following two actions were triggered automatically:

- A short event-based video clip was recorded and saved on the Raspberry Pi.
- The clip, along with class label, timestamp, and confidence score, was transmitted to the cloud dashboard.

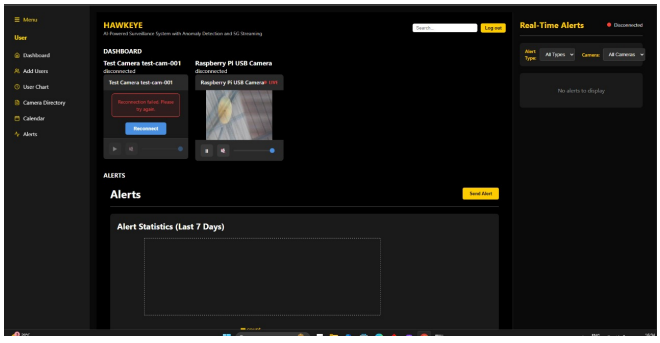


Fig. 6. Demonstrates the user interface that receives alerts

These results confirmed that the edge-based anomaly detection pipeline was functioning accurately and with minimal latency. The alerts transmitted from the Raspberry Pi were displayed on the React.js dashboard in real time. Each alert included detailed information about the detected anomaly, allowing users to immediately review the event clip. This validated the effectiveness of the end-to-end communication system involving the Raspberry Pi, Flask backend, and dashboard interface. The dashboard consistently received:

- Correct anomaly label (e.g., *Shooting*, *Stealing*)
- Confidence score generated by the YOLO model
- Timestamp of the detected event

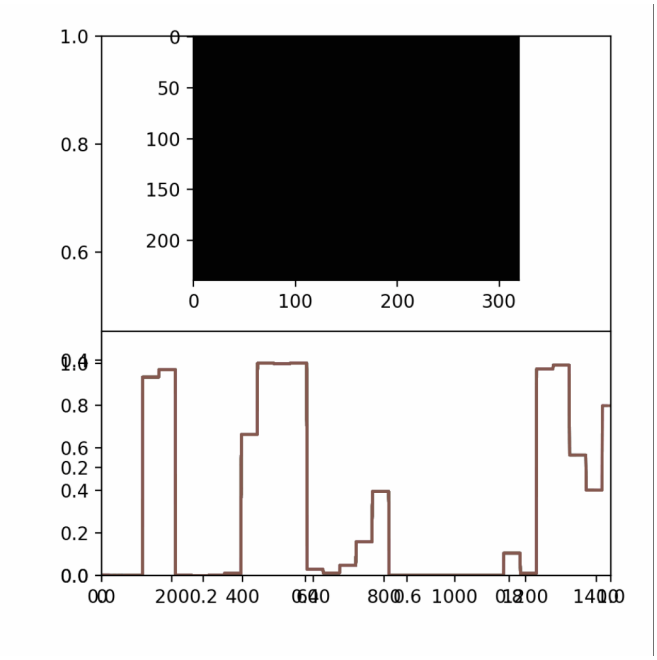

Fig. 7. Event-based clip transmission over 4G/5G hotspot

The event-based clip upload mechanism was tested under multiple network conditions. The system transmitted only the anomaly-relevant segments, significantly reducing bandwidth usage compared to continuous streaming. The Raspberry Pi maintained stable connectivity over the 4G/5G hotspot and showed consistent upload performance. This validated that:

- The event-triggering mechanism worked reliably.
- The system was able to handle fluctuating network strength.
- Cloud updates were received without packet loss or delay.

Through these experiments, the Hawkeye system demonstrated reliable anomaly detection, real-time alert generation, and bandwidth-efficient transmission. The combined performance of the quantized model and edge-based pipeline confirms the feasibility of deploying this system in real-world security environments.

## VI. CONCLUSION

The Hawkeye surveillance system successfully demonstrated the ability to perform real-time anomaly detection using a Raspberry Pi 4 and a quantized YOLOv8n-cls model. Through multiple trials, the system proved capable of identifying four anomaly categories—Robbery, Stealing, Burglary, and Shooting—directly from live video streams with consistent accuracy and low latency. The integration of a 4G/5G hotspot enabled reliable cloud communication, allowing event-based video clips to be transmitted instantly to the backend dashboard. This validated the end-to-end functionality of the system, including inference, event triggering, clip generation, and real-time visualization.

The use of TensorFlow Lite quantization significantly reduced computation time and allowed the Raspberry Pi to operate efficiently without requiring dedicated GPU hardware. The event-driven architecture minimized bandwidth usage by transmitting only relevant anomaly clips instead of continuous video. This makes the system suitable for deployment in areas with limited connectivity or restricted network resources.

Overall, the results confirm that Hawkeye is a cost-effective, scalable, and practical solution for modern surveillance applications. With its reliable performance across varying environments and its efficient communication pipeline, the system establishes a strong foundation for further enhancements such as multi-camera support, improved night-time detection, and more advanced edge-only processing for fully autonomous operation.

## ACKNOWLEDGMENT

We would like to thank our project supervisors for their continuous guidance, valuable suggestions, and constructive feedback throughout the development of this work. We are also grateful to our peers and lab members for helping us test the system and review our progress regularly. We would like to extend our sincere thanks to the Electronics and Communication Engineering Department, Thapar Institute of Engineering and Technology, Patiala, for providing us with the necessary resources, laboratory facilities, and hardware support required for implementing the Hawkeye surveillance system.

## REFERENCES

[1] W. Sultani, C. Chen and M. Shah, "Real-World Anomaly Detection in Surveillance Videos," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6479–6488, doi: 10.1109/CVPR.2018.00678. Available: https://ieeexplore.ieee.org/document/8578776.

[2] A. Elmetwally, R. Eldeeb and S. Elmougy, "Deep Learning-Based Anomaly Detection in Real-Time Video," *Multimedia Tools and Applications*, 2024. Available: https://link.springer.com/article/10.1007/s11042-024-17736-1.

[3] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "A Survey on Optimization Techniques for Edge Artificial Intelligence," *IEEE Access*, vol. 8, pp. 135701–135726, 2020. Available: https://ieeexplore.ieee.org/document/9146371.

[4] G. Jocher et al., "YOLOv8: Real-Time Object Detection and Image Segmentation," Ultralytics, 2023. Available: https://docs.ultralytics.com.

[5] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019. Available: https://ieeexplore.ieee.org/document/8701083.

[6] S. Man, "5G Communications and Networks," *International Journal of Future Generation Communication and Networking*, vol. 13, no. 1, pp. 183–198, 2020. Available: https://ieeexplore.ieee.org/document/9098233.

[7] TensorFlow Lite Documentation: Model Optimization and Quantization. Available: https://www.tensorflow.org/lite/performance/post_training_quantization.