A Hybrid AI Pipeline for Real-Time Aerial Video Analytics on Resource-Limited Edge Devices with Performance Profiling

S. J. Poornashree
School of ECE, REVA University, India
poornashreesj@gmail.com

Prashant V. Joshi
School of ECE, REVA University, India
prashanthvjoshi@reva.edu.in (corresponding author)

K. M. Sudharshan
School of ECE, REVA University, India
sudharshankm@reva.edu.in

Tony M. George
ADE, DRDO, India
tonymgeorge18@gmail.com

Received: 29 August 2025 | Revised: 16 September 2025 | Accepted: 24 September 2025

 DOI: https://doi.org/10.48084/etasr.14396

ABSTRACT

The increasing demand for real-time aerial video analytics across diverse applications poses significant challenges to the deployment in resource-constrained edge devices. This study proposes a hybrid AI method that integrates computer vision preprocessing with deep learning inference in a pipelined architecture. The proposed method facilitates stage-wise processing, intelligent frame selection, and lightweight inference, thus enhancing efficiency in embedded edge devices. The method involves implementing and profiling multiple deep learning models—such as DenseNet-121, Inception v3, MobileNet v2, EfficientNet B2, and YOLO v8 (s, m, l)—on a Raspberry Pi 5 edge device. AI performance was evaluated using accuracy, mean Average Precision (mAP), and model size, while hardware profiling metrics (Cycles, Instructions per Cycle, Cache Refs, CPU Time, Memory, and Time Latency) were obtained through the perf profiler. The results show that EfficientNet B2 outperforms all other models, achieving the highest mAP (97.2%), the lowest energy consumption (0.98 J/Inf), low time latency (278 ms), and a compact model size (4.34 MB), making it highly suitable for real-time aerial video analytics in constrained edge environments. Overall, this study demonstrates a cost-effective and energy-efficient solution for real-time video analytics on edge devices.

Keywords-AI; Unmanned Aerial Vehicles (UAVs); Tensorflow; YOLO

I. INTRODUCTION

The exponential growth in the use of Unmanned Aerial Vehicles (UAVs) across diverse domains, ranging from agriculture and surveillance to disaster management and urban planning, has generated a demand for real-time aerial video analytics. However, processing high-resolution video data in real time poses significant computational challenges, especially when executed on edge devices with limited hardware capabilities. Therefore, there is a critical need for innovative and resource-aware computational pipelines that can support complex analytics tasks directly on edge platforms without sacrificing accuracy or speed. The review in [1] highlights UAV-based real-time object detection applications in areas such as rescue and agriculture, focusing on edge computing, lightweight models, and the challenges of hardware limits and energy efficiency. Moreover, purely cloud-based solutions compromise the responsiveness and autonomy required in scenarios such as target tracking, anomaly detection, or rapid response operations [2]. These limitations underscore the importance of developing hybrid AI pipelines that intelligently balance the computational load between edge and cloud resources.

This study introduces a novel hybrid AI method specifically for aerial video analytics on resource-constrained edge platforms. The proposed method leverages a modular design that incorporates lightweight AI networks for on-device preprocessing, including object detection. The design also incorporates adaptive compression techniques and dynamic frame sampling strategies to optimize the trade-off between computational load and analytical accuracy. By orchestrating these elements, the method delivers robust real-time performance while operating within the strict limitations of edge computing environments.

Edge-AI video analytics has been extensively studied, with reviews highlighting its role in smart cities for surveillance, traffic management, and public safety, while also addressing challenges such as privacy, scalability, and device limitations, and recommending hybrid edge-cloud frameworks with lightweight models [3]. Edge video analytics integrates deep learning with edge computing to support applications such as surveillance, traffic control, and law enforcement [4]. Surveys in this domain review system architectures, frameworks, and datasets, while outlining key challenges and future research directions. Video analytics on the edge has also been thoroughly examined, particularly in overcoming limited on-device computing and cloud congestion, enabling low-latency, high-accuracy, and bandwidth-efficient performance [5]. Recent methods use deep learning combined with optimization techniques such as Mayfly Optimization to achieve high-accuracy object detection and classification in surveillance videos [6]. Research on deep neural network inference in resource-limited environments reviews methods for efficiently running models on constrained devices, focusing on trade-offs among latency, energy, and accuracy, while highlighting practical deployment strategies [7]. Edge-cloud video analytics frameworks have also been proposed to address latency and bandwidth constraints, improving accuracy for autonomous robotics applications [8].

Lightweight YOLO models have been evaluated for drone-based object detection, showing that TinyML-optimized versions achieve low latency, high accuracy, and efficient resource utilization on constrained platforms [9]. Similarly, in [10], an Edge-AI architecture was introduced to enable efficient real-time computer vision on smart devices, achieving higher accuracy and improved performance in applications such as smart surveillance. The latency-accuracy trade-offs in object detection on edge devices have been examined in detail, with strategies such as model compression and hardware acceleration shown to enable efficient real-time AI deployment on resource-constrained devices [11]. Further performance improvements have been demonstrated through multi-frame processing, which improves speed, memory efficiency, and power consumption in real-time object detection on edge devices [12]. Finally, energy-aware deep learning techniques using pruning, quantization, and hardware optimizations reduced energy consumption by up to 50% with minimal accuracy loss, supporting efficient real-time video analytics on edge devices [13].

Current research in aerial video analytics on edge devices often overlooks the critical need for hybrid AI methods that combine efficient video preprocessing with real-time inference, limiting their ability to handle complex video data under strict resource constraints. Additionally, although many studies focus on accuracy and speed, there is a significant lack of a comprehensive analysis of power consumption and energy-efficient strategies crucial for sustained operation on battery-powered edge platforms. Furthermore, existing evaluations of AI models and frameworks rarely provide comprehensive real-world analysis across multiple architectures on embedded hardware, hindering informed model selection for aerial analytics tasks. This work addresses these gaps by developing a hybrid AI method optimized for real-time aerial video analytics, systematically profiling power usage, and conducting detailed comparative evaluations of AI models, enabling energy-aware, robust, and high-performance deployment on resource-limited edge devices.

The main contributions of this work are as follows:

- Develops a hybrid AI method for real-time object detection and classification in aerial video streams, specifically optimized for execution on edge devices with constrained computational and memory resources.
- Investigates the detection accuracy and computational efficiency of deep learning models, including DenseNet-121, Inception v3, MobileNet v2, EfficientNet B2, and the YOLOv8 (s, m, l) family, through detailed hardware and performance metrics across AI models using metrics such as F1 score, accuracy, energy per inference, memory footprint, and total energy consumption on the embedded platform Raspberry Pi 5.
- Evaluates performance profiling of the edge device for AI inference and validates the proposed method using pretrained models and standard datasets in real-world aerial video analytics applications.

II. METHODOLOGY

The proposed hybrid AI–based aerial video analytics method, shown in Figure 1, begins with an input aerial video captured from UAVs or drones, containing diverse scenes such as urban infrastructure, traffic, or landscapes. This video is then decomposed into individual frames at a rate of 25 frames per second (fps). These frames undergo a sequence of preprocessing stages to enhance visual quality and optimize AI inference. Illumination inconsistencies are corrected using the Single Scale Retinex (SSR) model, improving feature visibility under varying lighting conditions. Noise is removed using Non-Local Means (NLM) denoising, preserving fine textures, followed by bilateral filtering to smooth images while retaining edges critical for detecting objects such as roads and buildings.

Quantitative evaluation of these enhancement techniques is performed using metrics such as Entropy, PSNR, SSIM, and CII, ensuring measurable improvements in clarity and detail. An entropy-based frame selection step discards low-information frames to reduce computational load, while saliency detection highlights visually significant regions, enabling ROI extraction to focus only on relevant content.

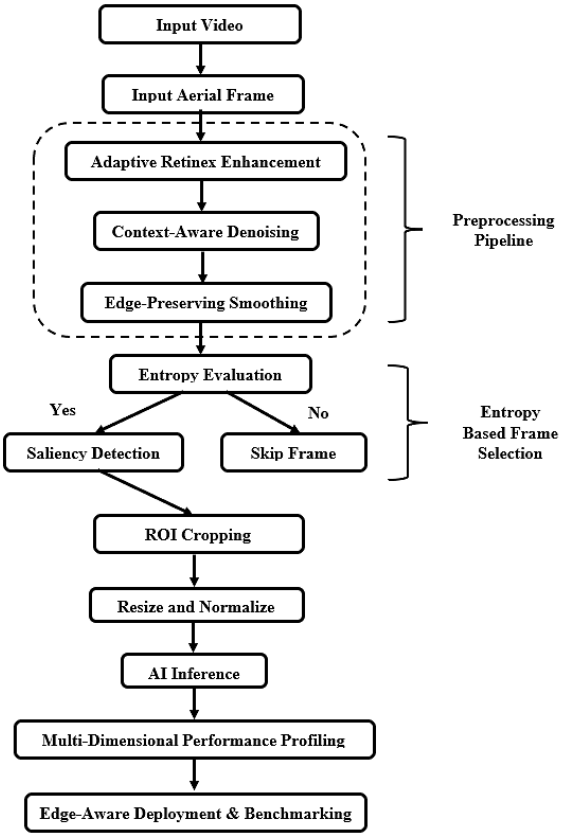

Fig. 1. Proposed method.

The extracted ROIs are resized, normalized, and fed into lightweight deep learning models such as MobileNet, EfficientDet, DenseNet, and YOLOv8 for efficient on-device inference via TensorFlow Lite. The models in .tflite and .pt formats for TensorFlow and YOLO models, respectively, were deployed on the Raspberry Pi 5, with the necessary dependencies installed to enable execution and performance profiling. Finally, detailed performance profiling evaluates models in terms of accuracy, mean Average Precision (mAP), model size, and energy consumption, guiding the selection of the most balanced model for edge deployment. Aerial object detection is achieved by optimizing algorithms, where edge-aware deployment involves techniques to ensure reliable performance under constraints such as limited power and connectivity [14].

The video dataset was obtained from Kaggle [15, 16], containing approximately 5,000 annotated aerial samples with a resolution of 650×480 at 25 fps. For model development, the dataset was divided into 70% for training, 20% for validation, and the remaining 10% for testing, ensuring a robust evaluation of model performance. The dataset includes diverse environmental conditions and five specific bridge types, annotated with bounding boxes to support both classification and localization tasks. The proposed method integrates preprocessing, intelligent frame filtering, ROI-focused AI inference, and model optimization to achieve real-time, resource-efficient aerial monitoring suitable for mission-critical applications.

A. Conventional vs Pipelining Processing

In the conventional AI processing flow, all operations—such as frame capture, enhancement, inference, and post-processing—are executed sequentially on each frame, resulting in a cumulative processing time per frame given by:

$$T_0 = t_{cap} + t_{prep} + t_{inf} + t_{post} \quad (1)$$

This sequential execution limits the system throughput, as the next frame cannot be processed until the current one completes the entire pipeline, as shown in:

$$T_1 = \max\left(t_{cap}, t_{prep}, p \cdot R \cdot t_{inf}, t_{post}\right) \quad (2)$$

where t_{cap} is the frame capture time, t_{prep} is the preprocessing time, t_{inf} is the inference time, t_{post} is the post-processing time, $p \in (0,1)$ is the frame filtering ratio and $R \in (0,1)$ is the ROI cropping factor.

Since stages are pipelined and inference operates on smaller, filtered inputs, the proposed method achieves significantly lower per-frame latency and higher throughput. For instance, if the conventional method takes $T_0 = 120$ ms per frame and the proposed method achieves $T_0 = 50$ ms per frame, this means that the hybrid pipeline can process frames about 58% faster than the sequential method. The proposed hybrid AI pipeline architecture adopts a true pipelined processing model, where different stages (e.g., entropy filtering, ROI detection, inference) are executed concurrently on different frames in a staged fashion. Once the pipeline is filled, a new output is produced at each stage interval.

III. PERFORMANCE METRICS

The studies in [17, 19] evaluated performance metrics on edge devices and were also considered in this work. Table I presents AI performance metrics evaluated using classification metrics, while Table II shows edge device performance analysis, conducted on a Raspberry Pi 5, which is equipped with a quad-core ARM Cortex-A76 CPU operating at a clock rate of 2.0 GHz. The reported metrics capture the computational load—reflected in CPU cycles, executed instructions, execution time, and energy per inference—and the efficiency of each AI model during runtime on this edge platform.

IV. RESULTS AND DISCUSSION

A. Performance Evaluation of Different AI Models

Figure 2 shows the input and output frames, respectively. The performance profiling of AI models is carried out by deploying them on the Raspberry Pi 5 (RPI) and analyzing their runtime behavior using the PERF profiler. This allows detailed measurement of key metrics on edge devices. The performance evaluation of different AI models based on these metrics is summarized in Table III [17]. The experiments were conducted using a dataset resolution of 640×480 with an input frame rate of 25 fps. Due to the complexity of the AI models and the hardware capabilities of the Raspberry Pi edge devices, the effective processing rate achieved was 4.60 fps.

TABLE I. PERFORMANCE METRICS OF AI MODELS

	Metric	Description	Formula / Unit	Significance
1	Accuracy	Overall correctness of the model predictions	$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$	Basic indicator of correct predictions
2	Precision	Correctness of positive predictions	$Precision = \frac{TP}{TP + FP}$	Important when false positives are costly
3	Recall	Coverage of actual positive instances	$Recall = \frac{TP}{TP + FN}$	Crucial when missing positives is risky (e.g., target detection)
4	F1-score	Harmonic mean of precision and recall	$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$	Balanced metric for uneven class distribution
5	Time latency	Total time required to process a single input to produce an output.	ms (milliseconds)	Low latency is critical for real-time applications
6	Frames per second (fps)	Number of frames processed per second	$\text{fps} = \frac{1}{\text{Inference time per frame (in seconds)}}$	Indicates real-time capability
7	Model size	Size of the trained model	MB (megabytes)	Affects deploy ability on memory-constrained edge devices
8	Memory usage	RAM used during model inference	MB / GB	Important for running efficiently on edge devices
9	mAP (mean Average Precision)	Average detection performance across classes and IoU thresholds	$mAP = \frac{1}{N} \sum_{c=1}^{N} AP_c$ (%)	Standard metric for object detection models

TABLE II. EDGE DEVICE PERFORMANCE ANALYSIS

	Metric	Description	Equation
1	Resolution	The number of pixels used to represent an image or video frame, expressed as width × height.	Resolution=Width (pixels)×Height (pixels)
2	Inferences Per Minute (IPM)	The number of times a model can process (infer) input data within one minute.	$IPM = \frac{Total\ Inferences}{Elapsed\ Time\ (s)} \times 6$
3	Frames per second	The number of video frames a model can process in one second.	$FPS = \frac{Total\ Frames\ Processed}{Elapsed\ Time\ (s)}$
4	CPU Cycles	Total cycles executed by the CPU during program runtime.	CPU Cycles = CPU Clock Rate × Program Execution Time
5	Instructions	Total number of instructions executed by the CPU.	Instructions = Sum of all executed instructions
6	Instructions Per Cycle (IPC)	Average number of instructions executed per CPU cycle.	$IPC = Instructions / CPU\ Cycles$
7	Time elapsed	Total wall-clock time taken to execute the program.	$Time\ Elapsed = End\ Time - Start\ Time$
8	Energy per Inference (J/Inf)	The average amount of energy consumed by the model to perform a single inference	$\frac{Total\ Energy\ Consumption}{Total\ Inferences}$

(a) (b)

Fig. 2. (a) Input frame, (b) Output frame.

Table III presents a detailed comparison of the models, with computational metrics obtained using the perf profiler on the Raspberry Pi 5 and power consumption measured using the Ruideng UM25C USB power meter to calculate the energy per inference. At 640×480 resolution, EfficientNet B2 proved to be the most balanced in accuracy, efficiency, resource utilization, and energy per inference, achieving accuracy of 99.7%, mAP of 0.972, F1-score of 0.967, compact size of 4.34 MB, lowest energy at 0.98 J/Inf, and efficient on-device profiling results 216 IPM, 4.60 FPS, 33M cycles, and 2.45% cache usage.

EfficientNet B2 is designed to optimize the balance between model size and feature extraction capability, enabling additional layers and filters to improve performance effectively without incurring excessive computational or memory overhead. Higher FPS allows faster detection of objects, but requires more computation and energy, while higher resolution improves detection accuracy by capturing more details but increases processing time and memory usage. In comparison, Inception V3 achieves strong multi-scale feature extraction but requires more cycles (61M) and 1.10 J/Inf. DenseNet121 improves recall (0.945) via dense connectivity but is larger (6.50 MB) and less energy-efficient (1.18 J/Inf). MobileNet V2 offers high throughput (287.2 IPM) through lightweight convolutions, but consumes 1.70 J/Inf. Finally, the YOLOv8 family, while competitive for detection, demands large memory (up to 420 MiB with 1075 MiB buffer/cache) and higher energy (1.39–1.94 J/Inf). Overall, EfficientNet B2 achieves the best balance of inference rate, resource usage, and energy consumption, making it the most suitable model for deployment on resource-constrained edge devices.

TABLE III. HARDWARE- PROFILING AND COMPUTATIONAL RESOURCE UTILIZATION OF AI MODELS

Model	Resolution	IPM	FPS	Cache refs (%)	Cycles	IPC	Time Latency(ms)	CPU time (µs)	Memory (MiB)	Buffer/ Cache (MiB)	Energy per inference (J/Inf)
EfficientNet B2	640×480	216.0	4.60	2.45	33,000,000	0.74	278	1.8	230	1040	0.98
Inception V3	640×480	177.9	2.96	2.95	61,000,000	0.58	338	1.6	215	835	1.10
DenseNet121	640×480	139.8	2.33	2.50	39,000,000	0.64	429	1.4	205	1030	1.18
MobileNet V2	640×480	287.2	4.79	2.89	56,000,000	0.61	209	1.3	220	820	1.70
YOLOv8s-cls	640×480	210.0	3.50	2.83	42,000,000	0.68	286	1.1	350	825	1.39
YOLOv8m-cls	640×480	165.2	2.75	2.70	39,500,000	0.70	364	1.4	355	830	1.52
YOLOv8l-cls	640×480	114.4	1.91	2.90	60,000,000	0.59	524	1.5	420	1075	1.94

B. Quantitative Evaluation of Image Enhancement Techniques

Quantitative evaluation is necessary to compare preprocessing methods and select the best balance between detail preservation, noise reduction, and visual quality. This ensures optimal input quality, directly enhancing detection and classification accuracy. Table IV provides a quantitative evaluation for Retinex enhancement, Non-local means denoising, and Bilateral filtering using entropy, PSNR, SSIM, and CII metrics. Retinex enhancement achieved the best overall performance, with the highest PSNR (10.86 dB), a relatively high SSIM (0.79), and a notable CII (43.38), demonstrating effective contrast enhancement with preserved structural details. Non-local means denoising produced slightly lower values, with a PSNR of 10.82 dB, SSIM of 0.71, and CII of 42.28, reflecting moderate noise suppression and acceptable detail retention. Bilateral filtering showed a comparable PSNR (10.77 dB), but the lowest SSIM (0.67) and CII (41.16), making it the least effective among the three methods.

TABLE IV. QUANTITATIVE EVALUATION OF PREPROCESSING TECHNIQUES

Metric	Original image	Retinex enhanced	Non-local means	Bilateral filtering
Entropy	5.16	4.84	4.67	4.73
PSNR (dB)	-	10.86	10.82	10.77
SSIM (0–1)	1	0.79	0.71	0.67
Contrast Improvement Index (CII)	64.59	43.38	42.28	41.16

C. Performance Comparison of AI Models

Table V compares the classification performance of EfficientNet B2, Inception V3, DenseNet121, MobileNet V2, and three YOLOv8 variants (s-cls, m-cls, l-cls), all trained on the same dataset. Accuracy, precision, recall, and F1-score assess classification effectiveness, model size indicates computational efficiency and edge deployment suitability, and mAP evaluates overall detection quality. EfficientNet B2 outperformed all models, achieving the highest accuracy (99.7%), precision (0.965), recall (0.970), F1-score (0.967), and mAP (0.972), with a compact size of 4.34 MB. Inception V3 and MobileNet V2 both reached 99.5% accuracy, with Inception V3 showing higher precision (0.927) and mAP (0.943). DenseNet121 provided strong recall (0.945) and F1-score (0.941), but was the largest (6.50 MB). Among YOLOv8 variants, YOLOv8l-cls achieved the best accuracy (98.6%) and F1-score (0.929). Overall, EfficientNet B2 offered the best balance, making it the most efficient for edge deployment.

TABLE V. COMPARISON OF CLASSIFICATION PERFORMANCE METRICS ACROSS AI MODELS

Model	Accuracy	Precision	Recall	F1-score	Model size (MB)	mAP
EfficientNet B2	99.7	0.965	0.97	0.967	4.34	0.972
Inception V3	99.5	0.927	0.938	0.932	6.1	0.943
DenseNet121	97.5	0.938	0.945	0.941	6.5	0.95
MobileNet V2	99.5	0.914	0.92	0.917	5.8	0.926
YOLOv8s-cls	95.2	0.899	0.903	0.901	4.9	0.911
YOLOv8m-cls	96.4	0.913	0.918	0.915	5.3	0.93
YOLOv8l-cls	98.6	0.925	0.933	0.929	6.7	0.942

D. Comparative Analysis of Existing Studies and Proposed Solution

In video analytics, a wide range of performance metrics can be evaluated, but this study focuses on the most significant parameters, including resolution, fps, latency, mAP, model size, energy per inference, and Inference Per Minute (IPM).

TABLE VI. COMPARATIVE ANALYSIS OF EXISTING STUDIES AND PROPOSED SOLUTION.

Ref./ Model	[18]	[19]	[20]	[21]	Proposed model
AI Model	Edge-YOLO	YOLOv8	YOLOv3-tiny	YOLOv7-tiny	EfficientNet B2
Resolution	-	640×640	-	640×480	640×480
fps	80.7	27	-	30	4.60
Time Latency(ms)	-	-	588	-	278
mAP (%)	78.8	86.20	64.59	-	97.2
AI model size (MB)	12.0	-	33.7	-	4.34
Energy per inference (J/Inf)	-	1.498	-	10	0.98
IPM	-	217	-	-	216
Edge device used	RK3588	Raspberry Pi 5	-	Jetson Xavier NX	Raspberry Pi 5

Table VI compares previous AI models with the proposed EfficientNet across these metrics. Edge-YOLO [16] achieves high fps (80.7) and moderate mAP (78.8%) with a 12 MB model on RK3588 but lacks efficiency and latency details. YOLOv8 [17] offers better mAP (86.20%) at 27 fps but higher energy use (1.498 J/inf). YOLOv3-tiny [18] had a lower performance overall, with low mAP (64.59%), high latency (588 ms), and a large model size (33.7 MB). YOLOv7-tiny [19] ran at 30 fps on Jetson Nano but consumed 10 J/inf without providing accuracy or latency data. In contrast, the proposed EfficientNet B2 achieves the highest mAP (97.2%)

with a small model size (4.34 MB), low latency (278 ms), low energy per inference (0.98 J/inf), and high inference rate (216 IPM) on a low-power Raspberry Pi 5, making it well-suited for real-time video analytics on edge devices.

V. CONCLUSION

This study demonstrated a structured hybrid AI method that integrates computer vision preprocessing with deep learning inference in a pipelined stage-wise architecture that effectively enables real-time aerial video analytics on resource-limited edge devices. By systematically profiling multiple AI models implemented using TensorFlow and YOLO on the Raspberry Pi 5 platform, this study establishes a clear performance–resource trade-off, demonstrating that EfficientNet B2 delivers the best overall balance with the highest mAP (97.2%), compact model size (4.34 MB), resolution (640×480), FPS (4.60), low time latency (278 ms), IPM (216.0), lowest energy consumption (0.98 J/Inf). The findings validate the pipeline's feasibility for field applications such as infrastructure monitoring, where low latency, low cost, energy efficiency, and high inference accuracy are critical. This work provides a practical deployment pathway for hybrid AI systems under stringent edge constraints, bridging the gap between theoretical AI capabilities and real-world operational demands. The proposed model is cost-effective, energy efficient, and well-suited for real-time video analytics on edge devices.

ACKNOWLEDGMENT

The authors acknowledge the support extended by ARDB, DRDO, in funding this research project titled "Design and Implementation of Image-Based Target Identification Using Open Core SOC on FPGA," Project No.1072066. In addition, the authors express their sincere gratitude to the School of Electronics and Communication Engineering, REVA University, for their invaluable support and guidance during this research.

REFERENCES

[1] Z. Cao, L. Kooistra, W. Wang, L. Guo, and J. Valente, "Real-Time Object Detection Based on UAV Remote Sensing: A Systematic Literature Review," *Drones*, vol. 7, no. 10, Oct. 2023, Art. no. 620, https://doi.org/10.3390/drones7100620.

[2] A. Koubaa, A. Ammar, M. Alahdab, A. Kanhouch, and A. T. Azar, "DeepBrain: Experimental Evaluation of Cloud-Based Computation Offloading and Edge Computing in the Internet-of-Drones for Deep Learning Applications," *Sensors*, vol. 20, no. 18, Jan. 2020, Art. no. 5240, https://doi.org/10.3390/s20185240.

[3] E. Badidi, K. Moumane, and F. E. Ghazi, "Opportunities, Applications, and Challenges of Edge-AI Enabled Video Analytics in Smart Cities: A Systematic Review," *IEEE Access*, vol. 11, pp. 80543–80572, 2023, https://doi.org/10.1109/ACCESS.2023.3300658.

[4] R. Xu, S. Razavi, and R. Zheng, "Edge Video Analytics: A Survey on Applications, Systems and Enabling Techniques," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2951–2982, 2023, https://doi.org/10.1109/COMST.2023.3323091.

[5] T. Bai, H. Zhao, L. Huang, Z. Wang, D. I. Kim, and A. Nallanathan, "A Decade of Video Analytics at Edge: Training, Deployment, Orchestration, and Platforms," *IEEE Communications Surveys & Tutorials*, 2025, https://doi.org/10.1109/COMST.2025.3563377.

[6] V. Saikrishnan and M. Karthikeyan, "Mayfly Optimization with Deep Learning-based Robust Object Detection and Classification on Surveillance Videos," *Engineering, Technology & Applied Science Research*, vol. 13, no. 5, pp. 11747–11752, Oct. 2023, https://doi.org/10.48084/etasr.6231.

[7] D. Ngo, H. C. Park, and B. Kang, "Edge Intelligence: A Review of Deep Neural Network Inference in Resource-Limited Environments," *Electronics*, vol. 14, no. 12, Jan. 2025, Art. no. 2495, https://doi.org/10.3390/electronics14122495.

[8] Y. Wang, W. Wang, D. Liu, X. Jin, J. Jiang, and K. Chen, "Enabling Edge-Cloud Video Analytics for Robotics Applications," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1500–1513, Apr. 2023, https://doi.org/10.1109/TCC.2022.3142066.

[9] M. Bakirci, "Performance evaluation of low-power and lightweight object detectors for real-time monitoring in resource-constrained drone systems," *Engineering Applications of Artificial Intelligence*, vol. 159, Nov. 2025, Art. no. 111775, https://doi.org/10.1016/j.engappai.2025.111775.

[10] R. M. A. Latif, K. Yoshigoe, N. Jamal, Y. Zhao, F. Ullah, and J. Ahmad, "Edge-AI Architecture for Real-Time Computer Vision in Smart MediaApplications," in *2025 IEEE International Conference on Pattern Recognition, Machine Vision and Artificial Intelligence (PRMVAI)*, June 2025, pp. 1–6, https://doi.org/10.1109/PRMVAI65741.2025.11108373.

[11] C. Joshua *et al.*, "Latency-Accuracy Trade-off Analysis in Edge-Based Object Detection Pipelines," *Advances in Engineering Software,* vol. 213, pp. 1–13.

[12] S. Kim, C. Kim, and S. Kim, "Improving Performance of Real-Time Object Detection in Edge Device Through Concurrent Multi-Frame Processing," *IEEE Access*, vol. 13, pp. 1522–1533, 2025, https://doi.org/10.1109/ACCESS.2024.3520240.

[13] M. E. Isenkul, "Energy-aware deep learning for real-time video analysis through pruning, quantization, and hardware optimization," *Journal of Real-Time Image Processing*, vol. 22, no. 3, June 2025, Art. no. 125, https://doi.org/10.1007/s11554-025-01703-0.

[14] P. V. Joshi, K. M. Sudharshan, M. G. Asuti, S. J. Poornashree, and S. K. Research, "Real-Time Aerial Image Edge Detection with MicroWatt Power ISA on FPGA," in *2025 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, Jan. 2025, pp. 1–6, https://doi.org/10.1109/IITCEE64140.2025.10915495.

[15] "Annotated (Yolov8) Bridge Image for Detection." Kaggle, [Online]. Available: https://www.kaggle.com/datasets/sarahhdd/bridge.

[16] "TDS-Satellite Images for Computer Vision Tasks." Kaggle, [Online]. Available: https://www.kaggle.com/datasets/dipensaini/sateelite-images-for-computer-vision-tasks.

[17] K. P. Nandini and G. Seshikala, "Efficient ECG Arrhythmia Detection on FPGA using Machine Learning and Fiducial Windowing," *Engineering, Technology & Applied Science Research*, vol. 15, no. 2, pp. 21100–21105, Apr. 2025, https://doi.org/10.48084/etasr.9589.

[18] J. Li and J. Ye, "Edge-YOLO: Lightweight Infrared Object Detection Method Deployed on Edge Devices," *Applied Sciences*, vol. 13, no. 7, Jan. 2023, Art. no. 4402, https://doi.org/10.3390/app13074402.

[19] L. Rey *et al.*, "A Performance Analysis of You Only Look Once Models for Deployment on Constrained Computational Edge Devices in Drone Applications," *Electronics*, vol. 14, no. 3, Feb. 2025, Art. no. 638, https://doi.org/10.3390/electronics14030638.

[20] B. Huang, H. Lin, Z. Hu, X. Xiang, and J. Yao, "An improved YOLOv3-tiny algorithm for vehicle detection in natural scenes," *IET Cyber-Systems and Robotics*, vol. 3, no. 3, pp. 256–264, 2021, https://doi.org/10.1049/csy2.12029.

[21] R. C. Camara de M. Santos, M. Coelho, and R. Oliveira, "Real-time Object Detection Performance Analysis Using YOLOv7 on Edge Devices," *IEEE Latin America Transactions*, vol. 22, no. 10, pp. 799–805, July 2024, https://doi.org/10.1109/TLA.2024.10705971.