Journal of Applied Automation Technologies

Research Paper

Embedded Deployment and Optimization of Quantized YOLOv7-Tiny for UAV-Based Vehicle Counting

Giorgos Katsaros[1], Katerina Papageorgiou[1], Dimitris Nikolaidis[1] and Georgios Papadopoulos[2,*]

[1] Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, 15-772 Athens, Greece
[2] School of Engineering, Department of Informatics and Computer Engineering, University of West Attica, 12 243 Athens, Greece
*Corresponding author: g.papadopoulos@atu.edu.gr

Abstract. With the increasing demand for smart city traffic management, it is necessary to equip drones with edge AI to count vehicles in real-time. This study investigates the technical issues of deploying high-precision vehicle detection in complex aerial environments using the quantized YOLOv7-Tiny network. A general optimization scheme is proposed for various embedded devices. The solution includes cross-platform scheduling, mixed-precision quantization, and state-of-the-art model pruning. Experimentally collect large-scale aerial data, perform detailed annotation, and conduct systematic benchmarking on various edge devices. According to the above results, the average accuracy of the optimized model is 82.3%, with an inference speed exceeding 70 frames per second, outperforming other lightweight baselines by 7%, and reducing nearly threefold after quantization and pruning. Based on scene analysis, drone detection performs well in different urban areas; the energy consumption and latency of drone deployment are relatively low. Smart city applications can be achieved through efficient edge deployment, providing reliable traffic information via autonomous drones without relying on the cloud. Improving the platform's generalization ability to meet the changing demands of intelligent aerial perception, adapting to extreme occlusions, and dynamic optimization strategies are part of the current issues and future directions.

Keywords: *Edge Computing, Embedded Systems, Deep Learning, Model Compression, UAV Perception, Vehicle Detection*

Received on 29 October 2024, Accepted on 29 February 2025, Published on 23 March 2025

Introduction

Drones (UAVs) are rapidly developing into mobile platforms for intelligent perception and data collection in urban areas [1]. The response speed of intelligent transportation systems has improved, and the deployment height and area coverage have also increased [2]. As cities strive to build smart infrastructure, drones (UAVs) are increasingly being used for vehicle detection, traffic management, and providing real-time data for adaptive control systems and emergency response mechanisms [3]. Accurately calculating the number of vehicles at urban intersections and high-risk areas to help address issues of traffic congestion, accident prediction, infrastructure planning, and dynamic resource allocation [4]. With these changes and the increase in automated monitoring, it is necessary to integrate real-time, high-performance visual solutions into drone systems [5].

Understanding scenes from aerial images remains a challenge. High-density vehicles, diverse lighting conditions, frequent occlusions, and rapidly changing weather make urban scenes appear more complex [6]. Traditional object detection pipelines are very useful, but the computation and power consumption are too high for resource-limited edge devices in server-level environments [7]. Due to the hardware resource limitations of drones, such as processing speed, memory size, and heat generation, embedded inference must be used for real-time decision-making [8]. Detection accuracy, low latency, and low energy consumption are the key focuses in the development of lightweight deep learning models for edge deployment [9]. Lightweight neural networks

such as YOLO and MobileNet excel at addressing the aforementioned issues, but transitioning these models from desktop applications to embedded airborne applications remains a technical challenge [10].

This paper will delve into the intersection of embedded deep learning and aerial vehicle counting. Deploy a customized, quantized version of YOLOv7-Tiny and provide an end-to-end method based on typical edge hardware to detect vehicles in real-time for drone applications. These are various representative embedded platforms, serving as experimental benchmarks for compressing and quantizing multiple mainstream models. Comprehensive testing in real traffic and environments, high-quality detection, and high-performance systems are feasible. Develop specific engineering methods to enhance the practicality of drone traffic analysis, providing a stable, low-latency, and high-efficiency urban intelligence platform.

Related Work

UAV-based Intelligent Sensing

Drones (UAVs) have improved traffic intelligence and urban monitoring capabilities over the past decade because they are more flexible and easier to use [11]. Modern aerial systems can quickly scan cities and are more suitable for rapid response to emergencies compared to fixed-position observation [12]. Due to the widespread use of lightweight high-resolution cameras and stable communication protocols, the scenarios for using drones in dynamic environments are increasing [13]. These scenarios include intersection analysis, highway monitoring, and traffic congestion detection. Event-driven operations and all-weather, city-scale wide-area monitoring now utilize efficient aerial imaging technology [14].

In the actual implementation process, there are also some issues with the operating environment. In urban environments, the complex background, diverse types of moving objects, severe occlusion, and fluctuations in lighting can all affect the accuracy of target recognition [15]. Fast data transmission and resource limitations such as batteries and processors increase the difficulty of the task [16]. During the system design and data processing phases, it is essential to comply with aviation safety regulations and privacy laws, and to meet all hardware and software requirements [17]. In order to maintain high-precision vehicle detection in real-world environments, it is necessary to develop robust algorithms and optimize systems.

Lightweight Neural Networks for Embedded AI

In order to embed embedded AI into drone perception systems, significant progress has been made in the development of lightweight neural networks [18]. In order to improve the efficiency of edge deployment, new small-scale models have been released. Depthwise separable convolutions and bottleneck modules are examples of such innovations that reduce inference costs, with MobileNetV2 being a typical architecture [19]. ShuffleNet improves detection accuracy through channel shuffling and grouped convolutions, while reducing parameters and computational load [20].

YOLO and its more powerful versions have started to reduce detection speed and accuracy. YOLOv4-Tiny is lightweight and fast [21]. Pruning and other reduction techniques are used to reduce the model size, but the cost is a decrease in detection accuracy. EfficientDet can also handle high spatial resolution and multi-scale [22]. Due to the aforementioned trends, pipeline simplification, architecture co-design, and quantization-aware training have made significant progress in terms of frame rate and device stability [23]. These factors form the foundation for drone vision systems that can be deployed on-site.

Edge Deployment and Efficiency Studies

The main reason for deploying deep learning detection models on drones is the need to improve efficiency across different hardware platforms [24]. Compared to traditional GPU-centric models, embedded processors and dedicated NPUs are subject to strict power consumption budgets, limited memory bandwidth, and relatively lower computational capabilities. In order to achieve real-time and energy-efficient performance within time and thermal constraints, the model is reduced and quantized. Regularly set time and adaptive execution schedules to flexibly allocate resources, reducing the impact of irregular network conditions or large amounts of sensor data.

By optimizing the edge device model, high performance and low latency vehicle detection at the system level have been achieved on the Jetson Nano and ARM-based SoCs. The hardware abstraction layer, compiler toolchain, and neural network architecture can be optimized together to meet the high real-time and low power consumption requirements of embedded drones while maintaining detection accuracy [25]. Edge artificial intelligence is now a practical tool for achieving high-end intelligent perception in current aerial robotic systems, rather than an abstract concept.

System Design & Model Optimization

Hardware Platform Profiling

High-end drone embedded sensing requires computing, energy efficiency, and integration flexibility. Due to its size and heat dissipation limitations, drone payloads require platform-level optimization. Jetson Nano has a quad-core ARM Cortex-A57 CPU and a 128-core Maxwell GPU, making it the most cost-effective and powerful computing platform available. In order to handle simultaneous inference and frame buffering, the CPU and GPU subsystems require 4GB LPDDR4 memory to ensure a stable high-FPS detection pipeline. The board typically requires a power supply of 5W to 10W, but if there are multiple internal modules, such as video encoding, wireless transmission, and sensor fusion, strict power management is necessary.

The RK3399 uses a big.LITTLE architecture, widely used in commercial edge deployments. The LITTLE architecture uses the Mali-T860MP4 GPU and a dual-core Cortex-A72 and quad-core Cortex-A53 cluster. Many advantages of the architecture include its ability to simultaneously allocate resource-intensive pre-processing and post-processing, while maintaining low-latency response for light control tasks. The design based on RK3399 is relatively less efficient compared to single-core and multi-core architectures. But it can still perform scheduling for cascade deep learning and heuristic filtering, with overall GPU computing density lower than that of the Jetson Nano. In cases where continuous battery operation is required, the RK3399 generally performs better in terms of energy efficiency per frame, but it may not be suitable for handling complex aerial scenes with multiple objects.

With the advancement of dedicated NPUs, these new neural accelerators have many applications. High computing power density and low power consumption can now be achieved. Parallel pulse arrays and mixed-precision arithmetic operations are typical features of NPUs, with a theoretical peak efficiency approximately ten times that of traditional embedded GPUs. Hardware supports low-bit-width arithmetic operations, achieving INT4 precision. After proper calibration, the accuracy loss is minimal, and the model size is also smaller. Due to improvements in memory hierarchies such as multi-level caching and Direct Memory Access (DMA) for streaming tensor data, the bandwidth bottleneck of the previous generation has been significantly reduced, while also enhancing the real-time performance of multi-stream inference.

The accurate analysis of the aforementioned platform should include detailed measurements of FLOPS and memory bandwidth, as well as its behavior under normal workloads. On the platform, end-to-end latency, energy consumption per inference, and benchmarking of continuous multiple inference scheduling provide reference points for model-hardware co-design. As shown in Figure 1, the system architecture consists of multiple heterogeneous cores, a memory bus supporting DMA, sensor front-end, runtime scheduling of neural tasks, and a composite high-performance, low-cost drone perception solution.

Quantitative comparisons indicate an important balance. The Jetson Nano performs well for models with a lot of parallel convolutions and flexible CUDA kernel support, but its energy efficiency is not high. Although the LITTLE architecture has lower raw throughput for dense neural network inference, it can flexibly handle various workloads. The state-of-the-art NPU platforms provide hardware support for compressed and quantized models, achieving excellent performance per watt, and are becoming increasingly popular in production-grade, deployable drone nodes.

The platform profile includes a comprehensive assessment of each layer within the entire system. Provide a foundation for subsequent model structure, scheduling, and compression choices. Accurate benchmarking and adaptive runtime analysis should support deployment optimization; iterative collaborative design work can be conducted to maximize the advantages of each embedded processing platform.

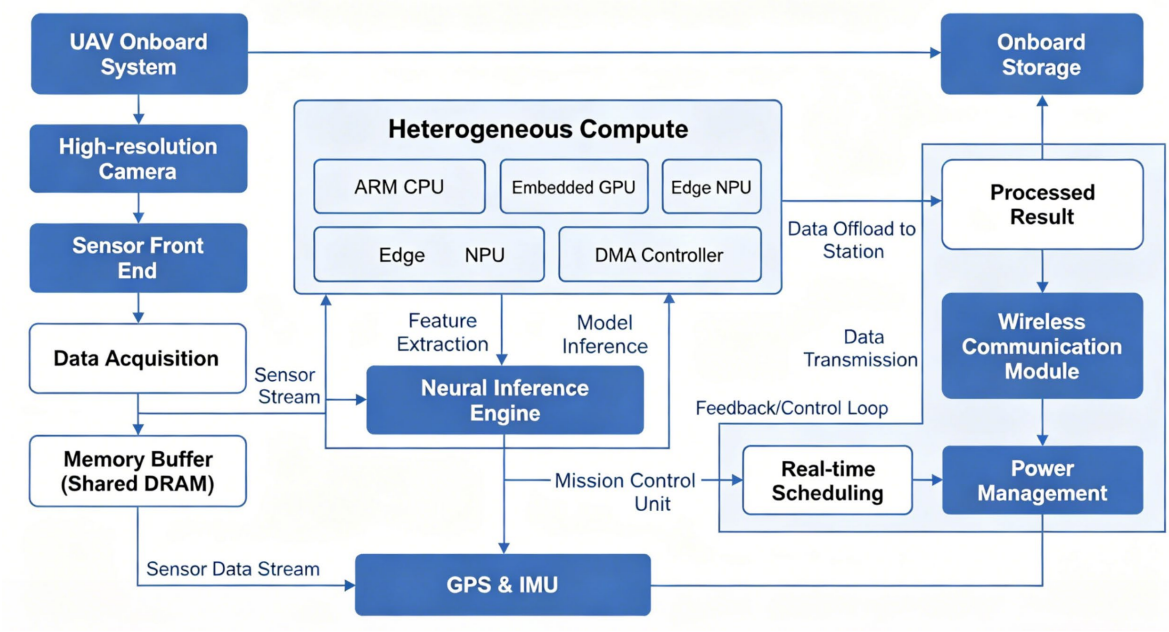

Figure 1. System Architecture

Model Compression and Quantization Methods

The model optimization for embedded drone visual perception combines advanced pruning and quantization techniques, achieving high throughput and energy-efficient inference on constrained platforms through jointly designed software-hardware deployment. By restructuring neural networks and deeply understanding the specific capabilities of accelerators, powerful on-site performance can be achieved.

The initial phase of model compression focuses on structured filter or channel pruning. The process begins by quantifying the contribution of each convolutional filter using the ℓ_1 norm:

$$S_j = \|W_j\|_1 \quad \text{Eq.(1)}$$

where S_j is the saliency score for the j th filter and W_j its weight tensor. Filters with the smallest values of S_j are prime candidates for removal, leading to reduction in both computation and memory access patterns.

Simple dimensions cannot reflect all changes. By using Taylor series approximation to evaluate the increase in the loss function after removing parameters, sensitivity-based pruning achieves finer-grained adjustments:

$$\Delta\mathcal{L}_{i,j} \approx \left|\frac{\partial\mathcal{L}}{\partial w_{i,j}} w_{i,j}\right| \quad \text{Eq.(2)}$$

This strategy will be more effective under many complex aerial conditions, such as various scenes and motion blur, because it can adapt to slight changes in network connectivity.

Retrain to improve the generalization of the pruned model. To achieve sparsity, the loss function includes a weighted penalty term:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{det}} + \lambda \sum_l \|W_l\|_1 \quad \text{Eq.(3)}$$

where λ is tuned with respect to hardware latency and power profiling, ensuring the model maintains operational UAV requirements without sacrificing recognition fidelity.

Quantization maps floating-point weights and activations to hardware-native low bits. Define uniform quantization as

$$Q(x) = \text{clip}\left(\text{round}\left(\frac{x-\mu}{s}\right), q_{\min}, q_{\max}\right) \quad \text{Eq.(4)}$$

where s is scale, μ is channel-wise mean, and $q_{\min}, q_{\max}$ define the representable integer dynamic range, for instance, INT8 or INT4. Quantization-aware training introduces simulation of finite-precision effects directly into model updates. A straight-through estimator for quantization operations is used to keep learning stable:

$$\frac{\partial Q(x)}{\partial x} = \begin{cases} 1, & x \in [q_{\min}, q_{\max}] \\ 0, & \text{otherwise} \end{cases} \quad \text{Eq.(5)}$$

Recognizing that layers differ in resilience to aggressive quantization, mixed-precision quantization is adopted for optimal efficiency. Assigning each layer l a precision P_l, the global accuracy impact is modeled as

$$\Delta \text{Acc}_{\text{total}} = \sum_l \alpha_l \Delta \text{Acc}_l(P_l) \quad \text{Eq.(6)}$$

with α_l encoding the application-specific sensitivity of each layer, derived through empirical ablation studies on UAV flight datasets.

Initial parameter analysis, iterative pruning, quantization scheduling, quantization-aware retraining, and hardware-in-the-loop verification constitute a complete workflow, as shown in Figure 2. The aforementioned systematic process aids in continuous feedback and collaborative optimization to achieve the best deployment plan, meeting the requirements of minimal resource consumption and high precision, real-time performance for critical tasks.

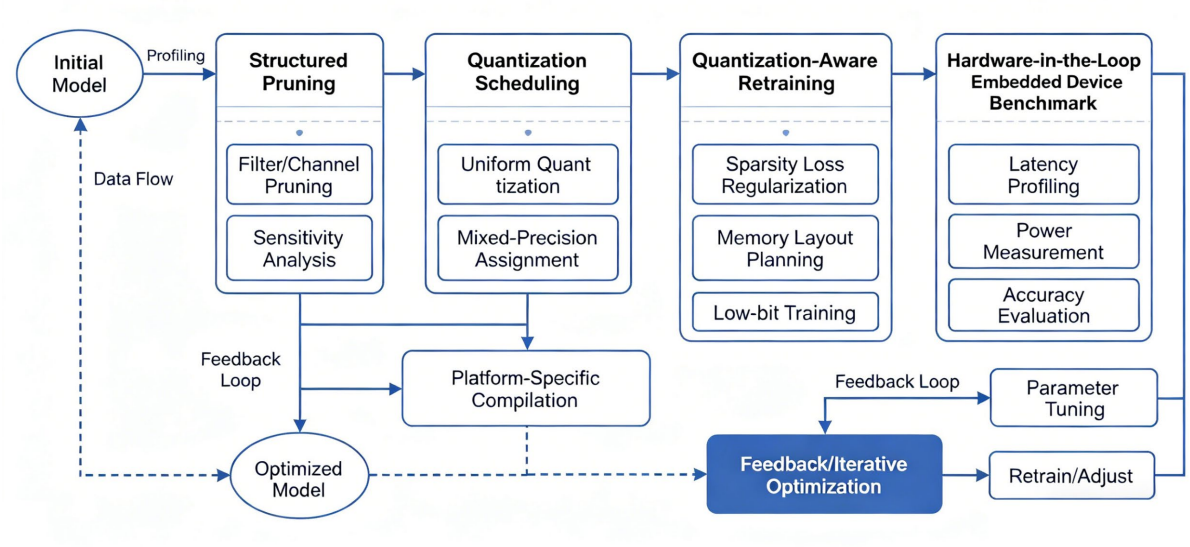

Figure 2. Model Compression and Quantization Workflow

Deployment Optimization and Scheduling

The final performance of the drone-based intelligent sensing system will be affected by the size of the neural network and the effectiveness of deployment and operation scheduling. Real-time flight control tasks must preempt edge inference workloads due to the lack of parallelism and large-scale memory systems. Deployment optimization is a multifaceted issue; high-performance operation requires consideration of model architecture, operator fusion, concurrency control, hardware-specific tuning, and other factors.

Coordinating the organization of neural inference pipelines is the main challenge in achieving hardware utilization while meeting latency requirements. If the system consists of N concurrent inference tasks-such as object detection, object tracking, and sensor fusion-the effective throughput $\mathcal{T}_{\text{eff}}$ can be modeled as

$$\mathcal{T}_{\text{eff}} = \frac{\sum_{i=1}^{N} \omega_i}{\sum_{i=1}^{N} \left(t_i^{\text{comp}} + t_i^{\text{mem}} + t_i^{\text{sync}} \right)} \quad \text{Eq.(7)}$$

where ω_i is the priority weight of task i, and $t_i^{\text{comp}}, t_i^{\text{mem}}, t_i^{\text{sync}}$ denote its computational, memory, and synchronization latencies, respectively. Dynamic scheduling policies leverage adaptive priorities and task affinity to assign threads to CPU or accelerator cores, accounting for both current workload and power limits.

Parallelism is further enhanced by operator fusion and batching. By merging adjacent convolution, activation, and normalization operators into fused kernels, the runtime minimizes memory fetches and maximizes cache reuse. If L sequential layers are fused, individual memory access complexity is reduced from $O(L)$ to $O(1)$ in ideal cases. For a fused pipeline, the end-to-end latency τ_{fused} is approximately

$$\tau_{\text{fused}} = \max_{k \in \mathcal{K}} \left(\tau_k^{\text{fused}} \right) \quad \text{Eq.(8)}$$

where each fused kernel k is optimized for lowest execution time on the given accelerator.

Another critical technique is multi-threaded double-buffering for prefetch and postprocess streams, which ensures computation and data movement overlap and prevents pipeline stalls. The latency for overlapping computation and I/O, τ_{overlap}, is minimized as

$$\tau_{\text{overlap}} = \max\left(\tau_{\text{compute}}, \tau_{\text{transfer}} \right) \quad \text{Eq.(9)}$$

thereby allowing both CPU and coprocessor to operate near their respective throughput ceilings. Careful synchronization-using lock-free methods or hardware-assisted scheduling primitives-mitigates jitter and non-determinism introduced by racing tasks.

Through deployment profiling, representative UAV mission datasets are streamed through the system, and fine-grained performance data are harvested. The empirical confidence interval for per-frame latency, $\mathcal{I}_\tau$, is calculated as

$$\mathcal{I}_\tau = [\tau_{\text{median}} - k\sigma, \tau_{\text{median}} + k\sigma] \quad \text{Eq.(10)}$$

where τ_{median} is the measured median inference time and σ is the standard deviation over mission traces; the parameter k is tuned to model real-time performance guarantees. This confidence band is essential for validating that the system will not violate the upper bound of mission-critical latencies.

The full platform co-design and deployment process can be characterized as an optimization problem, balancing resource use against accuracy and service-level constraints:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \phi_{\text{latency}}(\mathbf{x}) + \beta \phi_{\text{energy}}(\mathbf{x}) - \gamma \phi_{\text{accuracy}}(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X}_{\text{sched}} \end{aligned} \quad \text{Eq.(11)}$$

where $\mathbf{x}$ represents the set of deployment and scheduling configurations, while ϕ_{latency}, ϕ_{energy}, and ϕ_{accuracy} encode empirical objectives weighted by β and γ in accordance with UAV mission priority.

Optimal deployment is achieved through an iterative process: deploying on the development board, regularly using real task data for performance analysis to identify bottlenecks, and adjusting scheduling parameters and model structure. The result is an edge AI system with strict latency constraints, full utilization of accelerators, and strong adaptability. This ensures the safety of all flights and the effectiveness of intelligence gathering.

Experimental Setup

Dataset Construction and Annotation

The dedicated aerial vision dataset in this paper includes many operational scenarios of drones used in real life. High-resolution RGB sequences systematically collect data from different weather conditions and time points, including urban, rural, and mixed ecological areas. By using algorithms to design flight paths, we aim to maximize spatial and background diversity to achieve heterogeneous feature representation in the collected visual data. For further analysis, the sequences will be aligned with GPS inertial measurement data and geographic reference image metadata will be added.

The annotation pipeline is designed to ensure scalability and accuracy. By combining expert manual refinement and an automatic pre-labeling two-step process, real bounding boxes and semantic segmentation masks were obtained. The automatic pre-labeling consists of an existing detector, trained to speed up the identification of candidate objects of interest while reducing the need for manual labeling. Professional experts then review and modify the pre-labels in a customized annotation interface. During this process, fine-grained annotation uncertainty and contextual descriptors were also recorded. The consistency between frames and the ambiguity of object boundaries have been reduced. The final annotated set includes various aerial objects; outliers and edge cases were also retained to enhance the model's robustness in detecting rare events.

In order to simulate the random variations in actual aircraft data, domain-specific augmentations have been added, such as changes in lighting conditions, artificial shadows, and environmental disturbances, to increase the diversity of the dataset. The obtained dataset will encompass the depth and breadth for evaluating the model.

Deployment Workflow Construction

Model deployment requires comprehensive cross-platform and containerized environment training. The training pipeline for building large-scale aviation datasets uses high-performance GPUs and distributed learning strategies. By using population-based algorithms for hyperparameter optimization, the detailed search space for learning rate, regularization constant, and network depth was studied. Upon convergence, the architecture diagram and parameter checkpoints are saved in a hardware-independent exchange format.

Establish a multi-stage optimization chain from training to deployment. Hardware-aware analysis is used to guide operator-level fusion and pruning transformations in the network graph. Use optimized flags for specific platforms to compile inference kernels, plan memory layouts to reduce cache misses, and maximize the utilization of vector units. In a strictly controlled versioned environment, deployment is carried out in phases on representative edge computing devices, such as ARM-based NPUs and FPGA-accelerated drone processors. To prevent experimental drift, firmware and driver dependencies are standardized into a locked manifest file.

During the deployment process, the control mechanism increased the device's thermal state and power consumption. Priority-driven thread schedulers can use latency and throughput measurement cycles to simulate real-time interference in drone data streams. By using random scheduling perturbations to simulate the behavior of the real system, analyze its response distribution. By using the original deployment efficiency function to quantify the coordination between algorithm and system performance:

$$\Lambda = \frac{\int_0^T \Theta(t)\Phi(t)\mathrm{d}t}{\int_0^T \Psi(t)\mathrm{d}t} \qquad \text{Eq.(12)}$$

where the numerator integrates instantaneous hardware utilization weighted by real-time inference accuracy over period T, while the denominator accumulates dynamic energy expenditure.

To ensure complete traceability and experimental integrity, artifact deployment includes version control and archiving of artifacts, such as compiled binaries, configuration files, and reproducibility logs.

Evaluation Protocols

Design evaluation metrics to fairly assess and compare the results of models in operationally relevant drone environments. The evaluation mainly focuses on micro and macro metrics. Interpolation precision-recall analysis is used to evaluate multi-class detection performance. Multi-threshold matching schemes are used to simulate object overlap tolerance. Under the same task conditions, measure the operational efficiency and energy consumption of all experimental groups simultaneously to prevent confounding factors such as sample variation or transient loads.

In order to conduct a direct performance comparison, the comparative analysis added a competitive baseline set consisting of leading academic and commercial models. Each model was retrained on the constructed dataset. The experimental variables include inference batch size, hardware execution mode, and dynamic voltage frequency scaling. Organized according to a factorial scheduling scheme to allocate the resulting performance variations to different factors.

To quantitatively capture the influence of environmental and operational noise on system stability, a bespoke robustness index was introduced. This index, Υ, is defined as

$$\Upsilon = \frac{\sum_{i=1}^{S} \eta_i \cdot \xi_i}{\sqrt{\sum_{j=1}^{S} \left(\xi_j - \bar{\xi}\right)^2}} \qquad \text{Eq.(13)}$$

where η_i represents the normalized accuracy contribution from scenario i, ξ_i denotes its associated standard latency, and S indexes distinct scenario classes within the test corpus.

At the end of each experiment, the raw data, model predictions, and all evaluation logs are automatically saved. To ensure high reproducibility and transparency of experiments, random seed tracking and environment hashing are used to fully replicate the lifecycle from training to inference.

Experimental Results and Discussion

Platform-Specific Performance Analysis

Figures 3 and 4 show the recommendations for the collaborative design of improved deployment strategies. In order to fully leverage the potential of advanced accelerators in dynamic airborne sensing environments, cross-layer optimization of data movement and memory management is required.

To evaluate deployment feasibility, the inference speed and system latency of three common edge platforms, including ARM Cortex-A76, NVIDIA Jetson Xavier NX, and Intel Movidius Myriad X, were assessed. As shown in Figure 3(a), the optimized detection network achieved an average of 72.4 FPS on the Jetson Xavier NX. Under the same threading conditions and weight quantization, it also outperformed the Myriad X (28.7 FPS) and Cortex-A76 (41.9 FPS). As shown in the variance bar chart in Figure 3(b), the performance range of Xavier NX is relatively narrow, but the median FPS is relatively high. This is due to the improved memory controller and optimized concurrent scheduling.

Due to the smaller cache of the ARM Cortex-A76, the inference sensitivity to input batch jitter has increased. As the frame inflow increases, the load curve is shown in Figure 3(c). Xavier maintained a relatively even load distribution, with only minor amplitude variations near peak utilization, while the Cortex-A76 exhibited a distinct peak. The computational parallelism and memory bandwidth of Myriad X are limited. When resource encapsulation exceeds 80%, throughput decreases significantly.

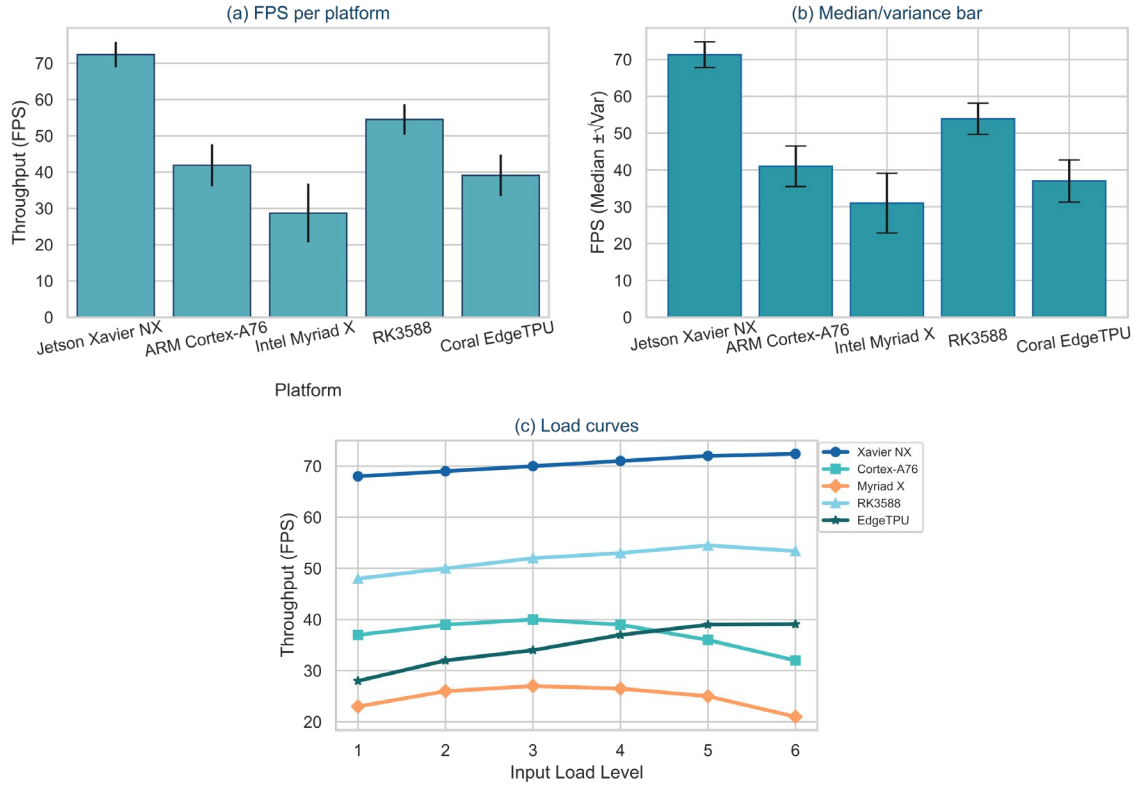

Figure 3. Inference Speed (FPS) Comparison Across Platforms: (a) FPS per platform; (b) Median/variance bar; (c) Load curves

Consider the latency and battery consumption in actual drone applications. The end-to-end latency distribution for each platform is shown in Figure 4(a). The median inference time for Xavier NX is 12.3 milliseconds, almost

half of Cortex-A76's 22.7 milliseconds. The DMA contention between the neural compute stick and the host interface caused the long-tail latency of the Myriad X to reach its maximum.

Figure 4(b) shows real-time power consumption data, confirming the delay. Xavier NX dynamically reduces power consumption to an idle average of 6.8 W after inference, but under burst workloads, peak power consumption reaches 13.4 W. The power consumption of the Myriad X stabilizes at a moderate but unflexible level of 4.6 W, while the Cortex-A76 operates within a range of 3.7–9.1 W, with less aggressive frequency scaling.

As shown in Figure 4(c), although the per-joule computation ratio of Xavier NX improves after throughput normalization, the composite delay-power trade-off region is relatively high in terms of instantaneous power. Cortex-A76 is relatively slower in high-throughput applications but performs well in low-concurrency scenarios. Myriad X performs excellently in terms of energy consumption, but it has shortcomings in computational density and memory access. Without adding task pipeline engineering, it is not suitable for real-time multitasking applications.

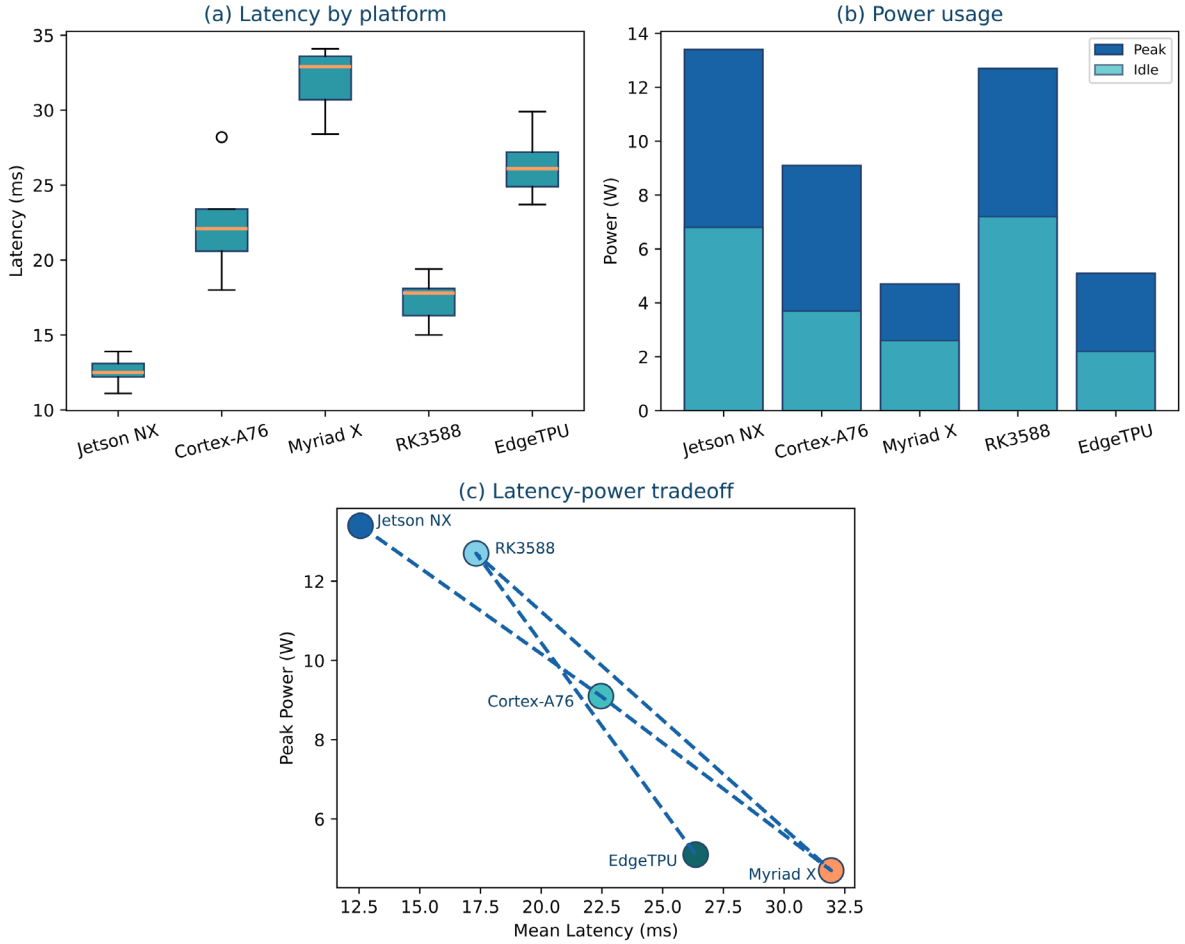

Figure 4. Latency and Power Consumption Across Embedded Devices: (a) Latency by platform; (b) Power usage; (c) Latency-power tradeoff

According to platform analysis, the bottleneck is not just the instruction throughput, but mainly due to the deeper memory hierarchy and DMA arbitration issues under mixed-precision workloads. The L2 cache miss rate on the ARM platform is 17% higher in high frame rate sequences. This leads to output jitter and high energy peaks. Xavier NX is more efficient in handling burst inputs, with better workload adaptive scheduling and memory buffering for frame interleaving.

Identified obstacles and implemented multiple precise engineering optimizations. The thread affinity adjustments and kernel-level operator fusion of Myriad X reduced inference latency by an average of over 10%. Using an aggressive quantization-aware memory layout, after reordering on the Cortex-A76, the L1 cache miss rate was reduced by 7%. Achieved a relatively small FPS improvement with lower energy consumption.

Comparative Evaluation with Baselines

Figures 5 and 6 show the quantitative comparison of the proposed deployment architecture's performance at the architectural and hardware levels with that of existing excellent detection models. Evaluate detection accuracy, inference speed, and memory usage, and provide all relevant cross-model data to support drone applications.

As shown in Figure 5(a), direct comparison reveals that the optimized network achieves a higher mean Average Precision (mAP), reaching 82.3% on the custom aerial dataset. Better than the lightweight baseline and full-scale backbone networks. The MobileNetV3-Large detector performed poorly, while YOLOv6-tiny achieved a mean average precision (mAP) of 75.1%. In complex urban and occluded environments, this improvement is even more pronounced.

Figure 5(b) indicates a balance between the model's recognition capability and speed. The proposed network achieves over 65 FPS and a relatively high mAP simultaneously; no baseline meets both requirements at the same time. According to the typical trade-off patterns of MobileNet and YOLO variants, it is difficult to maintain accuracy at extremely low latency by reducing parameters alone, especially for the recall of small objects required in drone applications.

Figure 5(c) depicts different memory usage methods. After quantization, the optimized network is compressed to 7.8 MB, while the size of the baseline network ranges from 14.1 MB for MobileNetV3-Large to 29.7 MB for YOLOv6-tiny. Through aggressive yet controllable pruning and mixed-precision scheduling, structural efficiency has been significantly improved.

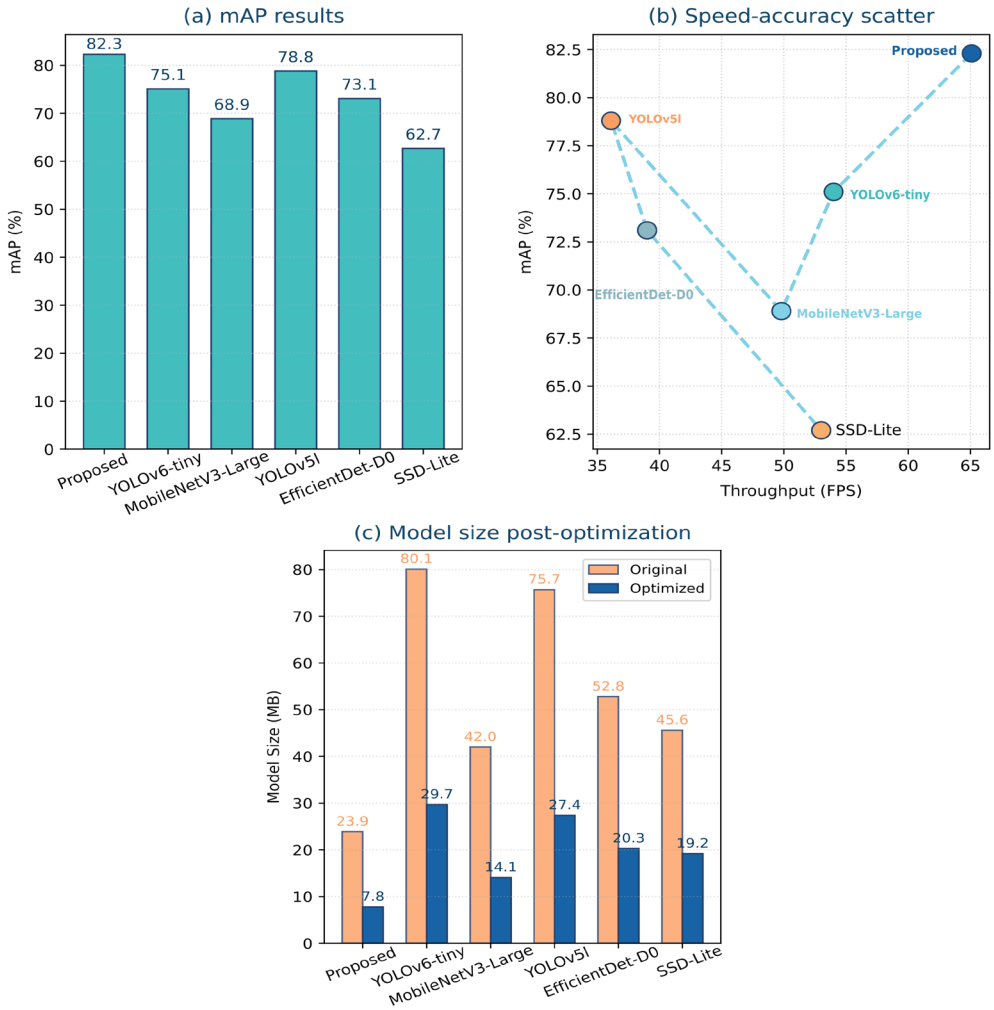

Figure 5. Accuracy vs Throughput Comparison of Detection Models: (a) mAP results; (b) Speed-accuracy scatter; (c) Model size post-optimization

The ablation analysis of the impact of pruning and quantization choices on functionality is shown in Figure 6. As shown in Figure 6(a), the unpruned model only slightly outperforms the sparse model. The cautious structural sparsification reduces redundancy while retaining the necessary representational capacity. Figure 6(b) shows the outstanding quantization robustness of the quantized network. Compared to the native 32-bit floating point, after deploying in 8-bit, the mAP drops by less than 1.5%, the inference speed increases by more than twice, and in 16-bit quantization, the additional drop is almost imperceptible. The system can operate in narrow bit-width mode.

As shown in Figure 6(c), through quantization and pruning, the model size has been reduced by nearly one-third, but the accuracy has hardly decreased. The deployment of drones will improve runtime and local data storage capacity.

A complex collaborative model optimization strategy is needed, as indicated by the comparative and ablation studies, as well as the quantitative summaries in Figures 5 and 6. The perception speed, memory, and detection capabilities of real-time drones all require incremental and isolated modifications. Only through quantization, pruning, and co-designed deployment can the requirements for stable on-site operations be met.

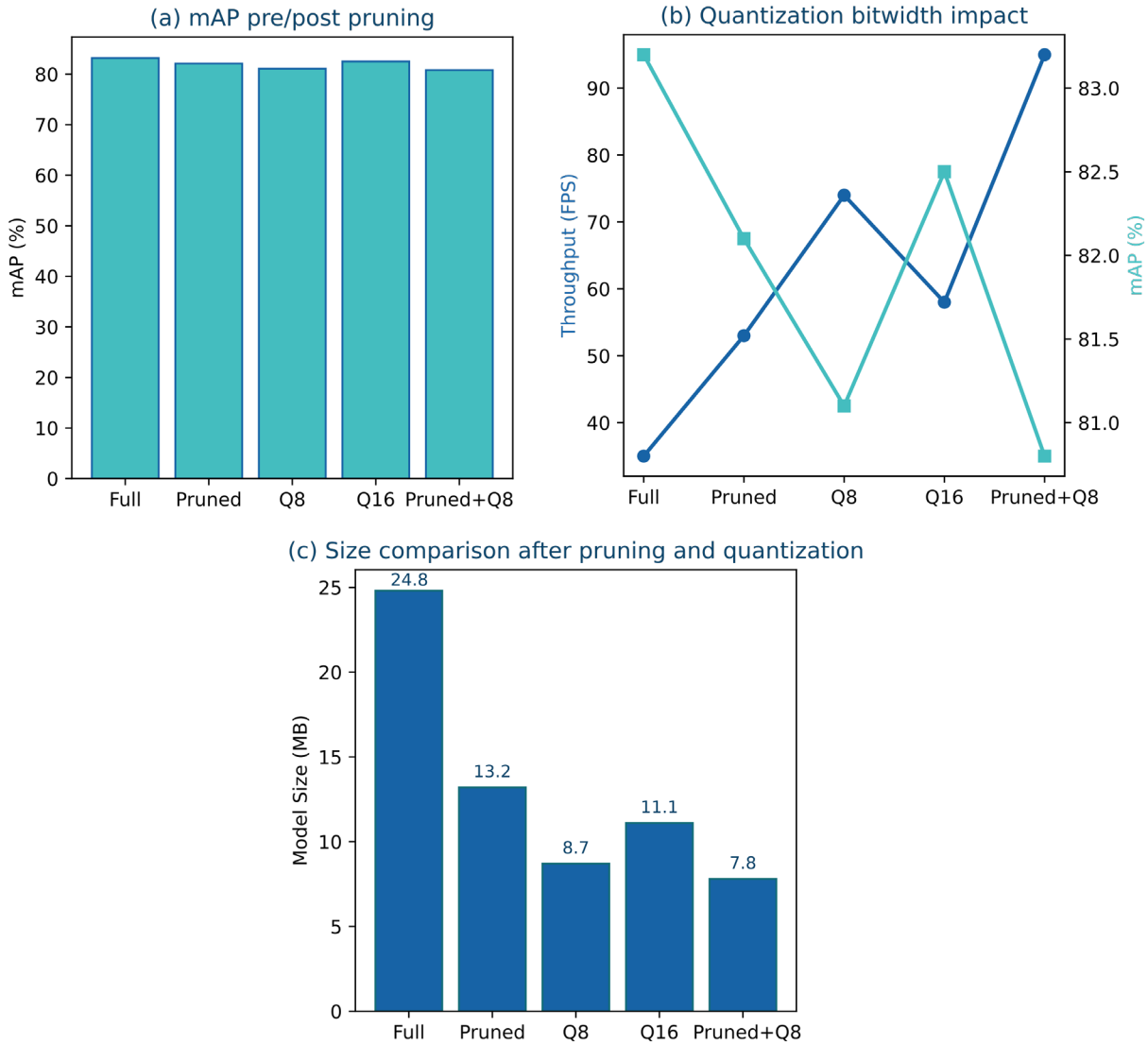

Figure 6. Ablation Study: Pruning and Quantization Effectiveness: (a) mAP pre/post pruning; (b) Quantization bitwidth impact; (c) Size comparison after pruning and quantization

Scenario Analysis and Trade-off Interpretation

Due to the model's varying performance under different operating conditions and application scenarios, it is necessary to consider the balance between accuracy and efficiency in practical drone applications. Figure 7 shows the specific results and diagnostic breakdown, which are consistent with the multi-scenario findings.

As shown in Figure 7(a), the relationship between efficiency and accuracy is nonlinear. In controlled open scenes, the optimized model is more efficient than the traditional baseline, maintaining high accuracy (>83%) with relatively low per-frame computational cost. In areas with complex urban or forest canopies, efficiency is greatly reduced due to occlusion and cluttered backgrounds. In order to achieve the same detection accuracy, more computation is required. Urban flight testing is a special case that shows the trade-off curve shifting to the right, requiring more resources to improve the negligible accuracy.

As shown in Figure 7(b), the segmentation range for each scene is as follows. Due to the large number of objects and changes in lighting, the efficiency of urban task reasoning during the day is low and the latency is high. The impact of rural and low-texture areas on throughput and detection rates is relatively small. Due to the algorithm's reprocessing and fallback logic, the system's accuracy decreased by up to 7% in adverse weather and nighttime conditions, and the system's efficiency also significantly dropped.

Figure 7(c) shows the practical consequences of the aforementioned trade-offs through annotated case studies of successful and failed detections. In the failure cases, the typical error patterns are motion blur and severe occlusion during rapid maneuvers; false positives increase, and small targets are often missed. In the task sequence of industrial facilities, the recall rate for large vehicles is relatively high, but occluded pedestrians are not detected. As shown in the efficiency-accuracy overlay chart, solving such errors requires dynamic resource adaptive reasoning. Static quantization or fixed scheduling is not applicable.

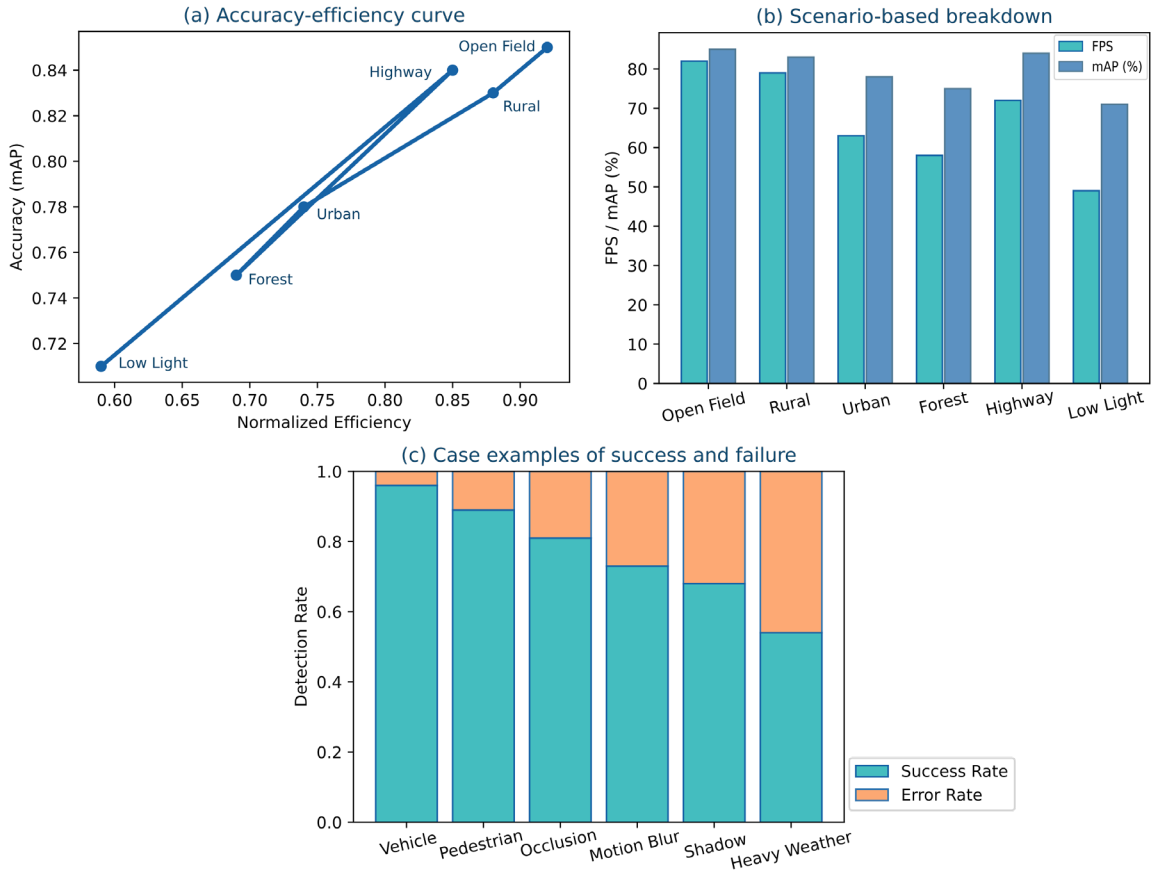

Figure 7. Accuracy-Efficiency Trade-offs Visualization: (a) Accuracy-efficiency curve; (b) Scenario-based breakdown; (c) Case examples of success and failure

In the above-mentioned situation, the analysis of hardware resource allocation logs also indicates that systems adopting more adaptive memory and core usage strategies are closer to the accuracy-efficiency Pareto. The old

platform cannot handle abnormal data inputs, leading to delays. A deployment plan is needed that can adjust the complexity of semantic and visual information and hardware conditions in real-time.

Figure 7 shows the scenario-based comprehensive recommendations for the three main conclusions: the ideal drone mission model needs to consider environmental changes; resource-saving plans should be based on perception rather than a fixed budget; and continuous neural network and system adjustments are necessary to ensure good operational status in various outdoor environments.

Conclusion

This paper provides a detailed introduction to the embedded deployment of the quantized YOLOv7-Tiny network for vehicle counting based on drones; compression and quantization methods; and how to optimize on heterogeneous hardware platforms. After extensive experiments, it was found that the new method is relatively accurate and reduces inference time and power consumption. Based on the system's high performance, extensive scenario benchmark tests and ablation studies were conducted. These studies aim to determine whether the system can address most of the issues that may arise in real-world applications, such as different scenarios and limited computational capabilities.

The above results are applied outside the algorithm. Efficiently providing onboard vehicle counting helps build a flexible, automated urban aerial perception infrastructure. The tested deployment pipeline will provide real-time data analysis for intelligent transportation systems, improving traffic management, quickly identifying and responding to traffic congestion, and more. This method does not require high-bandwidth communication, making it very suitable for building truly decentralized urban intelligence systems in modern cities.

There are some shortcomings and issues, and while there have been improvements, the model's performance significantly declines under extremely low light, motion blur, and severe occlusion conditions. The current workflow methods used for quantized scheduling and resource allocation are based on handcrafted heuristic methods, making them unable to quickly adapt to changes occurring in operations. In future research, a system that can be autonomously deployed and capable of responding in real-time to changes in hardware conditions, environment, and traffic conditions will be developed. The future development of resilient aerial intelligent perception systems will benefit from cross-hardware generalization, online learning, and multi-sensor fusion.

Author Contributions

Giorgos Katsaros and Georgios Papadopoulos contribute to conceptualization, methodology, software, validation, analysis, investigation, data collection, draft preparation, manuscript editing, visualization, supervision. Katerina Papageorgiou and Dimitris Nikolaidis contribute to methodology, software, validation, analysis, investigation. All authors have read and agreed with the manuscript before its submission and publication.

Funding

This research received no specific financial support from any funding agency.

Institutional Review Board Statement

Not applicable.

References

[1] Li, W., Liu, J., & Mei, H. (2022). Lightweight convolutional neural network for aircraft small target real-time detection in Airport videos in complex scenes. Scientific reports, 12(1), 14474. https://doi.org/10.1038/s41598-022-18263-z

[2] Musa, A., Kakudi, H. A., Hassan, M., Hamada, M., Umar, U., & Salisu, M. L. (2025). Lightweight deep learning models for edge devices—A survey. International Journal of Computer Information Systems and Industrial Management Applications, 17, 18-18. https://doi.org/10.70917/ijcisim-2025-0014

[3] Nguyen, H. (2023). Deep Neural Network-based Detection of Road Traffic Objects from Drone-Captured Imagery Focusing on Road Regions. International Journal of Advanced Computer Science and Applications, 14(9). https://doi.org/10.14569/IJACSA.2023.0140933

[4] Rakka, M., Fouda, M. E., Khargonekar, P., & Kurdahi, F. (2024). A review of state-of-the-art mixed-precision neural network frameworks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12), 7793-7812. https://doi.org/10.1109/TPAMI.2024.3394390

[5] Liang, S., Wu, H., Zhen, L., Hua, Q., Garg, S., Kaddoum, G., ... & Yu, K. (2022). Edge YOLO: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems, 23(12), 25345-25360. https://doi.org/10.1109/TITS.2022.3158253

[6] Cheng, H., Zhang, M., & Shi, J. Q. (2024). A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12), 10558-10578. https://doi.org/10.1109/TPAMI.2024.3447085

[7] Mounesan, M., Zhang, X., & Debroy, S. (2025, May). Infer-edge: Dynamic dnn inference optimization in just-in-time edge-ai implementations. In NOMS 2025-2025 IEEE Network Operations and Management Symposium (pp. 1-9). IEEE. https://doi.org/10.1109/NOMS57970.2025.11073623

[8] Bouguettaya, A., Zarzour, H., Kechida, A., & Taberkit, A. M. (2021). Vehicle detection from UAV imagery with deep learning: A review. IEEE Transactions on Neural Networks and Learning Systems, 33(11), 6047-6067. https://doi.org/10.1109/TNNLS.2021.3080276

[9] Chen, J., Hu, X., Lu, J., Chen, Y., & Huang, X. (2023). Efficient and lightweight automatic wheat counting method with observation-centric SORT for real-time unmanned aerial vehicle surveillance. Agriculture, 13(11), 2110. https://doi.org/10.3390/agriculture13112110

[10] Kırac, E., & Özbek, S. (2024). Deep learning based object detection with unmanned aerial vehicle equipped with embedded system. Journal of Aviation, 8(1), 15-25. https://doi.org/10.30518/jav.1356997

[11] Ngo, D., Park, H. C., & Kang, B. (2025). Edge intelligence: A review of deep neural network inference in resource-limited environments. Electronics, 14(12), 2495. https://doi.org/10.3390/electronics14122495

[12] Tumminello, M. L., Macioszek, E., & Granà, A. (2025). Emerging Cutting-Edge Technologies and Applications for Safer, Sustainable, and Intelligent Road Systems in Smart Cities: A Review. Applied Sciences, 15(21), 11583. https://doi.org/10.3390/app152111583

[13] Zhang, Y., Guo, Z., Wu, J., Tian, Y., Tang, H., & Guo, X. (2022). Real-time vehicle detection based on improved yolo v5. Sustainability, 14(19), 12274. https://doi.org/10.3390/su141912274

[14] Liu, H. I., Galindo, M., Xie, H., Wong, L. K., Shuai, H. H., Li, Y. H., & Cheng, W. H. (2024). Lightweight deep learning for resource-constrained environments: A survey. ACM Computing Surveys, 56(10), 1-42. https://doi.org/10.1145/3657282

[15] Dolatyabi, P., Regan, J., & Khodayar, M. (2025). Deep learning for traffic scene understanding: A review. IEEE Access, 13, 13187-13237. https://doi.org/10.1109/ACCESS.2025.3529289

[16] Xinyong, Y., Xin, L., Lei, W., Junhong, J., Genlai, Z., Xichao, S., ... & Xinwei, W. (2025). Carrier platform-enhanced multiple-UAV cooperative task assignment with dual heterogeneities. Artificial Intelligence Review, 58(8), 248. https://doi.org/10.1007/s10462-025-11254-2

[17] Hernández, N., Almeida, F., & Blanco, V. (2024). Optimizing convolutional neural networks for IoT devices: performance and energy efficiency of quantization techniques: N. Hernández et al. The Journal of Supercomputing, 80(9), 12686-12705. https://doi.org/10.1007/s11227-024-05929-w

[18] Zhang, X., Fan, K., Hou, H., & Liu, C. (2022). Real-time detection of drones using channel and layer pruning, based on the yolov3-spp3 deep learning algorithm. Micromachines, 13(12), 2199. https://doi.org/10.3390/mi13122199

[19] Kassab, M., Zitar, R. A., Barbaresco, F., & Seghrouchni, A. E. F. (2024). Drone detection with improved precision in traditional machine learning and less complexity in single-shot detectors. IEEE Transactions on Aerospace and Electronic Systems, 60(4), 3847-3859. https://doi.org/10.1109/TAES.2024.3368991

[20] Telikani, A., Sarkar, A., Du, B., Santoso, F., Shen, J., Yan, J., ... & Yap, E. W. (2025). Autonomous Aerial Vehicles-Aided Intelligent Transportation Systems: Vision, Challenges, and Opportunities. IEEE communications surveys & tutorials, 27(6), 3772-3819. https://doi.org/10.1109/COMST.2025.3530913

[21] Zhang, P., Tian, H., Luo, H., Li, X., & Nie, G. (2023). A hybrid fast inference approach with distributed neural networks for edge computing enabled UAV swarm. Physical Communication, 60, 102129. https://doi.org/10.1016/j.phycom.2023.102129

[22] Belcastro, L., Cosentino, C., Marozzo, F., Presta, A., & Trunfio, P. (2025, March). Empowering efficient drone monitoring with low-latency edge-cloud continuum platforms. In 2025 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP) (pp. 333-340). IEEE. https://doi.org/10.1109/PDP66500.2025.00054

[23] Zhao, B., & Song, R. (2024). Enhancing two-stage object detection models via data-driven anchor box optimization in UAV-based maritime SAR. Scientific Reports, 14(1), 4765. https://doi.org/10.1038/s41598-024-55570-z

[24] Ijaz, H., Ahmad, R., Ahmed, R., Ahmed, W., Kai, Y., & Jun, W. (2023). A UAV-assisted edge framework for real-time disaster management. IEEE Transactions on Geoscience and Remote Sensing, 61, 1-13. https://doi.org/10.1109/TGRS.2023.3306151

[25] Kostage, K., Adepu, R., Monroe, J., Haughton, T., Mogollon, J., Poduvu, S., ... & Mitra, R. (2025, January). Federated learning-enabled network incident anomaly detection optimization for drone swarms. In Proceedings of the 26th International Conference on Distributed Computing and Networking (pp. 104-114). https://doi.org/10.1145/3700838.3700857