DUST-YOLO: An Efficient and Lightweight Deployable UAV Swin Transformer YOLO for End-to-End Video Object Detection

Gongxun Lin , Jincheng Jiang , Jiaheng Cai , Xingjian Luo , Zihao Wang , Hao Sun * , Ziyuan Pu *

Posted Date: 21 April 2026

doi: 10.20944/preprints202604.1422.v1

Keywords: UAV; object detection; lightweight optimization; edge computing; Jetson

 doi:10.20944/preprints202604.1422.v1

Article

DUST-YOLO: An Efficient and Lightweight Deployable UAV Swin Transformer YOLO for End-to-End Video Object Detection

Gongxun Lin [1,†], Jincheng Jiang [1,†], Jiaheng Cai [1], Xingjian Luo [2], Zihao Wang [1], Hao Sun [1,*] and Ziyuan Pu [2,*]

[1] School of Electronic Science and Engineering, Southeast University, Nanjing 211189, China
[2] School of Transportation, Southeast University, Nanjing 211189, China
* Correspondence: hao.sun@seu.edu.cn (H.S.); ziyuanpu@seu.edu.cn (Z.P.)
† These authors contributed equally to this work.

Abstract

Real-time video object detection on unmanned aerial vehicles (UAVs) is essential for urban inspection and autonomous perception, yet its deployment on edge devices is severely constrained by the high computational cost of accurate detectors, the quantization sensitivity of hybrid convolution-attention networks, and the system-level latency of full video processing pipelines. To address these challenges, we present DUST-YOLO, a deployment-oriented algorithm-hardware co-design framework for lightweight and efficient UAV small-object detection on edge platforms. First, we introduce a multi-dimensional structured pruning strategy that applies asymmetric channel pruning to convolutional and feature-fusion modules while compressing the Swin Transformer prediction heads and bottleneck stacks, thereby reducing parameters and computation with limited impact on multi-scale representation capability. Second, we develop a hardware-aware mixed-precision quantization-aware training (QAT) scheme that maps computation-intensive backbone layers to INT8 while preserving the Transformer-related modules in FP16, improving inference efficiency while mitigating the accuracy loss caused by uniform low-bit quantization. Third, we compile the optimized network with TensorRT and integrate the resulting inference engine into a DeepStream-based asynchronous video pipeline on the edge platform, enabling end-to-end acceleration by reducing decoding, preprocessing, and memory-transfer overheads. Experimental results on the VisDrone2019-DET dataset and the NVIDIA Jetson Orin NX demonstrate that DUST-YOLO achieves 43.7% mAP@0.5 acuracy with an end-to-end latency of 36.3 ms and a throughput of 27.5 FPS. Compared with the state-of-the-art detector, DUST-YOLO reduces end-to-end latency by 56.9% and improves end-to-end video throughput by $\times$2.31, while lowering total energy consumption by 68.5%.

Keywords: UAV; object detection; lightweight optimization; edge computing; Jetson

1. Introduction

Real-time object detection on unmanned aerial vehicles (UAVs) has become a key enabling technology for low-altitude intelligent applications such as urban inspection, traffic monitoring, security patrol, emergency response, and remote sensing [1–6]. In these scenarios, UAVs continuously capture video streams and must perform on-board object recognition and localization under strict constraints on latency, power consumption, payload, and computing resources. Compared with ground-view perception, UAV-based detection is particularly challenging because targets are often small, sparsely distributed, and embedded in cluttered backgrounds with significant scale variation [7–10]. Therefore, achieving accurate and real-time video object detection on resource-constrained edge devices is of great practical importance for UAV autonomous perception systems.

 doi:10.20944/preprints202604.1422.v1

To improve aerial small-object detection accuracy, recent detectors have increasingly adopted multi-scale representation and attention-based mechanisms. Representative examples include TPH-YOLOv5 [11,12], which introduces a Transformer Prediction Head (TPH) [13] to enhance global context modeling and improve the representation of small targets. However, although such architectures offer higher detection accuracy, their efficient deployment on embedded edge platforms remains difficult. In particular, three challenges must be addressed. First, Transformer-enhanced detection models introduce substantial parameter redundancy and computational overhead, making them difficult to deploy within the limited compute and memory budgets of UAV edge devices [14]. Second, hybrid networks composed of convolutional and self-attention modules are highly sensitive to low-bit quantization, and conventional uniform quantization often causes severe accuracy degradation and poor operator fusion efficiency [15,21,22]. Third, even when the neural network itself is accelerated, the end-to-end video detection pipeline is still bottlenecked by video decoding, image preprocessing, memory transfer, and heterogeneous task scheduling, which significantly limits practical real-time performance [23].

Significant research effort has been dedicated to these challenges. In terms of architectural optimization, works such as EL-YOLO [16] and YOLO-UD [17] have proposed asymptotic feature pyramid networks (SCAFPN) [18] and dilated convolutions to capture richer contextual information for small targets [19]. However, these approaches inevitably increase parameters, FLOPs, and memory access overhead [20]. Regarding data quantization, conventional full-network uniform quantization applied to heterogeneous networks containing self-attention mechanisms [21,22] often leads to severe accuracy degradation and the failure of operator fusion. Regarding system deployment, traditional detection pipelines suffer from CPU-executed video decoding, image preprocessing, and memory copy overheads, which constitute major system I/O bottlenecks and hinder real-time end-to-end performance [23]. As a result, balancing small object detection accuracy and efficiency on edge devices remains a challenging open problem.

Significant efforts have been devoted to alleviating these issues. In terms of model architecture, methods such as EL-YOLO [16] and YOLO-UD [17] improve small-object detection by introducing stronger feature fusion, asymptotic feature pyramid structures, or dilated convolutions [18,19]. While these designs enhance contextual modeling, they typically increase parameter counts, floating-point operations, and memory access overhead, which weakens their suitability for embedded deployment [20]. Regarding model compression and quantization, existing low-bit deployment methods are mostly designed for homogeneous convolutional networks and often rely on full-network uniform quantization. Such strategies are less effective for detectors containing attention modules, because the different numerical sensitivities of convolutional and Transformer blocks frequently lead to unacceptable accuracy loss or limited acceleration benefits after deployment [21,22]. At the system level, traditional inference pipelines usually process video decoding and image preprocessing on the CPU, resulting in considerable I/O overhead and pipeline stalls. Consequently, despite substantial progress in detection algorithms, jointly optimizing model complexity, numerical precision, and end-to-end streaming efficiency for UAV edge deployment remains an open challenge.

To overcome these challenges, we propose DUST-YOLO, a deployment-oriented lightweight UAV detection framework built upon YOLOv5l for accurate and efficient small-object detection on edge platforms. The central idea of DUST-YOLO is to jointly optimize the detector at three tightly coupled levels: network structure, quantization strategy, and deployment pipeline. By integrating structured pruning, hardware-aware mixed-precision quantization-aware training, and DeepStream-based end-to-end acceleration on the NVIDIA Jetson Orin NX, the proposed framework achieves a favorable balance among detection accuracy, inference latency, and energy efficiency for real-world UAV video applications. Fig. 1 illustrates the overall end-to-end deployment pipeline on the Jetson edge platform. The main contributions of this paper are summarized as follows:

- **A multi-dimensional structured pruning strategy for lightweight UAV detection:** We design an asymmetric channel pruning scheme for deep convolutional layers and feature-fusion modules to

 doi:10.20944/preprints202604.1422.v1

remove redundant channels while preserving the multi-scale feature extraction capability required for aerial small-object detection. In addition, we compress the Swin Transformer prediction heads and reduce the number of bottleneck stacks, substantially decreasing model parameters and computational complexity with limited accuracy loss.

- **A hardware-aware mixed-precision QAT framework for hybrid CNN–Transformer detectors:** We introduce quantization-aware fine-tuning by dynamically inserting fake-quantization nodes during training, enabling the model to adapt to quantization noise before deployment. Furthermore, we map computation-intensive backbone layers to INT8 while retaining Transformer-related modules in FP16, thereby improving inference efficiency on edge hardware while maintaining operator fusion compatibility and detection accuracy.
- **An efficient DeepStream-integrated end-to-end deployment system on Jetson Orin NX:** We perform TensorRT explicit quantization compilation to precompute weights, fuse operators across layers, and exploit high-throughput INT8 kernels for convolutional computation. The resulting engine is integrated into an asynchronous DeepStream video pipeline that leverages hardware units such as NVDEC and VIC to reduce decoding, preprocessing, and memory-transfer overheads, significantly improving end-to-end throughput and power efficiency.

Figure 1. The end-to-end video object detection pipeline on the Jetson edge platform.

The remainder of this paper is organized as follows. Section II outlines the background and related work. Section III provides an overview of the proposed system architecture and its submodules. Section IV presents the detailed methodology, including the structured pruning strategy, the mixed-precision QAT, and the DeepStream-based embedded multi-object detection system deployment. Section V discusses the experimental results, ablation studies, and power consumption analysis. Finally, Section VI concludes the paper.

2. Related Work

This section is organized into three main subsections. Section 2.1 discusses lightweight optimization and model pruning techniques. Section 2.2 examines current data quantization strategies. Finally, Section 2.3 reviews edge deployment and system acceleration frameworks.

2.1. Lightweight Optimization and Model Pruning

Modern research has focused on reducing the redundancy of object detection models for edge deployment. Architectures like EL-YOLO and YOLO-UD have introduced feature pyramid optimizations and contextual information aggregation [24] to balance efficiency and accuracy. General lightweight models such as YOLO-LIGHT [25] and LSOD-YOLO emphasize limited edge computing scenarios [26]. To further categorize these efforts, lightweighting techniques generally bifurcate into unstructured and structured pruning. While unstructured pruning achieves high sparsity and significantly reduces theoretical parameter counts, it often relies on specialized sparse matrix multiplication libraries. This limits its practical deployment efficiency on standard embedded GPUs. Structured pruning, on the other hand, directly trims the physical dimensions of convolution kernels and feature maps, yielding immediate latency reductions.

However, a significant limitation of existing pruning techniques is that they often apply uniform compression across all network layers, which can disproportionately weaken the fine-grained responses

 doi:10.20944/preprints202604.1422.v1

of small objects. Furthermore, many structured pruning methods fail to consider hardware-level alignment, such as modulo-32 constraints, leading to memory fragmentation and sub-optimal CUDA parallel execution on platforms like the NVIDIA Jetson. Additionally, with the recent integration of Vision Transformers (ViTs) into aerial detection architectures, traditional CNN-centric pruning metrics struggle to accurately evaluate the redundancy within multi-head self-attention (MHA) modules. These unresolved gaps highlight the urgent need for a more comprehensive compression method. This directly motivates our multi-dimensional structured pruning strategy, which jointly optimizes convolutional channels and Transformer bottleneck depths while maintaining strict hardware modulo alignment to achieve actual on-device acceleration.

2.2. Quantization Strategies and Transformer Sensitivity

Quantization is a standard technique to reduce storage and memory access overhead by converting floating-point weights to integers. While Post-Training Quantization (PTQ) is widely used due to its simplicity, it frequently suffers from accuracy degradation in complex scenarios. QAT has emerged as a more robust alternative, allowing the model to adapt to quantization noise during fine-tuning. In recent years, mixed-precision quantization has gained traction as a superior alternative to uniform INT8 quantization.By selectively maintaining highly sensitive layers in higher precision (e.g., FP16) while mapping computationally intensive convolutional backbones to the INT8 domain, mixed-precision schemes aim to achieve an optimal balance between execution latency and mean Average Precision (mAP).

Despite these advancements, quantizing models with self-attention mechanisms, such as EDGS-YOLOv8 [27], remains highly challenging. Transformer modules are highly sensitive to activation ranges, especially within Softmax and LayerNorm operations. Forcible INT8 truncation in these paths often leads to gradient distortion and disrupts native MHA fusion in the inference engine. Recent studies on quantized Transformers reveal that extreme distribution outliers in activation functions are the primary culprits for precision degradation. If Q/DQ(Quantize/Dequantize) nodes are inserted blindly without hardware-aware boundary definitions, the low-level compiler will be forced to fragment continuous computational flows into isolated small kernels, potentially increasing rather than decreasing latency. Therefore, existing hardware-agnostic quantization methods fail to achieve optimal deployment for hybrid architectures. This limitation necessitates our hardware-aware mixed-precision QAT approach, which strategically isolates highly sensitive Transformer blocks in FP16 while heavily quantizing convolutional backbones to INT8 to protect MHA fusion integrity.

2.3. Edge Deployment and Acceleration

System-level optimization is crucial for achieving real-time performance in airborne vision tasks. Beyond model-specific inference, the total latency includes video decoding, image preprocessing, and post-processing. A common pitfall in current UAV object detection research is the isolated evaluation of pure model inference time, which deeply neglects the severe systemic overhead introduced by the surrounding video pipeline. High-resolution drone streams require intensive operations such as scaling, color space conversion, and memory allocation.

Traditional deployment methods often rely on native inference frameworks that encounter I/O bottlenecks and high CPU-GPU communication overhead during memory copying. While standalone execution environments like ONNX Runtime or basic TensorRT APIs provide robust tensor mathematical acceleration, they typically still require the host CPU to manage stream buffering, format conversions, and layout transformations. While frameworks like NVIDIA DeepStream have been explored for real-time video analysis [28,29], deep integration between the model architecture and the streaming media pipeline is often overlooked. Without hardware-software co-design—such as heterogeneous computing decoupling and tensor batching—the full potential of the Jetson's unified memory architecture remains underutilized [30–32]. To break through these system-level bottlenecks, it is imperative to move beyond pure algorithm optimization. This drives our deep integration with the DeepStream framework, which seamlessly maps decoding and inference tasks to dedicated hetero-

 doi:10.20944/preprints202604.1422.v1

geneous IP blocks (such as NVDEC and VIC) to establish a zero-copy memory paradigm, thereby fully unleashing the real-time processing potential of airborne edge intelligence.

3. System Overview

This section is organized into two main subsections. Section 3.1 discusses the overall network architecture of DUST-YOLO and details how data flows through the network during traditional video object detection tasks. Section 3.2 presents the comprehensive system framework, analyzing the series of lightweight optimizations and edge-accelerated deployment strategies proposed for DUST-YOLO in this study.

3.1. Network Architecture of DUST-YOLO

For UAV edge deployment in scenarios such as urban inspection, detection algorithms require high precision to handle complex backgrounds and extreme scale variations. Accordingly, we propose DUST-YOLO, an improved model based on YOLOv5l. It introduces a dedicated extra-small (xsmall) detection head to form a four-scale architecture and replaces conventional C3 modules in the Neck with C3STR blocks integrating Swin Transformer window attention. These architectural enhancements significantly improve the model's sensitivity to small objects and its overall robustness in complex aerial environments.The specific network architecture and complete data flow of DUST-YOLO are illustrated in Figure 2.

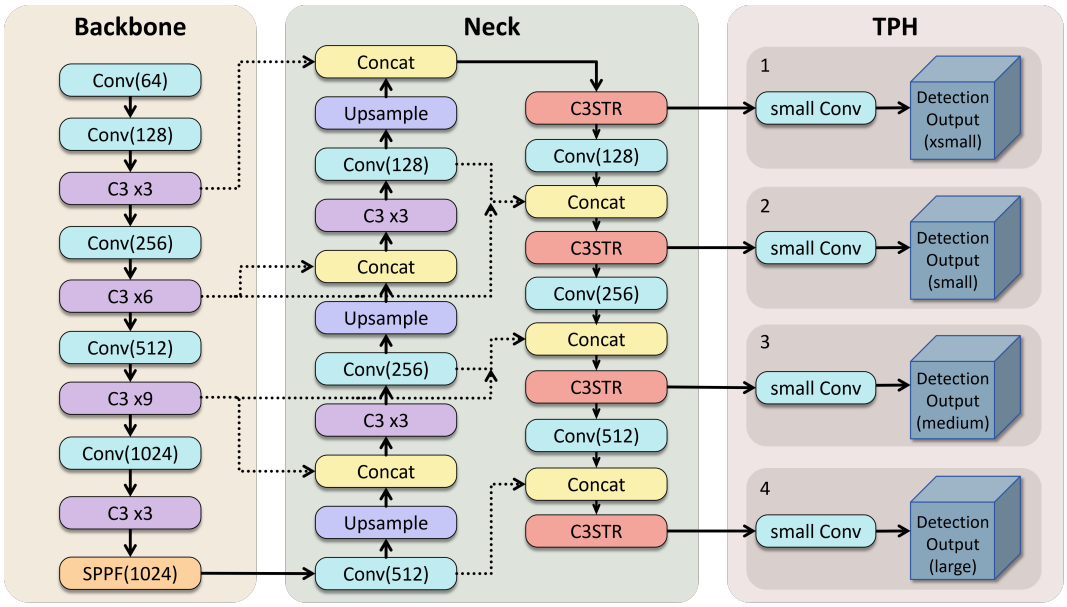

Figure 2. The structure diagram of DUST-YOLO.

During traditional UAV video object detection, the raw video stream is first decoded and pre-processed (e.g., scaled and normalized) into a standardized multi-dimensional RGB tensor matrix. This input tensor is then fed into the Backbone network, where it sequentially passes through multiple Convolutional (Conv) layers, C3 blocks, and a Spatial Pyramid Pooling - Fast (SPPF) module to hierarchically extract spatial and semantic features across varying receptive fields. Subsequently, these intermediate feature maps are routed into the Neck network. The Neck employs a bidirectional feature pyramid structure, utilizing Upsample operations and Concat layers to deeply fuse shallow fine-grained localization features with deep semantic features. Notably, the cascaded C3STR modules in this stage further refine the fused representations to prevent the loss of small object details. Finally, the enriched feature maps are directed into the TPH module. The four distinct detection branches (xsmall, small, medium, and large) apply small convolutional layers to generate dense prediction tensors containing bounding box coordinates, objectness scores, and class probabilities. Through post-processing operations, including confidence filtering and Non-Maximum Suppression (NMS),

 doi:10.20944/preprints202604.1422.v1

redundant boxes are eliminated, ultimately yielding the final precise target detection results for the UAV video stream.

3.2. System Module Design

The deep network structure of DUST-YOLO and the intensive matrix multiplication overhead introduced by the C3STR blocks result in extreme computational complexity, which significantly hinders its real-time performance on edge devices. Furthermore, traditional video detection pipelines face severe system I/O bottlenecks, primarily originating from CPU-executed video decoding, image preprocessing, and high CPU-GPU communication overhead caused by extensive memory copying. In the absence of effective hardware-software co-design, native inference flows encounter significant memory bandwidth limitations, resulting in suboptimal end-to-end latency that fails to meet the stringent task requirements of UAV inspections.

As illustrated in Figure 3, to address the aforementioned issues, this paper proposes an architectural design ranging from model lightweighting to edge-deployed video object detection acceleration:

- **The first module: Algorithmic Compression Layer.** This layer performs model lightweighting on the server side through multi-dimensional structured pruning and QAT. Pruning is customized for the network's heterogeneous modules: stripping redundant channels in convolutional layers and the feature pyramid, reducing the number of bottleneck stacks in deep C3 blocks, and compressing the dimensional scale of the fully connected layer matrices in C3STR blocks. After recovering accuracy through fine-tuning, the model undergoes QAT, which enables the network to learn loss and noise characteristics under INT8 precision by selectively inserting fake quantization nodes, thereby preparing a lightweight model for edge-side compilation.
- **The second module: Hardware Compilation Layer.** This module utilizes TensorRT to compile the optimized model from the ONNX intermediate format into a hardware-specific inference engine at the edge. During this compilation process, INT8+FP16 quantization is executed, and most fragmented small operators are fused into larger kernels. This strategy effectively reduces memory access overhead and significantly enhances the execution speed of the object detection algorithm on edge computing platforms.
- **The third module: System Deployment Layer.** This module focuses on the deployment of the model within an end-to-end video object detection pipeline based on the DeepStream framework. It utilizes DeepStream preprocessing modules, such as NVDEC, to preprocess the input video, then employs an inference plugin to accelerate the execution of the engine compiled and exported by the second module, and finally uses a post-processing plugin to efficiently output the processed video stream, thereby satisfying the real-time requirements of target detection.

 doi:10.20944/preprints202604.1422.v1

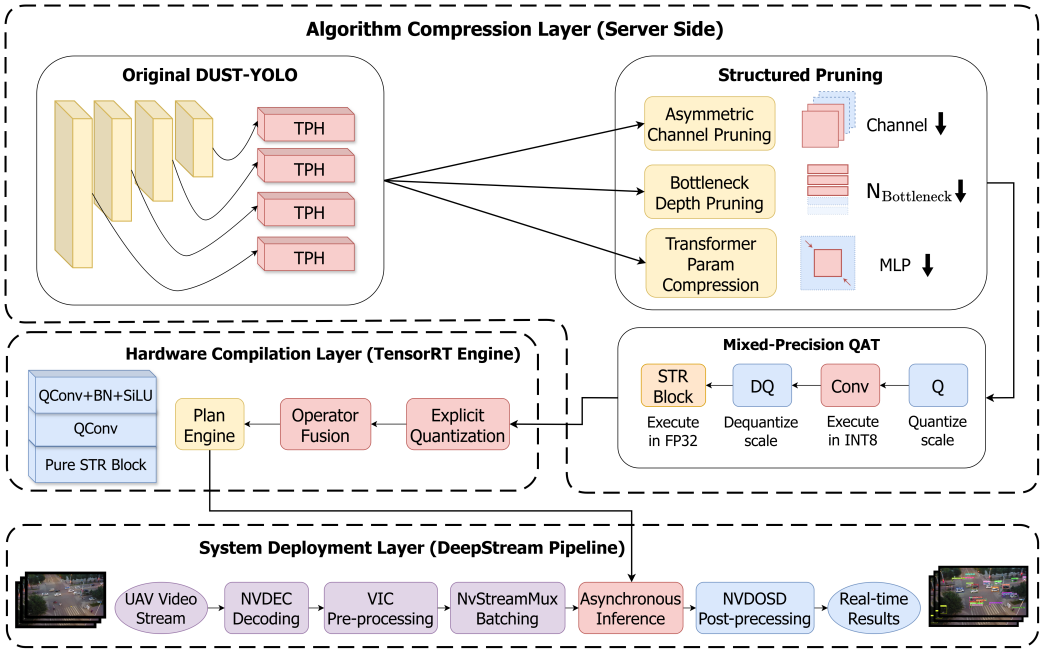

Figure 3. System overview.

4. Methodology

This section is organized into three main subsections to detail the proposed algorithm-hardware co-design framework. Section 4.1 discusses the multidimensional structured pruning strategy, explaining how feature channels, attention parameters, and network depth are jointly compressed to reduce algorithmic redundancy. Section 4.2 presents the hardware-aware mixed-precision QAT approach. Section 4.3 details the comprehensive system deployment framework based on NVIDIA DeepStream.

4.1. Structured Pruning and Model Compression Strategy

To deploy the high-accuracy DUST-YOLO model on resource-constrained edge computing devices, the primary challenge lies in its large parameter size (45.46M) and extremely high computational complexity (62.79G FLOPs). To break through the computation bottleneck, this paper proposes a multidimensional structured feature compression strategy based on sensitivity analysis [6]. This strategy jointly considers three key dimensions—feature channels, attention mechanisms, and network depth—to perform deep reconstruction and redundancy pruning on the model. While preserving the model's multi-scale feature extraction capability to the greatest extent possible, it achieves a highly hardware-friendly lightweight network design.

4.1.1. Asymmetric Channel Pruning and Hardware Alignment for Convolutional Layers

In general, feature redundancy in the model exhibits clear hierarchical differences. Specifically, shallow layers are mainly responsible for extracting fundamental spatial features such as edges and textures, and aggressive pruning of these layers would lead to a significant drop in detection accuracy. Therefore, our method adopts an asymmetric pruning mechanism and places the optimization emphasis on deep convolutional layers and feature pyramid modules. In implementation, we employ a dependency graph algorithm and introduce the L1 norm as the criterion for evaluating the importance of convolution kernels. Suppose that the weight matrix of the c-th convolution kernel in the l-th layer is denoted by $W_c^{(l)} \in \mathbb{R}^{N\times K\times K}$. Its importance score is defined as the sum of the absolute values of its weights:

$$I_c = \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{K}\left|W_{c,n,i,j}^{(l)}\right| \tag{1}$$

 doi:10.20944/preprints202604.1422.v1

Accordingly, redundant channels whose scores fall below a dynamic threshold are removed, making it possible to distinguish the redundancy levels of convolutional layers with different depths and computational costs and thus determine the pruning value of each layer. Meanwhile, in order to maximize CUDA core parallel efficiency on the Jetson platform and avoid memory-access latency caused by memory fragmentation, the objective of structured pruning is modeled as the following constrained optimization problem:

$$\min_{W} \mathcal{L}(f(x;W),y) \quad \text{s.t.} \quad \sum \mathbb{I}(I_c > 0) \leq C_{target} \tag{2}$$

where C_{target} denotes the target number of channels after pruning and satisfies $C_{target} \equiv 0 \pmod{32}$. This modulo-32 alignment constraint strictly guarantees that the pruned feature maps can perfectly match the vectorized storage and computation specifications of the underlying hardware.

4.1.2. Pruning and Parameter Compression of Transformer Modules

The C3STR module introduced in DUST-YOLO significantly enhances the global receptive field, but the built-in Swin Transformer structure also introduces extremely dense matrix multiplication overhead. To address this issue, we perform in-depth simplification on both the Self-Attention and MLP submodules. Figure 4 illustrates the structure of the C3STR module.

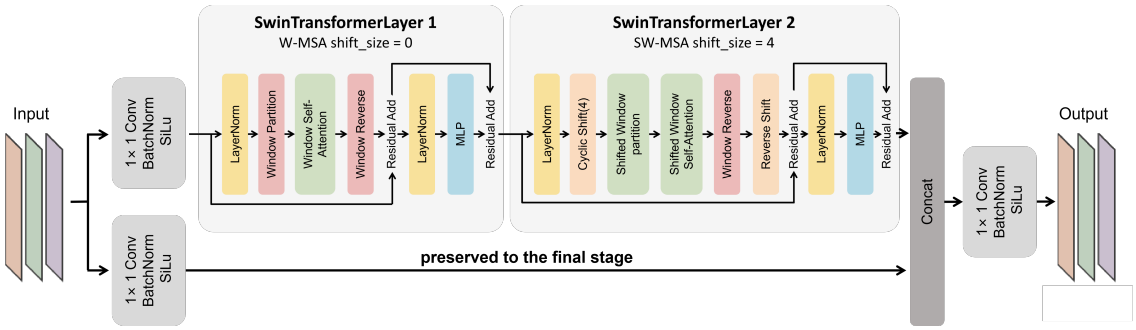

Figure 4. The structure diagram of C3STR.

In the architectural design of the Transformer block, the feature dimensions are subject to strict coupling relationships. Let D_{old} and D_{new} denote the embedding dimensions before and after pruning, respectively. Furthermore, let C_{skip} be the shortcut-connection dimension and γ be the expansion ratio of the feed-forward network (FFN). To guarantee the continuity of multi-branch feature concatenation and MHA computation, the internal concatenation dimension C_{concat}, hidden-layer dimension D_{MLP}, and number of attention heads H must satisfy the following constraints:

$$C_{concat} = D + C_{skip}, \quad D_{MLP} = \gamma \cdot D, \quad d = \frac{D_{new}}{H_{new}} = \frac{D_{old}}{H_{old}} = 32 \tag{3}$$

First, at the global optimization level, the expansion ratio γ in all scale detection heads is uniformly reduced from the constant 4 to 2, effectively eliminating the over-parameterized redundancy of nonlinear mappings in the FFN. Second, for the deep detection branches that dominate large-scale targets and possess wide receptive fields (the P4/P5 scales), this strategy significantly scales down the embedding dimension D by 75%; whereas for the shallow branches corresponding to tiny-object spatial features (the P2/P3 scales), either an unchanged or a mild 50% dimensionality reduction strategy is adopted to avoid accuracy loss caused by reduced resolution.

While reducing the number of parameters, this operation satisfies the constraints of the above equations, ensuring that the scaling factor $\sqrt{d}$ remains constant in the attention calculation:

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^{\top}}{\sqrt{d}}\right)V \tag{4}$$

 doi:10.20944/preprints202604.1422.v1

By maintaining this factor, we effectively prevent shifts in the attention distribution that might otherwise be caused by the pruning process.

In the Transformer's FFN, the standard computational process can be represented as:

$$FFN(x) = \text{Activation}(xW_1 + b_1)W_2 + b_2 \tag{5}$$

where $W_1 \in \mathbb{R}^{D \times \gamma D}$ and $W_2 \in \mathbb{R}^{\gamma D \times D}$. Consequently, the joint compression of γ and D results in the parameter count of each FFN being markedly reduced from $2(4D_{old}^2)$ to $2(2D_{new}^2)$, which significantly removes the redundancy within the fully connected layers.

4.1.3. Depth Simplification of Feature Cascades

Beyond the compression of channel width and attention dimensionality, we observed that the excessive stacking of bottleneck layers within deep C3 modules constitutes a major source of forward-propagation latency. While deep hierarchical structures are traditionally favored for capturing complex semantic information, they introduce significant computational overhead and memory access frequency on edge devices with limited bandwidth, such as the NVIDIA Jetson. In the specific context of UAV aerial imagery, where small objects dominate the scene, we found that over-abstracting features through extremely deep cascades often yields diminishing returns in detection accuracy and can even lead to the loss of fine-grained spatial cues necessary for precise localization.

To address this, we implement a depth-level pruning strategy that systematically reduces the number of bottleneck stacks in specific backbone feature-extraction stages from 9 to 6. Since this simplification is performed internally within the C3 blocks, it maintains the integrity of input and output channel dimensions, allowing the pruned modules to be seamlessly reintegrated into the DUST-YOLO architecture without disrupting the multi-scale fusion logic of the feature pyramid. By removing these redundant layers, we effectively shorten the physical path of feature transmission and reduce the cumulative I/O overhead. This structural optimization ensures that the network depth is strictly aligned with the hardware-level execution profile, further unleashing the real-time processing potential of the edge platform.

By jointly applying the above pruning strategies to channel width, attention parameters, and bottleneck depth, we successfully perform a hardware-friendly slimming and reconstruction of DUST-YOLO. The optimized lightweight architecture reduces the total number of parameters by approximately 48% and lowers the forward computational cost by roughly 26%. After multiple rounds of pruning and fine-tuning, the final pruned model fully recovers its detection capabilities, even showing a slight accuracy improvement over the pretrained baseline. This provides an efficient model foundation for the subsequent reduction of weight bit width through QAT and for TensorRT-based hardware-accelerated deployment.

4.2. Mixed-Precision QAT and Edge-Side Mixed-Precision Quantized Deployment

Although structured pruning substantially reduces the theoretical computational complexity of the object detection model, relying solely on floating-point computation (FP32/FP16) is still insufficient to fully unleash the potential of the underlying hardware Tensor Cores on edge devices with constrained computing power and memory bandwidth, such as the NVIDIA Jetson Orin NX. Quantizing the model to 8-bit integers (INT8) is a key approach to breaking through the memory-access bottleneck. However, in heterogeneous networks containing self-attention mechanisms, conventional full-network uniform quantization often leads to accuracy degradation and failure of operator fusion. To address this issue, this paper proposes a hardware-aware mixed-precision QAT training architecture, combined with TensorRT explicit quantization engine compilation on the edge side, thereby achieving efficient deployment from the perspective of algorithm–hardware co-design.

 doi:10.20944/preprints202604.1422.v1

4.2.1. QAT and Attention Fusion Protection

During the fine-tuning stage, QAT dynamically injects fake quantization nodes into the computational graph. This mechanism enables the network weights to adaptively absorb quantization noise. It requires training data and the insertion of fake quantization layers during the training stage, so that a model containing Q/DQ nodes can be obtained through training. During edge-side quantized deployment, quantization is then carried out according to these Q/DQ nodes. After the QAT process is completed, the fake quantization layers contain the quantization scales used to quantize the weights and activations for model inference, and therefore usually achieve higher accuracy than PTQ [5,9].

In this work, we adopt a symmetric uniform quantization scheme. For an arbitrary floating-point tensor X, its quantization scaling factor s, the INT8 quantized tensor X_q, and the dequantized approximation follow the mathematical mappings:

$$s = \frac{\max(|X|)}{127}, \quad X_q = \mathrm{Clip}\left(\mathrm{Round}\left(\frac{X}{s}\right), -127, 127\right), \quad X \approx X_q \cdot s \tag{6}$$

However, because the Transformer module involves intensive nonlinear normalization operations (such as Softmax and LayerNorm), it is highly sensitive to the dynamic range of activation values. In the self-attention mechanism, the attention computation (as previously defined in Equation (4)) is particularly vulnerable to discretized precision. If INT8 truncation is forcibly applied to the matrices within this mechanism, the discretized inner-product space will cause severe distortion of the gradients of the exponential terms. Moreover, in the TensorRT inference engine on the Jetson platform, the bottleneck of the MHA subgraph lies in memory scheduling rather than arithmetic throughput. If Q/DQ nodes are inserted into these highly sensitive paths, the low-level compiler will be forced to interrupt the native MHA fusion strategy, fragmenting the continuous computational flow into a large number of isolated small kernels. This significantly increases the frequency of memory read/write operations, causing the latency after quantization to increase rather than decrease.

Based on this underlying physical mechanism, this paper implements a mixed-precision QAT strategy. We explicitly define the quantization boundaries by locking the Transformer core subgraphs in the detection head to floating-point precision, while mapping only the computationally intensive backbone network and conventional convolutional layers into the INT8 domain. This not only significantly accelerates the convolution operators but also guarantees the complete fusion of the Transformer subgraphs. A schematic of the precision selection for different network modules and the insertion logic of fake quantization nodes is shown in Figure 5.

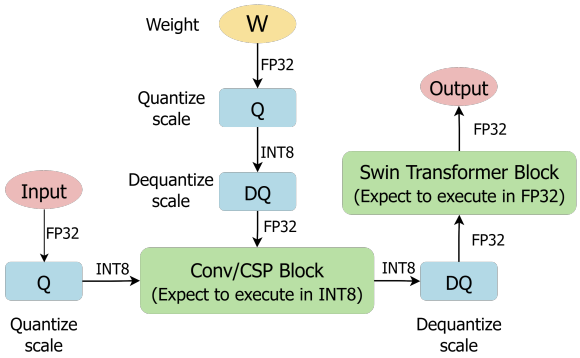

Figure 5. Schematic of network precision selection and fake quantization node insertion logic.

4.2.2. TensorRT Explicit Quantization Compilation and Continuous-Domain Optimization

After QAT training is completed and a computational graph containing explicit Q/DQ nodes is exported, extreme acceleration on the edge side relies on TensorRT's low-level explicit quantization compilation. Unlike conventional PTQ methods that rely on implicit calibration, the explicit compilation mechanism strictly follows the Q/DQ topological boundaries in the computational graph, fusing

 doi:10.20944/preprints202604.1422.v1

floating-point operators and reconstructing them within the INT8 domain. Layers for which Q/DQ nodes are not inserted during training are executed in FP16 precision by default.

During the engine construction stage, the compiler first performs constant folding and graph pruning, precomputing and absorbing nodes that do not participate in dynamic computation. For example, the Q node of a weight tensor is merged into the weight itself, ensuring that during actual inference, the FP32 weight no longer requires dynamic conversion, thereby reducing runtime VRAM access overhead. The compiler statically precomputes the convolution-layer weights together with the input/output scaling factors (s_w, s_x, s_y) and fuses the bias term b. The originally complex floating-point convolution is mathematically mapped at the hardware level into high-throughput INT8 integer multiply-accumulate instructions targeting the Jetson Orin Tensor Cores, expressed as:

$$Y_q = \mathrm{SAT8}\left(\mathrm{Round}\left(\frac{s_w \cdot s_x}{s_y} W_q * X_q + \frac{b}{s_y}\right)\right) \tag{7}$$

where $\mathrm{SAT8}(\cdot)$ denotes the 8-bit saturation truncation function. Notably, the coefficient $\frac{s_w \cdot s_x}{s_y}$ is precomputed as a constant, eliminating expensive floating-point math during forward propagation.

Furthermore, TensorRT performs cross-layer operator fusion, equivalently merging fragmented operators (e.g., Conv + BN + SiLU) into a single large kernel. This optimization minimizes the storage of intermediate feature maps in the GPU memory, significantly reducing I/O latency. For the Jetson Ampere architecture, the compiler executes a tactic search to select the fastest execution path. As illustrated in Figure 6, ordinary convolution layers are fused in the INT8 domain, while the Swin Transformer block operators are preserved in the FP16 domain to maintain MHA fusion integrity.

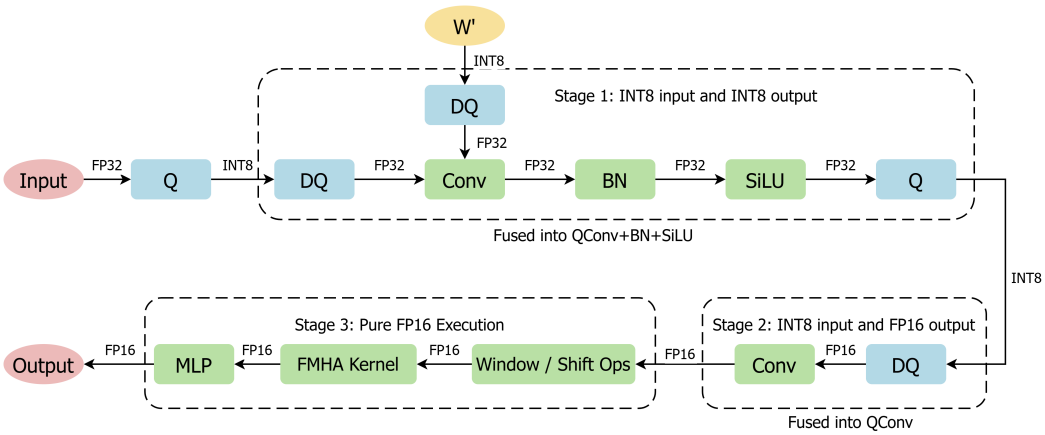

Figure 6. Schematic of operator fusion and edge-side mixed-precision quantization.

To further minimize latency, our method addresses the overhead of layout transformation nodes (Reformat / Transpose / Shuffle). The engine-building process is abstracted as a cost minimization problem:

$$\min \tau(G) = \sum_{v \in V} \tau_{\mathrm{kernel}}(v) + \sum_{e \in E_b} \tau_{\mathrm{reformat}}(e) \tag{8}$$

where E_b denotes the set of edges crossing precision or layout boundaries. Guided by this principle, we strategically shift quantization nodes upstream of the Concat operation into the INT8 domain. This forces the Concat operation to execute directly within the INT8 cache domain, enabling zero-copy memory concatenation and avoiding unnecessary Reformat operations triggered by precision conversion at quantization boundaries.

In summary, by introducing a sensitivity-driven mixed-precision QAT strategy alongside hardware-level TensorRT explicit compilation technology, this section establishes a highly efficient deployment pipeline from the algorithm level down to the Jetson hardware edge. This strategy allocates precision asymmetrically based on the distinct characteristics of Transformer and convolution opera-

 doi:10.20944/preprints202604.1422.v1

tors. While safeguarding the integrity of MHA fusion in the self-attention mechanism, it maximizes the length of the computational path executed in INT8 precision, significantly enhancing edge-side deployment speed with negligible accuracy degradation (less than 1.5%).

4.3. DeepStream-Based Multi-Object Detection System Deployment

Although model-level optimizations like pruning and mixed-precision QAT significantly reduce the computational intensity of DUST-YOLO, achieving peak real-time performance in aerial scenarios requires minimizing the systemic overhead introduced by video I/O and heterogeneous data scheduling. Traditional inference pipelines often suffer from CPU bottlenecks during decoding and memory copy latencies between the host and the accelerator. To address these system-level constraints, this section presents a high-throughput deployment framework based on DeepStream, which establishes a zero-copy memory paradigm and leverages hardware-specific acceleration units to achieve efficient end-to-end video analytics.

4.3.1. System Architecture Design

To construct an end-to-end, high-throughput UAV video analytics system, we utilized the DeepStream tool to integrate the optimized DUST-YOLO model into a highly efficient streaming media processing pipeline. DeepStream is not solely an inference engine but a heterogeneous computing platform based on the GStreamer framework, capable of achieving full-pipeline hardware acceleration from video capture, decoding, and preprocessing to inference and post-processing.

The overall system architecture, constructed on the Jetson end-to-end aerial computing platform, is illustrated in Figure 7. This architecture encompasses the complete data flow from front-end capture to application output.

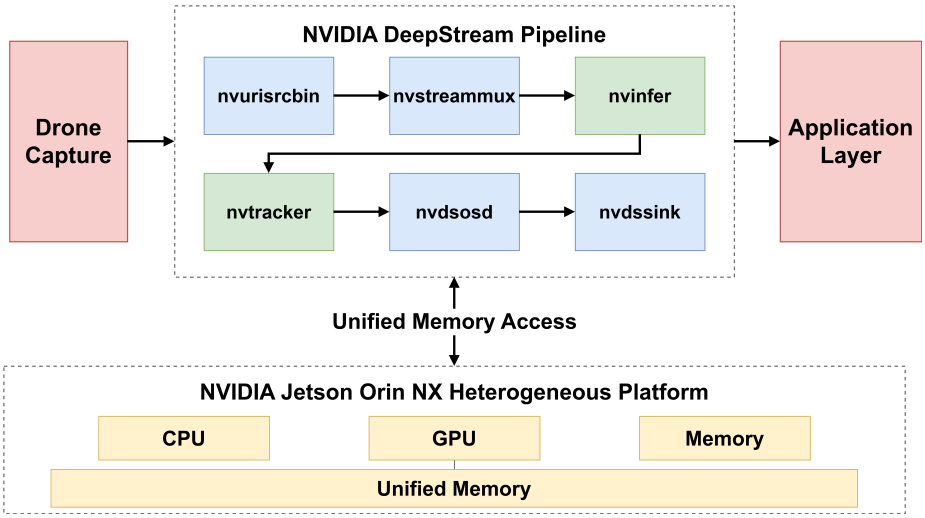

Figure 7. Deployment diagram of the multi-object detection system.

In the core processing pipeline, the high-resolution video stream captured by the UAV is first ingested through a video source plugin. Subsequently, it undergoes hardware-accelerated tensor batch preprocessing via stream multiplexing and batching plugins such as nvstreammux. The processed tensors are then fed into the core inference plugin, nvinfer, to execute highly efficient DUST-YOLO object detection inference. The system subsequently cascades the nvtracker and nvdsosd plugins to accomplish object trajectory tracking and bounding box rendering. Finally, the data is transmitted through an output node to the application layer for statistics, counting, and alarm decision-making, forming a complete AI-enabled smart surveillance paradigm.

The core advantage of this end-to-end architecture lies in its deep utilization of the physical unified memory architecture of the Jetson. This hardware-software co-design substantially reduces the scheduling overhead and bus communication latency of the host CPU. It achieves seamless heterogeneous decoupling of computational tasks and hardware resources, thereby practically guaranteeing

 doi:10.20944/preprints202604.1422.v1

the real-time performance of object detection tasks on the airborne platform under constrained power and computing conditions.

4.3.2. Hardware Acceleration and Parallel Mechanisms

The acceleration effect of DeepStream on DUST-YOLO primarily stems from system-level optimizations across the following three dimensions:

Heterogeneous Computing Decoupling

As shown in Figure 8, the front-end encoded video is decoded into raw frames using the dedicated hardware decoder (NVDEC). Subsequently, the Video Image Compositor (VIC) directly executes scaling and color space conversion within the VRAM, outputting RGB tensors for inference. By offloading these compute-intensive preprocessing tasks to specific hardware units, data flow remains strictly resident in the VRAM, entirely circumventing the memory copy overhead typical of traditional architectures.

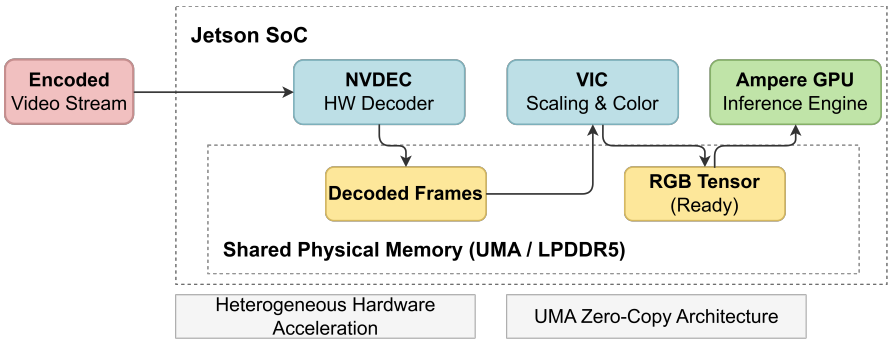

Figure 8. Schematic diagram of heterogeneous computing decoupling.

Tensor Batching

To process high-resolution UAV streams, the system utilizes the nvstreammux plugin to aggregate discrete frames into a four-dimensional inference tensor (Figure 9). This batching strategy maximizes the occupancy of GPU parallel compute units. By reducing the CUDA Kernel launch frequency from N to 1, system-level scheduling latency is significantly mitigated, guaranteeing real-time inference under high throughput.

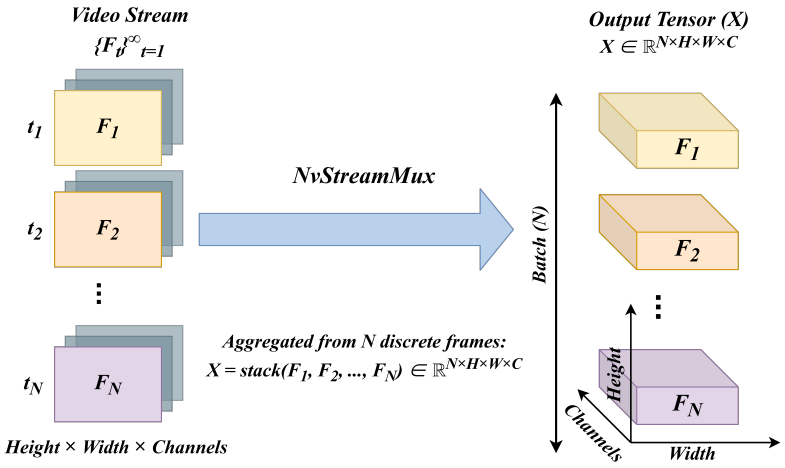

Figure 9. Schematic diagram of tensor batching.

 doi:10.20944/preprints202604.1422.v1

Highly Concurrent Pipeline

Leveraging an asynchronous pipeline architecture, the system achieves four-stage parallelism (decoding, preprocessing, inference, and post-processing). Specifically, the implementation of CUDA Streams enables the overlapping execution of the current frame's inference with the previous frame's post-processing. This design effectively masks I/O wait times, maximizing overall system throughput.

5. Experimental Results

5.1. Experimental Settings

Hardware and Software Environments: The experimental evaluation of the proposed framework was conducted across two distinct environments to simulate both high-performance training and resource-constrained edge deployment. The server-side operations, encompassing model training, structured pruning, fine-tuning, and QAT, were performed on a workstation equipped with an AMD Ryzen 9 9950X CPU and an NVIDIA GeForce RTX 4090 D GPU (24 GB VRAM), running on Ubuntu 22.04.5 LTS with CUDA 12.4, Python 3.8.18, and PyTorch 2.4.1. For edge-side deployment and inference benchmarking, we utilized the Jetson end-to-end aerial computing platform (Orin NX 16GB), which features an 8-core Arm Cortex-A78AE CPU and an Ampere-architecture GPU. This edge platform operated under a software environment comprising CUDA 12.6, Python 3.10.18, PyTorch 2.6.0, and TensorRT 10.3.0.

Dataset: To validate the effectiveness of the DUST-YOLO model within these environments, the VisDrone2019-DET dataset was employed for training and precision assessment [33,34]. This benchmark consists of 6,471 training images, 548 validation images, and 1,610 test images, covering ten distinct object categories including pedestrians, cars, and various types of tricycles. The dataset provides a comprehensive representation of complex urban scenarios, characterized by significant variations in UAV flight altitudes, illumination conditions, and occlusion levels, thereby offering a rigorous testbed for evaluating detection robustness and multi-scale feature extraction in real-world aerial surveillance.

Training Details and Evaluation Metrics: Building upon this data foundation, all models were trained at a standardized input resolution of 640×640 pixels using the Adam optimizer for 300 epochs with a batch size of 4. Performance evaluation was conducted through a dual-metric approach focusing on both detection accuracy and system-level efficiency. Accuracy was strictly validated on the VisDrone2019-DET validation set, while speed metrics were categorized into pure inference time and end-to-end pipeline latency.Specifically, pure inference time was measured based on TensorRT engine execution on the Jetson platform. In contrast, end-to-end latency was assessed across the complete system workflow—comprising video decoding, image preprocessing, and model inference—to reflect total operational delay. This evaluation was performed using a 211-second UAV aerial video sample captured at a fixed frame rate of 30 FPS, ensuring a consistent benchmark for all compared schemes and ablation stages.

5.2. Comparative Experiments

To evaluate the performance of various methods, we tested the deployed models on the Jetson platform across accuracy metrics (mAP@0.5 and mAP@0.5:0.95) on the VisDrone2019 validation set, as well as speed metrics (TensorRT pure inference time, end-to-end video detection latency, and end-to-end video detection frame rate). Our optimally compressed model achieved an mAP@0.5 of 43.7% and an mAP@0.5:0.95 of 25.9%, with a pure inference latency of 18.9 ms, an end-to-end video detection latency of 36.3 ms, and an end-to-end video detection frame rate of 27.5 FPS. The comparison results between our approach and other object detection works are summarized in Table 1.

 doi:10.20944/preprints202604.1422.v1

Table 1. Performance Comparison on Jetson.

Model	Resolution (pixels)	mAP @0.5 (%)	mAP @0.5:0.95 (%)	Inference Time (ms)	End-to-End Latency (ms)	End-to-End FPS	Speedup Ratio
YOLOv8s [35]	640 × 640	38.6	23.1	33.7	94.3	10.6	×1.00
YOLOv9s [36]	640 × 640	39.5	23.5	26.0	87.7	11.4	×1.08
YOLOv10s [37]	640 × 640	38.2	22.9	26.8	86.4	11.6	×1.09
YOLOv11s [38]	640 × 640	38.2	22.7	21.8	84.1	11.9	×1.12
YOLOv12s [39]	640 × 640	38.2	22.8	27.5	86.2	11.6	×1.09
FE-YOLO [40]	640 × 640	34.9	-	37.3	96.9	10.3	×0.97
YOLO-UD-s [17]	640 × 640	**45.5**	-	24.6	84.2	11.9	×1.12
DUST-YOLO (Ours)	640 × 640	43.7	**25.9**	**18.9**	**36.3**	**27.5**	**×2.60**

In recent years, the YOLO object detection family has undergone rapid architectural evolution [41, 42], continuously pushing the boundaries of speed and accuracy. Typically, from the classical YOLOv5 to the subsequent generations of YOLO algorithms (such as YOLOv8, YOLOv9, YOLOv10, YOLOv11, and YOLOv12), pre-trained weights are offered in various architectural scales (e.g., nano, small, medium, large, extra large). Among them, small-scale (s-tier) models often serve as representatives of high-speed models.

As a model derived from the YOLOv5l architecture, our approach significantly outperforms the recent versions of s-tier YOLO models in detection accuracy: the mAP@0.5 is generally 4.2% higher, and the mAP@0.5:0.95 is at least 2.4% higher than those of the s-tier baselines (compared to the most accurate baseline, YOLOv9s, which yielded 39.5% and 23.5%, respectively). Following a series of lightweighting procedures, including structural pruning and QAT, our final model inference time and end-to-end latency are substantially lower than those of the s-tier variants. Compared to the fastest recent baseline, YOLOv11s (21.8 ms), our inference time is reduced to 18.9 ms (an 86.7% relative duration); compared to YOLOv8s (33.7 ms), it requires only 56.1% of the execution time.

When evaluated against recent improved YOLO architectures, such as FE-YOLO (mAP@0.5=34.9%, inference time=37.3 ms), our model delivers an 8.8% absolute increase in mAP@0.5 while halving the inference time (50.7%). Compared to YOLO-UD-s (mAP@0.5=45.5%, inference time=24.6 ms), our model experiences a minor 1.8% drop in mAP@0.5 but cuts inference time down to 76.8%. These results indicate that our lightweighting methodology effectively boosts inference speed while maintaining robust accuracy, providing an excellent benchmark for meeting real-time requirements in edge-based end-to-end object detection.

By leveraging DeepStream to accelerate end-to-end video object detection, our system demonstrates a substantial throughput advantage. The end-to-end latency is reduced to 36.3 ms, significantly outperforming all evaluated algorithms. This lead in system-level latency is further amplified: our processing time is merely 37.5% of FE-YOLO's (96.9 ms) and 43.1% of YOLO-UD-s's (84.2 ms). In terms of end-to-end FPS, our work achieves 27.5 frames per second. Compared to the end-to-end performance of other YOLO series and lightweight object detection works (which typically operate around 10-12 FPS), our method achieves 2.3 to 2.6 times their frame rate. This validates that our pipeline optimizes the entire real-time detection workflow—from video capture, decoding, and preprocessing to inference and post-processing.

In summary, when navigating complex scenarios like VisDrone2019, the proposed algorithm and deployment framework successfully combine the baseline model with algorithmic lightweighting and embedded hardware acceleration. While maintaining competitive precision metrics, the pure model inference latency is restricted to 18.9 ms, the system-level end-to-end latency is controlled at 36.3 ms, and the end-to-end frame rate reaches 27.5 FPS, achieving a state-of-the-art trade-off among accuracy, inference speed, and overall system throughput.

 doi:10.20944/preprints202604.1422.v1

5.3. Ablation Study

To dissect the contribution of each component, an ablation study was performed. For clarity in the subsequent analyses and within Table 2, the fundamental optimization techniques are denoted by specific abbreviations. Specifically, **Opt. 1** denotes the application of FP16 quantization to maximize the utilization of hardware Tensor Cores; **Opt. 2** indicates the implementation of QAT to map compute-intensive layers to INT8 precision; and **Opt. 3** represents the proposed scale-aware structured pruning strategy. These techniques were evaluated both independently and synergistically to quantify their impacts on inference latency, model accuracy, and overall system throughput.

Comparing the baseline model with Opt. 3, introducing the proposed scale-aware structured pruning while maintaining FP32 precision significantly reduced the single-frame inference latency from 62.7 ms to 48.4 ms. Paradoxically, the pruned model's mAP@0.5 and mAP@0.5:0.95 improved to 45.1% and 27.3%, respectively. This phenomenon is attributed to the iterative "pruning-finetuning-repruning-refinetuning" strategy adopted in this study. By cyclically stripping redundant feature dimensions from deep networks and immediately restoring precision, this process not only alleviated the computational load but also acted as a strong regularizer, effectively suppressing the model's overfitting tendencies within the complex perspectives of UAV inspections.

To elucidate the source of this latency reduction, we quantified the parameter and computational cost drop brought by the varied pruning strategies. The network's total parameter count decreased from 45.46M to 23.66M, and the computational load dropped from 62.79G to 43.56G FLOPs. Notably, the most significant compression occurred within the four C3STR modules. As illustrated in Figure 10, the proposed strategy achieves a substantial reduction in redundant parameters for these Transformer detection heads. Specifically, for the tiny- (P2) and small-object (P3) detection heads, a relatively conservative pruning strategy is employed, yielding reductions in parameter count and computational cost of approximately 20% and 60%, respectively. In contrast, for the medium- (P4) and large-object (P5) detection heads, a more aggressive reduction is achieved, with both metrics decreasing by approximately 90%.

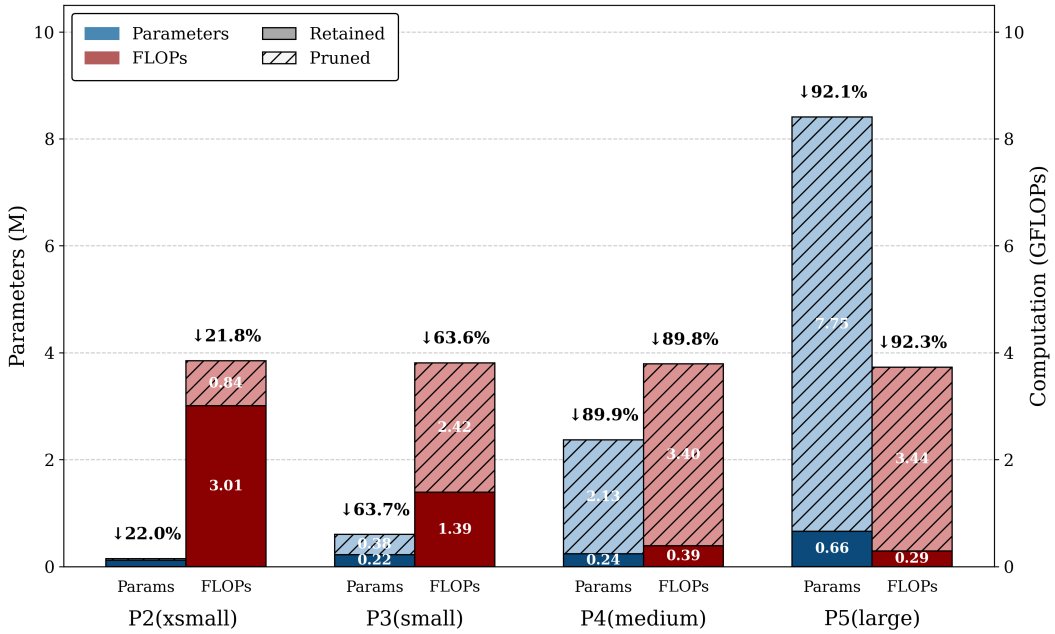

Figure 10. Comparison chart of parameter quantity and computational cost before and after C3STR pruning.

 doi:10.20944/preprints202604.1422.v1

Table 2. Ablation Study of Optimization Methods on Jetson Orin NX.

Method	FP16 Quantization	QAT (INT8)	Structured Pruning	DeepStream Deployment	mAP @0.5 (%)	mAP @0.5:0.95 (%)	Inference Time (ms)	End-to-End FPS	Speedup Ratio
Baseline					44.9	27.2	62.7	8.4	1.00×
Opt. 1	✓				44.9	27.2	28.9	11.3	1.35×
Opt. 1+2	✓	✓			43.6	25.8	23.3	12.2	1.45×
Opt. 3			✓		45.1	27.3	48.4	9.3	1.11×
Opt. 1+3	✓		✓		45.1	27.3	23.7	12.2	1.45×
Opt. 1+2+3	✓	✓	✓		43.7	25.9	**18.9**	14.1	1.68×
DUST-YOLO (Ours)	✓	✓	✓	✓	43.7	25.9	**18.9**	**27.5**	**3.27**×

 doi:10.20944/preprints202604.1422.v1

To fully exploit the underlying compute capability of Tensor Cores, Opt. 1 enabled FP16 quantization, decreasing latency to 28.9 ms with zero precision loss. Building upon this, Opt. 1+2 introduced QAT and edge-side mixed-precision deployment, mapping compute-intensive convolutional layers to the INT8 domain. Experiments show this strategy further compressed latency to 23.3 ms with negligible accuracy degradation (only a 1.3% drop compared to the baseline), yielding a 1.45× end-to-end framerate speedup.

As the optimal algorithmic lightweighting solution in this study, Opt. 1+2+3 integrated multidimensional pruning with mixed-precision QAT, achieving the lowest inference latency of 18.9 ms on the Jetson platform and an end-to-end speedup ratio of 1.68. This scheme realized the optimal balance between algorithmic complexity and hardware execution efficiency while maintaining high accuracy.

In traditional detection pipelines, CPU-executed video decoding, image preprocessing, and memory copy overheads constitute major system I/O bottlenecks. The final pipeline (Ours) integrated the model into the DeepStream asynchronous inference framework. By leveraging its heterogeneous computing decoupling, tensor batching, and concurrent pipeline mechanisms, non-inference time overheads were drastically reduced. Empirical data proves that this system-level optimization achieved an end-to-end framerate of 27.5 FPS without altering the intrinsic model accuracy, delivering a 3.27× speedup and completely resolving the edge-side performance bottlenecks in real-time UAV inspection tasks.

Beyond exceptional throughput, the paramount advantage of the proposed DUST-YOLO framework lies in its profound energy efficiency for battery-constrained UAVs. By synergizing extreme algorithmic compression with the DeepStream asynchronous architecture, our end-to-end aerial computing platform fundamentally reshapes the system's energy profile. While maximizing hardware utilization slightly elevates the average power plateau, the drastically shortened active duration (a 69.7% reduction in total execution time) effectively negates this increase. Consequently, our fully optimized framework achieves a staggering 68.5% reduction in cumulative total energy consumption compared to deploying the baseline model on a traditional synchronous pipeline.

The energy efficiency advantages are further substantiated by the power consumption trajectories. As illustrated in Figure 11, the power curve of the DeepStream-accelerated scheme (Figure 11a) exhibits superior steady-state characteristics compared to the native TensorRT approach (Figure 11b), and concludes with a step-like descent upon task completion. Given that the time-domain integral of power equates to total energy consumption, the significantly reduced area under the curve quantitatively verifies the minimization of energy dissipation. This demonstrates that the proposed algorithm-hardware co-design effectively establishes a high-throughput, low-latency computational paradigm, providing a reliable, endurance-optimized deployment strategy for battery-constrained UAV platforms during real-time inspections.

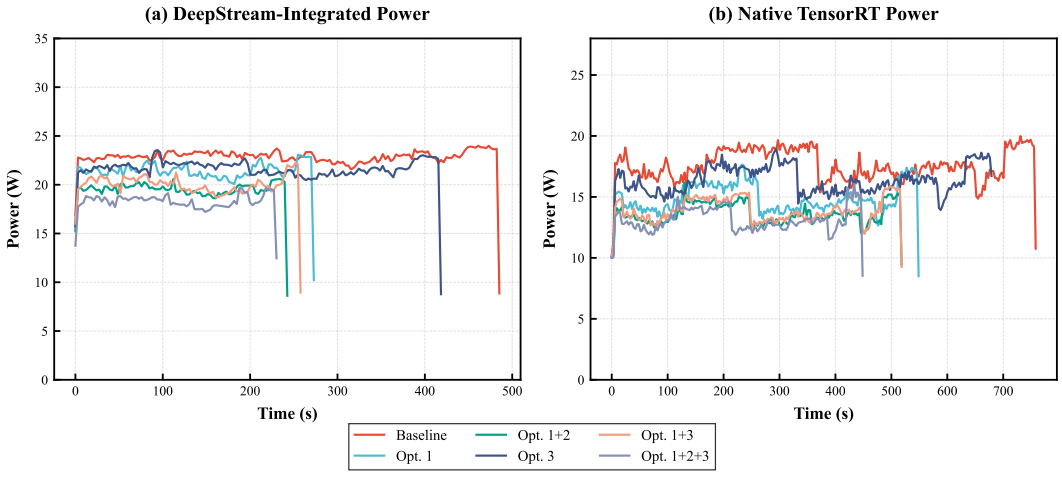

Figure 11. Power consumption trajectories of the object detection models under different acceleration frameworks. (**a**) Power consumption curve for the DeepStream implementation. (**b**) Power consumption curve for the native TensorRT approach.

 doi:10.20944/preprints202604.1422.v1

6. Conclusion

This paper presents an end-to-end framework to enable real-time, high-accuracy object detection on resource-constrained UAV edge devices. First, multidimensional structured pruning on the DUST-YOLO model reduces parameters by 48% and computational cost by 26% while preserving multi-scale feature extraction. Second, a mixed-precision QAT and TensorRT deployment strategy overcomes memory-access bottlenecks, optimally balancing algorithmic complexity and hardware efficiency. Finally, integrating the optimized model into the DeepStream framework enables full system-level hardware acceleration through tensor batching and heterogeneous computing decoupling.Experimental results on the Jetson Orin NX platform demonstrate significant improvements. The proposed framework reduces pure inference latency to 18.9 ms (a 70% reduction) and boosts the end-to-end video detection rate to 27.5 FPS (over 3 times the baseline). Furthermore, energy consumption is substantially reduced to 31.5% of the baseline, all with negligible accuracy loss (maintaining 43.7% mAP@0.5 on VisDrone2019). Ultimately, this method achieves an optimal balance of accuracy, speed, and power efficiency, providing a highly practical edge-computing solution for the emerging low-altitude economy and intelligent UAV inspections.

Author Contributions: Conceptualization, G.L., J.J., J.C., H.S. and Z.W.; methodology, G.L., J.J., J.C, and Z.W.; software, G.L. and J.J.; validation, G.L., J.C. and X.L.; formal analysis, G.L. and Z.W.; investigation, J.J. and X.L.; resources, H.S. and X.L.; data curation, G.L., J.J., J.C.; writing—original draft preparation, G.L., J.J. and J.C.; writing—review and editing, H.S. and Z.P.; visualization, G.L., J.C. and J.J.; supervision, H.S. and Z.P.; project administration, H.S. and Z.P.; funding acquisition, H.S. and Z.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Southeast University Interdisciplinary Research Program for Young Scholars (Grant No. 2024FGC1006), the National Natural Science Foundation of China (Grants 52525204 and 52572354), the Jiangsu International Collaborative Research Project (Grant No. BZ2024055), and the National Undergraduate Training Programs for Innovation (Grant No. 202510286051).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article or supplementary material.

Acknowledgments: We sincerely thank our advisors, Prof. Hao Sun (School of Electronic Science and Engineering) and Prof. Ziyuan Pu (School of Transportation), for their invaluable guidance, methodological insights, and generous provision of computational resources, hardware platforms, and funding support. We also extend our gratitude to our senior peers for their technical mentorship in algorithmic compression and edge deployment.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Echchidmi, M.; Bouayad, A. TinyML for sustainable edge intelligence: Practical optimization under extreme resource constraints. *Technologies* **2026**, *14*, 215. https://doi.org/10.3390/technologies14040215
2. Shih, W.C.; Wang, Z.Y.; Kristiani, E.; Chen, M.Y.; Shih, W.C. The construction of a stream service application with DeepStream and simple realtime server using containerization for edge computing. *Sensors* **2025**, *25*, 259. https://doi.org/10.3390/s24248102
3. Gong, J.; Yuan, Z.; Li, W.; Li, W.; Guo, Y.; Guo, B. A Lightweight Upsampling and Cross-Modal Feature Fusion-Based Algorithm for Small-Object Detection in UAV Imagery. *Electronics* **2026**, *15*, 298. https://doi.org/10.3390/electronics15020298
4. Jiang, Z.; Li, C.; Qu, T.; He, C.; Wang, D. MSQuant: Efficient post-training quantization for object detection via migration scale search. *Electronics* **2025**, *14*, 504. https://doi.org/10.3390/electronics14030504
5. Tian, L.; Wang, P. An effective mixed-precision quantization method for joint image deblurring and edge detection. *Electronics* **2025**, *14*, 1767. https://doi.org/10.3390/electronics14091767
6. Sun, J.; Gao, H.; Yan, Z.; Qi, X.; Yu, J.; Ju, Z. Lightweight UAV Object-Detection Method Based on Efficient Multidimensional Global Feature Adaptive Fusion and Knowledge Distillation. *Electronics* **2024**, *13*, 1558.

 doi:10.20944/preprints202604.1422.v1

7. Yang, R.; Li, W.; Shang, X.; Zhu, D.; Man, X. KPE-YOLOv5: An improved small target detection algorithm based on YOLOv5. *Electronics* **2023**, *12*, 817. https://doi.org/10.3390/electronics12040817
8. Wang, K.; Zhou, H.; Wu, H.; Yuan, G. RN-YOLO: A Small Target Detection Model for Aerial Remote-Sensing Images. *Electronics* **2024**, *13*, 2383. https://doi.org/10.3390/electronics13122383
9. Li, Z.; Xiao, J.; Yang, L.; Gu, Q. RepQ-ViT: Scale Reparameterization for Post-Training Quantization of Vision Transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023; pp. 17181–17190. https://doi.org/10.1109/ICCV51070.2023.01580
10. Zhu, W.; Chen, K. Real-time object detection for unmanned aerial vehicles based on vision transformer and edge computing. *Sci. Rep.* **2026**, *16*, 6814. https://doi.org/10.1038/s41598-026-37938-5
11. Jocher, G.; Stoken, A.; Borovec, J.; Chaurasia, A.; Qiu, J.; et al. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation. *Zenodo* **2022**. https://doi.org/10.5281/zenodo.7347926
12. Zhu, X.; Lyu, S.; Wang, X.; Zhao, Q. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), Montreal, QC, Canada, 11–17 October 2021; pp. 2778–2788. https://doi.org/10.1109/ICCVW54141.2021.00312
13. Roy, A.M.; Bhaduri, J. A computer vision enabled damage detection model with improved YOLOV5 based on transformer prediction head. *Ecol. Inform.* **2023**, *77*, 102124. https://doi.org/10.1016/j.ecoinf.2023.102124
14. Shi, H.; Cheng, X.; Mao, W.; Wang, Z. P^2-ViT: Power-of-two post-training quantization and acceleration for fully quantized vision transformer. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2024**, *32*, 1704–1717. https://doi.org/10.1109/TVLSI.2024.3401234
15. Alhrgan, H.M.; Alharbi, N.A.; Alshaya, A.M.; Alhrgan, S.O.; Alanezi, R.A.; Alanezi, M.S.; Alshaya, S.S.; Sharif, N.U.; Alkanhal, R. Benchmarking YOLOv8 variants for object detection efficiency on Jetson Orin NX for edge computing applications. *Computers* **2025**, *15*, 74. https://doi.org/10.3390/computers15020074
16. Xue, C.; Xia, Y.; Wu, M.; Yun, L. EL-YOLO: An efficient and lightweight low-altitude aerial objects detector for onboard applications. *Expert Syst. Appl.* **2024**, *238*, 122204. https://doi.org/10.1016/j.eswa.2023.122204
17. Wu, J.; Meng, H.; Yuan, M.; Liu, C.; Lu, Z. Enhanced feature representation for real time UAV image object detection using contextual information and adaptive fusion. *Sci. Rep.* **2025**, *15*, 33711. https://doi.org/10.1038/s41598-025-54321-x
18. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125. https://doi.org/10.1109/CVPR.2017.227
19. Zhao, Z.; Liu, X.; He, P. PSO-YOLO: A contextual feature enhancement method for small object detection in UAV aerial images. *Earth Sci. Inform.* **2025**, *18*, 258. https://doi.org/10.1007/s12145-024-01567-w
20. Cai, S.; Wu, Z.; Liu, K.; Zhang, T.; Weng, W.; Zheng, X. LSOD-YOLO: A visual object detection method for AGV perception systems based on a lightweight backbone and detection head. *Technologies* **2026**, *14*, 173. https://doi.org/10.3390/technologies14030173
21. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual Event, 3–7 May 2021. https://doi.org/10.48550/arXiv.2010.11929
22. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 11–17 October 2021; pp. 10012–10022. https://doi.org/10.1109/ICCV48922.2021.00986
23. Hakani, R.; Rawat, A. Edge computing-driven real-time drone detection using YOLOv9 and NVIDIA Jetson Nano. *Drones* **2024**, *8*, 680. https://doi.org/10.3390/drones8110680
24. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 8759–8768. https://doi.org/10.1109/CVPR.2018.00913
25. Mi, Q. YOLO11s-UAV: An advanced algorithm for small object detection in UAV aerial imagery. *J. Imaging* **2026**, *12*, 69. https://doi.org/10.3390/jimaging12030069
26. Fan, Q.; Li, Y.; Deveci, M.; Zhong, K.; Kadry, S. LUD-YOLO: A novel lightweight object detection network for unmanned aerial vehicle. *Inf. Sci.* **2025**, *686*, 121366. https://doi.org/10.1016/j.ins.2024.121366
27. Huang, M.; Mi, W.; Wang, Y. EDGS-YOLOv8: An improved YOLOv8 lightweight UAV detection model. *Drones* **2024**, *8*, 337. https://doi.org/10.3390/drones8070337

 doi:10.20944/preprints202604.1422.v1

28. Xie, S.; Deng, G.; Lin, B.; Jing, W.; Li, Y.; Zhao, X. Real-time object detection from UAV inspection videos by combining YOLOv5s and DeepStream. *Sensors* **2024**, *24*, 3862. https://doi.org/10.3390/s24123862
29. Barthelemy, J.; Iqbal, U.; Qian, Y.; Amirghasemi, M.; Perez, P. Safety after dark: A privacy compliant and real-time edge computing intelligent video analytics for safer public transportation. *Sensors* **2024**, *24*, 8102. https://doi.org/10.3390/s24248102
30. Yue, M.; Zhang, L.; Huang, J.; Zhang, H. Lightweight and efficient tiny-object detection based on improved YOLOv8n for UAV aerial images. *Drones* **2024**, *8*, 276. https://doi.org/10.3390/drones8060276
31. Liu, C.; Gao, G.; Huang, Z.; Hu, Z.; Liu, Q.; Wang, Y. YOLC: You only look clusters for tiny object detection in aerial images. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 1–14. https://doi.org/10.1109/TITS.2024.3356789
32. Ma, C.; Fu, Y.Y.; Wang, D.; Guo, R.; Zhao, X.; Fang, J. YOLO-UAV: Object detection method of unmanned aerial vehicle imagery based on efficient multi-scale feature fusion. *IEEE Access* **2023**, *11*, 126857–126878. https://doi.org/10.1109/ACCESS.2023.3332154
33. Zhu, P.; Wen, L.; Bian, X.; Ling, H.; Hu, Q. Vision meets drones: A challenge. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 7380–7395. https://doi.org/10.1109/TPAMI.2021.3068176
34. Cao, Y.; He, Z.; Wang, L.; Wang, W.; Yuan, Y.; Zhang, D.; Zhang, J.; Zhu, P.; Van Gool, L.; Han, J.; et al. VisDrone-DET2021: The vision meets drone object detection challenge results. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), Montreal, QC, Canada, 11–17 October 2021; pp. 2847–2854. https://doi.org/10.1109/ICCVW54141.2021.00319
35. Jocher, G.; Chaurasia, A.; Qiu, J. Ultralytics YOLOv8. **2023**. Available online: https://github.com/ultralytics/ultralytics (accessed on 15 April 2026).
36. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. YOLOv9: Learning what you want to learn using programmable gradient information. In Proceedings of the European Conference on Computer Vision (ECCV), Milan, Italy, 29 September–4 October 2024; pp. 1–21. https://doi.org/10.48550/arXiv.2402.13616
37. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. YOLOv10: Real-Time End-to-End Object Detection. *arXiv* **2024**, arXiv:2405.14458.
38. Jocher, G.; Qiu, J. Ultralytics YOLO11. **2024**. Available online: https://github.com/ultralytics/ultralytics (accessed on 15 April 2026).
39. Tian, Y.; Ye, Q.; Doermann, D. YOLOv12: Attention-Centric Real-Time Object Detectors. *arXiv* **2025**, arXiv:2502.12524.
40. Gao, F.; An, J.; Zhang, M.; Chen, X.; Zhao, Q. FE-YOLO: A Traffic Target Detection Network Based on YOLOv11n. In *Proceedings of the 2025 44th Chinese Control Conference (CCC)*, Chongqing, China, 2025; pp. 8845–8850. https://doi.org/10.23919/CCC64809.2025.11179290
41. Jiang, P.; Ergu, D.; Liu, F.; Cai, Y.; Ma, B. A review of YOLO algorithm developments. *Procedia Comput. Sci.* **2022**, *199*, 1066–1073. https://doi.org/10.1016/j.procs.2022.01.135
42. Terven, J.; Cordova-Esparza, D. A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Front. Robot. AI* **2023**, *10*, 1212874. https://doi.org/10.3389/frobt.2023.1212874