Intelligent Systems with Applications 25 (2025) 200484

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

# LEAF-YOLO: Lightweight Edge-Real-Time Small Object Detection on Aerial Imagery

Nghiem Van Quang, Nguyen Huy Hoang *, Hoang Minh Son

*Hanoi University of Science and Technology, Hanoi, 10000, Viet Nam*

ARTICLE INFO

*Keywords:*
Aerial imagery
UAV imagery
Small object detection
Edge-real-time algorithm
You only look once (YOLO)

ABSTRACT

Advances in Unmanned Aerial Vehicles (UAVs) and deep learning have spotlighted the challenges of detecting small objects in UAV imagery, where limited computational resources complicate deployment on edge devices. While many high-accuracy deep learning solutions have been developed, their large parameter sizes hinder deployment on edge devices where low latency and efficient resource use are essential. To address this, we propose LEAF-YOLO, a lightweight and efficient object detection algorithm with two versions: LEAF-YOLO (standard) and LEAF-YOLO-N (nano). Using Lightweight-Efficient Aggregating Fusion along with other blocks and techniques, LEAF-YOLO enhances multiscale feature extraction while reducing complexity, targeting small object detection in dense and varied backgrounds. Experimental results show that both LEAF-YOLO and LEAF-YOLO-N outperform models with fewer than 20 million parameters in accuracy and efficiency on the Visdrone2019-DET-val dataset, running in real-time (>30 FPS) on the Jetson AGX Xavier. LEAF-YOLO-N achieves 21.9% $AP_{.50:.95}$ and 39.7% $AP_{.50}$ with only 1.2M parameters. LEAF-YOLO achieves 28.2% $AP_{.50:.95}$ and 48.3% $AP_{.50}$ with 4.28M parameters. Furthermore, LEAF-YOLO attains 23% $AP_{.50}$ on the TinyPerson dataset, outperforming models with ≥ 20 million parameters, making it suitable for UAV-based human detection.

## 1. Introduction

Detecting small objects in aerial photographs acquired by drones, Unmanned Aerial Vehicles (UAVs), and other airborne platforms is crucial for various applications, such as construction monitoring (Tao, Zheng, Wang, Qiu, & Stojanovic, 2024; Zhu, Zhu, Bu, & Gao, 2022), precision agriculture (Zualkernan, Abuhani, Hussain, Khan, & ElMohandes, 2023), violence detection (Nguyen, Trung Le, Nghiem, Son Hoang, & Pham, 2023), simultaneous localization and mapping (SLAM) (Li, Shen, Fu, & Wang, 2024), and pedestrian surveillance (Nguyen et al., 2021). However, UAV aerial images present several unique challenges that complicate the detection process (Du, Zhu, & Wen, 2019). One of the primary issues is scale variation: objects appear at drastically different sizes depending on factors such as altitude, camera angle, and distance from the UAV (Li, Wan, Cheng, Meng, & Han, 2020). For instance, vehicles, people, or equipment on the ground may occupy only a few pixels, which limits the model's ability to learn distinct features for accurate classification. Furthermore, small objects in low-resolution images often blend into the background, leading to missed detections or false positives, especially in cluttered environments.

The complexity of backgrounds in aerial imagery is another significant challenge (Xia et al., 2018). Unlike controlled settings, UAV images often contain varied landscapes – such as urban areas, agricultural fields, and natural environments – each with its own textures, colors, and lighting conditions. This variety introduces high background noise and may cause small objects to be overshadowed by larger, more complex background elements. Additionally, objects of interest may be camouflaged due to similar colors or textures, making it difficult for detection algorithms to distinguish them from their surroundings. Finally, the dense distribution of objects within UAV images intensifies the difficulty. In applications such as crowd monitoring or vehicle tracking, a high concentration of objects may lead to significant overlap between them, further complicating object separation and localization. These challenges demand robust models capable of distinguishing small, densely packed objects within noisy backgrounds while effectively handling scale and perspective changes. Given these issues, developing a model optimized for small object detection with enhanced feature extraction and spatial awareness is essential for improving detection accuracy and reliability in UAV applications.

Traditional methods for small object detection in aerial imagery often rely on handcrafted features and shallow learning algorithms (Dalal & Triggs, 2005; Pool & Vatsavai, 2018). These methods typically involve a multi-step process, starting with extracting low-level features

* Corresponding author.
*E-mail addresses:* quang.nv203547@sis.hust.edu.vn (Nghiem V.Q.), Hoang.nguyenhuy@hust.edu.vn (Nguyen H.H.), son.hm193081@sis.hust.edu.vn (Hoang M.S.).

https://doi.org/10.1016/j.iswa.2025.200484
Received 31 August 2024; Received in revised form 11 January 2025; Accepted 19 January 2025
Available online 1 February 2025

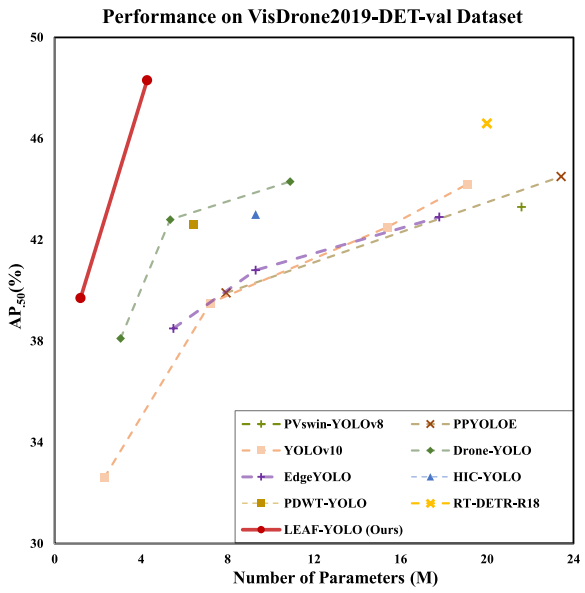


**Fig. 1.** Comparisons with others in terms of size-accuracy on Visdrone2019-DET-val. All models are train-from-scratch on Visdrone2019-DET-train.

such as edges, corners, and textures, then constructing higher-level representations and applying machine learning classifiers such as multilayer perceptrons (MLPs), support vector machines (SVMs), or decision trees. While traditional methods have been successful in specific scenarios, they often struggle with complex backgrounds, scale variation, and occlusion, limiting their performance in real-world applications. Furthermore, the handcrafted features used in these methods may not generalize well across different datasets and imaging conditions, making them less robust and adaptable than deep learning approaches.

In recent years, deep learning techniques, particularly Convolutional Neural Networks (CNNs), have emerged as powerful tools for small object detection in aerial imagery. These CNN-based approaches can automatically learn discriminative features directly from the data, eliminating the need for handcrafted feature engineering and improving detection performance. Two main categories of CNN-based methods for small object detection are two-stage and one-stage methods. Two-stage methods, exemplified by Region-based Convolutional Neural Networks (R-CNN) (Girshick, Donahue, Darrell, & Malik, 2014), first propose a set of candidate object regions and then classify and refine these proposals to generate final detections. However, the processing time of these models is often very long. According to Gavrilescu, Zet, Foşalău, Skoczylas, and Cotovanu (2018), Faster R-CNN is an improved two-stage model that uses a Region Proposal Network (RPN) to make object detection faster. However, the computational cost and number of parameters remain high, making such models complex to deploy for real-time detection problems.

On the other hand, single-stage detectors such as YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016) and SSD (Liu et al., 2016) eliminate the need to generate candidate regions, instead directly performing classification and localization tasks on the feature map, thereby reducing computational load. For instance, YOLOv7-Tiny (YOLOv7-T) (Wang, Bochkovskiy, & Liao, 2023) was chosen as the baseline model for real-time small object detection algorithms from UAVs, such as EdgeYOLO (Liu, Zha, Sun, Li, & Wang, 2023) and PDWT-YOLO (Zhang et al., 2023). YOLOv7-T is a one-stage model that provides processing speed while maintaining good accuracy, owing to its PAFPN (Liu, Qi, Qin, Shi, & Jia, 2018) architecture and Efficient Layer Aggregation Network (ELAN) (Wang et al., 2023). Consequently, single-stage detectors such as YOLOv7-T and other YOLO-based algorithms (Ge, Liu, Wang, Li, & Sun, 2021; Jocher, 2020; Jocher, Chaurasia, & Qiu, 2023; Li et al., 2023; Wang et al., 2024, 2024; Xu, Jiang, Chen, Huang, Zhang, & Sun, 2023), which balance detection accuracy, speed, and model size, are better suited for object detection tasks in drone aerial scenes.

In aerial images captured by UAVs, the small size of targets, significant scale variations, and the prevalence of complex interference backgrounds pose unique challenges to object detection algorithms. Numerous datasets, including VisDrone2019-DET (Du et al., 2019), TinyPerson (Yu, Gong, Jiang, Ye, & Han, 2020), and UAVDT (Yu et al., 2020), have been published to address these challenges. Several solutions and research efforts (Nguyen, Nghiem, Hoang, Nghiem, & Dang, 2024; Zhao, Liu, Lyu, Wang, & Zhang, 2023; Zhao & Zhu, 2023; Zhu, Lyu, Wang, & Zhao, 2021) have been tested on VisDrone2019. However, a common limitation of these studies is their sole focus on accuracy, resulting in models with a very large number of parameters, making deployment on drones at the edge impractical. Other studies have attempted to develop lightweight models, such as PDWT-YOLO (Zhang, Xiong, et al., 2023), HIC-YOLO (Tang, Zhang, & Fang, 2024), LE-YOLO (Yue, Zhang, Huang, & Zhang, 2024), or Drone-YOLO (Zhang, 2023). Nonetheless, these models lack suitable modules for effective object detection and fail to fully utilize the PAFPN architecture in YOLO, ultimately leading to lower accuracy. Furthermore, very few studies have deployed their "lightweight" models on edge devices while achieving real-time performance, highlighting a critical gap in this domain.

To address the challenges of small object detection on edge devices, we propose a novel lightweight solution called LEAF-YOLO. This algorithm is designed for real-time processing and includes two versions: LEAF-YOLO (standard) and LEAF-YOLO-N (nano-size). Both models have been extensively trained and evaluated on the VisDrone2019-DET dataset, demonstrating superior performance over all models with fewer than 20 million parameters, as shown in Fig. 1. Specifically, LEAF-YOLO-N achieves 21.9% $AP_{.50:.95}$ and 39.7% $AP_{.50}$ with only 1.2 million parameters and 5.6 GFLOPs, while LEAF-YOLO attains 28.2% $AP_{.50:.95}$ and 48.3% $AP_{.50}$ with 4.28 million parameters and 20.9 GFLOPs. Both models maintain real-time performance, exceeding 30 FPS on the Jetson AGX Xavier edge device. Additionally, LEAF-YOLO demonstrates a 23% $AP_{.50}$ improvement on the TinyPerson dataset, outperforming larger models with more than 20 million parameters, making it particularly effective for human detection in UAV-captured imagery. The code, hyperparameters, and model weights are publicly available at https://github.com/highquanglity/LEAF-YOLO.

The following are the main contributions of this work:

1. **Optimized Network Design for Small Object Detection in Aerial Imagery:** We developed a Small Object Detection Head and restructured feature extraction blocks within the PAFPN (Liu, Qi, et al., 2018) architecture to enhance the detection of small objects in aerial images. Additionally, we designed the LEAF and LEAF-T modules to create multiscale feature maps, enabling the model to effectively learn diverse features of small objects while maintaining efficiency. To further improve spatial awareness during upsampling, we integrated a Coordinate Block, combining CoordConv (Liu et al., 2018) and Coordinate Attention (CoordATT) (Hou, Zhou, & Feng, 2021). This block helps retain spatial information by embedding positional context into the feature maps, enabling the model to focus more effectively on object locations, especially for small objects.
2. **Parameter-Efficient Modules for Real-Time Edge Deployment:** We introduced the MaxPooling—Ghost Convolution (MGC) module and Partial Convolutions (PConv) (Chen, Kao, He, & Zhu, 2023) into the network's architecture. These modules effectively reduce the parameter count and computational load without sacrificing accuracy. Additionally, the Spatial Pyramid Pooling Receptive Field Module (SPPRFM) was designed to replace the traditional SPP (He, Zhang, Ren, & Sun, 2014), enhancing receptive field utilization for improved feature capture.

3. **Enhanced Bounding Box Regression:** To improve bounding box regression accuracy, we implemented the InnerShape-IoU loss, which helps the model generate bounding boxes that closely match the shapes and sizes of small objects. This is particularly beneficial in high-density object detection scenarios.
4. **Empirical Evaluation and Real-Time Edge Deployment on Jetson AGX Xavier:** We deployed both LEAF-YOLO and LEAF-YOLO-N on the Jetson AGX Xavier, leveraging TensorRT to achieve real-time performance (> 30 FPS). Extensive empirical evaluations on the VisDrone2019-DET (Du et al., 2019) and TinyPerson (Yu, Gong, et al., 2020) datasets demonstrate that both LEAF-YOLO models outperform comparable lightweight models with fewer than 20 million parameters in terms of mAP and speed, confirming the efficacy and efficiency of our approach.

The document is divided into five parts: Section 2 reviews related research. Section 3 provides a detailed description of the proposed method. Section 4 outlines the experimental process and evaluates the results, including comparative ablation experiments and analysis of experimental outcomes. Finally, Section 5 summarizes the paper.

## 2. Related works and background

### *2.1. Previous works on real-time and lightweight algorithms for object detection on aerial imagery*

With the rapid advancement of UAV technology, object detection in aerial imagery has become an increasingly significant research topic. Approaches in this domain can be broadly categorized into two directions: (1) maximizing detection accuracy, often at the cost of model complexity and latency (Zhao et al., 2023; Zhao & Zhu, 2023; Zhu et al., 2021), and (2) creating lightweight models that achieve efficient accuracy for real-time deployment on edge devices (Liu et al., 2023; Tahir, Long, Zhang, Asim, & ELAffendi, 2024; Tang et al., 2024; Xu, Zheng, Liu, Li, & Wang, 2023; Yue et al., 2024; Zhang, 2023; Zhang, Xiong, et al., 2023). This study primarily focuses on the second direction, aiming to design a lightweight and efficient model suitable for real-time edge deployment.

Many existing models designed for real-time performance build upon YOLO architectures due to their balance of accuracy and speed. Enhancements in these models often focus on three key components: feature extraction, attention mechanisms, and detection heads. For example, HIC-YOLO (Tang et al., 2024) and PVswin-YOLOv8s (Tahir et al., 2024) both employ the Convolutional Block Attention Module (CBAM) (Woo, Park, Lee, & Kweon, 2018) to refine attention to relevant features while eliminating redundant information. This attention mechanism is particularly effective for improving the detection of small and distant objects. Additionally, PVswin-YOLOv8s integrates a Swin Transformer (Liu et al., 2021) block to enhance global feature extraction, while HIC-YOLO introduces an Involution (Li et al., 2021) block to strengthen channel feature representations. PDWT-YOLO (Zhang, Xiong, et al., 2023), an adaptation of YOLOv7-T, also incorporates advanced techniques to improve small object detection. It replaces the Implicit Head (Wang, Yeh, & Liao, 2021) with a Decoupled Head and introduces the WIoU (Tong, Chen, Xu, & Yu, 2023) loss function to improve bounding box regression. However, models like HIC-YOLO and PDWT-YOLO primarily focus on enhancing output heads and loss functions, leaving the feature extraction process less optimized, which can limit their performance in challenging aerial imagery.

Drone-YOLO (Zhang, 2023) adopts a novel sandwich-fusion module to improve feature integration across multiple stages of the network. By concatenating lower-stage features using depthwise convolutions, mid-stage features directly, and higher-stage features via upsampling, it optimizes the PAFPN structure for better feature preservation. Additionally, Drone-YOLO enhances downsampling layers in the backbone using RepVGG (Ding et al., 2021), which reduces parameter complexity while maintaining strong feature extraction capabilities. Similarly, the Lightweight and Efficient Tiny-Object Detection method based on YOLOv8n (LE-YOLO) (Yue et al., 2024) introduces the LHGNet backbone with depthwise separable convolutions and channel shuffling to establish a deep feature extraction network. The model also leverages the LGS bottleneck and LGSCSP fusion module, which use concatenation operations to preserve channel information and improve multi-scale feature integration. Furthermore, by optimizing the detection head and feature map sizes, this method achieves substantial accuracy gains in detecting tiny objects while maintaining a lightweight architecture suitable for real-time deployment.

Other methods focus on improving attention mechanisms and activation functions to enhance small object recognition. For example, an improved YOLOv5-based algorithm (YOLOv5s-pp) (Xu, Zheng, et al., 2023) introduces the CA attention mechanism to integrate lightweight channel and spatial attention, as well as the Meta-ACON (Ma, Zhang, Liu, & Sun, 2021) activation function, which dynamically adjusts nonlinearity based on input data. Additionally, the SPD Conv module reduces the loss of fine-grained information during cross-layer convolution, improving the detection of small targets.

EdgeYOLO (Liu et al., 2023) prioritizes latency reduction, making it one of the most efficient models for real-time applications. It employs an anchor-free lightweight decoupled head and unique data augmentation techniques to enhance small object detection accuracy. By using non-standard channel numbers (e.g., 360, 240, 120, 60, 24) rather than the power-of −2 channels typical in YOLO architectures, EdgeYOLO reduces both latency and parameter count without compromising detection performance.

These innovations reflect a broader trend toward balancing efficiency and accuracy in lightweight object detection models. However, while many of these approaches focus on specific components—such as attention mechanisms, detection heads, or backbone structures—they often lack holistic optimization across the entire architecture. In contrast, this study takes an integrated approach, combining lightweight feature extraction blocks, advanced attention mechanisms, and optimized detection heads to achieve state-of-the-art accuracy and efficiency for small object detection in UAV applications.

### *2.2. Theoretical background*

#### *2.2.1. YOLOv7-T (Wang et al., 2023)*

YOLOv7-T (Wang et al., 2023), a streamlined version of YOLOv7 (Wang et al., 2023) optimized for edge deployment, is structured into three main components: Backbone, Neck, and Prediction Head. The Backbone and Neck feature the Path Aggregation Feature Pyramid Network (PAFPN) paired with ELAN blocks, which extract features with fewer parameters compared to the Extended ELAN (E-ELAN) blocks (Wang et al., 2023). The PAFPN structure (shown in Fig. 2) constructs a hierarchical pyramid of feature maps, enabling multi-scale object detection. The ELAN blocks use $1 \times 1$ convolutions to adjust network width and $3 \times 3$ convolutions to add depth, producing feature maps at multiple scales, making them well-suited for small object detection.

In the Prediction Head, YOLOv7-T employs an Implicit Head, inspired by YOLOR (Wang et al., 2021). This design combines explicit knowledge (from input features) with implicit knowledge (learned during training) to improve prediction accuracy without increasing model complexity. As illustrated in Fig. 3, the Implicit Head introduces a Normal Distribution to the input feature map, enabling the model to learn both explicit and implicit patterns, thereby enhancing predictive performance.

Fig. 2 illustrates the general architecture of YOLOv7-T. PAFPN constructs hierarchical pyramid feature maps, allowing YOLOv7-T to simultaneously detect objects at different scales. Furthermore, within each stage of the PAFPN pyramid, YOLOv7-T employs ELAN blocks

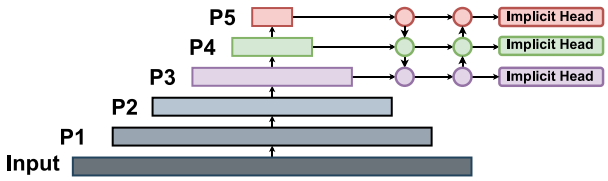


**Fig. 2.** PAFPN structure in YOLOv7-T.

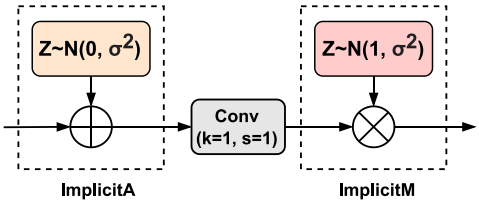


**Fig. 3.** Implicit Head of YOLOv7-T.

(Fig. 7), which use 1 × 1 convolution layers to scale the network's width and modify the number of feature map channels. This is followed by 3 × 3 convolution layers to increase network depth and create multi-scale feature maps. The combination of PAFPN and ELAN blocks makes YOLOv7-T an effective architecture for addressing small object detection challenges.

In the Prediction Head, YOLOv7-T integrates an Implicit Head inspired by YOLOR (Wang et al., 2021). In YOLOR, explicit knowledge refers to features learned from the input, while implicit knowledge encompasses patterns recognized during training independently of the input. As shown in Fig. 3, the input to the Implicit Head consists of explicit knowledge extracted from the Backbone, while implicit knowledge is introduced through vectors incorporated via Normal Distributions. Specifically, the ImplicitA component adds a Normal Distribution with a mean of 0 and a standard deviation $\sigma$ to the input features. Following this, the ImplicitM component multiplies the feature map (now processed with ImplicitA and passed through a 1 × 1 convolution layer) by a Normal Distribution with a mean of 1. This process produces an output feature map that effectively combines explicit and implicit knowledge, significantly enhancing the accuracy of YOLOv7-T's Prediction Head without increasing the network's parameter count or complexity.

#### 2.2.2. YOLOv7-T loss function

The YOLOv7-T algorithm's loss function, similar to YOLOv5 (Jocher, 2020), consists of three main components: classification loss ($L_{\text{cls}}$), bounding box regression (BBR) loss ($L_{\text{BBR}}$), and object loss ($L_{\text{obj}}$). The overall loss function is expressed as:

$$L = \lambda_1 L_{\text{cls}} + \lambda_2 L_{\text{obj}} + \lambda_3 L_{\text{BBR}}, \tag{1}$$

where $\lambda_1$, $\lambda_2$, and $\lambda_3$ are balance coefficients that control the relative contributions of each loss term. These coefficients are fine-tuned to ensure the loss terms appropriately address the challenges of detecting small objects.

To handle objects of varying sizes, the object loss ($L_{\text{obj}}$) is further weighted across scales:

$$L_{\text{obj}} = 4.0 \times L_{\text{obj}}^{\text{small}} + 1.0 \times L_{\text{obj}}^{\text{medium}} + 0.4 \times L_{\text{obj}}^{\text{large}}, \tag{2}$$

where $L_{\text{obj}}^{\text{small}}$, $L_{\text{obj}}^{\text{medium}}$, and $L_{\text{obj}}^{\text{large}}$ represent the object losses for small, medium, and large objects, respectively. This scale-specific weighting prioritizes small object detection, addressing their higher difficulty in UAV imagery.

For bounding box regression, YOLOv7-T adopts the Complete IoU (CIoU) (Zheng et al., 2022) loss, which incorporates multiple factors – IoU, center point distance, and aspect ratio – to optimize bounding box alignment comprehensively:

$$L_{\text{CIoU}} = 1 - \text{IoU} + \frac{\rho^2(b, b^{\text{gt}})}{c^2} + \alpha v, \tag{3}$$

where $\rho(b, b^{\text{gt}})$ denotes the Euclidean distance between the center points of the predicted box $b$ and ground truth box $b^{\text{gt}}$, and $c$ is the diagonal length of the smallest enclosing box covering both $b$ and $b^{\text{gt}}$. IoU is defined as:

$$\text{IoU} = \frac{|B \cap B^{\text{gt}}|}{|B \cup B^{\text{gt}}|}, \tag{4}$$

where $B = (x, y, w, h)$ represents the predicted bounding box, and $B^{\text{gt}} = (x^{\text{gt}}, y^{\text{gt}}, w^{\text{gt}}, h^{\text{gt}})$ is the ground truth box. The term $v$ accounts for aspect ratio consistency between $b$ and $b^{\text{gt}}$:

$$v = \frac{4}{\pi^2}\left(\arctan\frac{w^{\text{gt}}}{h^{\text{gt}}} - \arctan\frac{w}{h}\right)^2, \tag{5}$$

$$\alpha = \frac{v}{1 - \text{IoU} + v}. \tag{6}$$

Here, $w$ and $h$ denote the width and height of the predicted box, while $w^{\text{gt}}$ and $h^{\text{gt}}$ represent those of the ground truth box.

### 2.3. Limitations of existing methods and motivation for the proposed model

While existing lightweight and real-time models for small object detection in aerial imagery have shown promising results, they still face notable limitations. For instance, HIC-YOLO (Tang et al., 2024) and PDWT-YOLO (Zhang, Xiong, et al., 2023) enhance small object detection by adding specialized output heads. However, they fail to adequately refine the feature extraction blocks to fully exploit the additional output heads. As a result, their performance is often constrained in high-density or cluttered aerial scenes where precision and feature differentiation are critical. Similarly, EdgeYOLO (Liu et al., 2023) reduces latency by limiting network width, but this compromise can lead to reduced accuracy, particularly in complex detection tasks involving small or occluded objects.

Recent models have introduced innovations to improve small object detection, but they remain insufficient for optimal performance. Drone-YOLO (Zhang, 2023) and LE-YOLO (Yue et al., 2024) propose advanced modules for feature fusion and backbone optimization. However, their PAFPN architectures are not fully optimized, with certain blocks remaining non-lightweight, making them less practical for edge deployment where computational efficiency is paramount. Likewise, PVswin-YOLOv8s (Tahir et al., 2024) leverages a Swin Transformer block to enhance global feature extraction, significantly improving accuracy for small and distant objects. However, the increased computational complexity of this approach makes deployment on resource-constrained devices, such as drones or embedded systems, challenging.

Moreover, while attention mechanisms such as CBAM and CA have been incorporated into models like PVswin-YOLOv8s and YOLOv5s-pp (Xu, Zheng, et al., 2023), they are not always well-suited to the unique challenges of small object detection in aerial imagery. These mechanisms often fail to adequately focus on spatial information, which is crucial for detecting objects that occupy only a few pixels in the image. To address this, our proposed model integrates CoordBlock, a spatially aware attention mechanism that embeds positional context directly into feature maps. This enables the model to better retain and emphasize spatial information, improving its ability to precisely localize small objects even in cluttered environments.

Another challenge faced by models such as YOLOv7-T (Wang et al., 2023) and other YOLO-based networks is their reliance on large datasets like COCO (Lin et al., 2014), which include a broad range of object scales but lack specialization for small object detection. Consequently, these models often underperform in detecting small objects in aerial imagery, where objects can be densely packed or occupy minimal pixel areas. While YOLOv7-T provides a strong foundation for real-time performance due to its streamlined architecture and fast inference, it

lacks specialized components tailored for small object detection.

To address these issues, our proposed model, LEAF-YOLO, builds upon YOLOv7-T while incorporating targeted improvements to enhance small object detection capabilities. A dedicated Small Object Detection Head is introduced, alongside optimized feature extraction blocks designed to capture fine-grained details and spatial context. Modules such as the MaxPooling—Ghost Convolution (MGC) and LEAF blocks are strategically integrated to maximize efficiency, ensuring high detection accuracy while maintaining a lightweight architecture for real-time edge deployment.

Additionally, to overcome limitations in bounding box regression (BBR), we propose a novel loss function called InnerShape-IoU. Unlike traditional loss functions like CIoU (Zheng et al., 2022) and WIoU (Tong et al., 2023), which focus on overlap and distance metrics, InnerShape-IoU adjusts the shape of predicted bounding boxes to better fit the true shape and size of objects. This is particularly beneficial for small and irregularly shaped objects in aerial imagery, as it reduces the discrepancy between ground truth and predicted boxes, improving both localization accuracy and robustness.

By combining lightweight architectural improvements, spatially focused attention mechanisms like CoordBlock, and the innovative InnerShape-IoU loss, LEAF-YOLO provides a holistic solution to the challenges of small object detection in UAV applications. These enhancements enable LEAF-YOLO to achieve superior accuracy and efficiency, ensuring real-time performance on edge devices while addressing the shortcomings of existing methods.

## 3. Methodology

With the above analysis, YOLOv7-T (Wang et al., 2023) is chosen as the baseline for development to create our proposed model, called LEAF-YOLO. The detailed architecture of our LEAF-YOLO is shown in Fig. 4:

- **Backbone**: Incorporates LEAF and MGC blocks for feature extraction.
- **Neck**: Uses SPPRFM, LEAF-T, and CoordBlock blocks to enhance features and reduce the loss of spatial information. Additionally, Partial Convolution (PConv) is applied at the end of each branch in the Neck before feature mapping to the head, reducing the number of parameters.
- **Head**: Includes an additional output head for small object detection, with each output head utilizing an Implicit Head.
- **Loss**: Binary Cross Entropy (BCE) is used for classification and object loss. For bounding box regression (BBR) loss, InnerShape-IoU is employed instead of CIoU.

LEAF-YOLO-N shares the same architecture as LEAF-YOLO while maintaining the same depth. However, it scales the network width by a factor of 0.5, meaning that in LEAF-YOLO-N, the number of channels in each layer is halved compared to LEAF-YOLO.

More details about the architectural improvements of LEAF-YOLO compared to YOLOv7-T will be presented in the following sections.

### 3.1. Small object detection head

In YOLO and object detection algorithms, the PAFPN and FPN architectures generate feature pyramids by reducing the spatial size of the input square image (width = height) through division by a power of 2. If the spatial size of the input image is $x$, the output size of the feature maps after passing through PAFPN is:

$$P_i = \frac{x}{2^i}, \quad i = 0, \ldots, n \tag{7}$$

In most current YOLO networks, the outputs P3, P4, and P5 are used. When the input image size is 640 × 640, the output sizes for P3, P4, and P5 are 80 × 80, 40 × 40, and 20 × 20, respectively. As per Table 1, for small objects (area $< 32^2$), passing through PAFPN results in the dimensions of small objects at P3, P4, and P5 being less than $4 \times 4$, $2 \times 2$, and $1 \times 1$, respectively. These extremely small dimensions make it challenging for the model to extract meaningful features from objects.

**Table 1**
Object size classifications based on the COCO dataset (Lin et al., 2014).

| Type of objects | Small | Medium | Large |
|---|---|---|---|
| Area | $< 32^2$ | $(32^2, 96^2]$ | $> 96^2$ |

Based on the above analysis and inspired by Tang et al. (2024), Zhang, Xiong, et al. (2023), Zhu et al. (2021), the proposed method incorporates a P2 head into the model. With an input image size of 640 × 640, the output spatial size of head P2 is 160 × 160, enabling the model to extract features more effectively using a higher-resolution feature map. Furthermore, to maintain the model's lightweight nature, the network branch containing the P2 head for small object detection includes more feature extraction blocks, while the branch containing the P5 head reduces these blocks. Additional details are illustrated in Fig. 4 and discussed further in Section 4.3.2.

### 3.2. PConv

According to Chen et al. (2023), regular convolution often creates feature maps that share high similarities among different channels. However, only a small subset of these channels fully utilizes this feature in a simple yet effective way, leading to redundancy. To address this, Partial Convolution (PConv) is employed in the LEAF architecture and the Neck part of our proposed model to simultaneously reduce computational redundancy and memory access.

Fig. 5 illustrates how PConv operates. Unlike regular convolution, which applies computations to all input channels, PConv performs convolution operations on only a subset of the input channels for spatial feature extraction, leaving the remaining channels untouched. Specifically, suppose a regular convolution with kernel size $k$ receives input $X$ with $c_{in}$ channels and produces output $Y \in \mathbb{R}^{c_{out} \times h \times w}$, where $c_{in} = c_{out} = c$. The FLOPs (Floating-point Operations Per Second) of this convolution are given by:

$$\text{FLOPs}_{\text{Conv}} = c^2 \times k^2 \times h \times w. \tag{8}$$

In contrast, PConv processes only $c_p$ channels from the input and generates an output with $c_p$ channels, where $c_p < c$. The FLOPs of PConv relative to regular convolution are:

$$\frac{\text{FLOPs}_{\text{PConv}}}{\text{FLOPs}_{\text{Conv}}} = \left(\frac{c_p}{c}\right)^2 = d^2. \tag{9}$$

In this article, we use $d = \frac{1}{4}$, which means that the FLOPs of PConv are only $\frac{1}{16}$ of those for regular convolution.

Additionally, PConv reduces memory access. The memory access for PConv is given by:

$$2 \times c_p \times h \times w + k^2 \times c \times c_p \approx 2 \times c_p \times h \times w, \tag{10}$$

which is only $\frac{1}{4}$ of the memory access required for regular convolution when $d = \frac{1}{4}$.

It is important to note that in PConv, the remaining $(c - c_p)$ channels are not discarded but are retained in the output feature map. This is because these channels are still useful for subsequent processing, allowing the extracted feature information to flow effectively through all channels. This approach ensures that the channel characteristics of the feature map are preserved, making PConv suitable for tasks like small object detection where detailed feature extraction is critical.

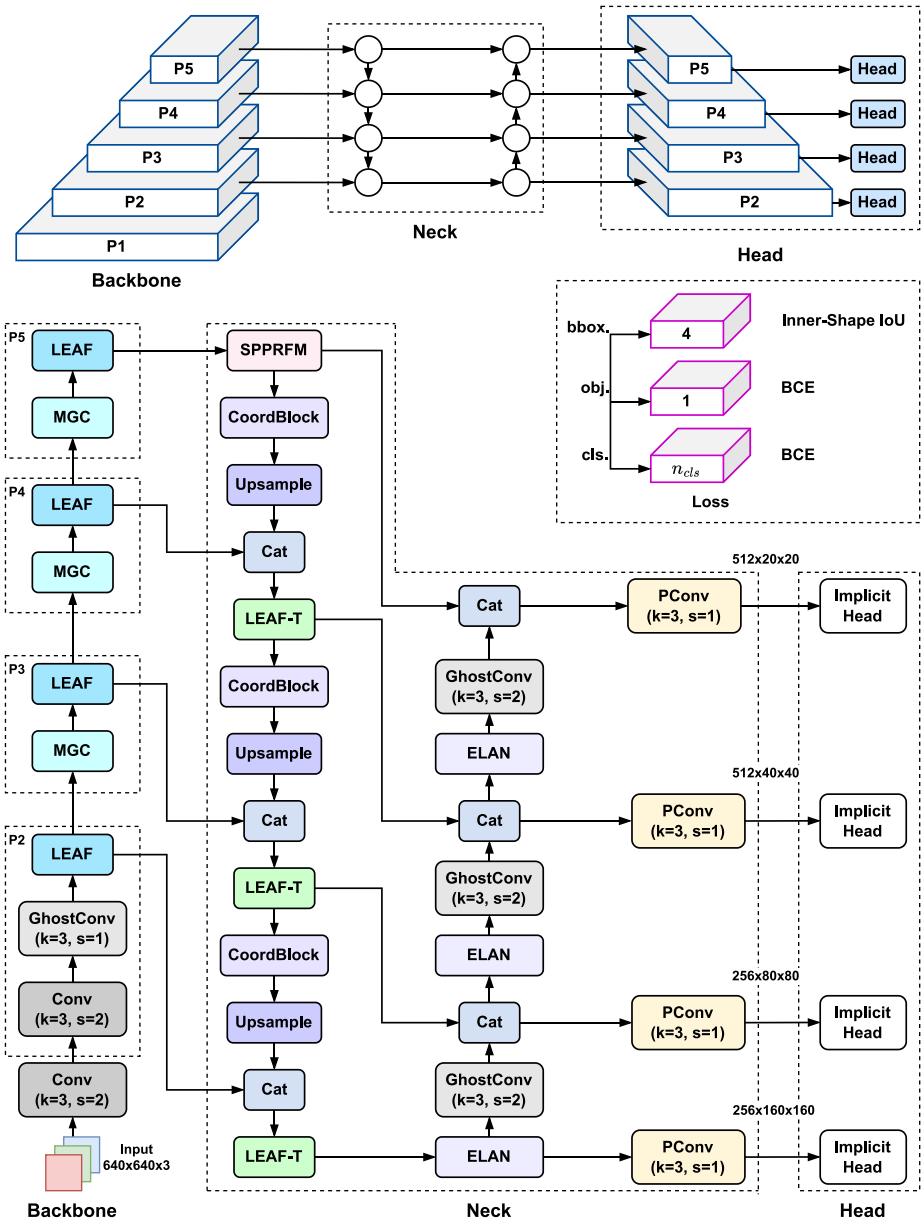


**Fig. 4.** LEAF-YOLO: Network model structure.

### 3.3. MGC block

In this section, the MaxPooling–Ghost Convolution (MGC) module is introduced as an effective method for downsampling feature maps. YOLOv5 (Jocher, 2020) and YOLOv8 (Jocher et al., 2023) perform downsampling using a convolutional layer with a kernel size ($k$) of 3 and a stride ($s$) of 2, the Backbone of YOLOv7-T employs a MaxPooling layer with $k = 2$ and $s = 2$ to downsample the feature map. Convolutional downsampling allows the network to extract and compress feature maps from the previous layer, while MaxPooling offers a parameter-free method of downsampling, retaining essential information from the previous feature map.

The MGC architecture combines these two techniques: MaxPooling and Ghost Convolution (Han et al., 2020)—to enhance feature extraction efficiency while preserving the information contained in the feature maps. As illustrated in Fig. 6, the first branch of MGC applies a MaxPooling layer with $k = 2$ and $s = 2$ to downsample the input feature map. This is followed by a $1 \times 1$ Convolution layer, which adjusts the network width and improves the extraction of feature maps within each channel.

The second branch begins by compressing the number of channels in the feature map using a $1 \times 1$ Convolution layer. The resulting feature map is then passed through a Ghost Convolution layer with $k = 3$ and $s = 2$, which reduces the spatial size of the feature map by half. Finally, the outputs of the two branches are concatenated to produce the final feature map.

Using $1 \times 1$ Convolution to compress the network width, combined with lightweight layers such as Ghost Convolution (as depicted in Fig. 5) and MaxPooling, ensures that MGC remains efficient without significantly increasing the number of parameters or computational complexity.

In summary, MGC provides a robust yet efficient approach to downsampling feature maps by integrating MaxPooling and Ghost Convolution. This method allows for effective feature extraction while

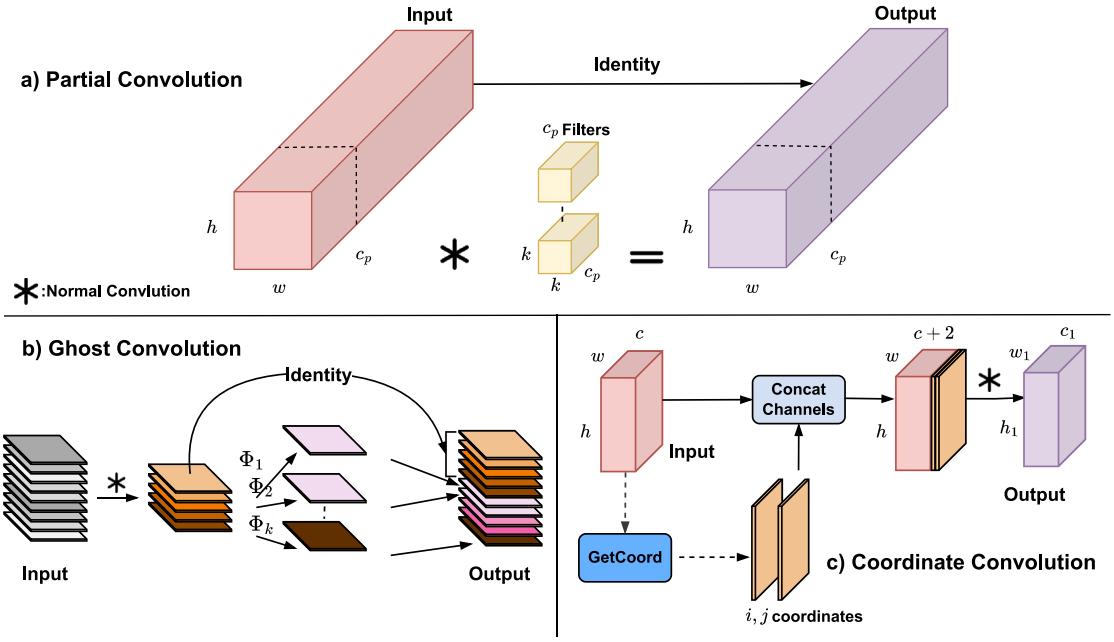


**Fig. 5.** (a) Partial Convolution only performs Convolution in the $c_p$ channel of input, the remaining part (c-$c_p$) is untouched. (b) In Ghost Convolution, only part of the feature map goes through Convolution, and then a simple linear transformation is applied to create the output feature. (c) Coordinate Convolution creates a feature map with c+2 channels from the input with c channels and then performs Convolution calculation.

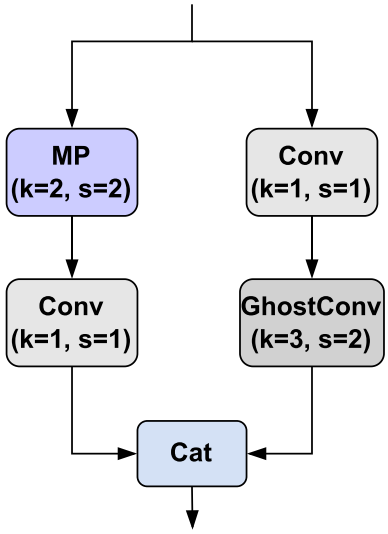


**Fig. 6.** Architecture of MGC block.

minimizing computational and parameter overhead. Such an approach is particularly advantageous in addressing the challenges of small object detection within high-density, cluttered backgrounds, as it retains critical spatial information that might otherwise be lost with standard downsampling techniques.

### 3.4. LEAF and LEAF-T block

This section clarifies the details of the LEAF and LEAF-T blocks. The LEAF block is used in the Backbone of the model to extract features from small objects more effectively by creating multiscale feature maps. In contrast, the LEAF-T block is utilized in the Neck part of the model to extract and enhance features obtained after the Backbone.

#### 3.4.1. LEAF

Inspired by the ELAN block of YOLOv7-T and the CSPBottleNeck block of CSPNet (Wang et al., 2020), the LEAF block, as shown in Fig. 7, is constructed with two consecutive components: LELAN and CSPRes2Block.

(a) **LELAN**: Similar to ELAN, LELAN's branches utilize 1 × 1 Convolution layers to compress the feature map along the channel direction, thereby reducing the network's width. In the second branch, subsequent 3 × 3 Convolution layers create a multiscale feature map to extract spatial features across multiple object sizes. The key difference between LELAN and ELAN is shown in Fig. 7. In the second branch, LELAN employs four 3 × 3 PConv layers instead of two sequential 3 × 3 Convolution layers. This modification improves network efficiency while reducing the model's parameter count and FLOPs. During the merging stage, LELAN concatenates the first branch with interleaved Convolution outputs of the second branch rather than all Convolution outputs. This approach creates a rich feature map that retains information from previous layers.

(b) **CSPRes2Block**: Small object recognition in aerial images captured by UAVs is characterized by dense and complex backgrounds, making multiscale representations critical. To address this, the CSPRes2Block replaces the CSPBottleNeck's Bottleneck block with a Res2Block (R2B), as shown in Fig. 8. This replacement enables efficient feature extraction without increasing computational costs.

The Res2Net paper (Gao et al., 2021) introduces a novel method for constructing feature extraction networks by splitting feature maps into smaller subsets and connecting multiple filter sets in a hierarchical residual-like manner. The Res2Block (R2B) incorporates residual-like connections within a single residual block, as illustrated in Fig. 9. After a 1 × 1 Convolution layer, the feature map is split into four subsets, denoted as $x_i$, where $i = 1 \ldots 4$. Each subset $x_i$ has the same spatial size but only $\frac{1}{4}$ of the input feature map's channels.

With the exception of $x_1$, each subset $x_i$ is processed by a 3 × 3 Convolution operator $F_i()$, producing output $y_i$. The feature

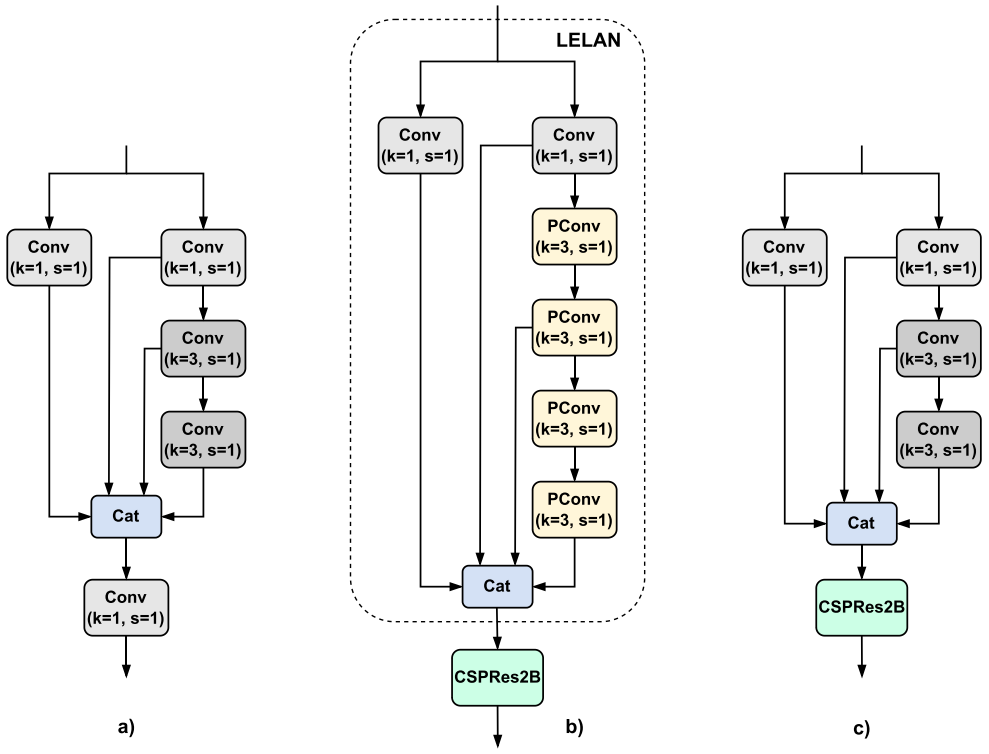


**Fig. 7.** (a) ELAN block in YOLOv7-T. (b) LEAF block (Ours). (c) LEAF-T block (Ours).

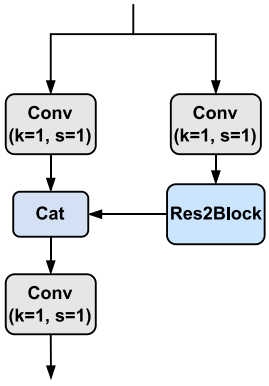


**Fig. 8.** The structure of CSPRes2Block.

subset $x_i$ is combined with the output of $F_{i-1}()$ and then passed to $F_i()$. To reduce parameters, the $3 \times 3$ Convolution layer for $x_1$ is omitted. The output $y_i$ is given by:

$$\mathbf{y}_i = \begin{cases} \mathbf{x}_i & \text{if } i = 1; \\ \mathbf{F}_i(\mathbf{x}_i) & \text{if } i = 2; \\ \mathbf{F}_i(\mathbf{x}_i + \mathbf{y}_{i-1}) & \text{if } 2 < i \leqslant 4. \end{cases} \tag{11}$$

Each $3 \times 3$ Convolution operator $F_i()$ receives feature information from all preceding subsets $x_j$ $(j < i)$. As these feature splits pass through the $3 \times 3$ Convolution operators, their receptive fields grow larger than their original subset size. This combinatorial effect ensures that the output of the R2B contains a wide range of receptive field sizes and scales. To effectively fuse information across these scales, all splits are concatenated and passed through a final $1 \times 1$ Convolution layer. This split-and-concatenation strategy forces the Convolutions to process features more effectively. By omitting the Convolution for the first split, the parameter count is reduced, further enhancing efficiency while reusing features. Replacing the Bottleneck with the Res2Block, the CSPRes2Block offers superior performance compared to the CSPBottleNeck by creating multiscale feature representations. This enhancement is particularly useful for detecting small objects in complex, cluttered scenes.

Overall, the LEAF architecture, combining LELAN and CSPRes2Block, optimizes feature extraction by balancing both network width and depth. LELAN enables efficient layer aggregation, capturing both low-level and high-level features essential for small object detection. Meanwhile, CSPRes2Block provides multiscale representation within each block to handle diverse object sizes and densities. Together, these components improve feature fusion, capturing critical details and contextual information without introducing redundant computations. This results in an efficient, lightweight backbone that supports real-time performance on edge devices, making LEAF ideal for detecting small objects in complex aerial images.

#### 3.4.2. LEAF-T

With the same purpose as LEAF, LEAF-T retains the architecture of YOLOv7-T's ELAN block and replaces the $1 \times 1$ Convolution layer after concatenation with the CSPRes2Block introduced earlier, as shown in Fig. 7. LEAF-T is used instead of LEAF in the Neck part to prevent an increase in the network's depth, which would otherwise lead to longer feed-forward times.

### 3.5. CoordBlock

In PAFPN architectures, the upsampling feature map is as important as the downsampling feature map. While downsampling compresses the spatial dimensions of the feature map by half, upsampling performs the reverse operation, doubling the spatial size of the feature map. This process can result in significant spatial information loss, particularly for features extracted from the Backbone.

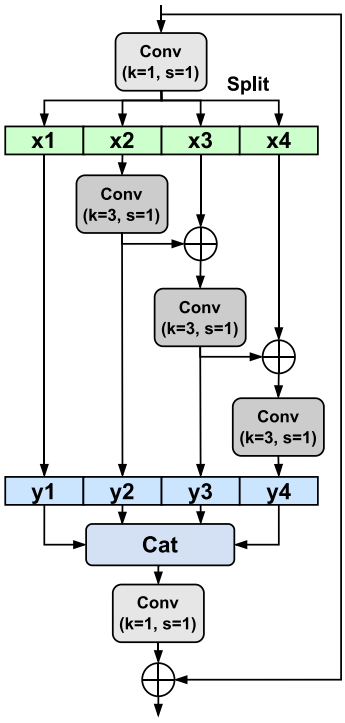


**Fig. 9.** Res2Block's architecture.

In the context of detecting objects in aerial images captured by UAVs, additional challenges such as rain, fog, and lighting conditions exacerbate the loss of information about small objects, leading to ineffective model predictions. To address this issue, the Coordinate Block (CoordBlock) is proposed. CoordBlock is integrated into the Neck of the model before each upsampling layer, replacing regular Convolution layers. It consists of two interconnected components: Coordinate Convolution (CoordConv) (Liu, Lehman,et al., 2018) to enhance spatial characteristics and Coordinate Attention (CoordAtt) (Hou et al., 2021) to help the model focus on spatial information.

### 3.5.1. CoordConv

The CoordConv method introduces coordinate channels to provide convolutional access to Cartesian spatial information, unlike regular Convolution. While maintaining the computational and parametric efficiency of standard Convolution, CoordConv enables the network to decide between complete translational invariance or varying levels of translational dependence based on the specific task. Translational invariance ensures that the network generates consistent responses regardless of how its inputs are translated in image space. This capability allows CoordConv to more accurately capture spatial information of targets while minimizing interference from factors such as angle and position transformations in multiscale targets.

CoordConv is constructed by extending standard Convolution with two additional channels that incorporate coordinate information. As illustrated in Fig. 5, the process adds $i$ and $j$ coordinate channels. Specifically: - The $i$ coordinate channel consists of an $h \times w$ rank-1 matrix, with the first row set to 0, the second row to 1, the third row to 2, and so on. - Similarly, the $j$ coordinate channel fills constants along the columns.

These coordinate values are then linearly scaled to the range $[-1, 1]$. A third additional channel, the $r$ coordinate, is computed to encode the distance from the center of the feature map:

$$r = \sqrt{\left(i - \frac{h}{2}\right)^2 + \left(j - \frac{w}{2}\right)^2}. \tag{12}$$

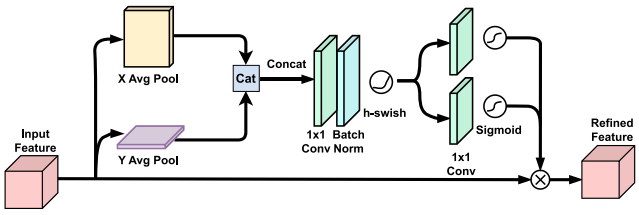


**Fig. 10.** Coordinate Attention mechanism.

The $i$, $j$, and $r$ channels are concatenated with the input feature map, providing enhanced spatial context to the network. CoordConv improves the perception of spatial information, offering the flexibility to adapt translational invariance based on the network's learning requirements. This method operates similarly to residual connectivity, enhancing the model's generalization capabilities.

In terms of parameters, a standard Convolution with a kernel size $k$ and $c$ channels contains $c^2 \times k^2$ parameters. In comparison, a CoordConv layer includes weights of $(c + 2) \times c \times k^2$, resulting in only a minimal increase in parameters. The additional bias parameters introduced by CoordConv are negligible, ensuring that the computational overhead remains low.

### 3.5.2. CoordAtt

The generation of feature maps during network feature extraction relies heavily on spatial and channel characteristics. While attention mechanisms like Squeeze-and-Excitation (SE) (Hu, Shen, & Sun, 2018) may lose spatial location information during global encoding, and Bottleneck Attention Module (BAM) (Park, Woo, Lee, & Kweon, 2018) and Convolution Block Attention Module (CBAM) (Woo et al., 2018) may fail to capture comprehensive range dependence information, CoordATT provides an effective solution. CoordATT addresses these issues by decomposing global encoding from channel attention into one-dimensional parallel encodings along horizontal and vertical directions. By using a coordinate attention mechanism, CoordATT aggregates position-aware features in each direction for an input feature map. This results in location-aware features with cross-channel dependencies along the feature map in each direction.

CoordATT operates in two main stages: embedding information and generating attention. As shown in Fig. 10, CoordATT replaces global pooling with directional encoding pooling along the horizontal and vertical axes. Pooling kernels of size $(H, 1)$ and $(1, W)$ are used to encode the input feature map $X \in \mathbb{R}^{C \times H \times W}$. The feature map is then processed along the $x$ and $y$ axes, with spatial location information combined across these dimensions. The resulting feature maps, $z_c^h$ and $z_c^w$, are expressed as follows:

$$Z_c = \frac{\sum_{i=1}^{H} \sum_{j=1}^{W} x_c(i, j)}{H \times W}, \tag{13}$$

$$z_c^h(h) = \frac{1}{W} \sum_{j=1}^{W} x_c(h, j), \tag{14}$$

$$z_c^w(w) = \frac{1}{H} \sum_{i=1}^{H} x_c(i, w). \tag{15}$$

CoordATT captures the long-term relationship between spatial orientations and retains positional data along the complementary spatial axis. The feature maps $z_c^h$ and $z_c^w$ capture the overall sensory field of the input feature map while preserving precise spatial location details. These maps are then combined along the spatial dimension. Using a $1 \times 1$ Convolution, the number of channels is reduced to produce the attention feature map $f$, as defined in:

$$f = \delta(F_1[z_c^h, z_c^w]), \tag{16}$$

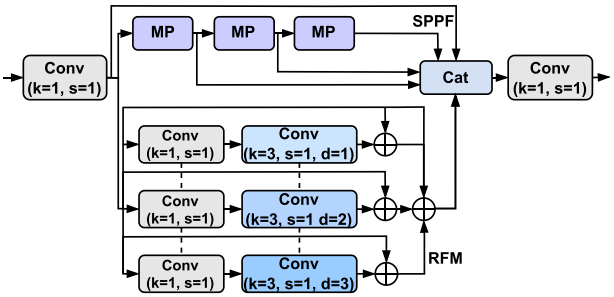


**Fig. 11.** The structure of SPPRFM.

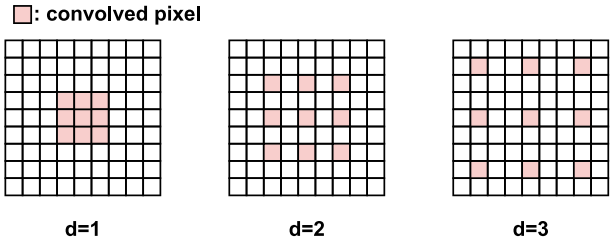


**Fig. 12.** Illustration of dilated convolution with different dilation rates (d). As the dilation rate increases, the receptive field expands, allowing the convolution to capture a broader context without increasing the kernel size.

where [.,.] represents the concatenation operation along the spatial dimension, and $\delta$ refers to the hswish activation function:

$$\delta(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6}. \tag{17}$$

The feature map $f \in \mathbb{R}^{\frac{C\times(H+W)}{r}}$ encodes spatial information in both horizontal and vertical directions using batch normalization and nonlinear transformations. The tensor $f$ is then split into two separate tensors, $f^h$ and $f^w$, along the spatial dimension. Two $1 \times 1$ Convolution layers, $F_{1h}$ and $F_{1w}$, transform $f^h$ and $f^w$ into the same number of channels as the input feature map $X$:

$$g_c^h = \sigma(F_{1h}(f^h)), \tag{18}$$

$$g_c^w = \sigma(F_{1w}(f^w)), \tag{19}$$

where $\sigma$ denotes the sigmoid function. The outputs $g_c^h$ and $g_c^w$ are combined with the original input feature map to produce the final result:

$$y_c(i,j) = x_c(i,j) \times g_c^h(i) \times g_c^w(j). \tag{20}$$

CoordATT integrates location information into channel attention, enabling smaller networks to focus on broader regions while avoiding substantial computational overhead. This makes it well-suited for lightweight model requirements.

As a result, CoordATT is integrated after CoordConv to form the CoordBlock. By combining CoordConv's ability to enhance spatial characteristics with CoordATT's attention mechanism, the CoordBlock improves the model's understanding of spatial location information, addressing the issue of incomplete spatial details effectively.

### 3.6. SPPRFM

The connection between the Backbone and Neck in the PAFPN architecture is critical, as it determines whether the features extracted by the Backbone can be effectively retained and enhanced when passed to the Neck. In the specific problem of detecting small objects, even if the Backbone successfully extracts multiscale feature maps, their utility diminishes without a robust mechanism to transmit them through the Neck. To address this issue, the Spatial Pyramid Pooling Receptive Field Module (SPPRFM) is proposed. SPPRFM comprises two main branches: the first branch utilizes the Spatial Pyramid Pooling Faster (SPPF) (Jocher, 2020) architecture for efficient feature extraction with minimal computation and fast feed-forward time. The second branch employs the Receptive Field Module (RFM), which establishes a parallel spatial feature extraction mechanism to minimize the loss of spatial information.

In the first branch of SPPRFM, shown in Fig. 11, the SPPF architecture integrates three consecutive MaxPooling layers with a kernel size of 5 to capture features across multiple scales. While (Jocher, 2020) highlights that SPPF is twice as fast as SPP while maintaining similar efficiency, two key issues remain. First, consecutive pooling can propagate incorrect information and lead to the loss of small target features. Second, the direct combination of pooled features overlooks the complex relationships between features at different scales, hindering the detection of multiscale targets and high-resolution images, which ultimately affects detection accuracy.

To mitigate these limitations of SPPF, a lightweight RFM module is added to the second branch to complement the first branch. Inspired by the Trident Block from (Li, Chen, Wang, & Zhang, 2019), the RFM employs four branches with different dilation rates, as illustrated in Fig. 12, to capture multiscale information. These branches share weights, with the primary distinction being the unique receptive fields of the dilated convolutions. This shared-weight design prevents overfitting and optimizes parameter usage. Following a $1 \times 1$ Convolution to extract channel-wise features, the multibranch RFM applies dilation rates of 1, 2, and 3 in separate branches, all using fixed $3 \times 3$ Convolutions. Compared to the Trident Block, our RFM introduces a residual connection to mitigate exploding and vanishing gradient issues during training. Additionally, the three $1 \times 1$ Convolutions typically found after the dilated Convolution in each branch have been removed to reduce the parameter count. Finally, the branches are combined to synthesize features and weights from each branch, producing a unified feature representation.

By integrating the SPPF and RFM modules, SPPRFM establishes a multiscale architecture with both serial and parallel connections. This design enables the multiscale features extracted by the LEAF block in the Backbone to be transmitted to the Neck efficiently, minimizing feature loss and ensuring effective enhancement of spatial and channel information.

### 3.7. Inner-Shape IoU

Bounding box regression (BBR) loss is a crucial component of the detector's localization branch and is essential for object detection tasks. Many existing BBR methods only consider the geometric relationship between the Ground Truth (GT) box and the predicted bounding box, computing the loss based on their relative position and shape. However, they often neglect inherent properties such as the shape and scale of the bounding boxes. This limitation is particularly problematic in scenarios involving small targets in aerial images. Current IoU loss methods (Gevorgyan, 2022; Rezatofighi et al., 2019; Zheng et al., 2022) cannot dynamically adjust to create bounding boxes suitable for objects at different scales. To address these shortcomings and fully exploit the potential of BBR loss, Inner-Shape IoU is proposed to control the scale and shape of the auxiliary bounding boxes, resulting in a more accurate BBR.

First, Inner-IoU (Zhang, Xu, & Zhang, 2023) introduces a mechanism for scaling the size of bounding boxes. The GT box and anchor box are denoted as $B^{gt}$ and $B$, respectively, as shown in Fig. 13. The center points of the GT box and the inner GT box are represented by $(x_c^{gt}, y_c^{gt})$, while $(x_c, y_c)$ represents the center points of the anchor and inner anchor. The width and height of the GT box are denoted as $w^{gt}$ and

$h^{gt}$, respectively, while the width and height of the anchor box are represented as $w$ and $h$. The *ratio* parameter, which serves as the scaling factor within the range [0.5, 1.5], allows for fine-tuning how closely the anchor box fits the GT box. This adjustment enables better handling of high-density or sparsely distributed objects.

- **When** $ratio < 1$**:** The bounding boxes are scaled down (as shown in (a) of Fig. 13). This is helpful in high-density environments where multiple objects are close together, avoiding overlaps with neighboring objects.
- **When** $ratio > 1$**:** The bounding boxes are scaled up (as shown in (b) of Fig. 13). This scaling helps in cases where objects are more sparsely distributed, ensuring that the model focuses on a broader area around the object.

Calculation details are through the following steps:

- **Adjust GT box size:**

$$b_l^{gt} = x_c^{gt} - \frac{w^{gt} * ratio}{2}, b_r^{gt} = x_c^{gt} + \frac{w^{gt} * ratio}{2} \tag{21}$$

$$b_t^{gt} = y_c^{gt} - \frac{h^{gt} * ratio}{2}, b_b^{gt} = y_c^{gt} + \frac{h^{gt} * ratio}{2} \tag{22}$$

- **Adjust anchor box size:**

$$b_l = x_c - \frac{w * ratio}{2}, b_r = x_c + \frac{w * ratio}{2} \tag{23}$$

$$b_t = y_c - \frac{h * ratio}{2}, b_b = y_c + \frac{h * ratio}{2} \tag{24}$$

The formulas provided for adjusting the bounding box size (Eqs. (21), (22), (23), (24)) ensure that the scaling is symmetrical around the center of the object. This symmetrical scaling preserves the object's spatial context, which is important in aerial imagery where the relative position of objects (such as vehicles or pedestrians) carries critical information.

After scaling, the Inner-IoU computes the intersection area and the union area between the GT and anchor boxes using standard IoU principles. The use of a scaled version of the GT and anchor boxes adds a degree of flexibility, particularly in complex aerial scenes, where object sizes can be inconsistent across different areas of an image. By adjusting the box sizes, the model can adapt to both small and large objects, ensuring more precise localization:

- **Intersection Calculation (Eq. (25)):** This step calculates the overlapping region between the scaled GT and anchor boxes.

$$inter = (min(b_r^{gt}, b_r) - max(b_l^{gt}, b_l)) * (min(b_b^{gt}, b_b) - max(b_t^{gt}, b_t)) \tag{25}$$

- **Union Calculation (Eq. (26)):** The union is derived similarly to traditional IoU, but using the scaled box areas. This step ensures that the total object area is appropriately captured, improving the model's accuracy when detecting objects of various scales.

$$union = (w^{gt} * h^{gt}) * (ratio)^2 + (w * h) * (ratio)^2 - inter \tag{26}$$

- **Calculate Inner-IoU:**

$$IoU^{inner} = \frac{inter}{union} \tag{27}$$

The Inner-IoU (Eq. (27)) is defined as the ratio of the intersection to the union of the scaled boxes. This is a refined IoU metric that focuses on how well the anchor box fits the core area of the object, rather than just the outer boundaries. The design of Inner-IoU is particularly suited for aerial images, where detecting smaller or partially occluded objects can be challenging. By focusing on the "inner" region of the bounding box, this loss function reduces the likelihood of false positives caused by overlapping objects.

Next, Shape-IoU (Zhang & Zhang, 2024) is used to focus on the shape of the bounding box and further refines the loss by incorporating

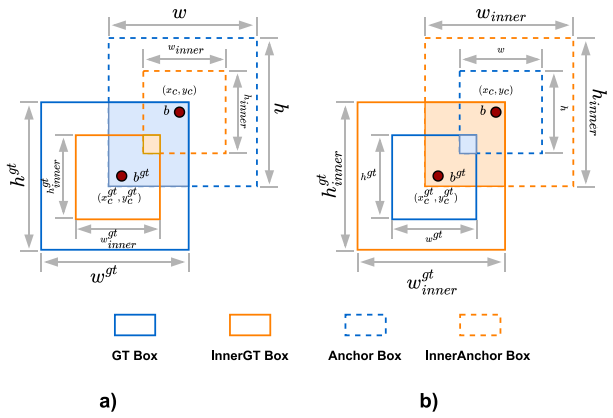


**Fig. 13.** InnerShape-IoU. (a) The $ratio < 1$. (b) The $ratio > 1$.

a distance metric between the shapes of the GT and anchor boxes (Eq. (28)). This distance measure ensures that the model not only aligns the anchor box with the GT box in terms of position but also in terms of shape. This is critical in aerial imagery, where objects such as cars, trucks, and pedestrians can have different aspect ratios. The formula for Shape-IoU can be driven from Fig. 13.

- **Shape Distance (Eq. (28)):** This calculates the normalized distance between the centers of the GT and anchor boxes, penalizing cases where the shapes are mismatched.

$$distance^{shape} = \frac{(x_c - x_c^{gt})^2}{c^2} + \frac{(y_c - y_c^{gt})^2}{c^2} \tag{28}$$

where $c$ is the diagonal distance of the minimum enclosing bounding box between $B$ and $B^{gt}$.

- **Shape Cost (Eq. (29)):** This cost function ensures that the shapes of the bounding boxes are consistent with the objects being detected, which is especially useful in aerial imagery where varying object sizes and shapes can lead to misdetections.

$$\Omega^{shape} = \sum_{t=w,h} (1 - e^{-\omega_t})^{\theta}, \theta = 4 \tag{29}$$

where:

$$\begin{cases} \omega_w = \frac{|w - w^{gt}|}{\max(w, w^{gt})} \\ \omega_h = \frac{|h - h^{gt}|}{\max(h, h^{gt})} \end{cases} \tag{30}$$

Finally, the InnerShape-IoU loss (Eq. (31)) combines these terms to penalize both poor overlaps (via $IoU^{inner}$), misalignments in the center and shape (via $distance^{shape}$), and shape mismatches (via $\Omega^{shape}$).

$$L_{InnerShape-IoU} = 1 - IoU^{inner} + distance^{shape} + 0.5 \times \Omega^{shape} \tag{31}$$

So, by balancing these penalties, the loss encourages the model to predict bounding boxes that accurately match the core position, shape, and aspect ratio of objects in aerial imagery.

**Discussion:** The Inner-Shape IoU loss is specifically designed to tackle the challenges of aerial imagery, where objects may be small, densely packed, or irregularly shaped. By incorporating both inner box scaling and shape alignment, this loss function provides several advantages:

- **Small Object Detection:** The use of inner box scaling helps the model better localize smaller objects, which are common in aerial datasets such as VisDrone2019 and TinyPerson.
- **High-Density Object Detection:** In crowded scenes, the scaling mechanism reduces the chances of anchor boxes overlapping with neighboring objects, improving precision.

**Table 2**
Comparison between COCO and Visdrone2019-DET.

| | COCO Dataset | Visdrone2019-DET |
|---|---|---|
| Maximum resolution | 640 × 640 | 2000 × 1500 |
| Object per image | 7 | 53 |

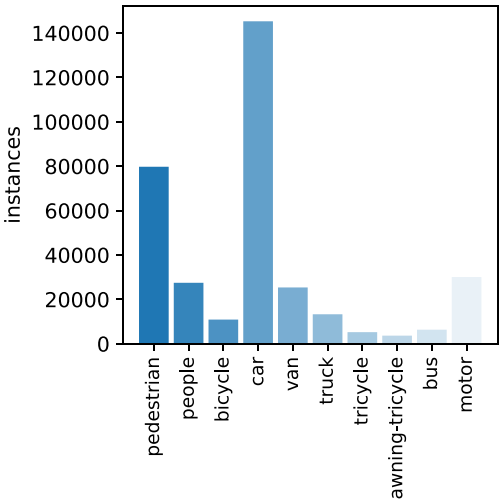


**Fig. 14.** Number objects per class in VisDrone2019-DET.

- **Improved Shape Matching:** The shape distance and cost terms ensure that the bounding box shape is more aligned with the actual object, reducing errors caused by mismatched aspect ratios.

This design ultimately leads to better overall performance, especially when dealing with complex aerial images with diverse object distributions.

## 4. Experimental results

In this section, datasets, parameters configuration, ablation studies to obtain the final model, as well as benchmark results on the VisDrone2019-DET and TinyPerson datasets are presented.

### 4.1. Dataset

#### 4.1.1. Visdrone2019-DET

VisDrone2019-DET (Du et al., 2019), developed by AISKYEYE at Tianjin University's Lab of Machine Learning and Data Mining, is a comprehensive benchmark designed to facilitate the integration of drone technology and visual perception. It includes 288 video clips consisting of 261,908 frames and 10,209 static images captured by drones from various angles, distances, lighting conditions, and locations along thousands of kilometers in China. The 10,209 static images, with resolutions ranging from 950 × 540 pixels to 2000 × 1500 pixels, were processed to yield 2.6 million target annotations. These are split into 6471 training samples, 548 validation samples, 1610 testing samples, and 1580 images for challenge testing. This article focuses exclusively on the labeled sets, including training, validation, and testing. The challenge test set is excluded as it lacks labels for evaluation.

More detailed information about the dataset is provided in Fig. 14 and Table 2. The dataset comprises 10 classes, with the class "car" representing the majority of objects, accounting for nearly 42% of the total objects. In contrast, the "awning tricycle" (AwTri) class constitutes less than 1% of the total samples. When compared to the COCO dataset (Lin et al., 2014), which has a maximum resolution of 640 × 640, the VisDrone2019-DET dataset boasts a significantly higher maximum resolution of 2000 × 1500. Additionally, while the COCO dataset averages 7 objects per image, VisDrone2019-DET contains an average of 53 objects per image. Moreover, nearly 10% of VisDrone2019-DET's objects have annotated bounding box areas smaller than 3 pixels.

This poses a considerable challenge for object detection algorithms that are typically developed and optimized for the COCO dataset. Consequently, specialized modifications to the model architecture are required to address the problem of small object detection effectively.

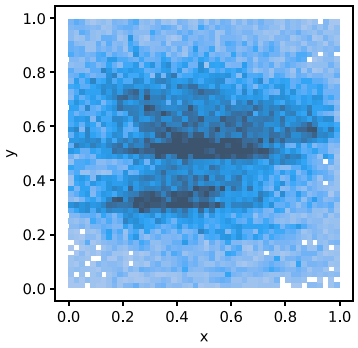


**Fig. 15.** Location distribution of objects in TinyPerson Dataset.

#### 4.1.2. TinyPerson

The TinyPerson dataset (Yu, Gong, et al., 2020) was specifically created for detecting tiny individuals in UAV-captured imagery, presenting several unique challenges due to its composition and object distribution. The dataset comprises high-resolution images collected every 50 frames from various online video sources, resulting in a set of 1610 labeled images and 759 unlabeled images, annotated with a total of 72,651 bounding boxes. In this work, we utilize only the labeled portion: 794 images for training and 816 images for validation and testing, in alignment with the original dataset distribution.

One of the key challenges of TinyPerson is its categorization of human objects into two distinct classes: "sea person" and "earth person", differentiating individuals on land from those in water. Additionally, as illustrated in Fig. 15, the object distribution within each image is highly dispersed, with a notable clustering of objects toward the center. This uneven distribution poses significant challenges for model training and validation, particularly in cases where objects overlap, leading to bounding box interference and affecting detection accuracy.

These dataset-specific characteristics necessitate careful model design and training strategies to effectively address the high object density, uneven distribution, and occlusion challenges posed by TinyPerson.

### 4.2. Experimental parameter configuration

#### 4.2.1. Implementation configuration

(a) **Dataset.** As mentioned above, VisDrone2019-DET and TinyPerson are the two datasets used in this study. However, the primary results and ablation studies in Sections 4.3 and 4.4 focus on VisDrone2019-DET. For TinyPerson, this dataset is used and evaluated only in Section 4.5.

(b) **Training.** The experimental setup for this study includes Ubuntu 20.04 OS, one NVIDIA GeForce RTX 3090 GPU with 25 GB of VRAM, CUDA 11.4, PyTorch 1.11.0, and pycocotools 2.0.7. The training processes utilize the repositories and formats of YOLOv5 (Jocher, 2020) and YOLOv7 (Wang et al., 2023). Before training, the anchors are recalculated on VisDrone2019-DET using K-means clustering to ensure the most suitable recall for each dataset. All models are trained using 640 × 640 input images, SGD optimizers, a learning rate of 0.01, 800 training iterations, and a batch size of 16. Further training parameters, detailed in

**Table 3**
Hyperparameter Settings for training LEAF-YOLO.

| $hsv_h$ | $hsv_s$ | $hsv_v$ | Translate | Scale | fliplr | Mosaic | Mixup | Pastein |
|---|---|---|---|---|---|---|---|---|
| 0.015 | 0.7 | 0.4 | 0.1 | 0.5 | 0.5 | 1 | 0.1 | 0.05 |

Table 3, are manually optimized and selected. Notably, all models are trained from scratch using Automatic Mixed Precision (AMP) and do not employ transfer learning or knowledge distillation techniques, ensuring an unbiased and honest evaluation. Additional hyperparameters and training configurations can be found at https://github.com/highquanglity/LEAF-YOLO.

(c) **Inference.** After training, all models are evaluated using PyTorch with weights in floating-point 16 precision (fp16) to measure accuracy and latency per image on a single NVIDIA GeForce RTX 3090 GPU with a batch size of 1. The inference performance is first tested on the RTX 3090 to establish a baseline for comparison. In Section 4.4.3, the LEAF-YOLO and LEAF-YOLO-N models are deployed on an edge computing device, the NVIDIA Jetson AGX Xavier, which features 512 CUDA cores operating in MAXN mode. Unlike the high-performance RTX 3090, the Jetson AGX Xavier is specifically designed for edge computing, with significantly lower computational power, resulting in reduced raw throughput. The performance on the Jetson AGX Xavier is measured in terms of frames per second (FPS), with the model optimized using TensorRT 8.5.2 and converted to fp16 format. This comparison underscores the trade-offs between leveraging high-performance GPUs for training and addressing the challenges of deploying models in resource-constrained edge environments.

#### 4.2.2. Performance metrics

This study adopts Average Precision (AP) as the primary metric to evaluate model performance, where a higher AP value indicates a more accurate detection capability. The AP is computed from the area under the Precision-Recall (PR) curve, as expressed in Eq. (32):

$$AP_{IoU} = \int_0^1 \mathrm{P}(R)\, dR \tag{32}$$

Here, $IoU$ represents the Intersection Over Union between the predicted and ground truth bounding boxes. $P$ denotes Precision, and $R$ denotes Recall. Precision and Recall are calculated using True Positives (TP), False Positives (FP), and False Negatives (FN), as shown in Eqs. (33) and (34):

$$P = \frac{TP}{TP + FP} \tag{33}$$

$$R = \frac{TP}{TP + FN} \tag{34}$$

In this study, we use the COCO API via the pycocotools 2.0.7 library to compute accuracy metrics. Specific metrics include:

- $AP_{.50}$: AP at an IoU threshold of 0.5 for all classes.
- $AP_{.75}$: AP at an IoU threshold of 0.75 for all classes.
- $AP_{.50:.95}$: The average AP across IoU thresholds from 0.50 to 0.95 with a step size of 0.05.
- $AP_S$, $AP_M$, and $AP_L$: AP at IoU thresholds of 0.5 to 0.95, calculated for small, medium, and large objects based on their areas as defined in Table 1.

To evaluate model complexity, additional metrics are considered:

- #Param.(M): The total number of parameters in the model (million).
- Floating Point Operations per Second (FLOPs): The number of mathematical operations performed by the model.

**Table 4**
Results of ablation experiments highlighting the contribution of each LEAF-YOLO component.

| # | Model | #Param.(M) | FLOPs(G) | $AP^{val}_{.50}$(%) |
|---|---|---|---|---|
| | base | 6.0 | 13.1 | 37.2 |
| 1 | +P2+PConv | 4.5 | 18.9 | 41.2 |
| 2 | +MGC+GhostConv | 5.0 | 20.2 | 42.4 |
| 3 | +LEAF | 4.58 | 20.7 | 45.0 |
| 4 | +LEAF-T | 4.62 | 21.2 | 45.8 |
| 5 | +SPPRFM | 4.27 | 20.9 | 46.3 |
| 6 | +CoordBlock | 4.28 | 20.9 | 46.6 |
| **7** | +InnerShape-IoU | 4.28 | 20.9 | 48.3 |

- Latency$^{fp16.pt}_{b=1}$(ms): The per-image latency of the model, measured in milliseconds, using PyTorch with fp16 weights and a batch size of 1.

### 4.3. Ablation study on VisDrone2019-DET-val

#### 4.3.1. Comprehensive analysis for model improvement

As mentioned in the Methodology, YOLOv7-T (Wang et al., 2023) was chosen as the baseline to create LEAF-YOLO. The impact of the added modules and blocks on the number of parameters (#Param.), model operations (FLOPs), and model accuracy on VisDrone2019-DET-val is shown in Table 4. The addition of a P2 head for small object detection and PConv reduces the number of parameters while increasing FLOPs. However, this trade-off yields significant improvements, with $AP_{.50}$ rising from 37.2% at baseline to 41.2%.

Next, replacing MaxPooling in the Backbone with MGC and substituting normal Convolution in the Neck with Ghost Convolution prevents a significant increase in parameters, while $AP_{.50}$ improves further to 42.4%, a 1.2% gain. Steps (**#3**) and (**#4**) demonstrate the superior feature extraction and enhancement capabilities of LEAF and LEAF-T. Compared to ELAN in step (**#2**), LEAF and LEAF-T in (**#4**) achieve higher accuracy while utilizing fewer parameters.

Step (**#5**) highlights that the SPPRFM architecture optimizes model complexity more effectively than SPP-YOLOv7 (Wang et al., 2023). In step (**#6**), replacing Normal Convolution in the Neck with the Coordinate Block before the upsampling feature map reduces spatial information loss while incurring minimal additional parameter cost. Finally, to complete LEAF-YOLO, InnerShape-IoU is introduced in step (**#7**), replacing CIoU, which further improves accuracy to 48.3% $AP_{.50}$ without increasing the parameter count.

To provide more detailed insights, the following sections analyze and compare the modules added at each step to explain their selection and impact.

#### 4.3.2. Comparative study on integrating small detection head to PAFPN

This study evaluates the impact of adding a small detection head and adjusting the number of ELAN blocks in the PAFPN of YOLOv7-T (base) to align with the improvements made from the baseline to our proposed model.

In Table 5, the Output Head configurations are defined as follows:

- **[p3, p4, p5] (base)**: Includes 1 ELAN block in branch P2 (1 in Backbone), 2 ELAN blocks in branch P3 (1 in Backbone, 1 in Neck), 3 ELAN blocks in branch P4 (1 in Backbone, 2 in Neck), and 2 ELAN blocks in P5 (1 in Backbone, 1 in Neck).
- **[p2, p3, p4, p5]**: Includes 2 ELAN blocks in branch P2 (1 in Backbone, 1 in Neck), 3 ELAN blocks in branch P3 (1 in Backbone, 2 in Neck), 3 ELAN blocks in branch P4 (1 in Backbone, 2 in Neck), and 2 ELAN blocks in P5 (1 in Backbone, 1 in Neck).
- **[p2, p3, p4, p5] + PConv (#1)**: Includes 3 ELAN blocks in branch P2 (1 in Backbone, 2 in Neck), 3 ELAN blocks in branch P3 (1 in Backbone, 2 in Neck), 3 ELAN blocks in branch P4 (1 in Backbone, 2 in Neck), and 1 ELAN block in P5 (1 in Backbone). Additionally, replaces the $3 \times 3$ Convolution at the end of each branch in the Neck with PConv.

**Table 5**
Comparison on restructure PAFPN with YOLOv7-T as a baseline.

| Output head | No.ELANs per branch | | | | #Param.(M) | FLOPs(G) | $AP^{val}_{.50}$(%) |
|---|---|---|---|---|---|---|---|
| | P2 | P3 | P4 | P5 | | | |
| [p3,p4,p5] (base) | 1 | 2 | 3 | 2 | 6.0 | 13.1 | 37.2 |
| [p2,p3,p4,p5] | 2 | 3 | 3 | 2 | 6.43 | 20.9 | 40.8 |
| **[p2,p3,p4,p5] +PConv (#1)** | **3** | **3** | **3** | **1** | **4.45** | **18.9** | **41.2** |

**Table 6**
Comparison of downsampling methods.

| Module | #Param.(M) | FLOPs(G) | $AP^{val}_{.50}$(%) |
|---|---|---|---|
| MP+Conv (#1) | 4.5 | 18.9 | 41.2 |
| MP+MP | 4.1 | 17.8 | 40.9 |
| Conv+Conv | 5.2 | 20.3 | 41.7 |
| **MGC+GhostConv (#2)** | **5.0** | **20.2** | **42.4** |

**Table 7**
Performance comparison of detectors with different SPP structure.

| Module | #Param.(M) | FLOPs(G) | $AP^{val}_{.50}$(%) |
|---|---|---|---|
| SPP-YOLOv7-T (Wang et al., 2023) (#4) | 4.62 | 21.2 | 45.8 |
| SPPCSPC (Wang et al., 2023) | 5.87 | 22.2 | 46.3 |
| GhostSPPCSPC (Wang et al., 2023) | 5.0 | 21.5 | 46.0 |
| SPP-YOLOv3 (Redmon & Farhadi, 2018) | 4.26 | 20.9 | 45.5 |
| SPPF (Jocher, 2020) | 4.26 | 20.9 | 45.8 |
| **SPPRFM (#5)** | **4.27** | **20.9** | **46.3** |

**Table 8**
Impact of different pre-upsampling methods on model accuracy.

| Module | #Param.(M) | FLOPs(G) | $AP^{val}_{.50}$(%) |
|---|---|---|---|
| Conv (#5) | 4.27 | 20.9 | 41.2 |
| Conv+SE (Hu et al., 2018) | 4.29 | 20.9 | 40.9 |
| Conv+CBAM (Woo et al., 2018) | 4.29 | 20.9 | 41.7 |
| **CoordBlock (#6)** | **4.28** | **20.9** | **42.4** |

**Table 9**
Performance comparison of bounding box regression loss algorithms on LEAF-YOLO.

| Method | $AP^{val}_{.50}$(%) | $AP^{val}_{.50:.95}$(%) |
|---|---|---|
| CIoU (Zheng et al., 2022) (#6) | 46.6 | 27.3 |
| SIoU (Gevorgyan, 2022) | 47.0 | 27.7 |
| GIoU (Rezatofighi et al., 2019) | 46.7 | 27.4 |
| EIoU (Yang, Wang, & Li, 2021) | 47.0 | 27.8 |
| WIoU (Tong et al., 2023) | 47.2 | 27.9 |
| Shape-IoU (Zhang & Zhang, 2024) | 47.3 | 27.9 |
| **InnerShape-IoU (#7)** | **48.3** | **28.2** |

**Table 10**
Impact of varying the *ratio* parameter on model accuracy.

| Method | $AP^{val}_{.50}$(%) | $AP^{val}_{.50:.95}$(%) |
|---|---|---|
| Shape-IoU | 47.3 | 27.9 |
| InnerShape-IoU(ratio=0.95) | 47.0 | 27.7 |
| InnerShape-IoU(ratio=1.05) | 47.5 | 27.9 |
| InnerShape-IoU(ratio=1.11) | 47.9 | 28.2 |
| **InnerShape-IoU(ratio=1.13)** | **48.3** | **28.2** |
| InnerShape-IoU(ratio=1.15) | 48.1 | 28.2 |
| InnerShape-IoU(ratio=1.19) | 47.8 | 28.0 |

It can be observed that adding a small detection head P2 significantly improves the model's accuracy by enabling it to detect a larger number of small objects. Moreover, in the configuration **[p2, p3, p4, p5] + PConv**, increasing the number of ELAN blocks in branch P2 at the Neck by one, reducing the number of ELAN blocks in branch P5 at the Neck by one, and incorporating PConv into the Neck collectively enhance the model's $AP_{.50}$. At the same time, these adjustments reduce the total number of parameters from 6.43M to 4.45M compared to the **[p2, p3, p4, p5]** configuration.

#### 4.3.3. Comparative study on downsample module

This section evaluates the effectiveness of different downsampling methods based on the results shown in Table 6. It can be observed that using MaxPooling layers for downsampling in both the Backbone and Neck reduces the number of parameters compared to the model (**#1**). However, this approach results in a decrease in accuracy due to less effective feature extraction.

In contrast, our proposed downsampling method (**#2**), which combines MGC in the Backbone with lightweight Ghost Convolution in the Neck, strikes a balance between complexity and performance. This method achieves lower model complexity compared to using normal Convolution throughout the network while improving accuracy, with $AP_{.50}$ increasing to 42.4% compared to 41.7%.

#### 4.3.4. Comparative study on spatial pyramid pooling

Spatial Pyramid Pooling (SPP), which transfers the feature map from the Backbone to the Neck of the model, is a crucial component in YOLO's PAFPN architecture. Table 7 compares several configurations for building model (**#4**) into model (**#5**). Notable in this comparison is SPPCSPC (Wang et al., 2023), the SPP architecture of YOLOv7, which achieves an accuracy of 46.3% $AP_{.50}$ but increases the number of parameters by more than 1.2 million compared to using SPP-YOLOv7-T (Wang et al., 2023).

Using SPPF (Jocher, 2020) reduces model parameters to 4.26M compared to 4.62M with SPP-YOLOv7-T while maintaining the same accuracy. Our proposed SPPRFM, which combines SPPF and RFM as described above, achieves the same accuracy as SPPCSPC (46.3% $AP_{.50}$) while significantly reducing the parameter count to 4.27M, a difference of nearly 1.6 million parameters compared to SPPCSPC (5.87M).

#### 4.3.5. Comparative study on pre-upsample block

This section evaluates various methods for the pre-upsampling block, including standard Convolution (**#5**), Convolution with SE attention (Hu et al., 2018), Convolution with CBAM attention (Woo et al., 2018), and our proposed CoordBlock (**#6**).

Table 8 demonstrates that adding attention blocks after Convolution slightly increases the number of parameters, while FLOPs remain unchanged. Using Conv + CBAM increases $AP_{.50}$ to 41.7%, a gain of 0.5% compared to standard Convolution. However, our CoordBlock, which integrates CoordConv and CoordATT, further minimizes spatial information loss during the upsampling of feature maps. As a result, $AP_{.50}$ improves to 42.4%, while the parameter count is slightly lower than that of Conv + CBAM.

#### 4.3.6. Comparative study on bouding box regression loss

To evaluate the effectiveness of our proposed InnerShape-IoU compared to other methods, we use $AP_{.50}$ and $AP_{.50:.95}$ as performance metrics. Table 9 shows that SIoU and WIoU achieve better results for small object detection than CIoU (**#6**), but their performance still falls short of ShapeIoU. The advantage of ShapeIoU lies in its ability to adjust bounding boxes based on the object's shape. However, with the added scale-up functionality through the adjustment of the *ratio* variable, InnerShape-IoU (**#7**) surpasses all other methods, achieving 48.3% $AP_{.50}$ and 28.2% $AP_{.50:.95}$. Table 10 provides detailed insights into the impact of the *ratio* parameter on InnerShape-IoU's performance.

As shown in Table 10, setting *ratio* = 0.95 results in inner bounding boxes smaller than the original bounding boxes, leading to a decline in model accuracy. Increasing the *ratio* above 1 scales up the original

**Table 11**
Comparison with other methods on Visdrone2019-DET-val. "p2" stand for model using output P2 head for small object detection. All results below are without additional advanced training techniques such as transfer learning or knowledge distillation for fair comparisons.

| Model | #Param.(M) | FLOPs(G) | $AP^{val}_{.50:.95}$(%) | $AP^{val}_{.50}$(%) | $AP^{val}_{.75}$(%) | $AP^{val}_{S}$(%) | $AP^{val}_{M}$(%) | $AP^{val}_{L}$(%) |
|---|---|---|---|---|---|---|---|---|
| YOLOv5-N r7.0 (Jocher, 2020) | 1.7 | 4.2 | 13.2 | 26.2 | 11.8 | 7.0 | 20.1 | 28.3 |
| YOLOv5-S r7.0 (Jocher, 2020) | 7.0 | 15.8 | 17.6 | 32.7 | 16.4 | 10.2 | 26.0 | 34.7 |
| YOLOv5-M r7.0 (Jocher, 2020) | 20.8 | 48.0 | 20.8 | 36.5 | 20.1 | 12.5 | 30.7 | 44.0 |
| YOLOv5-p2-N r7.0 (Jocher, 2020) | 1.8 | 4.9 | 15.8 | 30.6 | 14.1 | 9.1 | 23.2 | 30.2 |
| YOLOv5-p2-S r7.0 (Jocher, 2020) | 7.1 | 18.7 | 20.6 | 37.5 | 19.8 | 12.6 | 29.5 | 36.4 |
| YOLOv5u-N (Jocher et al., 2023) | 2.5 | 7.1 | 18.2 | 31.9 | 17.5 | 9.0 | 28.2 | 43.0 |
| YOLOv5u-S (Jocher et al., 2023) | 9.1 | 23.8 | 22.5 | 38.3 | 22.5 | 13.1 | 33.9 | 40.5 |
| YOLOv6-N v3.0 (Li et al., 2023) | 4.63 | 11.34 | 15.2 | 27.2 | 14.7 | 6.6 | 23.9 | 45.4 |
| YOLOv6-S v3.0 (Li et al., 2023) | 18.51 | 45.18 | 21.3 | 36.6 | 21.1 | 11.3 | 33.2 | 53.8 |
| YOLOv7-T (Wang et al., 2023) | 6.0 | 13.1 | 19.9 | 37.2 | 18.3 | 11.7 | 29.1 | 41.9 |
| YOLOv8-N (Jocher et al., 2023) | 3.0 | 8.1 | 19.0 | 33.2 | 18.6 | 9.9 | 29.3 | 43.5 |
| YOLOv8-S (Jocher et al., 2023) | 11.1 | 28.5 | 22.8 | 38.7 | 22.5 | 13.7 | 34.0 | 45.0 |
| YOLOv8-M (Jocher et al., 2023) | 25.8 | 78.7 | 25.4 | 41.8 | 25.9 | 15.8 | 37.8 | 48.5 |
| YOLOv8-p2-N (Jocher et al., 2023) | 2.9 | 12.2 | 21.3 | 36.8 | 21.0 | 13.2 | 30.8 | 38.7 |
| YOLOv8-p2-S (Jocher et al., 2023) | 10.6 | 36.7 | 25.4 | 42.9 | 25.3 | 16.9 | 36.0 | 41.8 |
| YOLOv9-T (Wang, Yeh, & Liao, 2024c) | 2.0 | 7.7 | 19.7 | 34.3 | 19.2 | 10.0 | 31.2 | 44.4 |
| YOLOv9-S (Wang, Yeh, & Liao, 2024c) | 7.1 | 26.4 | 23.5 | 39.8 | 23.1 | 14.1 | 35.8 | 45.0 |
| YOLOv9-M (Wang, Yeh, & Liao, 2024c) | 20 | 76.3 | 26.6 | 44.4 | 27.1 | 16.2 | 40.3 | 53.6 |
| YOLOv10-N (Wang, Chen, et al., 2024) | 2.3 | 6.7 | 18.5 | 32.6 | 18.0 | 9.8 | 28.8 | 37.0 |
| YOLOv10-S (Wang, Chen, et al., 2024) | 7.2 | 21.6 | 22.9 | 39.5 | 22.8 | 13.8 | 34.1 | 47.8 |
| YOLOv10-M (Wang, Chen, et al., 2024) | 15.4 | 59.1 | 25.3 | 42.5 | 25.4 | 16.0 | 37.4 | 48.6 |
| YOLOv10-B (Wang, Chen, et al., 2024) | 19.1 | 92.0 | 26.6 | 44.2 | 26.9 | 17.0 | 39.0 | 47.1 |
| YOLOX-S (Ge et al., 2021) | 8.94 | 26.78 | 20.7 | 36.9 | 20.0 | 12.2 | 30.5 | 43.9 |
| YOLOX-M (Ge et al., 2021) | 25.29 | 73.8 | 22.4 | 39.3 | 22.0 | 13.6 | 32.9 | 48.4 |
| Gold YOLO-N (Wang, He, et al., 2024) | 5.61 | 12.1 | 18.8 | 33.2 | 18.3 | 10.2 | 29.1 | 39.8 |
| Gold YOLO-S (Wang, He, et al., 2024) | 21.51 | 46.0 | 21.1 | 36.2 | 21.2 | 12.0 | 32.4 | 42.4 |
| DAMO YOLO-T (Xu, Jiang, et al., 2023) | 8.59 | 18.32 | 23.8 | 40.4 | 24.0 | 14.2 | 35.9 | 50.2 |
| DAMO YOLO-S (Xu, Jiang, et al., 2023) | 16.37 | 38.21 | 25.6 | 43.4 | 26.4 | 16.2 | 38.3 | 51.0 |
| RT-DETR-R18 (Zhao et al., 2024) | 20.0 | 60.0 | 27.2 | 46.6 | 27.0 | 18.4 | 37.6 | 53.6 |
| EdgeYOLO-T (Liu et al., 2023) | 5.5 | 27.24 | 21.8 | 38.5 | 21.3 | 12.4 | 32.5 | 45.8 |
| EdgeYOLO-S (Liu et al., 2023) | 9.3 | 45.32 | 23.5 | 40.8 | 23.1 | 13.8 | 34.8 | 49.1 |
| EdgeYOLO-M (Liu et al., 2023) | 17.8 | 70.74 | 24.9 | 42.9 | 24.7 | 15.2 | 36.2 | 49.1 |
| EdgeYOLO (Liu et al., 2023) | 40.5 | 126.41 | 26.8 | 45.4 | 26.7 | 17.0 | 38.1 | 55.3 |
| PPYOLOE-S (Xu et al., 2022) | 7.93 | 17.36 | 23.5 | 39.9 | 23.6 | 13.9 | 35.7 | 52.5 |
| PPYOLOE-M (Xu et al., 2022) | 23.43 | 49.91 | 26.1 | 44.5 | 26.2 | 16.3 | 38.1 | 54.5 |
| Drone-YOLO-N (Zhang, 2023) | 3.05 | _ | 22.7 | 38.1 | 22.8 | _ | _ | _ |
| Drone-YOLO-T (Zhang, 2023) | 5.35 | _ | 25.6 | 42.8 | 26.2 | _ | _ | _ |
| Drone-YOLO-S (Zhang, 2023) | 10.9 | _ | 27.0 | 44.3 | 27.5 | _ | _ | _ |
| LE-YOLO (Yue et al., 2024) | 2.1 | 13.1 | 22.7 | 39.3 | _ | _ | _ | _ |
| YOLOv5s-pp (Xu, Zheng, et al., 2023) | 10.5 | _ | _ | 41.7 | _ | _ | _ | _ |
| PVswin-YOLOv8 (Tahir et al., 2024) | 21.6 | _ | 26.4 | 43.3 | _ | _ | _ | _ |
| HIC-YOLO (Tang et al., 2024) | 9.3 | 30.9 | 24.9 | 43.0 | 24.4 | 15.8 | 35.2 | 42.8 |
| PDWT-YOLO (Zhang, Xiong, et al., 2023) | 6.44 | 24.5 | 24.3 | 42.6 | 23.6 | 15.9 | 33.4 | 40.9 |
| **LEAF-YOLO-N (Ours)** | **1.2** | **5.6** | **21.9** | **39.7** | **20.9** | **14.0** | **30.6** | **41.3** |
| **LEAF-YOLO (Ours)** | **4.28** | **20.9** | **28.2** | **48.3** | **27.6** | **20.0** | **38.0** | **46.5** |

bounding box. At *ratio* = 1.13, the model achieves optimal accuracy, with the best results in both $AP_{.50}$ and $AP_{.50:.95}$. However, as the *ratio* continues to increase, the inner boxes become excessively large compared to the original bounding box, resulting in a drop in accuracy.

### 4.4. Performance benchmarking of different algorithms on VisDrone 2019-DET

In this section, we compare our proposed methods, LEAF-YOLO and LEAF-YOLO-N, with two groups of object detection models on the VisDrone2019-DET-val and VisDrone2019-DET-test datasets. The first group (**Group 1**) consists of current State-of-the-Art (SOTA) real-time object detection models, including YOLOv5 (Jocher, 2020), YOLOv6 (Li et al., 2023), YOLOv7 (Wang et al., 2023), YOLOv8 (Jocher et al., 2023), YOLOv9 (Wang, Yeh, & Liao, 2024c), YOLOv10 (Wang, Chen, et al., 2024), YOLOX (Ge et al., 2021), Gold YOLO (Wang, He, et al., 2024), DAMO-YOLO (Xu, Jiang, et al., 2023), and RT-DET-R18 (Zhao et al., 2024) (a Transformer-based model). The second group (**Group 2**) includes prior works specifically targeting lightweight small object detection in aerial imagery, such as EdgeYOLO (Liu et al., 2023), PPYOLOE (Xu et al., 2022), Drone-YOLO (Zhang, 2023), LE-YOLO (Yue et al., 2024), YOLOv5s-pp (Xu, Zheng, et al., 2023), PVswin-YOLOv8 (Tahir et al., 2024), HIC-YOLO (Tang et al., 2024), and PDWT-YOLO (Zhang, Xiong, et al., 2023).

The comparison highlights the performance of both LEAF-YOLO models against nano, tiny, and small variants with fewer than 20 million parameters, as well as selected medium-sized models, such as YOLOX-M and RT-DET-R18. As shown in Tables 11 and 13, LEAF-YOLO and LEAF-YOLO-N achieve superior accuracy while maintaining efficiency, demonstrating their effectiveness for real-time small object detection in UAV-captured imagery.

#### 4.4.1. Result on VisDrone2019-DET-val set

**In terms of Accuracy-Parameters.** Table 11 and Fig. 1 illustrate the trade-off between accuracy ($AP_{.50}$) and the number of parameters for all models evaluated.

- **Compare with State-of-the-Art Models.** Among the models this group, medium-sized models like YOLOv8-M (41.8% $AP_{.50}$, 11.1M parameters) and YOLOv10-M (42.5% $AP_{.50}$, 12.6M parameters) exhibit high accuracy. However, they require significantly more parameters compared to our proposed LEAF-YOLO, which achieves 48.3% $AP_{.50}$ with only 4.28M parameters. In particular, the lightweight model YOLOv8-N achieves 29.1% $AP_{.50}$ with 3.0M parameters, but this is still lower than the 39.7% $AP_{.50}$ of LEAF-YOLO-N, which uses only 1.2M parameters.
- **Compare with Lightweight Small Object Detection Models for Aerial Imagery.** Within this group, LEAF-YOLO-N demonstrates superior performance among nano-sized models, achieving 39.7% $AP_{.50}$ with 1.2M parameters, surpassing YOLOv5-N (28.3% $AP_{.50}$, 4.2M parameters) and YOLOv8p2-N (36.8% $AP_{.50}$, 2.8M parameters). Similarly, LEAF-YOLO achieves the best balance among small and tiny-sized models, outperforming PDWT-YOLO (42.6% $AP_{.50}$, 6.4M parameters) and HIC-YOLO (43.0% $AP_{.50}$, 9.3M parameters) with significantly fewer parameters.

**In terms of Accuracy vs. Complexity.** Fig. 16 and Table 11 provide an analysis of accuracy ($AP_{.50:.95}$ and $AP_S$) against computational complexity (FLOPs) for the evaluated models.

- **Compare with State-of-the-Art Models.** In this group, YOLOv9-N achieves an $AP_{.50:.95}$ of 19.7% and 9.9% $AP_S$ with 6.7 GFLOPs, while Gold YOLO-N achieves an $AP_{.50:.95}$ of 18.8% and 10.2% $AP_S$ with 12.1 GFLOPs. Similarly, DAMO-YOLO-T attains 23.8% $AP_{.50:.95}$ and 14.2% $AP_S$ with 18.3 GFLOPs. In comparison, our LEAF-YOLO achieves 28.2% $AP_{.50:.95}$ and 20.0% $AP_S$ with just 20.9 GFLOPs, outperforming all three models in both overall accuracy and small object detection. Notably, while YOLOX-S achieves 23.5% $AP_{.50:.95}$ and 12.2% $AP_S$ at 26.78 GFLOPs, it lags behind LEAF-YOLO in both metrics despite requiring higher computational complexity. This demonstrates the superior efficiency of LEAF-YOLO for real-time small object detection.
- **Compare with Lightweight Small Object Detection Models for Aerial Imagery.** EdgeYOLO-T achieves 21.8% $AP_{.50:.95}$ and 12.4% $AP_S$ with 5.5 GFLOPs, while PP-YOLOE-S achieves 23.5% $AP_{.50:.95}$ and 13.9% $AP_S$ with 17.36 GFLOPs. Drone-YOLO-N achieves 22.7% $AP_{.50:.95}$ and 8.3% $AP_S$ with 4.5 GFLOPs, but its $AP_S$ lags significantly compared to LEAF-YOLO-N's 14.0% $AP_S$ at a similar complexity of 5.6 GFLOPs. LEAF-YOLO-N achieves 21.9% $AP_{.50:.95}$ and 14.0% $AP_S$, outperforming EdgeYOLO-T and Drone-YOLO-N in small object detection while maintaining a competitive complexity. LEAF-YOLO, with 28.2% $AP_{.50:.95}$ and 20.0% $AP_S$ at 20.9 GFLOPs, further surpasses PP-YOLOE-M in both metrics despite requiring fewer FLOPs.

**Detailed Comparison for Each Class.** For a detailed evaluation of the performance of LEAF-YOLO-N among nano-size models and LEAF-YOLO among tiny and small-size models, Table 12 presents $AP_{.50}$ for each of the 10 object classes on the VisDrone2019-DET-val dataset.

- **Nano-Size Models.** Among the nano-size models, our LEAF-YOLO-N demonstrates superior performance across all classes. For the Awning-Tricycle (AwTri) class, which contains the fewest objects, LEAF-YOLO-N achieves 13.0% $AP_{.50}$, significantly outperforming YOLOv8p2-N (9.93% $AP_{.50}$). Similarly, LEAF-YOLO-N achieves strong results for other classes, such as Pedestrian (47.2% $AP_{.50}$), People (40.2% $AP_{.50}$), and Motor (50.2% $AP_{.50}$), surpassing YOLOv8p2-N, which only achieves 42.3%, 38.1%, and 48.6% $AP_{.50}$, respectively. These results highlight the robustness of LEAF-YOLO-N in detecting objects across diverse categories, even with a minimal parameter size and complexity.
- **Tiny and Small-Size Models.** For tiny and small-size models, LEAF-YOLO consistently achieves the highest $AP_{.50}$ scores across most object classes, particularly for small object categories such as Pedestrian, People, and Motor. LEAF-YOLO achieves 57.7%, 49.5%, and 59.7% $AP_{.50}$ in these classes, outperforming HIC-YOLO by 7.19%, 10.29%, and 9.19%, respectively, and PDWT-YOLO by 6.63%, 6.33%, and 7.43%, respectively. Additionally, LEAF-YOLO shows exceptional performance in larger object categories, such as Van and Bus, achieving 50.3% and 64.1% $AP_{.50}$, respectively, further demonstrating its versatility across object sizes and categories.

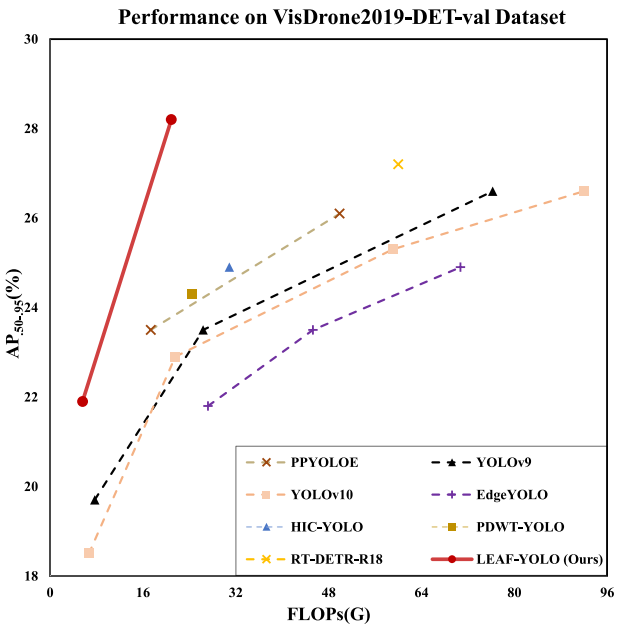


**Fig. 16.** Comparisons with others in terms of complexity-accuracy using Visdrone2019-DET-val.

These results confirm that both LEAF-YOLO-N and LEAF-YOLO provide state-of-the-art performance within their respective size categories, offering robust and accurate detection across all object classes in the VisDrone2019-DET-val dataset.

#### 4.4.2. Result on VisDrone2019-DET-test set

LEAF-YOLO-N and LEAF-YOLO have been proven in the last section as models with high accuracy and the optimal number of parameters on the validation set of VisDrone2019-DET. In this part, the VisDrone2019-DET-test set is used to test the practical usability of the models. Table 13 and Fig. 17 provide a comprehensive analysis of accuracy and latency for models evaluated on the VisDrone2019-DET-test set. The results are divided into the same two aforementioned groups: SOTA models and current work on lightweight small object detection models for aerial imaging, with accuracy metrics ($AP_{.50}$, $AP_{.50:.95}$, $AP_{.75}$, $AP_S$, $AP_M$, and $AP_L$) evaluated along with latency (measured in milliseconds) using fp16 precision on an RTX 3090 GPU.

When compared to the group of SOTA object detection models, LEAF-YOLO demonstrates an exceptional balance between accuracy and latency. It achieves an $AP_{.50}$ of 39.1% and $AP_{.50:.95}$ of 22.0%, outperforming YOLOv9-S, which achieves 38.7% $AP_{.50}$ and 20.5% $AP_{.50:.95}$. Despite its competitive accuracy, YOLOv9-S has a higher latency of 29.3 ms compared to LEAF-YOLO's 21.7 ms. Similarly, Gold YOLO-S achieves 35.2% $AP_{.50}$ and 19.7% $AP_{.50:.95}$ but is hindered by a much higher latency of 32.8 ms. When evaluating small object detection ($AP_S$), LEAF-YOLO achieves 12.1%, surpassing YOLOv9-S and Gold YOLO-S, which score 10.8% and 10.0%, respectively. Furthermore, LEAF-YOLO also outperforms DAMO-YOLO-T in both $AP_{.50:.95}$ and small object detection, with DAMO-YOLO-T achieving only 19.3% $AP_{.50:.95}$ and 9.4% $AP_S$ and also having a slightly higher latency of

**Table 12**
$AP_{.50}$(%) of each category on Visdrone2019-DET-val.

| Model | Pedestrian | People | Bicycle | Car | Van | Truck | Tricycle | AwTri | Bus | Motor | All |
|---|---|---|---|---|---|---|---|---|---|---|---|
| YOLOv5-N r7.0 (Jocher, 2020) | 29.98 | 24.68 | 8.29 | 66.78 | 25.08 | 20.98 | 12.68 | 6.88 | 37.08 | 29.58 | 26.2 |
| YOLOv5-p2-N r7.0 (Jocher, 2020) | 35.4 | 28.8 | 10.8 | 73.3 | 29 | 25.6 | 15.7 | 7.2 | 44.1 | 36.1 | 30.6 |
| YOLOv5u-N (Jocher et al., 2023) | 32.89 | 25.79 | 9.88 | 72.59 | 35.09 | 29.29 | 21.39 | 9.49 | 48.19 | 34.39 | 31.9 |
| YOLOv8-N (Jocher et al., 2023) | 34.43 | 27.93 | 10.6 | 73.33 | 37.23 | 31.13 | 21.23 | 9.53 | 49.63 | 36.93 | 33.2 |
| YOLOv8-p2-N (Jocher et al., 2023) | 42.93 | 35.73 | 13.8 | 77.83 | 40.73 | 28.43 | 22.03 | 9.93 | 52.83 | 43.73 | 36.8 |
| YOLOv9-T (Wang, Yeh, & Liao, 2024c) | 34.16 | 28.36 | 10.8 | 74.76 | 38.66 | 32.36 | 25.76 | 11.76 | 47.76 | 38.36 | 34.3 |
| YOLOv10-N (Wang, Chen, et al., 2024) | 34.24 | 28.44 | 10.5 | 74.94 | 37.04 | 29.04 | 19.34 | 9.44 | 44.94 | 37.64 | 32.6 |
| LE-YOLO (Yue et al., 2024) | 46.7 | 39.1 | 12.0 | 82.3 | 43.1 | 31.4 | 24.5 | 13.8 | 52.9 | 46.8 | 39.3 |
| **LEAF-YOLO-N (Ours)** | **47.2** | **40.2** | **13.8** | **80.6** | **41.5** | **33.5** | **24.4** | **13.0** | **52.9** | **50.2** | **39.7** |
| YOLOv5-S r7.0 (Jocher, 2020) | 37.06 | 30.26 | 13.0 | 71.26 | 33.36 | 30.66 | 17.86 | 8.56 | 48.36 | 36.66 | 32.7 |
| YOLOv5-p2-S r7.0 (Jocher, 2020) | 44.19 | 34.39 | 16.3 | 77.79 | 38.99 | 33.09 | 22.39 | 11.69 | 53.89 | 42.29 | 37.5 |
| YOLOv5u-S (Jocher et al., 2023) | 41.02 | 32.62 | 17.2 | 76.22 | 41.12 | 35.52 | 27.32 | 13.32 | 55.32 | 43.32 | 38.3 |
| YOLOv7-T (Wang et al., 2023) | 40.73 | 37.73 | 12.3 | 77.23 | 38.33 | 31.83 | 24.23 | 12.73 | 50.33 | 46.53 | 37.2 |
| YOLOv8-S (Jocher et al., 2023) | 42.12 | 32.12 | 18.3 | 76.22 | 41.02 | 38.12 | 25.92 | 12.12 | 57.72 | 43.32 | 38.7 |
| YOLOv8-p2-S (Jocher et al., 2023) | 51.33 | 41.43 | 21.6 | 80.13 | 44.83 | 37.93 | 28.13 | 13.93 | 57.93 | 51.73 | 42.9 |
| YOLOv9-S (Wang, Yeh, & Liao, 2024c) | 43.04 | 33.74 | 17.4 | 77.54 | 41.94 | 37.94 | 29.54 | 14.04 | 57.64 | 43.54 | 39.8 |
| YOLOv10-S (Wang, Chen, et al., 2024) | 42.34 | 33.64 | 16.3 | 78.24 | 42.94 | 36.64 | 27.84 | 15.04 | 56.94 | 44.94 | 39.5 |
| YOLOv5s-pp (Xu, Zheng, et al., 2023) | 51.7 | 39.6 | 19.0 | 82.1 | 44.1 | 36.0 | 26.3 | 14.7 | 55.3 | 48.2 | 41.7 |
| PVswin-YOLOv8 (Tahir et al., 2024) | 45.9 | 35.7 | 16.4 | 81.5 | 49.1 | 42.4 | 32.8 | 17.7 | 62.9 | 48.2 | 43.3 |
| HIC-YOLO (Tang et al., 2024) | 50.51 | 39.21 | 21.4 | 80.61 | 43.61 | 38.21 | 29.01 | 14.81 | 62.11 | 50.51 | 43.0 |
| PDWT-YOLO (Zhang, Xiong, et al., 2023) | 51.07 | 43.17 | 14.4 | 82.57 | 44.07 | 37.87 | 26.77 | 15.17 | 58.67 | 52.27 | 42.6 |
| **LEAF-YOLO (Ours)** | **57.7** | **49.5** | **23.6** | **85.4** | **50.3** | **42.0** | **33.9** | **16.8** | **64.1** | **59.7** | **48.3** |

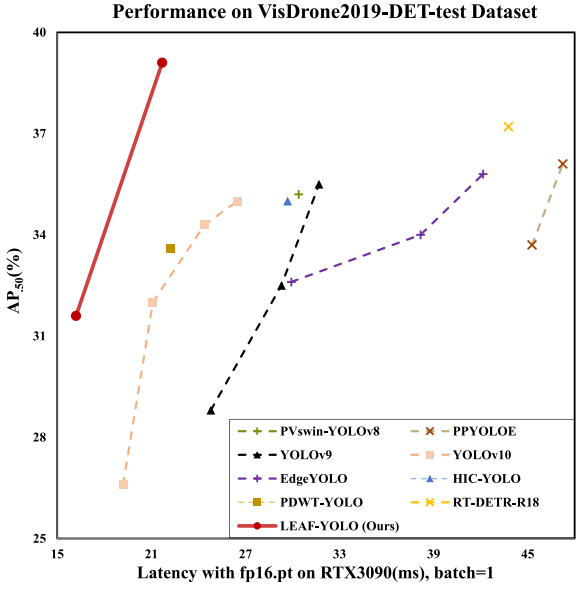


**Fig. 17.** Comparisons with others in terms of latency-accuracy using Visdrone2019-DET-test.

22.33 ms. These results illustrate that LEAF-YOLO is not only more accurate but also faster than most SOTA models, making it ideal for real-time UAV applications.

When compared to the previous works on Lightweight Small Object Detection Models for Aerial Imagery, LEAF-YOLO-N sets itself apart with strong accuracy and low latency. It achieves an $AP_{.50}$ of 36.1%, $AP_{.50:.95}$ of 16.9%, and $AP_S$ of 8.6%, while maintaining an impressive latency of 21.6 ms. Comparatively, EdgeYOLO-T achieves 32.6% $AP_{.50}$, 16.2% $AP_{.50:.95}$, and 8.5% $AP_S$, with a latency of 29.93 ms, indicating that LEAF-YOLO-N delivers better accuracy and faster inference. Similarly, PP-YOLOE-S achieves 42.8% $AP_{.50}$ and 19.3% $AP_{.50:.95}$ but suffers from a high latency of 47.25 ms, more than double that of LEAF-YOLO-N. While PP-YOLOE-S excels in large object detection with 50.6% $AP_L$, LEAF-YOLO-N achieves better performance in small object detection with 8.6% $AP_S$ compared to PP-YOLOE-S's 8.0%. Another lightweight competitor, Drone-YOLO-N, achieves only 31.0% $AP_{.50}$ and 7.6% $AP_S$, falling short of LEAF-YOLO-N's performance in every metric. Despite having slightly lower complexity, Drone-YOLO-N is outperformed across all accuracy categories, showcasing the robustness of LEAF-YOLO-N.

Overall, the results on the test set confirm the efficiency of both LEAF-YOLO and LEAF-YOLO-N in achieving higher accuracy while maintaining low latency. LEAF-YOLO surpasses SOTA models in both overall accuracy ($AP_{.50:.95}$) and small object detection ($AP_S$) with a significantly faster inference speed. Similarly, LEAF-YOLO-N outperforms other lightweight models, offering superior performance in both accuracy and latency, making it a powerful choice for real-time UAV-based applications.

**Visual Analysis of Small Object Detection Performance.** To visually demonstrate the capability of small object detection in aerial imagery, Figs. 18 and 19 present a comparison using EigenGrad-CAM (Muhammad & Yeasin, 2020) heat maps in the first row and the corresponding prediction results in the second row. These visualizations highlight the differences in feature extraction and prediction accuracy between our proposed models and other state-of-the-art methods.

In Fig. 18, LEAF-YOLO-N is compared with YOLOv8-p2-N, both of which are nano-size models equipped with an additional P2 head to enhance small object detection. The EigenGradCAM heat maps reveal that LEAF-YOLO-N demonstrates superior feature fusion compared to YOLOv8-p2-N. This improvement can be attributed to the inclusion of the SPPRFM architecture and CoordBlock, which enhance the Neck's ability to retain spatial information and focus on small objects. Consequently, LEAF-YOLO-N captures a greater number of objects with higher accuracy, as evident in the prediction results.

Similarly, in Fig. 19, LEAF-YOLO is compared with HIC-YOLO. The visualizations show that LEAF-YOLO is able to detect a larger number of small objects, particularly those located farther away, which HIC-YOLO fails to identify. This advantage is due to the innovative design of the LEAF and LEAF-T blocks, which generate rich multiscale feature maps, and the InnerShape-IoU loss function, which enables the model to adapt to objects of varying shapes and sizes. These enhancements allow LEAF-YOLO to consistently outperform HIC-YOLO in scenarios involving small and distant objects.

### *4.4.3. Real-time on edge with TensorRT*

In this section, LEAF-YOLO and LEAF-YOLO-N are converted end-to-end, including the EfficientNMS module, to fp16 format using TensorRT 8.5.2 and deployed on the Jetson AGX Xavier board. The models' performance is evaluated using frames per second (FPS) with a throughput of 1 ($FPS_{b=1}$) and a throughput of 16 ($FPS_{b=16}$). These metrics determine

**Table 13**
Comparison with other methods on Visdrone2019-DET-test. "p2" stand for model using output P2 head for small object detection.

| Model | $AP^{test}_{.50:.95}$(%) | $AP^{test}_{.50}$(%) | $AP^{test}_{.75}$(%) | $AP^{test}_{S}$(%) | $AP^{test}_{M}$(%) | $AP^{test}_{L}$(%) | $Latency^{fp16.pt}_{b=1}$(ms) |
|---|---|---|---|---|---|---|---|
| YOLOv5-N r7.0 (Jocher, 2020) | 11.0 | 22.2 | 9.8 | 4.5 | 17.1 | 26.7 | 19.7 |
| YOLOv5-S r7.0 (Jocher, 2020) | 14.0 | 26.6 | 13.3 | 6.3 | 21.6 | 32.8 | 24.4 |
| YOLOv5-M r7.0 (Jocher, 2020) | 16.1 | 30.0 | 15.8 | 7.8 | 25.0 | 36.0 | 26.2 |
| YOLOv5-p2-N r7.0 (Jocher, 2020) | 11.8 | 24.1 | 10.3 | 5.2 | 18.3 | 27.3 | 22.5 |
| YOLOv5-p2-S r7.0 (Jocher, 2020) | 15.4 | 29.7 | 14.5 | 7.4 | 23.5 | 35.2 | 26.2 |
| YOLOv5u-N (Jocher et al., 2023) | 14.5 | 25.9 | 14.5 | 5.7 | 22.7 | 36.0 | 21.3 |
| YOLOv5u-S (Jocher et al., 2023) | 17.3 | 30.4 | 17.4 | 7.8 | 26.7 | 38.9 | 21.6 |
| YOLOv6-N v3.0 (Li et al., 2023) | 13.5 | 24.2 | 13.5 | 4.7 | 21.0 | 38.9 | 21.3 |
| YOLOv6-S v3.0 (Li et al., 2023) | 18.0 | 31.1 | 18.4 | 7.8 | 27.9 | 47.7 | 24.14 |
| YOLOv7-T (Wang et al., 2023) | 16.0 | 30.3 | 15.1 | 7.7 | 24.2 | 35.1 | 16.8 |
| YOLOv8-N (Jocher et al., 2023) | 14.8 | 26.4 | 14.9 | 5.9 | 23.4 | 34.2 | 21.6 |
| YOLOv8-S (Jocher et al., 2023) | 17.7 | 30.8 | 17.9 | 8.0 | 27.5 | 40.7 | 22.8 |
| YOLOv8-M (Jocher et al., 2023) | 19.1 | 32.9 | 19.5 | 9.3 | 29.2 | 42.2 | 28.0 |
| YOLOv8-p2-N (Jocher et al., 2023) | 15.4 | 28.1 | 15.1 | 7.8 | 23.6 | 30.6 | 22.0 |
| YOLOv8-p2-S (Jocher et al., 2023) | 19.1 | 33.6 | 19.2 | 10.2 | 28.3 | 36.4 | 26.4 |
| YOLOv9-T (Wang, Yeh, & Liao, 2024c) | 16.0 | 28.8 | 15.7 | 6.2 | 25.4 | 39.7 | 24.8 |
| YOLOv9-S (Wang, Yeh, & Liao, 2024c) | 18.4 | 32.5 | 18.4 | 8.4 | 28.7 | 42.4 | 29.3 |
| YOLOv9-M (Wang, Yeh, & Liao, 2024c) | 20.3 | 35.5 | 20.5 | 9.7 | 31.4 | 46.2 | 31.7 |
| YOLOv10-N (Wang, Chen, et al., 2024) | 14.2 | 26.6 | 13.7 | 6.0 | 22.5 | 32.2 | 19.2 |
| YOLOv10-S (Wang, Chen, et al., 2024) | 17.7 | 32.0 | 17.4 | 8.4 | 27.5 | 40.0 | 21.1 |
| YOLOv10-M (Wang, Chen, et al., 2024) | 19.6 | 34.3 | 19.7 | 9.8 | 29.7 | 46.1 | 24.4 |
| YOLOv10-B (Wang, Chen, et al., 2024) | 20.1 | 35.0 | 20.2 | 10.1 | 30.3 | 44.1 | 26.5 |
| YOLOX-S (Ge et al., 2021) | 16.8 | 30.9 | 16.3 | 8.1 | 25.6 | 40.7 | 20.34 |
| YOLOX-M (Ge et al., 2021) | 16.9 | 30.4 | 16.4 | 8.4 | 25.7 | 38.8 | 25.76 |
| Gold YOLO-N (Wang, He, et al., 2024) | 15.3 | 27.6 | 15.2 | 6.7 | 23.7 | 34.9 | 21.83 |
| Gold YOLO-S (Wang, He, et al., 2024) | 16.9 | 29.7 | 17.1 | 7.6 | 26.3 | 36.8 | 23.83 |
| DAMO YOLO-T (Xu, Jiang, et al., 2023) | 19.1 | 33.6 | 19.3 | 9.4 | 29.0 | 43.6 | 20.5 |
| DAMO YOLO-S (Xu, Jiang, et al., 2023) | 20.7 | 36.4 | 21.0 | 10.8 | 31.6 | 45.1 | 22.33 |
| RT-DETR-R18 (Zhao et al., 2024) | 20.5 | 37.2 | 20.3 | 10.8 | 30.3 | 45.9 | 43.78 |
| EdgeYOLO-T (Liu et al., 2023) | 18.0 | 32.6 | 17.7 | 8.5 | 27.4 | 37.6 | 29.93 |
| EdgeYOLO-S (Liu et al., 2023) | 18.9 | 34.0 | 18.9 | 9.6 | 28.5 | 42.2 | 38.18 |
| EdgeYOLO-M (Liu et al., 2023) | 20.1 | 35.8 | 20.1 | 10.0 | 30.0 | 40.9 | 42.17 |
| EdgeYOLO (Liu et al., 2023) | 21.4 | 36.8 | 21.6 | 10.9 | 31.3 | 45.7 | 46.2 |
| PPYOLOE-S (Xu et al., 2022) | 19.3 | 33.7 | 19.8 | 9.3 | 29.4 | 42.8 | 45.29 |
| PPYOLOE-M (Xu et al., 2022) | 21.1 | 36.1 | 21.3 | 10.2 | 30.6 | 45.8 | 47.25 |
| Drone-YOLO-N (Zhang, 2023) | 17.5 | 31.0 | 17.6 | _ | _ | _ | _ |
| Drone-YOLO-T (Zhang, 2023) | 19.1 | 33.7 | 19.4 | _ | _ | _ | _ |
| Drone-YOLO-S (Zhang, 2023) | 20.4 | 35.6 | 20.6 | _ | _ | _ | _ |
| PVswin-YOLOv8 (Tahir et al., 2024) | 20.4 | 35.2 | _ | _ | _ | _ | 30.4 |
| HIC-YOLO (Tang et al., 2024) | 19.2 | 35.0 | 18.7 | 9.3 | 28.7 | 39.7 | 29.7 |
| PDWT-YOLO (Zhang, Xiong, et al., 2023) | 18.3 | 33.6 | 17.8 | 9.4 | 27.1 | 36.0 | 22.2 |
| **LEAF-YOLO-N (Ours)** | **16.9** | **31.6** | **16.1** | **8.6** | **24.9** | **38.0** | **16.2** |
| **LEAF-YOLO (Ours)** | **22.0** | **39.1** | **21.4** | **12.1** | **31.6** | **42.9** | **21.7** |

whether the models achieve real-time performance (FPS > 30) on edge devices with an input image size of 640 × 640. Additionally, our proposed method is compared against two state-of-the-art edge-real-time small object detection models: EdgeYOLO-T (Liu et al., 2023) and EdgeYOLO-S (Liu et al., 2023).

To provide a more comprehensive analysis, we address the types of uncertainties encountered in real-time edge applications. In this context, uncertainties can be categorized as follows:

- **Internal vs. External:** Internal uncertainties arise from model parameters and processing capabilities, such as quantization errors during model conversion to TensorRT. External uncertainties stem from environmental factors, including varying lighting conditions and occlusions in input images during inference.
- **Parametric vs. Non-parametric:** Parametric uncertainties relate to model parameters, such as weights and biases, which may vary due to hardware-specific optimizations or differences in TensorRT calibration. Non-parametric uncertainties, on the other hand, include unpredictable variations in image data, which can impact model performance.
- **Constant vs. Random:** Constant uncertainties, such as fixed rounding errors from fp16 precision, are consistently present across runs. Random uncertainties, including image noise or hardware-induced timing variations, may fluctuate unpredictably.

Determining and addressing these uncertainties is particularly challenging in real-time edge applications due to limited computational resources and strict latency constraints. Our approach focuses on minimizing parametric uncertainties by optimizing model architecture and using efficient quantization techniques. However, external uncertainties like variations in input data remain a challenge and may affect real-time performance.

In Table 14, EdgeYOLO-T and EdgeYOLO-S achieved 65 and 54 FPS with a throughput of 16, closely aligning with the original results of 67 and 53 FPS reported in the EdgeYOLO paper (Liu et al., 2023). Additionally, when tested with a throughput of 1, EdgeYOLO-T and EdgeYOLO-S achieved 27 and 21 FPS, respectively. In comparison, LEAF-YOLO-N and LEAF-YOLO demonstrated significantly higher performance. With a throughput of 1, LEAF-YOLO-N achieved 56 FPS and LEAF-YOLO achieved 32 FPS, approximately 2 and 1.5 times faster than EdgeYOLO-T and EdgeYOLO-S, respectively. When the throughput was

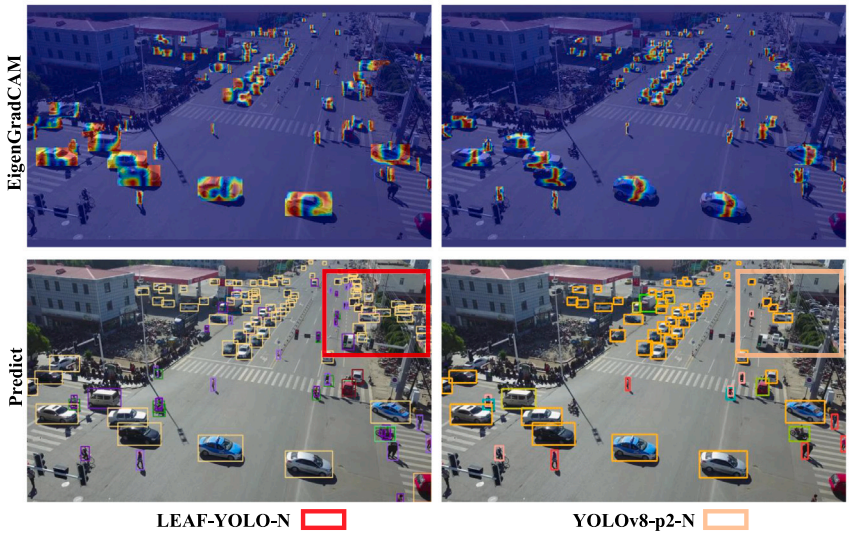


**Fig. 18.** LEAF-YOLO-N (Ours) vs. YOLOv8-p2-N: EigenGradCAM heat map and Prediction results in comparison. (left and right, respectively).

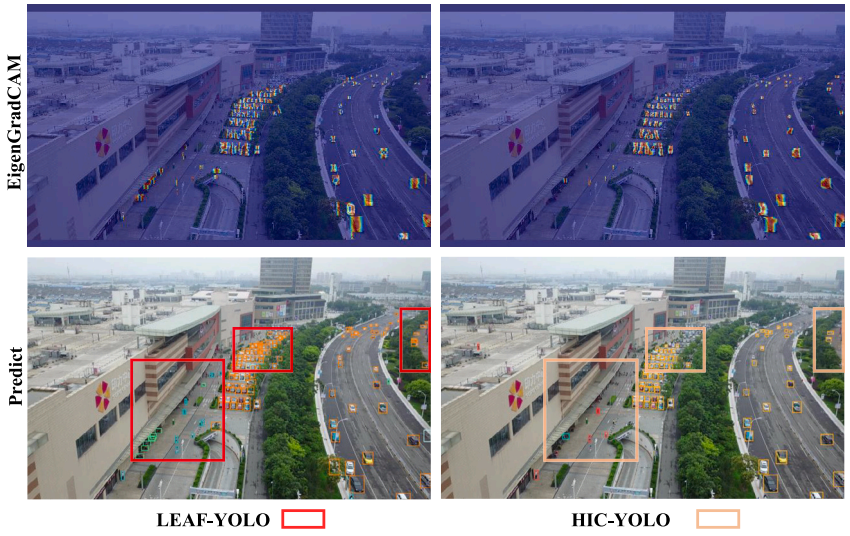


**Fig. 19.** LEAF-YOLO (Ours) vs. HIC-YOLO: EigenGradCAM heat map and Prediction results in comparison. (left and right, respectively).

**Table 14**
Real-time performance comparison of models on edge devices.

| Model | $AP^{val}_{.50}$(%) | $FPS^{fp16.trt}_{b=1}$ | $FPS^{fp16.trt}_{b=16}$ |
|---|---|---|---|
| EdgeYOLO-T (Liu et al., 2023) | 38.5 | 27 | 65 |
| EdgeYOLO-S (Liu et al., 2023) | 40.8 | 21 | 54 |
| **LEAF-YOLO-N (Ours)** | **39.7** | **56** | **147** |
| **LEAF-YOLO (Ours)** | **48.3** | **32** | **87** |

increased to 16, LEAF-YOLO-N ran at 147 FPS, nearly 2.3 times faster than EdgeYOLO-T, while LEAF-YOLO achieved 87 FPS, approximately 1.6 times faster than EdgeYOLO-S. These results clearly demonstrate that our proposed methods outperform EdgeYOLO in both speed and accuracy for small object detection.

Moreover, both LEAF-YOLO and LEAF-YOLO-N meet the real-time performance requirement on the edge-computing device Jetson AGX Xavier. This is achieved through model optimization and quantization, addressing the inherent uncertainties of edge deployment and ensuring consistent performance.

### 4.5. More comparison on TinyPerson dataset

To further validate the capability of recognizing small objects, this section compares the models with the highest $AP_{50}$ scores in Table 11 on the TinyPerson dataset. After training these models on the TinyPerson training set, Table 15 presents their evaluation results on the TinyPerson validation set. The accuracy metrics, including $AP_{.50:.95}$, $AP_{.50}$, $AP_{.75}$, $AP_S$, and $AP_M$, are used to evaluate the models. The metric $AP_L$ is excluded because the TinyPerson dataset does not contain large objects. Regarding the number of parameters and GFLOPs, these metrics are reported similarly to Table 11, providing a comprehensive comparison of model efficiency and computational complexity. Table 15 demonstrates that the AP metrics on the TinyPerson dataset are significantly lower compared to the Visdrone2019-DET dataset. This disparity arises from the unique challenges posed by the TinyPerson dataset, where objects are extremely small, densely distributed, and often form overlapping crowds. Despite these challenges, LEAF-YOLO outperforms other models with 7.6% $AP_{.50:.95}$, 23.0% $AP_{.50}$, and 3.2% $AP_{.75}$.

Notably, the incorporation of modules specifically designed for small object detection enables LEAF-YOLO to achieve an $AP_S$ of 7.5%,

**Table 15**
Performance of various object detectors on the TinyPerson dataset.

| Model | $AP_{.50:.95}$ | $AP_{.50}$ | $AP_{.75}$ | $AP_S$ | $AP_M$ |
|---|---|---|---|---|---|
| YOLOv5-M r7.0 (Jocher, 2020) | 6.3% | 20.6% | 2.1% | 6.2% | 24.1% |
| YOLOv5-p2-S r7.0 (Jocher, 2020) | 6.0% | 19.5% | 2.1% | 5.8% | 26.7% |
| YOLOv5u-S (Jocher et al., 2023) | 5.1% | 16.3% | 2.1% | 5.0% | 24.2% |
| YOLOv6-S v3.0 | 6.0% | 17.9% | 2.6% | 5.7% | **41.4%** |
| YOLOv7-T (Wang et al., 2023) | 5.3% | 17.1% | 1.5% | 5.2% | 19.2% |
| YOLOv8-M (Jocher et al., 2023) | 5.5% | 17.2% | 2.2% | 5.3% | 26.5% |
| YOLOv8-p2-S (Jocher et al., 2023) | 6.7% | 20.6% | 2.6% | 6.3% | 27.6% |
| YOLOv9-M (Wang, Yeh, & Liao, 2024c) | 6.0% | 18.5% | 2.5% | 5.7% | 34.5% |
| YOLOv10-B (Wang, Chen, et al., 2024) | 6.2% | 18.7% | 2.6% | 5.9% | 31.6% |
| YOLOX-M (Ge et al., 2021) | 6.1% | 18.8% | 2.5% | 5.9% | 34.5% |
| Gold YOLO-S (Wang, He, et al., 2024) | 4.5% | 14.5% | 1.8% | 4.4% | 21.7% |
| DAMO YOLO-S (Xu, Jiang, et al., 2023) | 5.2% | 16.6% | 1.9% | 4.9% | 34.4% |
| EdgeYOLO (Liu et al., 2023) | 7.1% | 22.2% | 2.8% | 7.0% | 33.8% |
| PPYOLOE-M (Xu et al., 2022) | 4.8% | 14.8% | 1.9% | 4.7% | 29.6% |
| RT-DETR-R18 (Zhao et al., 2024) | 6.5% | 20.0% | 2.3% | 6.0% | 27.4% |
| HIC-YOLO (Tang et al., 2024) | 6.9% | 22.0% | 2.4% | 6.7% | 28.6% |
| PDWT-YOLO (Zhang, Xiong, et al., 2023) | 6.7% | 20.6% | 2.7% | 6.6% | 22.9% |
| **LEAF-YOLO (Ours)** | **7.6%** | **23.0%** | **3.2%** | **7.5%** | 29.1% |

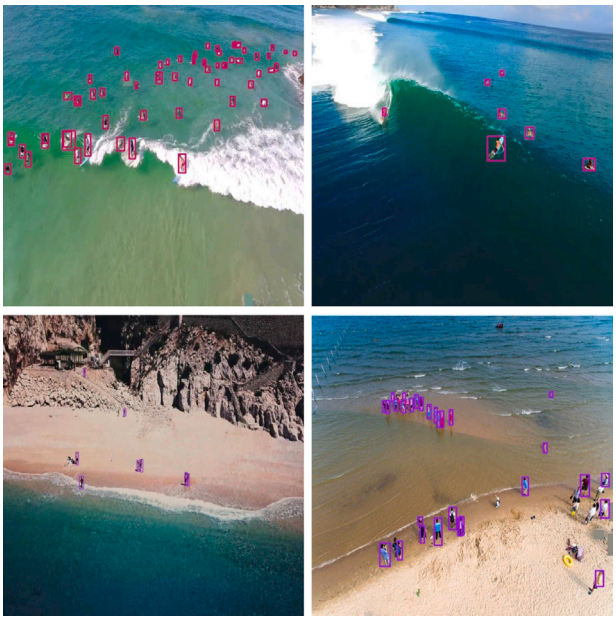

**Fig. 20.** Detection results of LEAF-YOLO on TinyPerson.

surpassing all other models in the comparison. However, its $AP_M$ is only 29.1%, which is lower than models with a larger number of parameters, such as YOLOv6-S, YOLOX-M, DAMO-YOLO-S, and PPYOLOE-M. This discrepancy can be attributed to the architectural design of LEAF-YOLO, which reallocates feature extraction blocks from branch P5 to branch P2, prioritizing the detection of smaller objects over medium-sized ones.

Overall, with general metrics such as $AP_{.50:.95}$ and $AP_{.50}$, LEAF-YOLO consistently outperforms the other methods mentioned in this article, while maintaining a significantly smaller parameter count. Additionally, Fig. 20 highlights LEAF-YOLO's exceptional performance in detecting tiny humans, showcasing its potential for applications requiring small object detection.

## 5. Conclusions and discussions

In this study, we proposed LEAF-YOLO, a real-time and lightweight algorithm specifically designed for small object detection in UAV aerial images. By incorporating an additional detection head for small objects and novel architectural blocks such as LEAF, MGC, CoordBlock, and SPPRFM—LEAF, so YOLO effectively addresses the challenges of multiscale feature fusion and feature extraction in complex backgrounds. The InnerShape-IoU, replacing CIoU, further enhances bounding box predictions for small objects by better aligning box shapes with object dimensions.

**Performance Analysis:** The significant improvement in LEAF-YOLO's accuracy, particularly on small object detection, can be attributed to its tailored architecture. The additional detection head allows the model to capture more details for small-scale objects, which standard object detection heads often miss. Similarly, InnerShape-IoU's shape alignment provides more accurate bounding box regression, especially in cases of dense object distributions. The CoordBlock helps retain spatial information in feature maps, which is critical for UAV-captured images where object localization is key. The SPPRFM module also aids in handling complex, cluttered backgrounds by improving feature pyramid fusion, making LEAF-YOLO effective even in visually complex scenes.

**Technical Comparisons:** LEAF-YOLO and LEAF-YOLO-N outperform other models with fewer than 20 million parameters on the Visdrone2019-DET dataset, achieving 21.9% and 39.7% $AP_{.50}$ for LEAF-YOLO-N, and 28.2% and 48.3% $AP_{.50}$ for LEAF-YOLO, with real-time capabilities exceeding 30 FPS on edge devices. This high efficiency results from the model's architectural optimizations, which effectively balance detection accuracy with computational efficiency. Furthermore, on the TinyPerson dataset, LEAF-YOLO surpasses other lightweight models with more than 20 million parameters, emphasizing the robustness of the proposed structure for small object detection in aerial images.

**Use Cases for LEAF-YOLO and LEAF-YOLO-N:** The two versions of LEAF-YOLO serve distinct purposes. LEAF-YOLO, with its higher accuracy and larger architecture, is well-suited for scenarios that prioritize detection performance over resource constraints, such as surveillance systems with edge-computing devices capable of handling higher computational loads. LEAF-YOLO-N, with its minimal parameter count and high efficiency, is ideal for deployment directly on UAVs or drones where lightweight models and real-time processing are critical. This versatility allows LEAF-YOLO to adapt to various operational requirements within the UAV domain.

**Limitations and Challenges:** Despite its high accuracy and efficiency, LEAF-YOLO may still face challenges under certain conditions. Extreme lighting variations, such as nighttime environments or complex shadows, can hinder feature extraction and reduce detection accuracy. Additionally, dense object distributions or scenes with overlapping small objects complicate bounding box prediction and object separation. While the InnerShape-IoU improves small object detection, further refinement is needed to handle irregular shapes and partial occlusions in high-density scenarios. These challenges emphasize the need for continued development to enhance robustness in demanding UAV environments.

**Future Research Directions:** Future research can expand on LEAF-YOLO's architecture by exploring its deployment on advanced edge-computing hardware, such as Neural Processing Units (NPUs) and Field Programmable Gate Arrays (FPGAs). NPUs would allow hardware-accelerated processing for increased efficiency and power conservation, ideal for resource-constrained UAVs (Chen, Hu, Li, Quan, & Chen, 2021). Additionally, a Spiking Neural Network (SNN) framework on FPGAs could provide an event-driven processing model for low-power, high-efficiency operation in real-time object detection tasks (Yuan, Zhang, Wang, Liu, Pan, & Tang, 2024). Beyond hardware optimization, further algorithmic refinement — such as advanced shape-matching methods or multi-frame analysis for better temporal consistency — could enhance LEAF-YOLO's performance in difficult environments. These research directions will support broader applications of LEAF-YOLO in UAV contexts, expanding its adaptability to more complex real-world scenarios.

Overall, LEAF-YOLO introduces effective approaches to small object detection in UAV aerial imagery by balancing detection accuracy with model efficiency. Future research will aim to expand the model's robustness in challenging conditions and enhance its performance for a wider range of UAV applications.

## CRediT authorship contribution statement

**Nghiem Van Quang:** Model Architecture, Validation, Writing. **Nguyen Huy Hoang:** Conceptualization of this study, Visualization, Editing. **Hoang Minh Son:** Deploying, Software.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## References


Chen, L., Hu, J., Li, X., Quan, F., & Chen, H. (2021). Onboard real-time object detection for UAV with embedded NPU. In *2021 IEEE 11th annual international cONFERENCE on CYBER technology in automation, control, and intelligent systems* (pp. 192–197). http://dx.doi.org/10.1109/CYBER53097.2021.9588193.

Chen, J., Kao, S.-h., He, H., & Zhu, X. (2023). FasterNet: Reducing small object misclassifications by enlarging feature maps in object detection networks. In *2023 IEEE cONFERENCE on computer vision and pattern recognition* (pp. 6231–6239). http://dx.doi.org/10.1109/CVPR52377.2023.00621.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society cONFERENCE on computer vision and pattern recognition* (pp. 886–893 vol. 1). http://dx.doi.org/10.1109/CVPR.2005.177.

Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., & Sun, J. (2021). RepVGG: Making VGG-style ConvNets great again. In *2021 IEEE/CVF conference on computer vision and pattern recognition* (pp. 13728–13737). Los Alamitos, CA, USA: IEEE Computer Society, http://dx.doi.org/10.1109/CVPR46437.2021.01352, URL https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.01352.

Du, D., Zhu, P., & Wen, L. o. (2019). VisDrone-DET2019: The vision meets drone object detection in image challenge results. In *2019 IEEE/CVF international conference on computer vision workshop* (pp. 213–226). http://dx.doi.org/10.1109/ICCVW.2019.00030.

Gao, S.-H., Cheng, M.-M., Zhao, K., Zhang, X.-Y., Yang, M.-H., & Torr, P. (2021). Res2Net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(2), 652–662. http://dx.doi.org/10.1109/TPAMI.2019.2938758.

Gavrilescu, R., Zet, C., Foşalău, C., Skoczylas, M., & Cotovanu, D. (2018). Faster R-CNN: An approach to real-time object detection. In *2018 International conference and exposition on electrical and power engineering* (pp. 0165–0168). http://dx.doi.org/10.1109/ICEPE.2018.8559776.

Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). YOLOX: exceeding YOLO series in 2021. arXiv:2107.08430cs.CV.

Gevorgyan, Z. (2022). SIoU loss: More powerful learning for bounding box regression. arXiv:2205.12740.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE conference on computer vision and pattern recognition* (pp. 580–587). http://dx.doi.org/10.1109/CVPR.2014.81.

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., & Xu, C. (2020). GhostNet: More features from cheap operations. In *2020 IEEE/CVF cONFERENCE on computer vision and pattern recognition* (pp. 1577–1586). http://dx.doi.org/10.1109/CVPR42600.2020.00165.

He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision – ECCV 2014* (pp. 346–361). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-10578-9_23.

Hou, Q., Zhou, D., & Feng, J. (2021). Coordinate attention for efficient mobile network design. In *2021 IEEE/CVF conference on computer vision and pattern recognition* (pp. 13708–13717). http://dx.doi.org/10.1109/CVPR46437.2021.01350.

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In *2018 IEEE/CVF cONFERENCE on computer vision and pattern recognition* (pp. 7132–7141). http://dx.doi.org/10.1109/CVPR.2018.00745.

Jocher, G. (2020). YOLOv5 by Ultralytics. http://dx.doi.org/10.5281/zenodo.3908559, URL https://github.com/ultralytics/yolov5, AGPL-3.0 License.

Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLO. URL https://github.com/ultralytics/ultralytics.

Li, Y., Chen, Y., Wang, N., & Zhang, Z.-X. (2019). Scale-aware trident networks for object detection. In *2019 IEEE/CVF international conference on computer vision* (pp. 6053–6062). http://dx.doi.org/10.1109/ICCV.2019.00615.

Li, D., Hu, J., Wang, C., Li, X., She, Q., Zhu, L., et al. (2021). Involution: Inverting the inherence of convolution for visual recognition. In *2021 IEEE/CVF conference on computer vision and pattern recognition* (pp. 12316–12325). http://dx.doi.org/10.1109/CVPR46437.2021.01214.

Li, C., Li, L., Geng, Y., Jiang, H., Cheng, M., Zhang, B., et al. (2023). YOLOv6 v3.0: A full-scale reloading. arXiv:2301.05586.

Li, Y., Shen, H., Fu, Y., & Wang, K. (2024). A method of dense point cloud SLAM based on improved YOLOV8 and fused with ORB-SLAM3 to cope with dynamic environments. *Expert Systems with Applications*, *255*, Article 124918. http://dx.doi.org/10.1016/j.eswa.2024.124918, URL https://www.sciencedirect.com/science/article/pii/S0957417424017858.

Li, K., Wan, G., Cheng, G., Meng, L., & Han, J. (2020). Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, *159*, 296–307. http://dx.doi.org/10.1016/j.isprsjprs.2019.11.023, URL https://www.sciencedirect.com/science/ARTICLE/pii/S0924271619302825.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., et al. (2014). Microsoft COCO: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), *Computer vision – ECCV 2014* (pp. 740–755). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-10602-1_48.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., et al. (2016). SSD: Single shot MultiBox detector. In B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), *Computer vision – ECCV 2016* (pp. 21–37). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-46448-0_2.

Liu, R., Lehman, J., Molino, P., Such, F. P., Frank, E., Sergeev, A., et al. (2018). An intriguing failing of convolutional neural networks and the CoordConv solution. In *Proceedings of the 32nd international cONFERENCE on neural information processing systems* (pp. 9628–9639). Red Hook, NY, USA: Curran Associates Inc., URL https://dl.acm.org/doi/10.5555/3327546.3327630.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., et al. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF international conference on computer vision* (pp. 9992–10002). Los Alamitos, CA, USA: IEEE Computer Society, http://dx.doi.org/10.1109/ICCV48922.2021.00986, URL https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00986.

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In *2018 IEEE/CVF conference on computer vision and pattern recognition* (pp. 8759–8768). http://dx.doi.org/10.1109/CVPR.2018.00913.

Liu, S., Zha, J., Sun, J., Li, Z., & Wang, G. (2023). EdgeYOLO: An edge-real-time object detector. In *2023 42nd Chinese control conference* (pp. 7507–7512). http://dx.doi.org/10.23919/CCC58697.2023.10239786.

Ma, N., Zhang, X., Liu, M., & Sun, J. (2021). Activate or not: Learning customized activation. In *2021 IEEE/CVF conference on computer vision and pattern recognition* (pp. 8028–8038). http://dx.doi.org/10.1109/CVPR46437.2021.00794.

Muhammad, M. B., & Yeasin, M. (2020). Eigen-CAM: Class activation map using principal components. In *2020 International joint conference on neural networks* (pp. 1–7). http://dx.doi.org/10.1109/IJCNN48605.2020.9206626.

Nguyen, H. H., Nghiem, V. Q., Hoang, M. S., Nghiem, T. K., & Dang, N. M. (2024). A novel variant of Yolov7-tiny for object detection on aerial vehicle images. In H. Sharma, V. Shrivastava, A. K. Tripathi, & L. Wang (Eds.), *Communication and intelligent systems* (pp. 253–265). Singapore: Springer Nature Singapore, http://dx.doi.org/10.1007/978-981-97-2053-8_19.

Nguyen, H. H., Ta, T. N., Nguyen, N. C., Bui, V. T., Pham, H. M., & Nguyen, D. M. (2021). YOLO based real-time human detection for smart video surveillance at the edge. In *2020 IEEE eighth international cONFERENCE on communications and electronics* (pp. 439–444). http://dx.doi.org/10.1109/ICCE48956.2021.9352144.

Nguyen, H. H., Trung Le, Q., Nghiem, V. Q., Son Hoang, M., & Pham, D. A. (2023). A novel violence detection for drone surveillance system. In *2023 international cONFERENCE on communication, circuits, and systems* (pp. 1–6). http://dx.doi.org/10.1109/IC3S57698.2023.10169405.

Park, J., Woo, S., Lee, J.-Y., & Kweon, I. S. (2018). BAM: bottleneck attention module. arXiv:1807.06514[cs.CV].

Pool, N., & Vatsavai, R. R. (2018). Deformable part models for complex object detection in remote sensing imagery. In *Proceedings of the 7th ACM sIGSPATIAL international workshop on analytics for big geospatial data (bigSpatial)*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3282834.3282843.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE cONFERENCE on computer vision and pattern recognition* (pp. 779–788). http://dx.doi.org/10.1109/CVPR.2016.91.

Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv:1804.02767.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. In *2019 IEEE/CVF conference on computer vision and pattern recognition* (pp. 658–666). http://dx.doi.org/10.1109/CVPR.2019.00075.

Tahir, N. U. A., Long, Z., Zhang, Z., Asim, M., & ELAffendi, M. (2024). PVswin-YOLOv8s: UAV-based pedestrian and vehicle detection for traffic management in smart cities using improved YOLOv8. *Drones*, *8*(3), http://dx.doi.org/10.3390/drones8030084, URL https://www.mdpi.com/2504-446X/8/3/84.

Tang, S., Zhang, S., & Fang, Y. (2024). HIC-YOLOv5: Improved YOLOv5 for small object detection. In *2024 IEEE international cONFERENCE on robotics and automation* (pp. 6614–6619). http://dx.doi.org/10.1109/ICRA57147.2024.10610273.

Tao, H., Zheng, Y., Wang, Y., Qiu, J., & Stojanovic, V. (2024). Enhanced feature extraction YOLO industrial small object detection algorithm based on receptive-field attention and multi-scale features. *Measurement Science and Technology*, *35*(10), Article 105023. http://dx.doi.org/10.1088/1361-6501/ad633d.

Tong, Z., Chen, Y., Xu, Z., & Yu, R. (2023). Wise-IoU: Bounding box regression loss with dynamic focusing mechanism. arXiv:2301.10051.

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 *2023 IEEE/CVF cONFERENCE on computer vision and pattern recognition* (pp. 7464–7475). http://dx.doi.org/10.1109/CVPR52729.2023.00721.

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., et al. (2024). YOLOv10: Real-time end-to-end object detection. arXiv:2405.14458.

Wang, C., He, W., Nie, Y., Guo, J., Liu, C., Han, K., et al. (2024). Gold-YOLO: Efficient object detector via gather-and-distribute mechanism. In *Proceedings of the 37th international cONFERENCE on neural information processing systems*. Red Hook, NY, USA: Curran Associates Inc., URL https://dl.acm.org/doi/10.5555/3666122.3668346.

Wang, C.-Y., Mark Liao, H.-Y., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., & Yeh, I.-H. (2020). Cspnet: A new backbone that can enhance learning capability of CNN. In *2020 IEEE/CVF conference on computer vision and pattern recognition workshops* (pp. 1571–1580). http://dx.doi.org/10.1109/CVPRW50498.2020.00203.

Wang, C.-Y., Yeh, I.-H., & Liao, H. (2021). You only learn one representation: Unified network for multiple tasks. *Journal of Information Science and Engineering*, *39*, 691–709. http://dx.doi.org/10.6688/JISE.202305_39(3).0015, URL https://homepage.iis.sinica.edu.tw/papers/liao/25702-F.pdf.

Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2024c). YOLOv9: Learning what you want to learn using programmable gradient information. arXiv:2402.13616.

Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (2018). CBAM: Convolutional block attention module. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer vision – ECCV 2018* (pp. 3–19). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-01234-2_1.

Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., et al. (2018). DOTA: A large-scale dataset for object detection in aerial images. In *2018 IEEE/CVF conference on computer vision and pattern recognition* (pp. 3974–3983). IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2018.00418.

Xu, X., Jiang, Y., Chen, W., Huang, Y., Zhang, Y., & Sun, X. (2023). DAMO-YOLO: A report on real-time object detection design. arXiv:2211.15444.

Xu, S., Wang, X., Lv, W., Chang, Q., Cui, C., Deng, K., et al. (2022). PP-YOLOE: An evolved version of YOLO. arXiv:2203.16250.

Xu, H., Zheng, W., Liu, F., Li, P., & Wang, R. (2023). Unmanned aerial vehicle perspective small target recognition algorithm based on improved YOLOv5. *Remote Sensing*, *15*(14), http://dx.doi.org/10.3390/rs15143583, URL https://www.mdpi.com/2072-4292/15/14/3583.

Yang, Z., Wang, X., & Li, J. (2021). EIoU: An improved vehicle detection algorithm based on VehicleNet neural network. *Journal of Physics: Conference Series*, *1924*(1), Article 012001. http://dx.doi.org/10.1088/1742-6596/1924/1/012001.

Yu, X., Gong, Y., Jiang, N., Ye, Q., & Han, Z. (2020). Scale match for tiny person detection. In *2020 IEEE winter cONFERENCE on applications of computer vision* (pp. 1246–1254). http://dx.doi.org/10.1109/WACV45572.2020.9093394.

Yu, H., Li, G., Zhang, W., Huang, Q., Du, D., Tian, Q., et al. (2020). The unmanned aerial vehicle benchmark: Object detection, tracking and baseline. *International Journal of Computer Vision*, *128*, 1141–1159. http://dx.doi.org/10.1007/s11263-019-01266-1.

Yuan, M., Zhang, C., Wang, Z., Liu, H., Pan, G., & Tang, H. (2024). Trainable spiking-YOLO for low-latency and high-performance object detection. *Neural Networks*, *172*, Article 106092. http://dx.doi.org/10.1016/j.neunet.2023.106092, URL https://www.sciencedirect.com/science/ARTICLE/pii/S0893608023007530.

Yue, M., Zhang, L., Huang, J., & Zhang, H. (2024). Lightweight and efficient tiny-object detection based on improved YOLOv8n for UAV aerial images. *Drones*, *8*(7), http://dx.doi.org/10.3390/drones8070276.

Zhang, Z. (2023). Drone-YOLO: An efficient neural network method for target detection in drone images. *Drones*, *7*(8), http://dx.doi.org/10.3390/drones7080526, URL https://www.mdpi.com/2504-446X/7/8/526.

Zhang, L., Xiong, N., Pan, X., Yue, X., Wu, P., & Guo, C. (2023). Improved object detection method utilizing YOLOv7-tiny for unmanned aerial vehicle photographic imagery. *Algorithms*, *16*(11), http://dx.doi.org/10.3390/a16110520, URL https://www.mdpi.com/1999-4893/16/11/520.

Zhang, H., Xu, C., & Zhang, S. (2023). Inner-iou: more effective intersection over union loss with auxiliary bounding box. arXiv:2311.02877[cs.CV].

Zhang, H., & Zhang, S. (2024). Shape-iou: more accurate metric considering bounding box shape and scale. arXiv:2312.17663[cs.CV].

Zhao, Q., Liu, B., Lyu, S., Wang, C., & Zhang, H. (2023). TPH-YOLOv5++: Boosting object detection on drone-captured scenarios with cross-layer asymmetric transformer. *Remote Sensing*, *15*(6), http://dx.doi.org/10.3390/rs15061687, URL https://www.mdpi.com/2072-4292/15/6/1687.

Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., et al. (2024). DETRs beat YOLOs on real-time object detection. arXiv:2304.08069.

Zhao, L., & Zhu, M. (2023). MS-YOLOv7:YOLOv7 based on multi-scale for object detection on uav aerial photography. *Drones*, *7*(3), http://dx.doi.org/10.3390/drones7030188, URL https://www.mdpi.com/2504-446X/7/3/188.

Zheng, Z., Wang, P., Ren, D., Liu, W., Ye, R., Hu, Q., et al. (2022). Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Transactions on Cybernetics*, *52*(8), 8574–8586. http://dx.doi.org/10.1109/TCYB.2021.3095305.

Zhu, X., Lyu, S., Wang, X., & Zhao, Q. (2021). TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In *2021 IEEE/CVF international conference on computer vision workshops* (pp. 2778–2788). http://dx.doi.org/10.1109/ICCVW54120.2021.00312.

Zhu, C., Zhu, J., Bu, T., & Gao, X. (2022). Monitoring and identification of road construction safety factors via UAV. *Sensors*, *22*(22), http://dx.doi.org/10.3390/s22228797, URL https://www.mdpi.com/1424-8220/22/22/8797.

Zualkernan, I., Abuhani, D. A., Hussain, M. H., Khan, J., & ElMohandes, M. (2023). Machine learning for precision agriculture using imagery from unmanned aerial vehicles (UAVs): A survey. *Drones*, *7*(6), http://dx.doi.org/10.3390/drones7060382, URL https://www.mdpi.com/2504-446X/7/6/382.