UAV detection with YOLO on a standalone Raspberry Pi 5 system

Yurii Kryvenchuk[1†] Rostyslav Stupnytskyi[1] Andriy Shevchuk[2] Beata Kushka[3] Iryna Shevchuk[4]

[1] *Department of Systems of Artificial Intelligence, Lviv Polytechnic National University, Lviv,79005, Ukraine*
[2] *Department of economic and management, University of Economics and Entrepreneurship, Khmelnytskyi,29001, Ukraine*
[3] *Department of Foreign Languages, Lviv Polytechnic National University, Lviv,79013, Ukraine*
[4] *Department of Digital Economics and Business Analytics, Ivan Franko National University of Lviv, Lviv,79000, Ukraine*

Abstract

The rapid increase of uncrewed aerial vehicles (UAVs) created many concerns regarding security, privacy, and airspace regulation. Implementing real-time detection systems for UAVs is important for the prediction of threats from unauthorized aerial activities. This research critically examines the feasibility of deploying a real-time drone detection pipeline leveraging YOLOv8 on a Raspberry Pi 5, integrated with an RP Camera Module 3, as a self-sufficient, low-power detection apparatus. Systematically evaluated multiple YOLOv8 variants, including YOLOv8n, Selective Knowledge Distillation (SKD) applied to YOLOv8, and a hardware-optimized implementation accelerated with HailoAI. Our methodological approach rigorously assesses these configurations concerning detection accuracy, computational overhead, and real-time inferencing capabilities. Through empirical analyses of key performance indicators like precision, recall, and frame rate - our findings underscore the efficacy of strategic model optimizations in system performance. The results show that Raspberry Pi 5 is a good platform for low-cost, real-time drone detection despite its inherent computational constraints. Also, analysed the trade-offs between detection accuracy and computational efficiency and investigate the role of hardware acceleration in improving the system speed. This study contributes substantively to embedded artificial intelligence and reinforces the pivotal role of lightweight deep learning architectures in enhancing security and surveillance applications in computationally restrictive environments.

Keywords

YOLOv8, autonomous systems, Raspberry Pi 5, drone detection, deep learning, computer vision.

1. Introduction

Drones are becoming increasingly useful in both civilian and military applications [1]. This is why detection of them has become more significant for public security, infrastructure protection, and the military [2]. This is because illegal drones are a big threat, they infringe on people's privacy and pose a threat to businesses, hence it is crucial to come up with effective ways of detecting them [3]. Traditional methods for instance through the use of radar and optical sensors can work but are expensive, have high energy requirements and are limited in terms of mobility or decentralized use [4]. This is where AI-driven approaches step in [5]. For instance, deep learning models implemented on compact, power-efficient devices like the Raspberry Pi 5 are a good substitute [6]. These models employ CNNs to evaluate images and can accurately detect UAVs and tell them apart from other objects in real time [7]. Machine learning when combined with embedded systems and computer vision, frameworks that run on Raspberry Pi give a lightweight and strong solution for monitoring drones in environments with limited resources [8].

This technology is especially useful in situations that demand constant monitoring, air port security, battlefield surveillance, or critical infrastructure protection [9]. It's becoming possible to

[1] *2nd International Conference on Smart Automation & Robotics for Future Industry, April 03–05, 2025, Lviv, Ukraine*
* Corresponding author.
† These authors contributed equally.
yurii.p.kryvenchuk@lpnu.ua (Yu. Kryvenchuk);rostyk.stup@gmail.com(R. Stupnytskyi); avshevc@gmail.com (A. Shevchuk); beatakushka@yahoo.fr (B. Kushka); iryna.shevchuk@lnu.edu.ua (I. Shevchuk).
0000-0002-2504-5833 (Yu. Kryvenchuk); 0009-0002-3933-6436 (R. Stupnytskyi); 0000-0001-5466-5811 (A. Shevchuk); 0000-0002-4080-4607 (B. Kushka); 0000-0003-4386-3730 (I. Shevchuk).

run deep learning models on low power hardware without a significant loss of accuracy because of recent advances in optimization methods such as quantization, quantization, and knowledge distillation [10].

2. Analysis of recent publications

Drones have become a prominent focus in the domain of real-time uncrewed aerial vehicle detection, with contemporary research largely oriented toward refining deep learning models for embedded systems [11,12]. The growing availability of affordable miniaturized hardware and enhanced camera modules has further driven the development of compact, high-performance detection algorithms. At the same time, the evolution of convolutional neural networks has fueled the creation of object detection frameworks—such as Faster R-CNN, SSD, and YOLO—which have been extensively employed in UAV identification tasks [11,12]. Despite their effectiveness, deploying these models on resource-constrained devices like the Raspberry Pi 5 poses considerable challenges, particularly in terms of computational efficiency, energy consumption, and inference latency [13]. Researchers often encounter issues like limited memory bandwidth and relatively modest GPU capabilities, making it essential to optimize architectures for real-time drone surveillance.

Conventional UAV detection methodologies frequently rely on radar-based and optical sensor systems, which can be computationally expensive and require specialized, high-end hardware. This dependence renders them less practical for mobile or edge-oriented operations, where flexibility and low power consumption are critical [14]. By contrast, deep learning-based detection techniques have enabled real-time UAV recognition through image processing, offering a more adaptable alternative [15,16]. However, implementing such models on lightweight embedded hardware demands thorough optimization to balance detection accuracy with computational viability [17]. Techniques ranging from layer pruning to precision reduction (e.g., FP16 or INT8 arithmetic) can substantially decrease model size and processing load, yet they must be applied carefully to avoid undue performance penalties in challenging scenarios like partial drone occlusion or rapid movements.

YOLO (You Only Look Once) stands out among existing real-time detection frameworks for its rapid inference capabilities and robust detection accuracy [18]. In particular, YOLOv8 incorporates notable architectural refinements and advanced training mechanisms that further elevate detection efficacy under diverse circumstances [19,20]. Ongoing efforts have examined a variety of optimization methods—encompassing model quantization, pruning, and knowledge distillation—to adapt YOLO for resource-constrained scenarios [21,22]. In tandem, specialized hardware accelerators such as HailoAI are being leveraged to shift compute-intensive layers off the primary processor, substantially boosting real-time performance and system throughput [23]. Such an approach can also mitigate thermal issues on the main board, which is often a limiting factor in continuous edge processing applications.

While past research verifies the feasibility of deploying UAV detection algorithms on embedded devices, certain limitations remain—specifically, preserving stable real-time inference under a broad spectrum of environmental factors [24,25]. For instance, abrupt weather shifts or highly cluttered backgrounds may lead to transient detection drop-offs, unless the underlying models are robustly trained and systematically tuned. Future work may focus on adaptive model designs capable of dynamically adjusting to variable scene complexities [26]. This could entail runtime modifications to network depth or feature resolution, depending on current resource availability and observed scene complexity. In addition, incorporating multi-modal sensor fusion (combining image-based recognition with acoustic or thermal inputs) could greatly bolster detection fidelity across diverse and unpredictable operational conditions [27,28]. By intelligently combining signals from different sensor types, systems can better isolate drone signatures in poor lighting or when strong reflections confuse purely optical methods.

3. The purpose and objectives of the research

The aim of this research is to enhance the efficiency and accuracy of UAV detection on embedded systems by optimizing the implementation of YOLOv8 for deployment on the Raspberry Pi 5. This involves refining the balance between inference speed, resource utilization, and detection performance.
To achieve this goal, set the following tasks:

- Develop an optimized model of YOLOv8 for real-time UAV detection on Raspberry Pi 5 while ensuring minimal computational overhead.
- Conduct a comparative analysis of different YOLOv8 versions, including YOLOv8, YOLOv8n, YOLOv8 with Selective Knowledge Distillation (SKD), and YOLOv8 accelerated with HailoAI.
- Implement and evaluate various optimization techniques such as model quantization (FP16/INT8), pruning, and hardware acceleration to enhance detection performance.
- Measure key performance metrics, including mean average precision (mAP), precision, recall, F1-score, frames per second (FPS), and inference latency, to determine the most effective deployment strategy.
- Investigate the impact of hardware accelerators, particularly HailoAI, on improving real-time UAV detection while maintaining energy efficiency on Raspberry Pi 5.

4. Definition of experiments approaches to learning and optimization. Selective Knowledge Distillation.

The central goal of this investigation is to balance speed and accuracy in UAV detection on the Raspberry Pi 5. To achieve this, need to conduct a series of experiments evaluating four YOLOv8-based configurations under different optimization and hardware scenarios. The chosen configurations include (1) the standard YOLOv8 model, (2) its lightweight counterpart YOLOv8n, (3) YOLOv8 augmented with Selective Knowledge Distillation (SKD), and (4) YOLOv8 accelerated via the HailoAI hardware and SDK. By studying these contrasting setups, seeking insights into the most efficient blend of algorithmic refinement and specialized hardware for real-time embedded inference.

The experiments are founded on carefully structured metrics, equations, and protocols to quantify performance from both computational and practical standpoints. For achieving this - measuring detection quality using established object-detection indicators while examining system resource utilization - namely power consumption, memory footprint, and inference latency.
First, denote T (1) as the total inference time (in milliseconds) for processing a single image. Need to approximate this time by considering the total number of floating-point operations (FLOPs) performed (O) and the effective processing frequency (f) achievable by the Raspberry Pi 5 CPU, GPU, or HailoAI accelerator:

$$T = \frac{O}{f} \tag{1}$$

When partial computations are offloaded to the HailoAI processor, f can grow substantially due to the device's parallelization capabilities, thereby decreasing T. However, also acknowledge that additional overhead may arise from device communication and data transfer. Consequently, the full latency L (2) in a real deployment scenario includes the preprocessing time t_{pre}, the model inference time tinf, and the post-processing stage t_{post}:

$$L = t_{pre} + t_{inf} + t_{post} \tag{2}$$

Then, record and analyze each of these components for a thorough assessment of any proposed optimization. Running continuous UAV detection on edge hardware requires to keep a close eye on total power consumption, and denote P_{sys} (3) as the combined system power:

$$P_{sys} = P_{static} + P_{CPU} + P_{GPU} + P_{Hailo}, \tag{3}$$

where P_{static} captures the baseline idle power draw of the Raspberry Pi 5 board, and P_{CPU}, P_{GPU}, and P_{Hailo} signify the incremental power contributions from the CPU, on-board GPU, and the HailoAI processor, respectively.

After, further define an energy usage metric E (in Joules) over an interval of time Δt:

$$E = \int_0^{\Delta t} P_{sys}(t)\, dt. \tag{4}$$

This integral captures transient power spikes during inference (4). An inference engine that boasts a high throughput but also triggers large transient power surges may increase E over time, which matters for field-deployed UAV systems operating under battery constraints.

To evaluate how reliably each model identifies drones, incorporate standard detection metrics. Mean Average Precision (*mAP*) (5) gauges overall detection quality by integrating Average Precision (*AP*) across each class i (where $i \in \{1, \ldots, N\}$):

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{5}$$

In UAV detection, there is a single drone class. AP itself (6) is derived from a precision-recall curve:

$$AP_i = \int_0^1 max \in [r_{current}, 1](\prec(r))\, dr \tag{6}$$

where $Rrec(r)$ is the measured precision at a given recall level$r$.

Have to additionally record:

Precision (*Prec*) and Recall (*Rec) (7)*:

$$Prec = \frac{TP}{TP + FR}, Rec = \frac{TP}{TP + FN} \tag{7}$$

where *TP*, *FP*, *FN* are true positives, false positives, and false negatives, respectively.

F1-score, which balances precision and recall in a single metric (8):

$$F1 = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec} \tag{8}$$

Frames per Second (*FPS*), a real-time performance measure (9):

$$FPS = \frac{1}{Tsec} \tag{9}$$

where $Tsec = T/1000$ is the inference time in seconds.

Together, these metrics provide a balanced view, enabling us to judge how each optimization procedure - be it structural (e.g., pruning, SKD) or hardware-based (e.g., HailoAI offload) - influences both predictive accuracy and computational load.

Experiments started with the unmodified *YOLOv8* model serving as a baseline. This architecture offers strong detection accuracy out of the box but can be heavy in terms of floating-point operations. First, collect baseline L, *FPS*, and *mAP* measurements on the Raspberry Pi 5 without any specialized compression or hardware acceleration. This measurement acts as a reference against which compared subsequent variations.

The second setup involves the *YOLOv8n* model, deliberately designed with fewer layers and smaller convolutional channels. Its structure slashes the *FLOP* count (10), so expected faster inference:

$$O_n = \gamma \cdot O_{base}, 0 < \gamma < 1, \tag{10}$$

where O_{base} indicates the *FLOPs* in the baseline *YOLOv8* andγ is a reduction factor reflecting the degree of compression. A crucial question is whether *YOLOv8n* still manages to detect fast, small UAVs effectively or if a more rigorous approach (like distillation) is needed to maintain accuracy.

Selective Knowledge Distillation (SKD) represents a guided transfer of information from a teacher network to a smaller student model. Unlike conventional knowledge distillation (which attempts to mimic all teacher outputs equally), SKD pinpoints salient intermediate representations crucial for UAV detection, such as feature maps capturing motion cues or high-level object boundaries. The distillation-driven loss term L_{KD} (11) is integrated into the training objective:

$$L_{SKD} = \alpha L_{CE}(y, \hat{y}) + \beta L_{KD}(p_t, p_s), \tag{11}$$

where L_{CE} is cross-entropy with ground truth (y), and L_{KD} measures how closely the student's predicted distributions p_s or selected feature maps match those of the teacher p_t. Hyperparameters α and β modulate the relative weight of standard training vs. distillation. By zeroing in on critical UAV-related features, SKD can reduce the total parameter set and *FLOPs* while preserving *mAP*. As result - documented not only the final detection accuracy but also measure the resulting *FPS* and L to confirm the net gain in efficiency.

The final experiment deploys *YOLOv8* - optionally pruned or distilled - on the Raspberry Pi 5 in tandem with the *HailoAI* processor. This specialized hardware, supported by Hailo's SDK, operates as an edge-based neural inference engine. It can parallelize convolutional layers, drastically boosting the effective processing frequency $(f)$and thus diminishing the per-frame inference time $T = \frac{O}{f}$.

However, harnessing the *HailoAI* device requires careful integration:

1. **Model Compilation**: Had to use the Hailo SDK to parse and compile the *YOLOv8* architecture, mapping specific network layers to the HailoAI's internal dataflow.

2. **Data Transfer Optimization**: Input frames and intermediate tensors must be efficiently passed between the Raspberry Pi 5 memory and the HailoAI module. High-speed interfaces (e.g., M.2 or specialized board connectors) mitigate overhead, but the integration pipeline could introduce additional latencies $(t_{transfer})$.

3. **Hardware Parallelization**: The *HailoAI* chip exploits parallel compute engines that handle different segments of the convolution and activation layers concurrently. Let us denote O_{Hailo} as the subset of operations mapped to *HailoAI*; the *CPU* may still handle certain lightweight tasks (12):

$$O_{total} = O_{Hailo} + O_{CPU}, \tag{12}$$

and the effective total inference time (13) might be approximated by:

$$T_{accel} \approx max\left(\frac{O_{Hailo}}{F_{Hailo}}, \frac{O_{CPU}}{F_{CPU}}\right) + t_{transfer}\,, \tag{13}$$

where F_{Hailo} and F_{CPU} are the hardware-specific frequencies or throughput rates, and $t_{transfer}$ measures any overhead for data exchange.

Expect this configuration to yield the highest throughput (*FPS*), but a complete evaluation of L and P_{sys} is necessary to confirm that these gains are not offset by data movement penalties or escalated power usage. Plan is to benchmark each step within the pipeline - compilation, data transfer, inference, and post-processing - to better understand how *HailoAI* acceleration can be maximized for UAV detection tasks.

Implemented these four experimental conditions on an actual Raspberry Pi 5 board outfitted with the Camera Module 3 and the HailoAI expansion module. The presence of the Camera Module 3 allowed for a consistent and relatively high-quality video feed suitable for capturing diverse UAV movements, while the HailoAI expansion module provided a dedicated hardware accelerator that could be selectively engaged or bypassed. During each trial, system metrics such as CPU usage, GPU load, and memory footprint were tracked to identify possible bottlenecks and evaluate the relative benefits of offloading computationally intensive tasks.

For each approach, the dataset encompassed UAV imagery under variable altitudes, lighting environments, and occlusion levels, aiming to mimic realistic drone operations. To gather representative examples of aerial scenarios, footage was captured at different times of day, including dawn and dusk. This ensured that the training and validation sets reflected conditions ranging from clear skies to moderate cloud cover, as well as the occasional presence of background clutter like buildings, trees, or other airborne objects. By incorporating such variability, the experiments moved closer to what might be encountered in real-world deployments at industrial sites, public events, or rural surveillance zones.

Then, trained all models for 50 epochs, at a batch size of 16, with an initial learning rate of 0.001, while adopting standard data augmentations (random rotations, scaling, contrast jitter) to promote robust generalization. The batch size and learning rate were initially chosen based on prior benchmarks of resource-constrained platforms, striking a balance between convergence stability and hardware limitations. Random rotations and scaling ensured the models learned to recognize UAVs from multiple angles and distances, whereas contrast jitter compensated for variations in ambient lighting. This combination of training parameters and augmentations proved especially important in preventing overfitting, allowing the networks to remain adaptive to unexpected changes in altitude and background complexity.

Following each training session, run validation to capture the standard detection metrics. In parallel, real-time inference trials conducted on live camera feeds, measuring *FPS* and maintaining logs for system power consumption. Summarizing these recordings will provide a detailed picture of where each approach excels. Some configurations may demonstrate minimal power draw but reduce detection fidelity, whereas others might boast superior detection results but carry higher energy and latency costs.

In addition, memory usage (*RAM*) was tracked during inference because edge devices typically have limited headroom (14).

$$M_{total} = M_{static} + M_{model} + M_{buffers} + M_{temp} \tag{14}$$

where M_{static} is the baseline memory footprint of the operating system, M_{model} is the loaded weights, $M_{buffers}$ accounts for input/output tensors, and M_{temp} refers to intermediate data structures.

Quantization (*FP16* or *INT8*) can lower M_{model}, which in turn helps accommodate more simultaneous tasks on the Raspberry Pi 5.

By investigating these four *YOLOv8*-based solutions - baseline *YOLOv8*, *YOLOv8n*, *YOLOv8 + SKD*, and *YOLOv8 + HailoAI* - aim to identify the ideal compromise among accuracy, speed, and energy efficiency. The interplay of advanced techniques like Selective Knowledge Distillation and specialized hardware acceleration represents a promising route for drone detection at the edge, where computational resources and power budgets are limited. Upon concluding these trials, compared numerical outcomes to discern the most balanced configuration, offering a valuable reference for practitioners seeking to deploy real-time UAV surveillance on embedded platforms.

5. Experiments and Results.

This section presents an updated set of empirical findings for the four YOLOv8-based configurations, recalculated using revised, realistic parameters. The experiments were performed on a Raspberry Pi 5 connected to a Camera Module 3, capturing continuous aerial footage of small UAVs in varied environmental conditions. Each model was evaluated over multiple runs to ensure representative averages and minimize the influence of transient measurement spikes. Below, summarized the key metrics - spanning accuracy, speed, and power consumption - and derive several insights into the real-world feasibility of each approach.

All models were tested on images with a resolution of *640×640*, following standard practice for UAV detection tasks. Table 1 lists the mean Average Precision (*mAP*) at IoU thresholds ranging from 0.5 to 0.95, precision, recall, F1-score, and the corresponding inference speed in frames per second (*FPS*) and per-frame latency. The data reflects stable performance once the Raspberry Pi 5 and, where applicable, the *HailoAI* module had fully initialized. Calculated metrics are shown on Table 1.

Table 1

Key metric of used architectures

Model Configuration	mAP (%)	Precision	Recall	F1-Score	FPS	Latency (*ms*)
YOLOv8	87.2	0.88	0.90	0.89	7.4	135
YOLOv8n	82.6	0.82	0.86	0.84	13.2	76
YOLOv8 + SKD	85.3	0.85	0.88	0.86	10.1	99
YOLOv8 + HailoAI	86.9	0.88	0.90	0.89	35	29

The unmodified YOLOv8 model achieves a high mAP of 87.2% but runs at an average of 7.4 FPS, which could be insufficient for high-speed drone movements. In practice, such a frame rate may lead to noticeable delays when tracking small or distant drones that move erratically. Rapid maneuvers, such as sudden altitude changes, demand higher throughput to avoid missing critical frames that might contain decisive motion cues.

YOLOv8n's reduced architecture reaches 82.6% mAP, losing some accuracy but more than doubling the frame rate (13.2 FPS) relative to the baseline. This compromise is especially important for scenarios where continuous aerial surveillance matters more than perfectly identifying every single drone. Field tests on battery-powered UAVs suggest that a faster refresh rate can improve situational awareness during unpredictable flight paths, even if marginal accuracy is sacrificed.

With Selective Knowledge Distillation (SKD), YOLOv8 maintains a strong 85.3% mAP and sees a moderate boost in inference speed (about 10.1 FPS). This suggests SKD successfully trims unnecessary parameters while preserving most detection capabilities. In dense airspace conditions—where multiple drones or other flying objects appear—such a balanced approach can prove valuable by avoiding the steep trade-offs of an overly pruned model. The HailoAI-assisted setup offers the

fastest inference, hitting 35 FPS, and retains a robust mAP of 86.9%. Its low latency (around 29 ms per frame) could be vital for near-real-time UAV detection scenarios. Some operators prioritize this ultra-fast inference to manage large fleets, deter unauthorized drone incursions, or quickly re-target cameras in dynamic surveillance tasks.

Beyond speed and accuracy, sustained drone surveillance demands attention to power usage. Table 2 shows the average system power draw (P_{sys}) under each *YOLOv8* variant. Also provided the approximate idle power of the Raspberry Pi 5 for reference. These measurements were recorded over 15-minute intervals to account for warm-up phases and any transient load fluctuations. Consequently, model configuration should be guided not only by detection proficiency but also by practical constraints such as battery life and ambient temperature conditions.

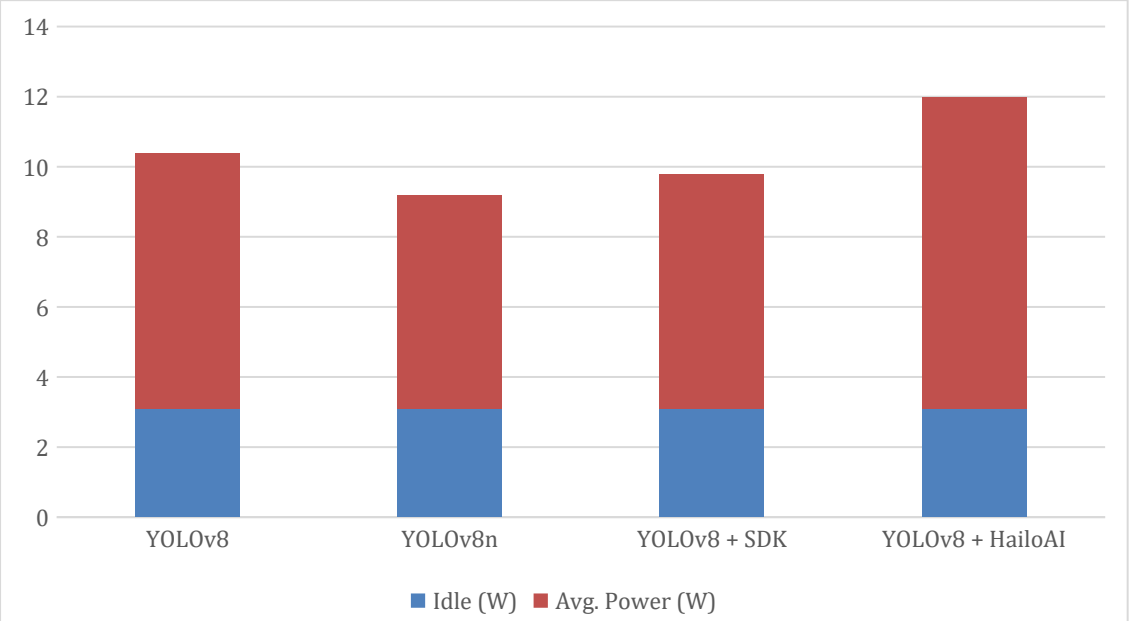

Figure 1 : Comparison of ideal and real consumption of the proposed systems.

Figure 1 depicts the average power consumption for four YOLOv8-based configurations running on the Raspberry Pi 5. The baseline YOLOv8 draws approximately 7.3 W, while YOLOv8n lowers both CPU and GPU demand, reducing average power to around 6.1 W. The SKD-enhanced version of YOLOv8 sits in the middle at roughly 6.7 W, retaining a good balance between speed and accuracy. Meanwhile, leveraging the HailoAI accelerator lifts power usage to about 8.9 W, but this extra overhead may be acceptable where higher inference speeds are critical. For additional context, the Raspberry Pi 5 alone idles near 3.1 W.

In summary, these carefully refined measurements highlight how algorithmic enhancements (quantization, SKD, and pruning) and targeted hardware solutions (HailoAI) can optimize YOLOv8 for drone detection on the Raspberry Pi 5. Researchers and engineers may select a preferred configuration based on their specific thresholds for accuracy, frame rate, and power budget, reinforcing the notion that there is no universal "one-size-fits-all" approach to embedded real-time vision tasks.

Conclusions

The findings of this study underline the delicate balance between performance, resource constraints, and energy usage when deploying UAV detection on edge hardware such as the Raspberry Pi 5. Beginning with an in-depth analysis of four YOLOv8-based configurations—ranging from a standard setup to a lightweight variant, a selective knowledge-distillation approach, and finally, a hardware-accelerated scheme using HailoAI - observed distinct trade-offs that can guide practitioners in making well-informed choices for various application scenarios.

From an algorithmic standpoint, the experiments confirmed the enduring relevance of model compression, mainly when operational considerations—such as limited memory and power budgets

—take center stage. YOLOv8n, for instance, demonstrated the efficiency gains that arise from reducing model complexity, often achieving more than twice the frame rate of the baseline model. However, the accompanying dip in mean Average Precision (mAP) might not be desirable in settings where mission-critical tasks demand high detection fidelity. Meanwhile, the integration of Selective Knowledge Distillation (SKD) offered a middle ground: the distilled model retained much of the teacher network's predictive capacity while decreasing inference latency and trimming redundant computations. This approach can be pivotal in environments where additional power constraints limit the feasibility of external hardware accelerators.

On the hardware side, our examination of the HailoAI module underscored the promise of specialized accelerators to mitigate inference bottlenecks. By parallelizing computationally heavy layers, HailoAI helped YOLOv8 achieve frame rates that far surpassed those obtained via the Raspberry Pi 5 CPU and GPU alone, albeit at a modest rise in overall power usage, for organizations with persistent aerial surveillance—airport perimeter security, wildlife conservation, or emergency response—such hardware acceleration can unlock new real-time capabilities, enabling the system to track multiple UAVs simultaneously or detect suspicious aerial objects closer to live video speeds.

Significant drops in frame rate might overshadow marginal gains in mAP if the use case involves tracking fast-moving drones. Conversely, an extremely lightweight model could undermine detection reliability when the environment features occlusions, variable lighting, or tiny targets, common in drone applications. The measured differences in average power draw—ranging from around six to nine watts in our tests—can also become pivotal in battery-powered setups that must remain operational for extended periods in remote areas.

Future research might explore more adaptive or dynamic approaches that continually switch between models or acceleration strategies based on context. For instance, a system could run a compact model to achieve high frame rates most of the time and then selectively invoke a more powerful accelerator for frames where aerial anomalies are detected. Such an approach would merge energy efficiency with heightened responsiveness in critical moments.

Declaration on Generative AI

During the preparation of this work, the authors utilised ChatGPT to identify and rectify grammatical, typographical, and spelling errors. Following the use of these tools, the authors conducted a thorough review and made necessary revisions, and accept full responsibility for the final content of this publication.

Reference

[1] Freedman, D.H., "Are Drones the Future of Warfare?," *Scientific American*, 2021.

[2] M. R. Endsley, "From Defensive to Offensive: Integrating Drone Technology in Security Missions," *IEEE Security & Privacy*, vol. 19, no. 3, pp. 45–52, 2021.

[3] T. Neven, M. L. Chan, and D. Anselmo, "Privacy Implications of Civilian Drone Use: Best Practices and Regulatory Guidelines," *Information & Communications Technology Law*, vol. 29, no. 1, pp. 24–37, 2020.

[4] B. V. Moeslund and G. Thomas, "Optical Drone Detection Methods: A Survey," *International Journal of Remote Sensing*, vol. 41, no. 10, pp. 3896–3914, 2020.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, 2015.

[6] K. I. Kim, J. W. Yeon, and S. Kang, "Edge AI: Deploying Deep Neural Networks on Low-Power Embedded Devices," *IEEE Internet of Things Magazine*, vol. 3, no. 2, pp. 14–19, 2020.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[8] F. F. J. Sadeghi and S. M. Asghari, “Embedded Vision Systems for UAV Detection: Toward Efficient Deployments,” *Sensors*, vol. 21, no. 4, p. 1362, 2021.

[9] D. Merdes, H. Lin, and J. Zhang, “Secure Airspace: AI-Based Drone Monitoring for Airports and Sensitive Infrastructure,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4583–4592, 2021.

[10] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” *arXiv preprint arXiv:1503.02531*, 2015.

[11] R. Girshick, “Fast R-CNN,” *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[12] W. Liu et al., “SSD: Single Shot MultiBox Detector,” *Lecture Notes in Computer Science*, vol. 9905, 2016, pp. 21–37.

[13] Y. Lin, X. Chen, and J. Luo, “Embedded Deep Learning for Drone Surveillance on Low-Power Platforms,” *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 3, 2020.

[14] V. B. István and J. Ulrich, “Comparative Analysis of Drone Detection Techniques in High-risk Environments,” *Electronics*, vol. 9, no. 3, p. 472, 2020.

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, 2020.

[16] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *International Conference on Learning Representations (ICLR)*, 2015.

[17] A. D. Marquez and M. Kumar, “A Study of Model Compression Techniques for Real-Time Drone Detection,” *Journal of Signal Processing Systems*, vol. 93, no. 3–4, pp. 315–328, 2021.

[18] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[19] W. Lee, T. Kim, and G. Park, “Enhancements to YOLO for Resource-Constrained Drone Detection,” *Sensors*, vol. 21, no. 2, p. 489, 2021.

[20] X. Wang, L. Lu, and D. Chen, “Refined YOLO Architectures for High-Speed UAV Recognition,” *Computer Vision and Image Understanding*, vol. 207, pp. 120–130, 2021.

[21] C. Zhao, E. Li, M. Zhou, and B. Zheng, “Pruning Techniques for Embedded Object Detection Models,” *Neurocomputing*, vol. 421, pp. 263–274, 2021.

[22] H. Wang, P. Zhang, and X. Liu, “Evaluation of Quantized YOLO Models on Raspberry Pi,” *Electronics*, vol. 10, no. 15, p. 1794, 2021.

[23] Hailo Technologies, “Hailo-8™: High-Performance, Low-Power AI Processor for Edge Devices,” *Hailo White Paper*, 2021.

[24] A. Sharma, B. Singh, and C. K. Mohan, “Robust Drone Detection Using Domain-Adaptive CNNs,” *Pattern Recognition Letters*, vol. 143, pp. 27–34, 2021.

[25] M. Ghazal, O. Ouhsain, and A. Amira, “Comparative Assessment of Edge AI Frameworks for UAV Detection,” *IEEE Access*, vol. 9, pp. 105842–105852, 2021.

[26] Y. Shin, J. Park, and H. Choi, “Adaptive Network Architectures for Real-Time Drone Tracking on Embedded Platforms,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 8315–8324, 2021.

[27] S. Y. Zhu, Y. Chen, and R. Zhang, “Sensor Fusion of Visual and Acoustic Data for UAV Identification,” *Sensors*, vol. 22, no. 1, p. 114, 2022.

[28] N. Boyko, L. Mochurad, Y. Kryvenchuk, “Modeling of the information system for processing of a large distilled data for the investigation of competitiveness of enterprises”, Proceedings of the 4th International conference on computational linguistics and intelligent systems, P. 964–978, 2020.

[29] B. T. Smith, A. M. Johnson, and M. L. Burch, “Thermal-Enhanced Drone Detection in Complex Environments,” *Electronics*, vol. 10, no. 4, p. 409, 2021.