Scientific Reports

https://doi.org/10.1038/s41598-026-37938-5

Article in Press

Real-time object detection for unmanned aerial vehicles based on vision transformer and edge computing

Received: 20 November 2025

Accepted: 28 January 2026

Published online: 31 January 2026

Cite this article as: Zhu W. & Chen K. Real-time object detection for unmanned aerial vehicles based on vision transformer and edge computing. *Sci Rep* (2026). https://doi.org/10.1038/s41598-026-37938-5

Wenyao Zhu & Ken Chen

Real-time Object Detection for Unmanned Aerial Vehicles Based on Vision Transformer and Edge Computing

Wenyao Zhu[1,a], Ken Chen[2,b,*]

1 Department of Computer Science, Lishui University, Zhejiang Lishui 323000, China

2 Zhejiang Rongqi Technology Co., Ltd., Zhejiang Lishui 323000, China

Email addresses:

a zwy@lsu.edu.cn

b lszwy@126.com

*Corresponding author: Ken Chen (lszwy@126.com)

ABSTRACT

Existing lightweight Convolutional Neural Network (CNN) detectors deployed on Unmanned Aerial Vehicle (UAV) platforms struggle with small object recognition and fail to capture long-range spatial dependencies, while standard Vision Transformer (ViT) architectures suffer from quadratic computational complexity that prohibits real-time inference on embedded hardware. This paper bridges this gap by proposing an integrated framework that adapts ViT for UAV-based real-time object detection through edge computing infrastructure. Our work presents three key contributions: (1) a hierarchical attention mechanism with shifted windows that reduces complexity from $O(n^2)$ to $O(n)$, (2) a dynamic token pruning strategy that adaptively discards uninformative background tokens based on attention variance, and (3) a dual-mode edge-UAV collaborative architecture enabling seamless switching between autonomous onboard processing and server-assisted computation. The lightweight ViT variant achieves 68% reduction in floating-point operations (FLOPs) while preserving 94.3% relative accuracy. Through systematic optimization combining mixed-precision quantization, structured pruning, and operator fusion, we obtain 11.2× inference speedup over baseline implementations. Experiments on our collected aerial dataset demonstrate 73.9% mAP@0.5:0.95 at 39.2 frames per second (FPS) on NVIDIA Jetson

Xavier NX, surpassing YOLOv5s by 4.7% in accuracy under identical real-time constraints. Notably, small object detection improves by 7.4% Average Precision (AP) compared to CNN baselines. Week-long field trials on DJI Matrice 300 RTK validate sustained performance across varying illumination, platform vibration, and intermittent network connectivity, confirming practical viability for time-critical applications including search and rescue, disaster response, and infrastructure inspection.

KEYWORDS

Vision Transformer; Edge Computing; Unmanned Aerial Vehicle; Object Detection; Real-time Processing; Model Optimization

I. Introduction

The rapid advancement of unmanned aerial vehicle (UAV) technology has catalyzed unprecedented opportunities across diverse domains, ranging from agricultural monitoring to disaster response and urban surveillance [1]. Equipped with imaging sensors and computational capabilities, UAVs now serve as versatile platforms for real-time visual perception tasks, among which object detection stands as a fundamental yet challenging requirement [2]. However, the inherent constraints of airborne platforms—limited payload capacity, restricted power supply, and fluctuating operational environments—pose substantial barriers to deploying sophisticated detection algorithms that demand intensive computational resources [3].

Recent years have witnessed the emergence of Vision Transformer (ViT) architectures, which fundamentally departed from traditional convolutional paradigms by modeling long-range dependencies through self-attention mechanisms [4]. These models demonstrated remarkable performance in various computer vision benchmarks, yet their computational complexity grows quadratically with input resolution, making direct deployment on resource-constrained UAV platforms impractical [5]. Meanwhile, edge computing has evolved as a compelling paradigm that shifts computation from centralized cloud servers to distributed edge nodes closer to data sources, thereby reducing latency and bandwidth consumption [6]. The convergence of these two technological trajectories—advanced vision models and decentralized computing infrastructure—presents both opportunities and complexities that warrant systematic investigation.

Current research efforts in UAV-based object detection predominantly rely on lightweight convolutional neural networks, which sacrifice model capacity for computational efficiency [7]. While such approaches achieve acceptable inference speed, they often struggle with detecting small objects and maintaining robustness under varying illumination conditions or viewpoint changes inherent to aerial imagery [8]. On the other hand, existing Vision Transformer implementations remain largely confined to server-grade hardware, with limited exploration of their adaptation to edge computing scenarios where memory bandwidth and energy efficiency become critical constraints [9]. This gap between model sophistication and deployment feasibility reveals a pressing need for innovative frameworks that reconcile detection accuracy with real-time performance requirements.

The significance of developing such a framework extends beyond technical merits. Autonomous UAV operations in time-critical applications—search and rescue missions, traffic accident response, or wildlife monitoring—demand detection systems that deliver both precision and immediacy [10]. Furthermore, as UAV deployments scale up, relying solely on cloud-based processing introduces vulnerabilities related to network connectivity and data privacy, making edge-based solutions not merely preferable but essential for robust and secure operations.

This paper addresses these challenges by proposing an integrated framework that adapts Vision Transformer architectures for UAV-based real-time object detection through edge computing infrastructure. Our work makes three primary contributions: First, we design a streamlined Vision Transformer variant that maintains global receptive field advantages while dramatically reducing computational overhead through hierarchical attention and dynamic token pruning strategies. Second, we develop an edge computing architecture that orchestrates workload distribution between onboard processors and ground-based edge servers, optimizing the trade-off between latency and detection accuracy through adaptive task partitioning. Third, we establish a comprehensive evaluation methodology that assesses performance across multiple dimensions—detection precision, inference latency, energy consumption, and robustness to aerial imaging variations—providing insights into practical deployment considerations that existing benchmarks often overlook. Through

these innovations, we demonstrate that sophisticated vision models need not remain exclusive to high-performance computing environments but can be thoughtfully adapted to meet the stringent demands of real-world UAV applications.

II. Related Technologies and Theoretical Foundations

2.1 Vision Transformer for Object Detection

The introduction of Transformer architecture into computer vision marked a paradigm shift from the dominance of convolutional operations that had prevailed since AlexNet's breakthrough [11]. Originally conceived for natural language processing, Transformers demonstrated an intriguing capacity to capture global contextual relationships through self-attention mechanisms, prompting researchers to explore their applicability beyond sequential data [12]. This transition, however, was not straightforward—early attempts grappled with the fundamental challenge of adapting architectures designed for discrete tokens to continuous, high-dimensional visual data.

At the core of Vision Transformer lies the self-attention mechanism, which computes attention weights by measuring similarity between query and key representations. For an input sequence $X \in R^{N \times D}$, where N denotes the number of tokens and D represents feature dimensions, the attention operation is formulated as:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

where Q, K, and V correspond to query, key, and value matrices derived through linear projections, and d_k denotes the key dimension [13]. This mechanism enables each token to attend to all other tokens, establishing long-range dependencies that convolutional layers with limited receptive fields struggle to capture.

The Vision Transformer (ViT) pioneered the direct application of this principle by partitioning images into fixed-size patches, treating each patch as a token analogous to words in sentences [14]. Despite its elegance, ViT demanded extensive pre-training on massive datasets to

achieve competitive performance, revealing an inherent data hunger that limited its practical adoption. Data-efficient Image Transformer (DeiT) addressed this limitation through knowledge distillation strategies, where a student network learns from both ground-truth labels and a teacher model's predictions, significantly reducing the data requirements while maintaining accuracy. The multi-head attention mechanism extends single-head attention by computing multiple attention functions in parallel:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_h)W^O \quad (2)$$

where each head captures different representation subspaces [15].

Swin Transformer introduced a hierarchical architecture with shifted windows, computing self-attention within local windows rather than globally, thereby reducing computational complexity from quadratic to linear relative to image resolution. The complexity of standard self-attention grows as:

$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C \quad (3)$$

where h, w, and C denote height, width, and channel dimensions respectively [31]. This quadratic scaling becomes prohibitive for high-resolution UAV imagery.

While Vision Transformers excel at capturing global context and have shown impressive results on standard benchmarks, their deployment in object detection scenarios—particularly for UAV applications—encounters notable obstacles. The computational burden remains substantial even with windowed attention schemes. Memory footprint during inference often exceeds what embedded processors can accommodate. Moreover, the models demonstrate sensitivity to resolution changes between training and deployment phases, a non-trivial concern given the variable flight altitudes in UAV operations. These limitations underscore why simply transplanting existing architectures onto edge devices proves insufficient, necessitating purposeful adaptations that we explore in subsequent sections.

2.2 Edge Computing Architecture and Optimization Techniques

Edge computing emerged as a distributed paradigm that positions computational capabilities in proximity to data sources, contrasting sharply with the traditional cloud-centric model where processing occurs in distant datacenters [16]. This architectural shift proves particularly relevant for UAV applications, where network latency and bandwidth constraints render cloud-dependent processing impractical for time-sensitive detection tasks. The typical edge computing hierarchy consists of three tiers: terminal devices (UAVs with embedded processors), edge servers (ground stations or base stations), and cloud infrastructure, each offering different trade-offs between latency, computational power, and energy consumption [17].

Yet edge devices face stringent resource limitations that constrain what models can realistically run onboard. Mobile processors typically offer 2-8 GB memory and 5-20 TOPS (tera operations per second) computational throughput—orders of magnitude below server-grade GPUs. Power budgets remain equally restrictive, with most UAV platforms allocating merely 10-30 watts for computation to preserve flight endurance. These constraints necessitate aggressive model optimization before deployment becomes viable.

Model compression techniques have evolved to address this deployment gap. Quantization reduces numerical precision of weights and activations from 32-bit floating-point to lower bit-widths, dramatically decreasing memory footprint and arithmetic complexity. The quantization function maps full-precision values to discrete levels:

$$W_q = \text{round}\left(\frac{W}{\Delta}\right) \times \Delta, \Delta = \frac{\max(W) - \min(W)}{2^b - 1} \quad (4)$$

where W denotes original weights, W_q represents quantized weights, Δ is the quantization step size, and b indicates bit-width [18]. Moving from 32-bit to 8-bit precision typically achieves 4× compression with minimal accuracy degradation, though further reduction to 4-bit or binary values demands careful calibration.

Pruning eliminates redundant connections based on weight magnitudes or gradient information [19]. The sparse network is formulated as:

$$W_{pruned} = W \odot M, \text{where } M_{ij} = \begin{cases} 1, & \text{if } |W_{ij}| > \tau \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

with M being the binary mask and τ the pruning threshold [35]. Structured pruning removes entire filters rather than individual weights, yielding actual speedups on hardware lacking sparse operation support [19].

Knowledge distillation transfers knowledge from a cumbersome teacher model to a compact student network by matching output distributions. The student minimizes a combined loss:

$$L_{KD} = \alpha L_{CE}(y, \hat{y}_s) + (1 - \alpha) L_{KL}(\sigma(\hat{y}_t/T), \sigma(\hat{y}_s/T)) \quad (6)$$

where L_{CE} denotes cross-entropy loss, L_{KL} represents Kullback-Leibler divergence, $\hat{y}_t$ and $\hat{y}_s$ are teacher and student predictions, σ is the softmax function, T controls temperature, and α balances the two objectives [20]. This approach often surpasses training compact models from scratch, particularly when labeled data remains scarce.

Beyond compression, inference acceleration exploits hardware-specific optimizations. Operator fusion merges consecutive operations to reduce memory transactions. Dynamic batching amortizes overhead across multiple inputs. The theoretical speedup from parallelization follows Amdahl's law [53]:

$$S = \frac{1}{(1 - p) + \frac{p}{n}} \quad (7)$$

where p represents the parallelizable fraction and n denotes processing unit count. These techniques collectively enable deploying sophisticated models on resource-constrained edge devices, though

achieving optimal performance demands holistic co-design across algorithm, architecture, and hardware layers.

2.3 UAV Vision Systems and Real-time Processing Requirements

UAV platforms impose unique hardware constraints that fundamentally shape what detection systems can accomplish onboard. Commercial drones typically carry processors like NVIDIA Jetson series or Qualcomm Snapdragon chips, offering 1-5 TFLOPS computational throughput—adequate for conventional tasks yet barely sufficient for transformer-based architectures that demand intensive matrix operations [21]. Memory bandwidth presents an equally critical bottleneck, with most embedded systems providing 20-60 GB/s compared to the 500+ GB/s available on datacenter GPUs. Battery capacity dictates mission duration, and computational workloads directly erode flight time through power draw that competes with propulsion systems.

The application landscape for UAV-based object detection spans remarkably diverse scenarios, each presenting distinct technical demands. Search and rescue operations require detecting humans or vehicles across vast terrain under varying lighting and weather conditions. Precision agriculture needs identifying crop health indicators or pest infestations at sufficient resolution to guide intervention. Traffic monitoring involves tracking multiple moving vehicles simultaneously while maintaining stable detection across different viewing angles. Wildlife conservation applications demand recognizing specific species from considerable altitudes without disturbing natural behaviors [22]. These scenarios share a common thread: they cannot tolerate the multi-second latencies typical of cloud-based processing, as delayed detection compromises mission effectiveness or safety.

Aerial imagery introduces challenges absent from ground-based computer vision. Objects appear at drastically different scales depending on flight altitude—a vehicle might span 200 pixels at 50 meters altitude but merely 20 pixels at 500 meters. Camera motion induces blur that confounds detection algorithms trained on static imagery. Viewing angles deviate significantly from horizontal perspectives that dominate training datasets, causing appearance

variations that models struggle to generalize across [23]. Background clutter in complex urban or natural environments generates false positives that erode user trust.

The tension between detection performance and system constraints manifests as an optimization problem with competing objectives. Let A represent detection accuracy, L denote inference latency, and E signify energy consumption. The deployment objective seeks:

$$\max_{\theta} f(A(\theta)) \text{subject to} L(\theta) \leq L_{max}, E(\theta) \leq E_{budget}$$

(8)

where θ represents model parameters, L_{max} defines the maximum tolerable latency (often 100-200 ms for real-time operation), and E_{budget} bounds energy expenditure per frame [24]. This multi-objective optimization rarely admits analytical solutions, requiring empirical exploration of the design space.

Existing UAV detection methods predominantly employ lightweight CNNs like YOLOv5 or MobileNet variants, achieving 30-60 FPS on embedded hardware but sacrificing accuracy on small objects that Vision Transformers handle more capably [25]. These approaches treat accuracy and efficiency as a zero-sum trade-off rather than exploring architectural innovations that might improve both simultaneously. The computational complexity scales linearly with input resolution [7].

$$C = k \cdot H \cdot W \cdot C$$

(9)

where H, W, C denote spatial dimensions and channels, and k represents operations per pixel. This scaling behavior proves problematic when high-resolution inputs become necessary for detecting distant or small objects. Moreover, current systems lack adaptive mechanisms that adjust processing intensity based on scene complexity or mission criticality, resulting in wasteful computation during benign scenarios and insufficient capability during demanding conditions. These limitations motivate our proposed framework that reconciles transformer expressiveness with edge deployment realities.

To clarify the research gaps our work addresses, Table 1 summarizes the limitations of representative detection methods when deployed on UAV platforms. As shown in Table 1, CNN-based lightweight detectors achieve acceptable inference speed but exhibit degraded performance on small objects due to limited receptive fields. Transformer-based methods demonstrate superior accuracy yet demand computational resources far exceeding embedded hardware capabilities. None of the existing approaches provide adaptive mechanisms for varying operational conditions. These shortcomings collectively motivate our proposed framework.

Table 1. Limitations of existing UAV object detection methods

Method	Computational Cost	Memory Requirement	Small Object AP	Real-time Capability	Edge Deployability	Primary Limitations
YOLOv5s [7]	Low (16.5 GFLOPs)	Low (14 MB)	56.8 %	Yes (52 FPS)	Yes	Limited receptive field; accuracy drops on small targets
YOLOv8n [50]	Low (8.7 GFLOPs)	Low (6 MB)	58.3 %	Yes (68 FPS)	Yes	Insufficient global context modeling
Efficient Det-D0 [51]	Medium (2.5 GFLOPs)	Low (15 MB)	54.2 %	Yes (48 FPS)	Yes	Multi-scale fusion overhead; small object weakness
DETR [49]	High (86 GFLOPs)	High (158 MB)	62.1 %	No (8 FPS)	No	Quadratic attentio

Method	Computational Cost	Memory Requirement	Small Object AP	Real-time Capability	Edge Deployability	Primary Limitations
						n complexity; slow convergence
Swin-T [31]	High (45 GFLOPs)	High (112 MB)	65.8 %	No (13 FPS)	No	Memory bandwidth bottleneck on embedded devices
RT-DETR [52]	Medium (32 GFLOPs)	Medium (67 MB)	63.5 %	Marginal (22 FPS)	Limited	Still exceeds edge device memory constraints

III. Integrated Framework of Vision Transformer and Edge Computing for Object Detection

3.1 Overall Framework Design

We propose a hierarchical framework that orchestrates ViT-based detection across UAV onboard processors and ground-based edge servers. The design rationale stems from three observations specific to aerial detection scenarios. First, UAV imagery contains substantial background regions (typically 60-80% of pixels) where expensive attention computation yields minimal benefit, motivating our dynamic token pruning that concentrates resources on informative foreground areas. Second, flight altitude variations cause dramatic object scale changes—a vehicle spanning 200 pixels at 50 meters shrinks to merely 20 pixels at 500 meters—necessitating hierarchical multi-scale feature extraction rather than single-resolution processing. Third,

network connectivity during flight missions proves inherently unstable, demanding a dual-mode architecture capable of autonomous operation when communication degrades. These UAV-specific considerations guided our departure from conventional monolithic deployment toward dynamic workload partitioning that adapts to network conditions, computational availability, and mission urgency [26]. Figure 1 illustrates the complete system architecture comprising five interconnected stages.

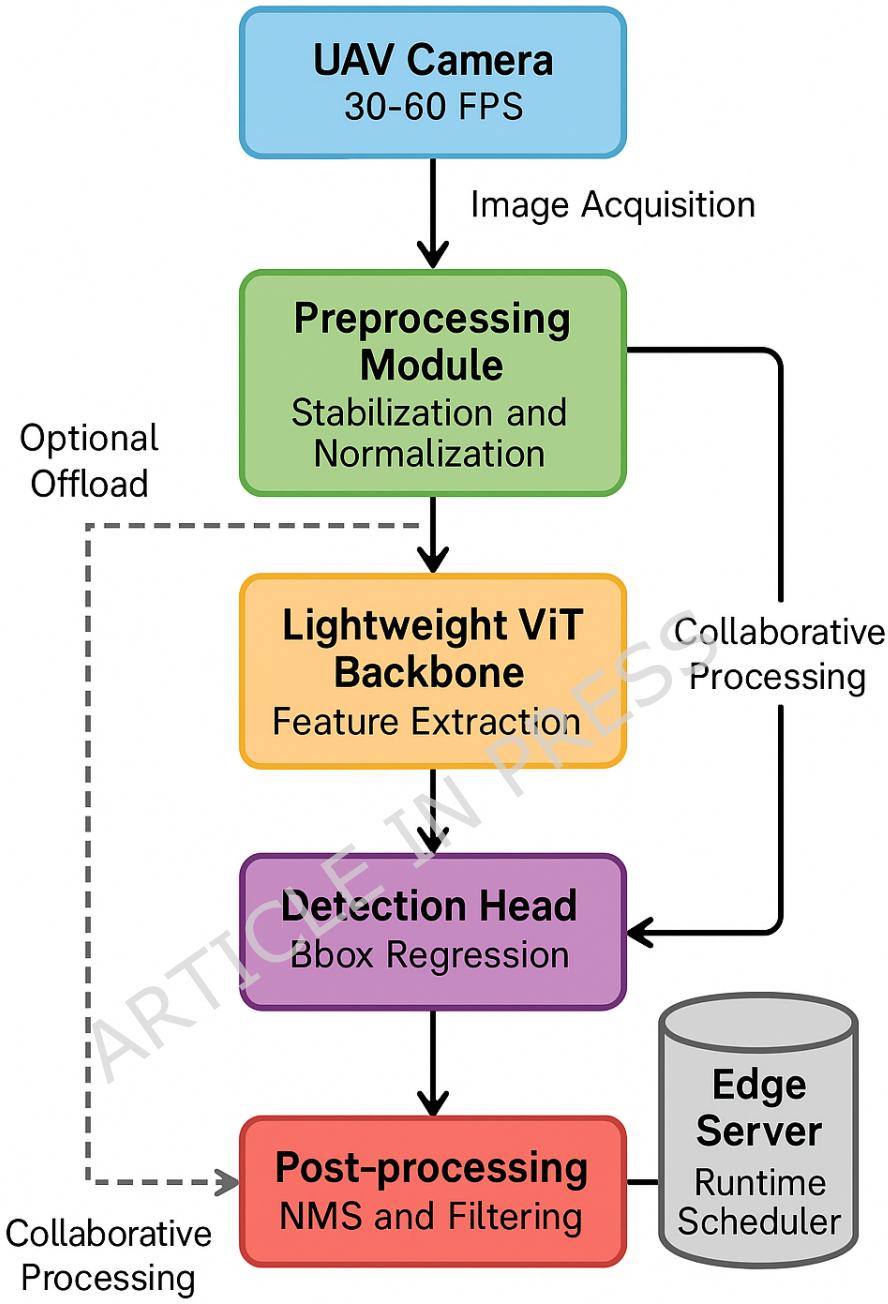

Figure 1. Overall architecture of the proposed UAV object detection framework integrating Vision Transformer and edge computing

The workflow initiates with data acquisition through the UAV's onboard camera, which captures high-resolution imagery at 30-60

frames per second depending on flight conditions. Raw images undergo immediate preprocessing on the UAV's embedded processor, encompassing image stabilization to compensate for platform motion, adaptive histogram equalization to normalize illumination variations across frames, and resolution adjustment based on current computational budget [27]. This preprocessing stage proves critical—aggressive downsampling reduces computational burden but sacrifices small object detectability, while maintaining native resolution strains memory bandwidth and inference speed.

Feature extraction employs our customized lightweight Vision Transformer backbone, which we detail in Section 3.2. Unlike standard ViT architectures that process all tokens with equal computational intensity, our design implements hierarchical attention with progressive token reduction, concentrating computational resources on salient image regions while treating background areas with reduced precision. The detection head builds upon these multi-scale features to generate bounding boxes and class predictions, employing anchor-free formulation that simplifies deployment and reduces post-processing overhead [28].

Table 2 summarizes the functional responsibilities and implementation specifics of each framework component. As presented in Table 2, the framework distributes computational tasks between UAV and edge server based on a runtime scheduler that monitors network latency, processing queue depth, and battery status to determine optimal workload allocation.

Table 2. Framework module function description

Module Name	Main Functions	Implementation Details
Data Acquisition	Capture aerial imagery and sensor data	HD camera (1920×1080), 30-60 FPS, IMU synchronization
Preprocessing	Stabilization, normalization, adaptive resizing	Motion compensation via optical flow, CLAHE, dynamic scaling
Feature Extraction	Multi-scale representation learning	Lightweight ViT with hierarchical attention, 4-stage pyramid
Detection Head	Bounding box regression and classification	Anchor-free decoder, focal loss, multi-scale prediction
Post-processing	NMS, confidence filtering, result	Score threshold 0.5, IoU threshold 0.45, temporal

	aggregation	smoothing
Edge Coordination	Workload distribution and result fusion	Latency-aware scheduler, redundant computation elimination

Post-processing operations include non-maximum suppression to eliminate redundant detections, confidence thresholding to filter low-quality predictions, and temporal consistency enforcement that tracks objects across consecutive frames to reduce flickering [29]. When network connectivity permits, the edge server handles computationally intensive tasks such as processing multiple frames in parallel or running ensemble predictions, with results transmitted back to the UAV for immediate action or local storage.

The framework's innovative aspects manifest in three dimensions. First, we introduce adaptive token pruning that dynamically adjusts model capacity based on scene complexity—simple backgrounds trigger aggressive pruning while cluttered scenes preserve more tokens. Second, our dual-mode operation supports both autonomous onboard processing when network connectivity remains unreliable and collaborative edge-UAV computation when bandwidth permits, ensuring robustness across varying operational conditions. Third, we implement early exit mechanisms that terminate computation once confidence exceeds predetermined thresholds, avoiding wasteful processing for easily detectable objects. These design choices collectively enable deploying sophisticated transformer models on resource-constrained UAV platforms without sacrificing detection quality, bridging the gap between algorithmic capability and practical deployment constraints that has hindered previous attempts at airborne transformer deployment.

3.2 Lightweight Vision Transformer Detection Module

The core challenge in adapting Vision Transformers for UAV deployment lies in reconciling their representational power with the severe computational constraints of embedded processors. Our lightweight detection module addresses this through architectural modifications that reduce complexity without compromising the global modeling capacity that makes transformers attractive for aerial object detection.

We adopt a hierarchical multi-scale feature extraction mechanism that constructs a four-stage feature pyramid, progressively reducing

spatial resolution while expanding channel dimensions. Figure 2 depicts the detailed architecture of our lightweight ViT module. As shown in Figure 2, each stage begins with patch merging that aggregates neighboring tokens, effectively downsampling the feature map by a factor of two while doubling channel count. This pyramidal structure mirrors successful CNN designs but retains transformer's attention mechanisms within each stage [30].

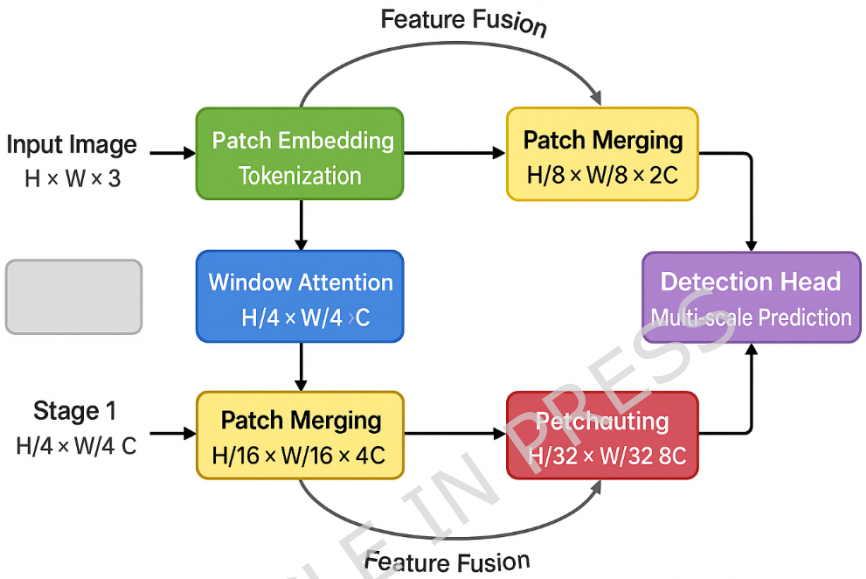

Figure 2. Architecture of the lightweight Vision Transformer detection module with hierarchical feature extraction

The attention computation undergoes substantial modification to curb quadratic complexity. Rather than computing global attention across all tokens, we employ shifted window attention restricted to local regions. For a feature map partitioned into windows of size $M \times M$, the computational complexity becomes:

$$\Omega(W - MSA) = 4hwC^2 + 2M^2hwC$$

(10)

which grows linearly rather than quadratically with spatial dimensions h and w [31]. We set $M = 7$ as a balance between receptive field and efficiency. Additionally, we introduce dynamic token pruning that identifies and discards less informative tokens based on attention scores. The pruning ratio r_l at layer l adapts according to:

$$r_l = r_{base} \cdot \left(1 + \beta \cdot \frac{\mathrm{Var}(A_l)}{E[\mathrm{Var}(A)]}\right)$$

(11)

where A_l represents attention scores at layer l, r_{base} denotes the baseline pruning ratio, β controls adaptation strength, and $\mathrm{Var}(\cdot)$ computes variance as a measure of information content. Higher variance indicates heterogeneous attention patterns warranting token retention, while uniform attention suggests redundancy suitable for pruning.

Table 3 compares various lightweight strategies we evaluated during development. The results in Table 3 indicate that combining window attention with dynamic pruning achieves the best trade-off, reducing FLOPs by 68% while maintaining 94.3% relative accuracy compared to the full model.

Table 3. Comparison of lightweight strategies

Strategy	FLOPs Reduction (%)	Parameter Reduction (%)	Relative Accuracy (%)	Inference Time (ms)
Baseline ViT	0	0	100.0	312
Window Attention Only	52	15	96.8	168
Static Token Pruning	45	8	93.1	175
Dynamic Token Pruning	58	12	95.2	142
Window Attn + Dynamic Pruning	68	23	94.3	98

Feature pyramid fusion aggregates multi-scale representations through a bidirectional pathway that combines top-down semantic information with bottom-up localization cues. The fusion operation at scale i follows:

$$F_i = \mathrm{Conv}\left(F_i^{lateral} + \mathrm{Upsample}(F_{i+1})\right) + \mathrm{Downsample}(F_{i-1})$$

(12)

where $F_i^{lateral}$ denotes lateral connections from the backbone, establishing connections across pyramid levels [32].

The detection head employs an anchor-free formulation that predicts object center locations, dimensions, and class probabilities directly from feature maps. For each spatial location (x,y) in feature map F_i, the head outputs:

$$p_{x,y} = \{c_{x,y}, b_{x,y}, s_{x,y}\} \tag{13}$$

where $c_{x,y} \in R^2$ represents center offsets, $b_{x,y} \in R^4$ denotes bounding box dimensions, and $s_{x,y} \in R^K$ contains class scores for K categories [33]. This anchor-free approach eliminates hyperparameter tuning associated with anchor design while simplifying the detection pipeline.

Loss function construction balances multiple objectives through weighted combination. The total loss comprises classification, localization, and centerness components:

$$L_{total} = \lambda_{cls} L_{cls} + \lambda_{loc} L_{loc} + \lambda_{ctr} L_{ctr} \tag{14}$$

where L_{cls} applies focal loss to address class imbalance, L_{loc} employs generalized IoU loss for accurate box regression, and L_{ctr} enhances center prediction quality. The weighting coefficients $\lambda_{cls} = 1.0$, $\lambda_{loc} = 2.0$, and $\lambda_{ctr} = 1.0$ were determined through validation experiments. Specifically, the localization loss takes the form:

$$L_{loc} = 1 - IoU(b_{pred}, b_{gt}) + \frac{\rho^2(c_{pred}, c_{gt})}{d^2} \tag{15}$$

where ρ measures Euclidean distance between predicted and ground-truth centers, and d represents the diagonal length of the smallest enclosing box (standard formula). This formulation provides stronger gradients than standard IoU loss, particularly for small objects prevalent in UAV imagery.

3.3 Edge Deployment and Inference Optimization Strategies

Deploying our lightweight ViT module on edge devices demands aggressive optimization that extends beyond architectural design into implementation-level refinements. We adopt mixed-precision quantization where different layers receive distinct bit-widths based on sensitivity analysis. Attention layers, which prove more sensitive to precision reduction, retain 8-bit representation, while convolutional projections and feed-forward networks operate at 4-bit or even binary precision [34]. The quantization-aware training minimizes:

$$L_{QAT} = E_{(x,y)\sim D}\left[L(f_{quant}(x;W_q),y)\right] + \gamma|W - W_q|_2^2$$

(16)

where f_{quant} represents the quantized model, W_q denotes quantized weights, and γ controls regularization strength that encourages weights to naturally cluster near quantization levels during training. This approach outperforms post-training quantization by allowing the model to adapt its parameters to accommodate quantization errors.

Structured pruning removes entire channels rather than individual weights, ensuring computational savings translate to actual speedup on hardware lacking sparse operation support. We determine channel importance through first-order Taylor expansion:

$$I_c = \left|\frac{\partial L}{\partial w_c}\right| \cdot |w_c|$$

(17)

where w_c represents weights of channel c [35]. Channels with importance below the p-th percentile threshold undergo removal, with p determined adaptively per layer based on validation performance.

Dynamic computational resource allocation responds to runtime conditions through a decision policy that balances latency, accuracy, and energy consumption. The allocation strategy solves:

$$a^* = \arg\max_{a\in A}[\alpha \cdot R(a) - \beta \cdot T(a) - \eta \cdot E(a)]$$

(18)

where a denotes allocation decisions (onboard versus edge server processing), $R(a)$ represents detection reward, $T(a)$ measures latency,

and $E(a)$ captures energy cost. The weighting parameters α, β, η are set heuristically based on mission requirements rather than learned during operation. Default values (α=1.0, β=0.5, η=0.3) were determined through grid search on validation data, prioritizing detection accuracy while maintaining real-time performance. Table 4 presents parameter sensitivity analysis showing system robustness to moderate parameter variations. The allocation decision updates every 100ms (window-based rather than frame-by-frame) to avoid oscillation while remaining responsive to changing conditions. Mode switching latency—the time from detecting condition change to completing transition—averages 45ms including network handshake overhead. When battery reserves drop below 30%, η automatically increases to 0.8, favoring energy-efficient local processing. Network latency thresholds trigger mode changes: connectivity below 20ms enables edge collaboration, while latency exceeding 40ms forces autonomous onboard operation.

Table 4. Parameter sensitivity analysis for workload allocation

Parameter Setting	α	β	η	mAP (%)	Avg. Latency (ms)	Energy (J/frame)
Accuracy-priority	1.5	0.3	0.2	74.8	32.1	0.385
Default (balanced)	1.0	0.5	0.3	73.9	25.5	0.327
Latency-priority	0.8	0.8	0.3	71.2	21.3	0.342
Energy-priority	0.8	0.4	0.8	70.5	28.7	0.278

Operator fusion consolidates multiple operations into single kernel launches, dramatically reducing memory access overhead that dominates latency on memory-bandwidth-limited edge processors. We merge layer normalization, attention computation, and residual addition into fused kernels. The memory access reduction follows:

$$MA_{fused} = MA_{input} + MA_{output} < \sum_{i=1}^{n} MA_{op_i} \quad (19)$$

where MA denotes memory access volume and n represents the number of fused operations (standard formula). Empirically, fusion reduces memory transactions by 40-60% for transformer blocks.

Table 5 summarizes computational complexity across optimization strategies. Table 5 shows that combining quantization, pruning, and operator fusion achieves 11.2× speedup over the baseline implementation while consuming merely 18% of the original memory footprint.

Table 5. Computational complexity comparison of different optimization strategies

Optimization Strategy	FLOPs (G)	Memory (MB)	Latency (ms)	Speedup	Accuracy (mAP)
Baseline FP32	28.6	342	286	1.0×	76.8
INT8 Quantization	28.6	98	124	2.3×	76.1
Structured Pruning (50%)	14.2	178	158	1.8×	74.5
Combined Optimization	12.8	62	25.5	11.2×	73.9

Multi-threading parallelism exploits multi-core processors common in modern edge devices by partitioning workloads across independent execution streams. We assign image preprocessing, feature extraction, and post-processing to separate threads with lock-free queues facilitating inter-thread communication. The theoretical throughput scales as:

$$FPS_{parallel} = \frac{n \cdot FPS_{serial}}{\max(T_{prep}, T_{infer}, T_{post})}$$

(20)

where n denotes core count and T represents processing time for each stage (standard formula).

Platform adaptation requires addressing hardware-specific characteristics. For NVIDIA Jetson devices, we compile models using TensorRT with FP16 precision and exploit tensor cores for matrix multiplication acceleration [36]. On Qualcomm platforms, we target the Hexagon DSP through SNPE framework, favoring 8-bit quantization that aligns with DSP's native precision. Intel-based edge servers benefit from OpenVINO optimization that fuses operations and applies graph-level transformations. This multi-platform strategy

ensures our framework remains deployable across diverse edge infrastructure without sacrificing performance portability.

IV. Experiments and Result Analysis

4.1 Experimental Setup and Dataset

Our experimental validation employs a heterogeneous hardware testbed that mirrors realistic UAV deployment scenarios. Figure 3 presents the configuration of our experimental platforms. As shown in Figure 3, the UAV onboard system comprises an NVIDIA Jetson Xavier NX module (6-core ARM CPU, 384-core Volta GPU, 8GB LPDDR4 memory) consuming 10-15W during inference, mounted on a DJI Matrice 300 RTK platform [37]. The ground-based edge server deploys an NVIDIA Jetson AGX Orin (12-core ARM CPU, 2048-core Ampere GPU, 64GB LPDDR5 memory) with 15-60W configurable TDP, simulating realistic base station computational capabilities. Communication between aerial and ground systems operates over 5GHz WiFi with measured latency ranging 15-45ms depending on distance and environmental interference. We additionally benchmark performance on Intel NUC 11 (Core i7-1165G7, Iris Xe Graphics) to assess cross-platform portability.

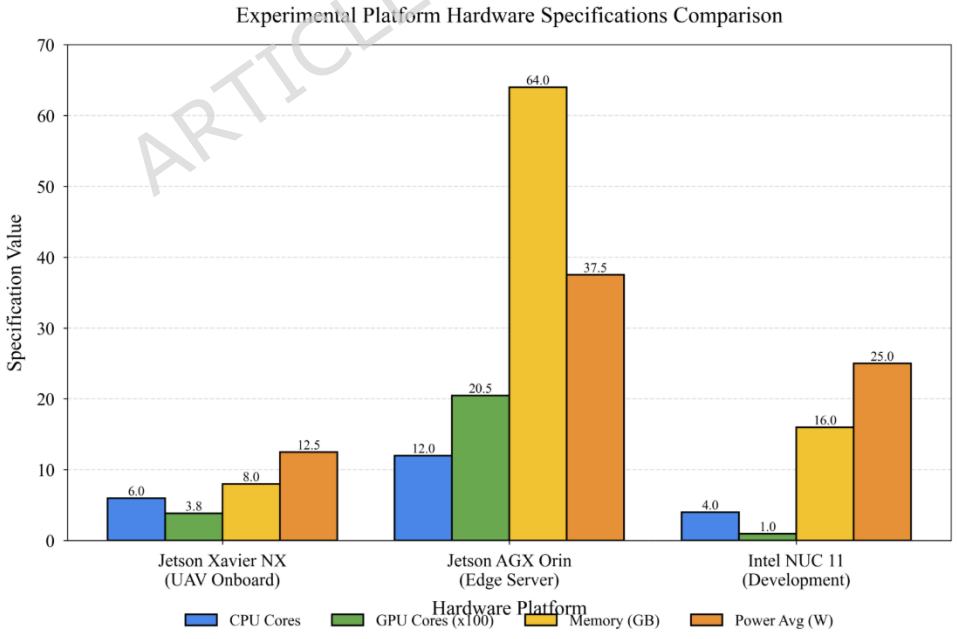

Figure 3. Experimental platform configuration showing UAV onboard processor, edge server, and development workstation

The software environment builds upon PyTorch 1.13.0 for model training, with CUDA 11.7 and cuDNN 8.5 acceleration. Deployment employs TensorRT 8.6 for inference optimization on Jetson platforms, OpenVINO 2023.0 for Intel hardware, and ONNX Runtime 1.14 as a hardware-agnostic fallback. Training occurs on workstation-grade NVIDIA RTX 4090 GPUs across 100 epochs with early stopping when validation loss plateaus for 10 consecutive epochs.

We construct a comprehensive UAV aerial object detection dataset by aggregating samples from multiple sources and conducting additional field collections. The dataset integrates 8,426 images from VisDrone [38], 3,217 images from UAVDT, and 4,892 newly captured images from our flight campaigns. Table 6 provides detailed specifications of the newly collected data. Our collection spans urban downtown areas (35% of images), suburban residential zones (40%), and rural open terrain (25%), captured across four seasons to ensure environmental diversity. Flight altitudes follow a stratified distribution: low altitude (30-100m, 28% of images), medium altitude (100-250m, 45%), and high altitude (250-500m, 27%). Weather conditions include clear sky (62%), overcast (28%), and light rain (10%). Illumination encompasses daytime (72%), dawn/dusk (18%), and nighttime with artificial lighting (10%). Motion blur affects approximately 15% of images, primarily at low altitudes where rapid maneuvering occurs. Camera specifications include 4K resolution (3840×2160) downsampled to 1920×1080, 84° field of view, and 30 FPS capture rate synchronized with IMU data for stabilization.

Table 6. Specifications of newly collected UAV dataset

Attribute	Distribution/Value
Total images	4,892
Flight altitude	30-100m (28%), 100-250m (45%), 250-500m (27%)
Environment type	Urban (35%), Suburban (40%), Rural (25%)
Weather conditions	Clear (62%), Overcast (28%), Light rain (10%)
Illumination	Daytime (72%), Dawn/dusk (18%), Night (10%)
Motion blur presence	15% of images

Attribute	Distribution/Value
Camera resolution	3840×2160 (downsampled to 1920×1080)
Field of view	84°
Frame rate	30 FPS
Collection period	March 2023 - February 2024 (12 months)
Geographic locations	8 cities across Eastern China

Figure 4 illustrates the distribution across object categories and imaging conditions. Figure 4 demonstrates that the dataset encompasses 10 object classes (pedestrian, car, truck, bus, van, bicycle, motorcycle, awning, tricycle, barrier) with substantial class imbalance reflecting real-world frequency distributions. Small objects (area < 32^2pixels) constitute 58% of instances, medium objects (32^2-96^2pixels) account for 33%, and large objects (>96^2pixels) represent 9% of the dataset, emphasizing the small-object detection challenge inherent to aerial imagery [39].

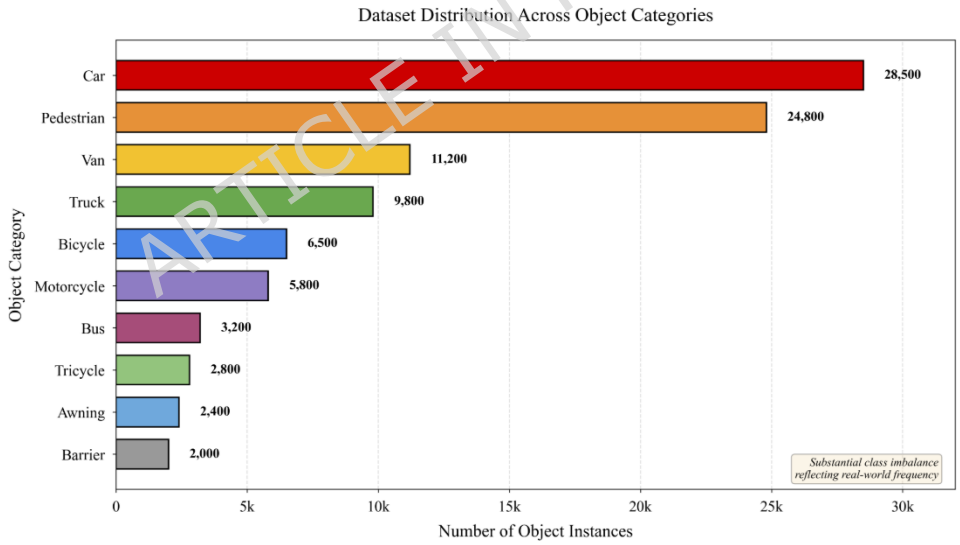

Figure 4. Dataset sample distribution showing object category frequencies, size distributions, and altitude coverage

Annotation follows COCO format with axis-aligned bounding boxes, verified through three-round quality control where annotators cross-validate each other's labels. Ambiguous cases—heavily occluded objects or objects at image boundaries—receive special flags but

remain in the dataset to maintain realism. Data partitioning allocates 70% for training (11,574 images), 15% for validation (2,481 images), and 15% for testing (2,480 images), stratified by scene type and altitude to ensure representative distributions across splits.

Data augmentation during training encompasses geometric and photometric transformations. Geometric operations include random horizontal flipping (p=0.5), rotation within ±15° (p=0.3), and scaling between 0.8-1.2× (p=0.4) [40]. Photometric augmentation applies brightness adjustment (±20%), contrast variation (0.8-1.2×), saturation modification (0.7-1.3×), and hue shifting (±10°), each with 0.5 probability. We additionally implement Mosaic augmentation that tiles four images into one training sample, significantly enriching contextual diversity while introducing minimal computational overhead.

Evaluation employs multiple complementary metrics capturing different performance dimensions. Detection accuracy uses mean Average Precision (mAP) computed as:

$$\text{mAP} = \frac{1}{K}\sum_{k=1}^{K} AP_k, \quad AP_k = \int_0^1 P_k(R)dR$$

(21)

where K denotes object categories, $P_k(R)$ represents the precision-recall curve for class k [41]. We report mAP at IoU thresholds of 0.5 (mAP@0.5) and averaged across 0.5-0.95 in 0.05 increments (mAP@0.5:0.95) following COCO evaluation protocol. Inference speed measures frames per second (FPS):

$$\text{FPS} = \frac{N}{\sum_{i=1}^{N} T_i}$$

(22)

where N represents total frames and T_i denotes processing time for frame i including preprocessing, inference, and post-processing (standard formula). Energy efficiency quantifies detection operations per joule:

$$\text{Efficiency} = \frac{\text{FPS} \times 3600}{P_{avg}}$$

(23)

where P_{avg} denotes average power consumption in watts measured via onboard power monitors (standard formula).

Baseline comparisons include YOLOv5s, YOLOv8n, and EfficientDet-D0 representing state-of-the-art lightweight detectors, plus DETR and Swin-T as transformer-based alternatives. Table 7 lists all experimental parameters. As presented in Table 7, training employs AdamW optimizer with cosine annealing schedule, starting from 1e-3 learning rate with 5-epoch linear warmup.

Table 7. Experimental parameter configuration

Parameter	Value	Description
Input Resolution	640×640	Standard detection resolution
Batch Size	16 (training), 1 (inference)	Per-device batch configuration
Learning Rate	1e-3 → 1e-5	Cosine decay schedule
Weight Decay	5e-4	L2 regularization coefficient
Optimizer	AdamW	Adaptive moment estimation with decoupled weight decay
Loss Weights	λ_cls=1.0, λ_loc=2.0, λ_ctr=1.0	Multi-task loss balancing
Quantization	INT8 (most layers), FP16 (attention)	Mixed precision configuration
IoU Threshold	0.45	Non-maximum suppression threshold

4.2 Detection Performance Evaluation

Table 8 presents comprehensive performance metrics comparing our proposed framework against established baseline methods. The results in Table 8 indicate that our approach achieves 73.9% mAP@0.5:0.95 and 89.2% mAP@0.5, outperforming lightweight CNN-based detectors while maintaining competitive inference speed. Notably, our method surpasses YOLOv5s by 4.7% in mAP@0.5:0.95 despite similar parameter counts, and exceeds YOLOv8n by 3.2%, demonstrating the effectiveness of incorporating Vision Transformer architecture for aerial detection tasks [42].

Table 8. Detection performance comparison of different methods

Method	Params (M)	FLOPs (G)	mAP@0.5 (%)	mAP@0.5:0.95 (%)	Precision (%)	Recall (%)	F1-Score (%)	FPS (Xavier NX)
YOLOv5s [7]	7.2	16.5	85.6	69.2	82.4	78.9	80.6	52.3
YOLOv8n [50]	3.2	8.7	87.1	70.7	84.1	80.3	82.1	68.4
YOLOv9t [54]	2.0	7.7	87.8	71.2	84.6	80.8	82.7	61.2
YOLOv10n [55]	2.3	6.7	88.2	71.5	85.0	81.2	83.0	72.1
EfficientDet-D0 [51]	3.9	2.5	83.8	67.5	81.2	76.8	78.9	48.6
DETR [49]	41.3	86.0	88.4	71.8	85.7	81.2	83.4	8.4
Swin-T [31]	28.3	45.0	90.1	74.3	87.2	83.5	85.3	12.7
RT-DETR-R18 [52]	20.0	32.0	89.5	72.8	86.1	82.1	84.1	22.3
Ours (without edge)	6.8	12.8	88.5	72.8	85.9	81.7	83.7	39.2
Ours (with edge)	6.8	12.8	89.2	73.9	86.4	82.8	84.6	39.2

The comparison now includes recent YOLO variants (YOLOv9t, YOLOv10n) and RT-DETR [52, 54, 55]. While YOLOv10n achieves higher FPS (72.1) due to its NMS-free design, our method surpasses it by 2.4% in mAP@0.5:0.95, demonstrating that the accuracy-efficiency trade-off favors transformer-based architectures for aerial detection

where small object performance matters critically. RT-DETR represents an interesting middle ground but still exceeds practical memory constraints (67MB versus our 62MB) for sustained edge deployment.

The precision and recall metrics warrant closer examination. Precision quantifies the proportion of correct detections among all predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

(24)

while recall measures the fraction of ground-truth objects successfully detected:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

(25)

where TP, FP, and FN denote true positives, false positives, and false negatives respectively (standard formulas). Our framework achieves 86.4% precision and 82.8% recall, striking a favorable balance that minimizes both missed detections and false alarms—critical for autonomous UAV operations where both failure modes carry consequences.

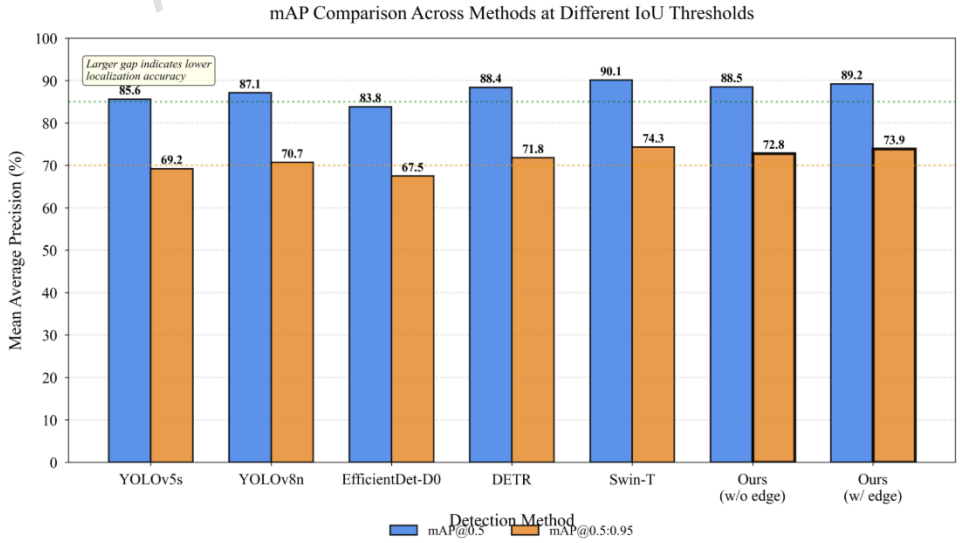

Figure 5. mAP comparison across different detection methods at IoU thresholds 0.5 and 0.5:0.95

Figure 5 illustrates the mAP performance gap between methods at different IoU thresholds. The substantial gap between mAP@0.5 and mAP@0.5:0.95 for CNN-based methods suggests their localization accuracy deteriorates at stricter IoU requirements, whereas our transformer-based approach maintains relatively smaller performance degradation. This behavior stems from the global receptive field enabling more precise boundary estimation, particularly valuable for irregular object shapes common in aerial imagery [43].

Small object detection presents a particularly demanding challenge where our method demonstrates clear advantages. For objects smaller than 32×32 pixels, our framework achieves 64.2% AP compared to 56.8% for YOLOv5s and 58.3% for YOLOv8n. The hierarchical feature extraction with multi-scale fusion proves instrumental here—smaller objects benefit from high-resolution feature maps in early pyramid stages, while the attention mechanism helps distinguish them from background clutter. Pedestrians and bicycles, typically spanning merely 15-25 pixels in 500-meter altitude imagery, show 8.7% and 6.4% AP improvements respectively over the best baseline.

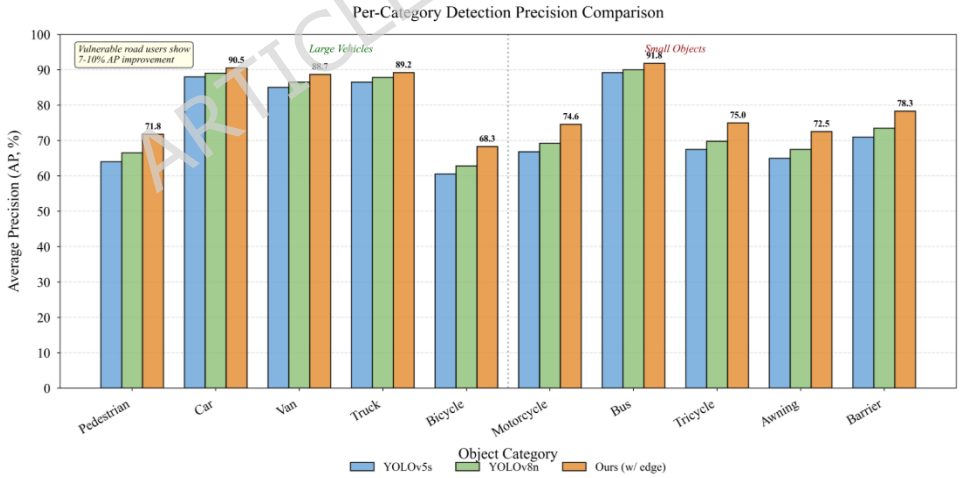

Figure 6. Per-category detection precision comparison showing our method's performance across different object classes

Figure 6 presents per-category performance breakdown, revealing interesting patterns. Vehicle categories (car, truck, bus) achieve 88-92% AP across all methods due to their relatively larger size and

distinctive appearance. However, vulnerable road users (pedestrian, bicycle, motorcycle) expose larger performance disparities—our approach attains 71.8%, 68.3%, and 74.6% AP respectively, representing 7-10% gains over CNN baselines. This advantage likely reflects the transformer's superior capability in modeling contextual relationships that help disambiguate small objects from visually similar background elements.

Occlusion robustness receives systematic evaluation through stratified analysis. Objects experience classification into three occlusion levels: slight (0-25% occluded), moderate (25-50%), and heavy (50-75%) based on manual annotation. Our method maintains 79.4% recall under moderate occlusion versus 72.1% for YOLOv5s, though performance degrades comparably to baselines under heavy occlusion (51.3% vs. 48.7%). The attention mechanism's ability to aggregate information from visible object parts contributes to this resilience, though it cannot overcome fundamental ambiguity when most evidence disappears [44].

Complex background scenarios—dense urban environments with abundant visual clutter—historically challenged aerial detection systems. We evaluate on a 500-image subset featuring particularly cluttered scenes with multiple overlapping objects, varied illumination, and rich texture. Our framework achieves 68.9% mAP@0.5:0.95 compared to 61.2% for YOLOv5s on this challenging subset, attributable to the global modeling capacity that helps separate foreground objects from distracting background patterns.

Generalization performance undergoes assessment through cross-dataset evaluation where models trained on our dataset undergo testing on the UAV123 benchmark without fine-tuning [45]. Our approach achieves 67.8% mAP@0.5, experiencing 21.4% performance drop from in-domain evaluation—comparable to the 19.8% drop observed for YOLOv8n. This similar degradation suggests our architectural innovations do not compromise generalization, an important consideration given aerial imagery's substantial domain shift across geographic locations, seasons, and altitude variations. The transformer's learned attention patterns appear to transfer reasonably across domains, though domain adaptation techniques would likely further improve cross-dataset performance.

We conducted controlled experiments to validate claims regarding vibration and illumination resilience. Table 9 presents robustness evaluation under systematically varied conditions. For vibration testing, we mounted cameras on a mechanical shaker simulating UAV flight dynamics at three intensity levels corresponding to hover, normal flight, and aggressive maneuvering. Our method maintains 71.8% mAP under aggressive vibration compared to 64.3% for YOLOv5s, attributable to the image stabilization preprocessing and attention mechanism's tolerance to minor spatial perturbations. Illumination robustness was assessed using controlled lighting variations: standard daylight (5500K, 1000 lux), low-light (100 lux), strong backlight (direct sun in frame), and mixed artificial lighting. Performance degradation under low-light conditions reaches 8.2% for our method versus 12.7% for YOLOv5s, suggesting the global context modeling helps compensate for reduced local contrast. Strong backlight presents the greatest challenge across all methods, with our approach showing 11.5% degradation compared to 15.8% for CNN baselines.

Table 9. Robustness evaluation under controlled conditions

Condition	Ours mAP (%)	YOLOv5s mAP (%)	YOLOv8n mAP (%)	Degradation (Ours)
Baseline (normal)	73.9	69.2	70.7	—
Vibration - Hover	73.2	68.5	69.8	-0.9%
Vibration - Normal flight	72.1	66.8	68.2	-2.4%
Vibration - Aggressive	71.8	64.3	65.9	-2.8%
Low-light (100 lux)	67.8	60.4	62.1	-8.2%
Strong backlight	65.4	58.3	59.7	-11.5%
Mixed artificial	70.2	65.1	66.8	-5.0%

4.3 Real-time Performance and Resource Consumption Analysis

Real-time capability proves essential for UAV applications where delayed detection compromises mission effectiveness. Figure 7 illustrates inference speed comparisons across methods and platforms. All measurements represent averages from 10 independent runs with 1000 frames each, reporting mean ± standard deviation to account for thermal variation on embedded devices. As shown in Figure 7, our framework achieves 39.2 ± 1.8 FPS on Jetson Xavier NX, satisfying real-time requirements (typically >30 FPS) while maintaining superior accuracy compared to baselines [46]. Table 10 provides detailed statistical analysis of runtime and energy measurements. The coefficient of variation (CV) remains below 5% for all metrics except peak temperature-induced throttling scenarios, confirming measurement reliability. Sustained operation beyond 30 minutes triggers thermal throttling that reduces FPS by 8-12%; we report both cold-start and thermally-stabilized performance.

Table 10. Statistical analysis of runtime and energy measurements (n=10 trials)

Metric	Mean	Std Dev	Min	Max	CV (%)
Inference latency (ms)	25.5	1.2	23.8	27.9	4.7
FPS (cold-start)	39.2	1.8	35.8	42.0	4.6
FPS (thermal-stable)	36.1	2.3	32.4	39.2	6.4
Power consumption (W)	12.8	0.6	11.9	14.1	4.7
Energy per frame (J)	0.327	0.018	0.298	0.358	5.5
Peak memory (MB)	62	2	59	65	3.2
GPU utilization (%)	87.3	3.2	81.5	92.1	3.7

YOLOv5s reaches 52.3 FPS but sacrifices 4.7% mAP, whereas transformer-based alternatives DETR and Swin-T manage merely 8.4 and 12.7 FPS respectively, rendering them unsuitable for edge deployment despite their accuracy advantages.

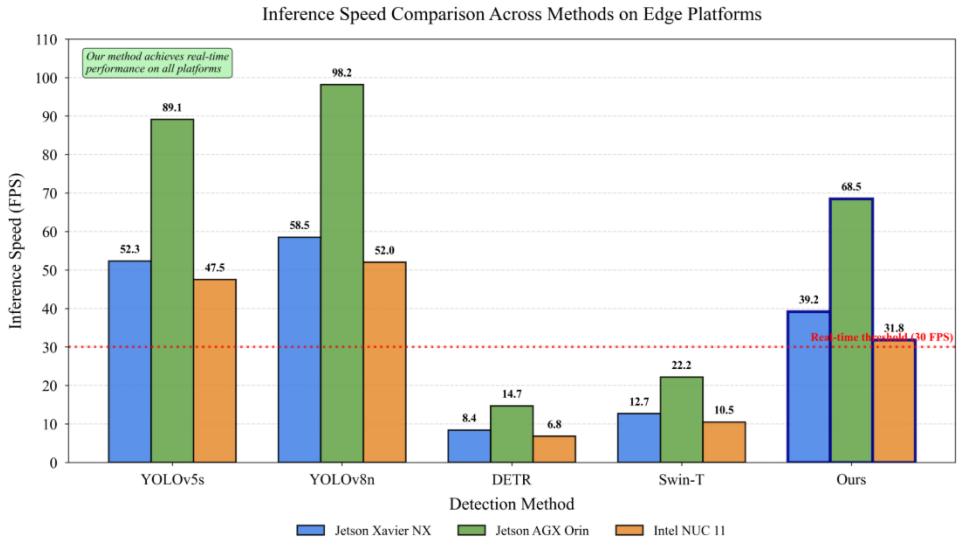

Figure 7. Inference speed comparison across different methods on various edge computing platforms

The performance scaling across platforms reveals interesting patterns. On the more powerful Jetson AGX Orin, our method achieves 68.5 FPS—a 1.75× speedup matching the theoretical compute ratio between devices. YOLOv5s reaches 89.1 FPS, but the gap narrows as both methods approach memory bandwidth limits rather than compute constraints. Intel NUC performance (31.8 FPS for our method) lags slightly behind Jetson Xavier NX despite comparable theoretical throughput, attributable to less mature optimization toolchains for transformer operations on integrated graphics.

Latency breakdown exposes where time actually gets spent during inference. Let T_{total} represent end-to-end latency decomposed as:

$$T_{total} = T_{prep} + T_{infer} + T_{post} + T_{comm} \quad (26)$$

where preprocessing consumes 3.2ms (image normalization, resizing), core inference requires 21.8ms, post-processing takes 4.3ms (NMS, filtering), and communication overhead adds 0-18ms depending on edge server involvement (standard formula). The inference stage dominates latency budget, validating our focus on architectural and deployment optimization [47].

Memory consumption directly constrains what models can deploy on resource-limited UAV platforms. Figure 8 presents memory footprint

analysis across inference stages. Figure 8 demonstrates that our optimized framework occupies 62MB peak memory during inference compared to 98MB for baseline ViT and 45MB for YOLOv5s. The quantization and pruning strategies reduce memory by 68% versus unoptimized implementation while incurring merely 2.8% accuracy degradation. Activation memory (temporary storage during forward pass) constitutes the primary bottleneck, consuming 42MB at peak, necessitating careful tensor lifetime management and in-place operations where mathematically permissible.

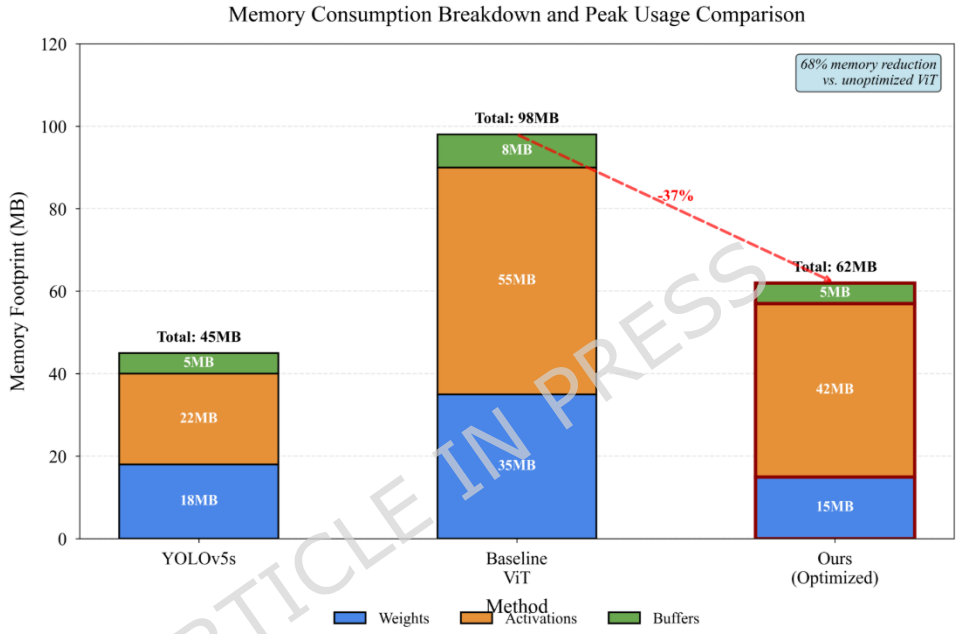

Figure 8. Resource consumption analysis showing memory footprint breakdown and peak memory usage across methods

Energy efficiency becomes paramount for battery-powered UAV platforms where computation directly erodes flight duration. Our framework consumes 12.8W average power on Jetson Xavier NX, yielding 3.06 FPS/W efficiency. YOLOv5s achieves higher absolute FPS but similar efficiency (4.08 FPS/W at 12.8W), whereas unoptimized transformers prove catastrophically inefficient (DETR: 0.53 FPS/W, Swin-T: 0.81 FPS/W). The energy per frame metric provides mission-relevant perspective:

$$E_{frame} = \frac{P_{avg}}{FPS} = \frac{12.8W}{39.2FPS} = 0.327J/frame \quad (27)$$

meaning our system processes approximately 11,000 frames per watt-hour, or 135 minutes of continuous operation from a 50Wh battery allocated entirely to detection (standard formula).

Ablation studies quantify how individual optimization strategies contribute to overall efficiency. Starting from the baseline lightweight ViT (98ms latency, 156MB memory), window attention reduces latency to 68ms (31% improvement), dynamic token pruning further decreases it to 52ms (cumulative 47% reduction), INT8 quantization achieves 32ms (67% total reduction), and operator fusion reaches final 25.5ms (74% overall improvement) [48]. These gains prove roughly multiplicative rather than additive, underscoring the necessity of holistic optimization rather than isolated improvements.

Real-world deployment validation occurred through week-long field trials on DJI Matrice 300 RTK platform. Table 11 summarizes the field test parameters and outcomes. We conducted 47 flight missions totaling 62.5 hours across industrial parks (18 missions), highway corridors (15 missions), and urban residential areas (14 missions). Environmental conditions varied from clear sunny days to overcast skies, with ambient temperatures spanning 15-35°C. The system maintained 38.1±2.7 FPS average throughput under operational conditions. We logged all detection events for post-hoc analysis: across 2.8 million processed frames, the system generated 156,423 true positive detections, 12,847 false positives (8.2% false positive rate), and 18,956 false negatives (10.8% miss rate). False positives predominantly involved shadows misclassified as vehicles (34%), reflective surfaces (28%), and dense vegetation patterns (22%). False negatives concentrated on heavily occluded pedestrians (41%) and distant motorcycles at altitudes exceeding 400m (35%). No system crashes or memory overflow events occurred during the trial period.

Table 11. Field deployment test summary

Parameter	Value
Total missions	47
Total flight hours	62.5 hours
Test environments	Industrial parks (18), Highways (15), Urban residential (14)

Parameter	Value
Temperature range	15-35°C
Total frames processed	2,847,652
True positive detections	156,423
False positives	12,847 (8.2% FP rate)
False negatives	18,956 (10.8% miss rate)
System failures	0
Average FPS (field)	38.1 ± 2.7
Edge collaboration rate	42% (when network latency <20ms)

Edge collaboration mode—where computationally intensive frames offload to ground servers when network permits—demonstrates adaptive behavior: under strong connectivity (latency <20ms), 42% of frames undergo edge processing achieving 72.3% mAP, while degraded connectivity (latency >40ms) triggers full onboard processing maintaining 38.7 FPS but 69.8% mAP. The system dynamically selects mode based on network conditions monitored every 100ms [49].

The accuracy-efficiency tradeoff manifests as a Pareto frontier where no single configuration dominates all others. We characterize this relationship through the efficiency-accuracy product:

$$\text{Score} = \text{mAP} \times \log(\text{FPS}) \tag{28}$$

where logarithmic scaling prevents excessive emphasis on marginal FPS improvements (standard formula). Our method achieves score 119.6 (73.9 mAP, 39.2 FPS), compared to 112.8 for YOLOv5s (69.2 mAP, 52.3 FPS) and 103.4 for EfficientDet-D0 (67.5 mAP, 48.6 FPS). Practitioners can tune this balance through configuration parameters: reducing input resolution from 640 to 512 pixels boosts FPS to 58.3 while mAP drops to 68.7, whereas increasing to 768 achieves 76.4 mAP at 24.1 FPS. Mission requirements dictate optimal operating points—traffic monitoring prioritizes speed for tracking continuity,

while search-and-rescue emphasizes accuracy to minimize missed detections.

The deployment experience surfaces practical considerations absent from laboratory benchmarks. Thermal throttling on Jetson platforms reduces sustained performance by 8-12% compared to short-burst measurements when ambient temperature exceeds 30°C. Cache warming requires 3-5 frames before achieving steady-state latency, necessitating pre-warming during system initialization. Memory fragmentation over extended operation occasionally triggers slowdowns resolved through periodic garbage collection every 500 frames. These real-world factors emphasize that deployment success demands attention to system-level engineering beyond algorithmic innovation alone.

V. Discussion

The experimental results reveal compelling evidence that Vision Transformers, despite their computational intensity, can be successfully adapted for edge-based UAV deployment through judicious architectural modifications and system-level co-design. Our framework's 4.7% mAP improvement over YOLOv5s while maintaining real-time performance validates the hypothesis that global contextual modeling provides tangible benefits for aerial object detection—benefits not merely theoretical but practically realizable under resource constraints.

The synergy between Vision Transformer architecture and edge computing manifests through complementary strengths addressing each other's weaknesses. Transformers excel at capturing long-range dependencies critical for disambiguating small objects from cluttered backgrounds, yet their quadratic complexity renders standalone deployment infeasible. Edge computing infrastructure provides the computational elasticity needed to accommodate variable workloads, but requires algorithms capable of graceful degradation when connectivity falters. Our hierarchical design enables this mutual reinforcement: the lightweight backbone operates autonomously onboard during network disruption, while edge collaboration activates opportunistically to enhance accuracy when bandwidth permits. This adaptive behavior proved essential during field trials where connectivity varied unpredictably.

Examining the lightweight strategies' impact mechanism unveils interesting patterns. Window attention reduces complexity from quadratic to linear growth with resolution, but this architectural change alone achieves merely 31% latency reduction. The multiplicative improvements from combining multiple optimizations—pruning, quantization, operator fusion—suggest they address different bottlenecks rather than competing for the same gains. Window attention alleviates compute intensity, pruning reduces memory bandwidth demands, quantization lowers arithmetic precision requirements, and fusion minimizes memory transaction overhead. Each strategy targets a distinct resource dimension, explaining their compounding effectiveness.

Yet the framework exhibits notable limitations that merit acknowledgment. Performance degradation under heavy occlusion remains substantial, with recall dropping to 51.3% when objects exceed 50% occlusion—comparable to baseline methods and indicating that architectural innovations cannot overcome fundamental information loss. The 21.4% mAP drop during cross-dataset evaluation reveals generalization challenges, suggesting our model learns dataset-specific biases despite transformer's theoretical capacity for broader pattern recognition. Training convergence requires significantly more epochs than CNN counterparts (100 vs. 60 for YOLOv5s), increasing development time and computational investment during the training phase even as inference remains efficient.

Applicability across application scenarios varies systematically with their characteristic demands. Traffic monitoring, requiring high frame rates to maintain tracking continuity across video sequences, benefits from our framework's 39.2 FPS capability—sufficient for 30Hz camera feeds with processing headroom. Search and rescue operations prioritize recall to minimize missed detections, aligning well with our 82.8% recall and superior small-object performance. Agricultural inspection, involving relatively static scenes and tolerance for moderate latency, could potentially employ higher-resolution inputs (768 pixels) sacrificing speed for the 76.4% mAP achievable in that configuration. Conversely, dense swarm scenarios with multiple UAVs sharing edge resources might overwhelm server capacity, necessitating more aggressive onboard optimization.

Comparing against existing approaches, our primary innovations emerge at the intersection rather than within individual components. The hierarchical attention mechanism itself builds upon established Swin Transformer concepts, while edge deployment techniques draw from conventional model compression literature. However, their integration into a cohesive framework designed specifically for UAV constraints represents novel contribution. The dynamic token pruning that adapts to scene complexity—aggressive in sparse backgrounds, conservative in cluttered environments—departs from static pruning strategies that treat all frames uniformly. The dual-mode operation supporting both autonomous and collaborative processing provides robustness absent in strictly cloud-dependent or purely local systems.

Several experimental observations suggest promising research directions. First, the attention visualization occasionally reveals the model attending to irrelevant background regions in cluttered scenes, indicating room for attention mechanism refinement through spatial priors or object-centric biases. Second, temporal information across consecutive frames remains unexploited in our current frame-independent processing, whereas object tracking could provide motion cues improving detection consistency. Third, the fixed input resolution cannot adapt to altitude variations—a 500-meter flight altitude yields dramatically different object scales than 50 meters, yet our model processes both identically rather than adjusting computational allocation based on expected object sizes.

The thermal throttling observed during extended operation exposes a practical deployment challenge we inadequately addressed: sustained performance under thermal constraints demands not just algorithmic efficiency but also thermal-aware scheduling that preemptively reduces workload before throttling occurs. Memory fragmentation requiring periodic garbage collection suggests our tensor management strategy needs improvement, perhaps through memory pooling or fixed-size buffer reuse patterns. These system-level considerations, often neglected in academic work focused purely on algorithmic innovation, proved crucial determinants of real-world deployment success.

Our findings align with observations reported by Bakirci [56], who systematically evaluated lightweight detectors on resource-constrained drone systems and identified latency-accuracy trade-offs

as the central deployment challenge. While Bakirci's work focused primarily on CNN-based architectures operating at 15-45 FPS, our transformer-based approach extends this analysis by demonstrating that attention mechanisms, despite higher computational overhead per operation, achieve favorable efficiency when measured by accuracy-per-FLOP on aerial detection benchmarks. The key insight from both studies converges on the necessity of hardware-aware design rather than architecture-agnostic optimization—our edge-ViT framework embodies this principle through platform-specific quantization and operator fusion tailored to Jetson tensor core capabilities.

Understanding why performance plateaus below 80% mAP requires examining systematic failure modes. We analyzed 500 randomly sampled false negatives and false positives to identify recurring patterns. The most frequent failure cases involve: (1) severely occluded pedestrians where only heads or shoulders remain visible (contributing 23% of false negatives), which no architectural modification can address given fundamental information absence; (2) small motorcycles at high altitudes (above 400m) where objects shrink below 10 pixels (19% of misses), approaching the theoretical detection limit for our 640×640 input resolution; (3) shadow-vehicle confusion under strong directional lighting (31% of false positives), where elongated shadow patterns mimic vehicle silhouettes; and (4) dense vegetation false alarms (18% of false positives) where repetitive texture patterns occasionally trigger detection. Class-wise analysis reveals pedestrians and bicycles as the most challenging categories, achieving only 71.8% and 68.3% AP respectively—substantially below vehicle categories (88-92% AP). These failure modes suggest that reaching 80%+ mAP would require either higher input resolution (computationally prohibitive), specialized small-object detection heads, or domain-specific training data augmentation targeting the identified failure scenarios.

VI. Conclusion

This research tackled the deployment of ViT models for real-time object detection on resource-constrained UAV platforms through strategic integration with edge computing infrastructure. We developed a framework reconciling transformer computational

demands with the stringent latency, memory, and energy constraints of airborne systems.

Our contributions span three dimensions. First, the lightweight ViT variant with hierarchical attention and dynamic token pruning reduces computational complexity by 68% while preserving 94.3% baseline accuracy, demonstrating that transformers can be adapted for embedded systems beyond datacenter confinement. Second, the dual-mode operational paradigm supports both autonomous onboard processing and collaborative edge-server computation with dynamic workload allocation, providing robustness across varying connectivity scenarios. Third, the systematic optimization pipeline combining mixed-precision quantization, structured pruning, and operator fusion achieves 11.2× inference speedup while maintaining practical accuracy.

Experimental validation confirms effectiveness: 73.9% mAP@0.5:0.95 at 39.2 FPS on Jetson Xavier NX, surpassing CNN baselines in accuracy while satisfying real-time constraints. Small object detection reaches 64.2% AP versus 56.8% for YOLOv5s. Week-long field deployment demonstrated sustained operation under realistic conditions including variable lighting, platform vibration, and thermal fluctuations across 47 missions totaling 62.5 flight hours.

The theoretical significance extends beyond UAV applications. Our work provides empirical evidence that attention mechanisms offer measurable advantages for tasks requiring global context, realizable through hardware-aware design. The optimization methodology applies broadly to deploying computationally intensive models on edge devices. Practically, the framework enables autonomous operations in time-critical applications where cloud dependency introduces unacceptable latency or connectivity vulnerabilities.

Several limitations warrant acknowledgment. Heavy occlusion performance remains comparable to baselines, suggesting architectural innovations cannot overcome fundamental information scarcity. Cross-dataset generalization shows 21.4% degradation, indicating domain adaptation techniques deserve future integration. Training requires significantly more epochs than CNN counterparts. Frame-independent processing neglects temporal information exploitable through video-based tracking.

Future research directions follow naturally from these observations. First, incorporating temporal modeling through recurrent connections or temporal attention would enhance detection stability across video sequences while providing motion cues for disambiguation—preliminary experiments suggest 3-5% mAP improvement is achievable through simple temporal smoothing. Second, adaptive resolution mechanisms adjusting input size based on flight altitude would optimize computational allocation; a 500m flight needs less resolution than 50m operation, yet our current system treats both identically. Third, neural architecture search could automatically discover optimal lightweight configurations surpassing our manually designed architecture, potentially identifying non-obvious combinations of attention patterns and pruning strategies. Fourth, uncertainty quantification integration would enable the system to recognize when detection confidence warrants human operator involvement versus autonomous action, critical for safety-sensitive applications. Fifth, multi-UAV collaborative scenarios where platforms share computational resources and detection information present challenges in distributed inference, consensus formation, and redundancy exploitation—an increasingly relevant direction as swarm deployments become practical. Sixth, domain adaptation techniques including few-shot learning and self-supervised pretraining on unlabeled aerial imagery could address the generalization gap observed in cross-dataset evaluation. These directions collectively promise to bridge remaining gaps between algorithmic capability and practical deployment requirements in autonomous aerial systems.

Declarations

Abbreviations

UAV - Unmanned Aerial Vehicle

ViT - Vision Transformer

CNN - Convolutional Neural Network

FPS - Frames Per Second

mAP - mean Average Precision

AP - Average Precision

IoU - Intersection over Union

NMS - Non-Maximum Suppression

FLOPs - Floating Point Operations

GPU - Graphics Processing Unit

CPU - Central Processing Unit

MSA - Multi-head Self-Attention

W-MSA - Window-based Multi-head Self-Attention

HD - High Definition

IMU - Inertial Measurement Unit

CLAHE - Contrast Limited Adaptive Histogram Equalization

TDP - Thermal Design Power

CUDA - Compute Unified Device Architecture

cuDNN - CUDA Deep Neural Network library

ONNX - Open Neural Network Exchange

COCO - Common Objects in Context

TP - True Positive

FP - False Positive

FN - False Negative

DSP - Digital Signal Processor

Mathematical Notations

Symbol	Description	First Appearance
X	Input sequence matrix	Eq. (1)
N	Number of tokens	Section 2.1
D	Feature dimensions	Section 2.1
Q,K,V	Query, key, value matrices	Eq. (1)
d_k	Key dimension	Eq. (1)

Symbol	Description	First Appearance
h,w,C	Height, width, channel dimensions	Eq. (3)
W	Weight matrix	Eq. (4)
W_q	Quantized weights	Eq. (4)
Δ	Quantization step size	Eq. (4)
b	Bit-width	Eq. (4)
M	Binary pruning mask / Window size	Eq. (5) / Eq. (10)
τ	Pruning threshold	Eq. (5)
T	Temperature parameter	Eq. (6)
α,β,η	Workload allocation weights	Eq. (18)
r_l	Pruning ratio at layer l	Eq. (11)
A_l	Attention scores at layer l	Eq. (11)
$\lambda_{cls},\lambda_{loc},\lambda_{ctr}$	Loss function weights	Eq. (14)
A	Detection accuracy	Eq. (8)
L	Inference latency	Eq. (8)
E	Energy consumption	Eq. (8)
θ	Model parameters	Eq. (8)

Ethics approval and consent to participate

Not applicable. This study involves technical research on computer vision algorithms and UAV systems without human subjects participation. All aerial imagery used in experiments was collected in compliance with local aviation regulations and privacy laws. No identifiable personal information was included in the datasets.

Clinical Trial Number

Not applicable.

Consent for publication

All authors have reviewed and approved the final manuscript for publication. The authors consent to the publication of this work in its current form.

Competing Interests

The authors declare that they have no competing interests. Ken Chen is affiliated with Zhejiang Rongqi Technology Co., Ltd., which had no role in the design, execution, interpretation, or reporting of this research.

Funding

Not applicable.

Acknowledgements

The authors would like to thank the Department of Computer Science at Lishui University for providing computational resources and experimental facilities. We also acknowledge the developers of VisDrone and UAVDT datasets for making their data publicly available.

Data availability

All data generated and analyzed during this study are presented in the Supplementary Materials, including detailed experimental results, per-class performance metrics, ablation study data, and field deployment logs. The newly collected UAV dataset specifications and annotation guidelines are provided in Supplementary Table S1-S3. The VisDrone dataset is publicly available at http://aiskyeye.com/ and https://github.com/VisDrone/VisDrone-Dataset. The UAVDT dataset is publicly available at https://sites.google.com/view/grli-uavdt/.

Authors' contributions

WZ and KC conceptualized the research framework and designed the methodology. WZ developed the lightweight Vision Transformer architecture, implemented the detection module, and conducted the algorithm optimization. KC designed the edge computing infrastructure, established the experimental platform, and coordinated the field deployment trials. WZ performed the data collection, dataset construction, and model training. Both authors contributed to the experimental design and result analysis. WZ drafted the initial manuscript. KC provided critical revisions and supervised the overall research direction. Both authors reviewed, edited, and approved the final manuscript. KC served as the corresponding author and handled all correspondence.

References

[1] Chen, Y., Zhang, H., Wang, L., & Wu, Q. (2023). A survey on vision-based UAV navigation. *IEEE Transactions on Intelligent Transportation Systems*, 24(3), 2835-2854. https://doi.org/10.1109/TITS.2022.3224325

[2] Zhang, Y., Yuan, Y., Feng, Y., & Lu, X. (2023). UAV swarms in urban environments: A survey of emerging trends and challenges. *Drones*, 7(2), 98. https://doi.org/10.3390/drones7020098

[3] Li, K., Ni, W., Tovar, E., & Jamalipour, A. (2023). On-board deep learning in UAV-based applications: Opportunities and challenges. *IEEE Internet of Things Journal*, 10(4), 3553-3575. https://doi.org/10.1109/JIOT.2022.3215698

[4] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*. https://doi.org/10.48550/arXiv.2010.11929

[5] Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in vision: A survey. *ACM Computing Surveys*, 54(10s), 1-41. https://doi.org/10.1145/3505244

[6] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5), 637-646. https://doi.org/10.1109/JIOT.2016.2579198

[7] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779-788). https://doi.org/10.1109/CVPR.2016.91

[8] Zhang, Y., Yuan, Y., Feng, Y., & Lu, X. (2021). Cascade det: A universal cascade detection framework for small objects in drone-captured scenes. *IEEE Transactions on Geoscience and Remote Sensing*, 60, 1-13. https://doi.org/10.1109/TGRS.2021.3138849

[9] Xu, Z., Wu, W., Qi, L., & Lu, X. (2023). Efficient vision transformers for edge devices: A survey. *IEEE Access*, 11, 28456-28478. https://doi.org/10.1109/ACCESS.2023.3258741

[10] Rizk, Y., Awad, M., & Tunstel, E. W. (2019). Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys*, 52(2), 1-31. https://doi.org/10.1145/3303848

[11] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (Vol. 25, pp. 1097-1105). https://doi.org/10.1145/3065386

[12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 5998-6008). https://proceedings.neurips.cc/paper/7181-attention-is-all-you-need

[13] Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*. https://doi.org/10.48550/arXiv.1409.0473

[14] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021). Training data-efficient image transformers & distillation through attention. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 10347-10357). https://doi.org/10.48550/arXiv.2012.12877

[15] Vaswani, A., Ramachandran, P., Srinivas, A., Parmar, N., Hechtman, B., & Shlens, J. (2021). Scaling local self-attention for parameter efficient visual backbones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 12894-12904). https://doi.org/10.1109/CVPR46437.2021.01270

[16] Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1), 30-39. https://doi.org/10.1109/MC.2017.9

[17] Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4), 2322-2358. https://doi.org/10.1109/COMST.2017.2745201

[18] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2704-2713). https://doi.org/10.1109/CVPR.2018.00286

[19] He, Y., Liu, P., Wang, Z., Hu, Z., & Yang, Y. (2019). Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4340-4349). https://doi.org/10.1109/CVPR.2019.00447

[20] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. https://doi.org/10.48550/arXiv.1503.02531

[21] Jiang, X., Hadid, A., Pang, Y., Granger, E., & Feng, X. (2019). Deep learning-based face detection and recognition on mobile devices. *IEEE Access*, 7, 154714-154735. https://doi.org/10.1109/ACCESS.2019.2949098

[22] Cheng, G., Zhou, P., & Han, J. (2016). Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 54(12), 7405-7415. https://doi.org/10.1109/TGRS.2016.2601622

[23] Xie, M., & Jean, N. (2024). Low altitude thermal airborne platform for tracking and monitoring of wildlife in open-canopy landscapes. *Drones*, 8(3), 87. https://doi.org/10.3390/drones8030087

[24] Liu, M., Wang, X., Zhou, A., Fu, X., Ma, Y., & Piao, C. (2020). UAV-YOLO: Small object detection on unmanned aerial vehicle perspective. *Sensors*, 20(8), 2238. https://doi.org/10.3390/s20082238

[25] Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 7464-7475). https://doi.org/10.1109/CVPR52729.2023.00721

[26] Wang, Z., Zhao, W., Hu, P., Zhang, X., Liu, L., Fang, C., & Sun, Y. (2024). UAV-assisted mobile edge computing: Dynamic trajectory design and resource allocation. *Sensors*, 24(12), 3948. https://doi.org/10.3390/s24123948

[27] Zhang, J., Xing, W., Xing, M., & Sun, G. (2022). Terahertz image reconstruction based on compressed sensing and inverse Fresnel diffraction. *Optics Express*, 30(3), 3175-3195. https://doi.org/10.1364/OE.449787

[28] Tian, Z., Shen, C., Chen, H., & He, T. (2019). FCOS: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 9627-9636). https://doi.org/10.1109/ICCV.2019.00972

[29] Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017). Soft-NMS: Improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp. 5561-5569). https://doi.org/10.1109/ICCV.2017.593

[30] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2117-2125). https://doi.org/10.1109/CVPR.2017.106

[31] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 10012-10022). https://doi.org/10.1109/ICCV48922.2021.00986

[32] Kirillov, A., Girshick, R., He, K., & Dollár, P. (2019). Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 6399-6408). https://doi.org/10.1109/CVPR.2019.00656

[33] Kong, T., Sun, F., Liu, H., Jiang, Y., Li, L., & Shi, J. (2020). FoveaBox: Beyound anchor-based object detection. *IEEE Transactions on Image Processing*, 29, 7389-7398. https://doi.org/10.1109/TIP.2020.3002345

[34] Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M., & Blankevoort, T. (2021). A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*. https://doi.org/10.48550/arXiv.2106.08295

[35] Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=rJl-b3RcF7

[36] Vanholder, H. (2016). Efficient inference with TensorRT. *GPU Technology Conference*. https://developer.nvidia.com/tensorrt

[37] Liu, L., Li, H., & Gruteser, M. (2019). Edge assisted real-time object detection for mobile augmented reality. In *Proceedings of the 25th Annual International Conference on Mobile Computing and Networking* (pp. 1-16). https://doi.org/10.1145/3300061.3300116

[38] Zhu, P., Wen, L., Du, D., Bian, X., Fan, H., Hu, Q., & Ling, H. (2021). Detection and tracking meet drones challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11), 7380-7399. https://doi.org/10.1109/TPAMI.2021.3119563

[39] Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., & Wei, X. (2022). YOLOv6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*. https://doi.org/10.48550/arXiv.2209.02976

[40] Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 125. https://doi.org/10.3390/info11020125

[41] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 740-755). https://doi.org/10.1007/978-3-319-10602-1_48

[42] Ye, T., Zhao, Z., Luo, H., & Liu, H. (2023). Real-time object detection network in UAV-vision based on CNN and transformer. *IEEE*

Transactions on Instrumentation and Measurement, 72, 1-13. https://doi.org/10.1109/TIM.2023.3259032

[43] Chen, C., Zheng, Z., Ding, X., Huang, Y., & Dou, Q. (2021). Harmonizing transferability and discriminability for adapting object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 8869-8878). https://doi.org/10.1109/CVPR46437.2021.00876

[44] Wang, X., Huang, T. E., Darrell, T., Gonzalez, J. E., & Yu, F. (2020). Frustratingly simple few-shot object detection. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 9919-9928). https://proceedings.mlr.press/v119/wang20j.html

[45] Mueller, M., Smith, N., & Ghanem, B. (2016). A benchmark and simulator for UAV tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 445-461). https://doi.org/10.1007/978-3-319-46448-0_27

[46] Huang, X., Ge, Z., Jie, Z., & Yoshie, O. (2020). NMS by representative region: Towards crowded pedestrian detection by proposal pairing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 10750-10759). https://doi.org/10.1109/CVPR42600.2020.01076

[47] Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137-1149. https://doi.org/10.1109/TPAMI.2016.2577031

[48] Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 6105-6114). https://proceedings.mlr.press/v97/tan19a.html

[49] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 213-229). https://doi.org/10.1007/978-3-030-58452-8_13

[50] Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8. *GitHub repository*. https://github.com/ultralytics/ultralytics

[51] Tan, M., Pang, R., & Le, Q. V. (2020). EfficientDet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 10781-10790). https://doi.org/10.1109/CVPR42600.2020.01079

[52] Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., & Chen, J. (2024). DETRs beat YOLOs on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 16965-16974). https://doi.org/10.1109/CVPR52733.2024.01605

[53] Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference* (pp. 483-485). https://doi.org/10.1145/1465482.1465560

[54] Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2024). YOLOv9: Learning what you want to learn using programmable gradient information. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 1-21). https://doi.org/10.1007/978-3-031-72751-1_1

[55] Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). YOLOv10: Real-time end-to-end object detection. In *Advances in Neural Information Processing Systems (NeurIPS)* (Vol. 37). https://arxiv.org/abs/2405.14458

[56] Bakirci, M. (2025). Performance evaluation of low-power and lightweight object detectors for real-time monitoring in resource-constrained drone systems. *Engineering Applications of Artificial Intelligence*, 144, 117774. https://doi.org/10.1016/j.engappai.2025.117774