sensors

MDPI

Article

Small Object Detection in UAV Images via Shared Semantic Modulation and Cross-Scale Gradient Decoupling

Chengyi Gong and Zhiyong Zhang *

School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen Campus, Shenzhen 518107, China; gongchy5@mail2.sysu.edu.cn
* Correspondence: zhangzhy99@mail.sysu.edu.cn

Abstract

Small objects in unmanned aerial vehicle (UAV) images are prone to detail loss and interference from background textures. To address these problems, we combine shared semantic guidance, gradient decoupling, and offset gating to coordinate the added cross-scale semantic paths and deformable sampling. Our method adds a high-resolution P2 detection branch to YOLO11s and combines detail and semantics through a two-input adaptively spatial feature fusion module (ASFF2). SG-GAFEM generates a shared semantic modulation map from P3 features to gate shallow feature enhancement, and SOG-DCN reuses this map to scale sampling offsets at each spatial location. Cross-scale semantic gradient decoupling (CSGD) preserves forward semantic information flow while blocking direct gradients from the two added semantic inputs back to P3. Across three seeds on the VisDrone validation set, our model achieves a mean mAP@0.5 of 46.672%, exceeding YOLO11s by 7.528 percentage points. After retraining on SOA-PD, the model achieves 75.738% mAP@0.5 on the independent pedestrian test set, the highest value in this evaluation. These results indicate that the method improves overall detection in aerial scenes while balancing accuracy and static computational cost.

Keywords: UAV imagery; small object detection; shared semantic guidance; gradient decoupling; feature enhancement; deformable convolution

1. Introduction

UAVs can rapidly capture aerial images over large areas for traffic monitoring, emergency response, and urban inspection. Because of flight altitude and viewing angle, objects such as pedestrians and vehicles often occupy only a few pixels and can occlude one another in crowded scenes. Accurately recognizing and localizing these small objects is the focus of this study.

Deep learning detectors commonly use region proposals, dense prediction, or end-to-end set prediction. Faster R-CNN [1] generates region proposals, then classifies them and regresses their bounding boxes. SSD [2], RetinaNet [3], and the YOLO family [4–11] predict object categories and locations directly from feature maps. RT-DETR [12] aggregates multiscale features through a hybrid encoder and produces predictions with a Transformer decoder. We use YOLO11s as the baseline.

Aerial and remote sensing detectors often preserve small-object details by extending prediction to additional scales, modeling context, or adapting feature fusion [13–16]. RT-DETR++ [17] preserves feature details through gated resampling and multiscale aggregation. UAVDet [18] uses a feature pyramid designed for extremely small objects

Check for updates

Academic Editor: Zhe-Ming Lu

Received: 20 October 2026
Revised: 19 November 2026
Accepted: 27 November 2026
Published: 29 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

to adjust cross-scale feature propagation. ESOD [19] concentrates high-resolution detection on candidate regions to reduce redundant computation in the background. ACD-DETR [20] introduces structural cues into cross-scale features through boundary-aware fusion, and GLSDet [21] uses deep semantics to guide shallow texture alignment. Building on these approaches, we examine how a single semantic map can modulate both local feature responses and actual sampling positions in deformable convolution, and how to restrict direct gradient flow through the added cross-scale semantic inputs.

We add a high-resolution P2 detection branch to YOLO11s and generate a shared semantic modulation map from P3 features. P2 and P3 denote feature pyramid levels with strides of 4 and 8 relative to the input image. For a 640×640 input, their feature map dimensions are 160×160 and 80×80, respectively. ASFF2 fuses P2 detail and P3 semantics. The shared semantic modulation map supports feature enhancement and offset gating, and CSGD controls backpropagation through the added semantic paths. We make three contributions.

(1) We design SG-GAFEM for feature enhancement with shared semantic guidance. It generates a shared semantic modulation map from deep P3 features and converts the map into a single-channel semantic bias for the P2 gating logits. This position-dependent control of enhancement strength helps limit the incidental amplification of background textures.
(2) We develop semantic offset-gated deformable convolution (SOG-DCN). It reuses the map from SG-GAFEM to generate a position-dependent scaling factor for deformable sampling offsets, giving feature enhancement and local geometric sampling a common semantic input while retaining the native DCNv2 modulation branch.
(3) We propose cross-scale semantic gradient decoupling (CSGD). A shared stop-gradient node at the P3 input to ASFF2 and SG-GAFEM blocks direct backpropagation from both added semantic inputs to P3. This limits the direct effect of the P2 detection loss on updates to the upstream parameters that produce P3 through these two paths. The approach preserves forward semantic information flow, backpropagation along the original feature pyramid paths, and learning in the added modules.

We compare models, conduct component ablations, and analyze the proposed mechanisms on the VisDrone validation set. We also retrain the model on SOA-PD and evaluate it on its independent aerial pedestrian test set. Section 2 reviews related work, Section 3 presents the method, and Section 4 reports the experiments and mechanism analyses. Section 5 discusses applicability and limitations, and Section 6 concludes the paper.

2. Related Work

This work investigates how deep semantics can improve high-resolution representations and local sampling for aerial small objects, and how added cross-scale semantic connections affect backpropagation. Section 2.1 reviews high-resolution detail preservation, cross-scale fusion, and selective feature enhancement. Section 2.2 examines semantic modulation of sampling positions and gradient control after adding cross-scale semantic connections.

2.1. High-Resolution Representations and Cross-Scale Feature Fusion

High-resolution representations can be improved at the prediction layers, through input reconstruction, or during feature propagation. TPH-YOLOv5 [13] adds a high-resolution prediction head. HRDNet [22] assigns inputs of different resolutions to backbones of different depths. Courtrai et al. [23] enhance details with detection-oriented super-resolution, and RemDet [24] improves detail propagation with channel-expanding downsampling, gated feed-forward processing, and reparameterization. To limit high-

resolution computation, QueryDet [25] uses low-resolution features to locate candidates for sparse high-resolution detection. UFPMP-Det [26] packs coarse detections for a second, finer detection stage, whereas ESOD [19] reuses the detector backbone to locate candidate regions and crops local feature patches for sparse detection.

FPN [27] propagates semantics top-down, PANet [28] adds a bottom-up path for localization information, and BiFPN [29] uses weighted bidirectional fusion to adjust each input's contribution. ASFF [30] aligns features at different scales and learns fusion weights at each spatial location. In aerial detection, R-AFPN [31] extracts shallow detail from P2 and fuses it across successive levels. RT-DETR++[17] preserves detail through channel-gated upsampling and downsampling. UAVDet [18] combines convolution and Mamba in the neck, using a feature pyramid designed for extremely small objects to adjust cross-scale information flow.

Cross-scale fusion can also use structural or semantic cues. ACD-DETR [20] supports fusion through edge enhancement and boundary calibration, and GLSDet [21] uses high-level semantics to guide shallow texture alignment. Because high-resolution features also contain background textures, local enhancement must be selective. SE [32] recalibrates channel responses, and CBAM [33] combines channel and spatial attention. In ARSOD-YOLO [15], AFEM generates features using GhostModule [34] and enhances them locally through sigmoid gating and a residual connection. Our ASFF2 applies the position-wise weighting of ASFF [30] to fuse P2 detail with P3 semantics. SG-GAFEM uses a single-channel semantic map from P3 to modulate its local gate.

2.2. Dynamic Sampling and Cross-Scale Semantic Interaction

DCN [35] uses learnable offsets to shift sampling positions away from a regular grid. DCNv2 [36] adds pointwise modulation weights to adjust sampling responses. DCNv3 [37] predicts offsets and weights by channel group and normalizes the weights across sampling points within each group. Offsets determine sampling positions, and modulation weights determine the contribution of each sampling response. Deformable DETR [38] uses query features to predict sampling offsets and attention weights around reference points for each attention head.

For aerial small object detection, AFE-YOLOv8 [16] uses deformable convolution in the backbone to extract multiscale features and adaptive fusion to enhance its small-object branch. I-YOLOv11n [39] uses predicted offset information in spatial attention to incorporate geometry into feature selection. HSAR-DETR [40] uses offset magnitude to modulate local responses while keeping the sampling grid fixed. Our SOG-DCN reuses the P3 semantic map from the enhancement stage to scale predicted offsets at each location and computes sampling coordinates from those scaled offsets. The independent DCNv2 modulation branch weights the sampling responses (Section 3.3.2).

With the cross-scale connections, the P2 detection loss can backpropagate directly to the top-down P3 feature through the added semantic inputs to ASFF2 and SG-GAFEM. Focal Loss [3] downweights the losses of easy examples. UGS [41] stabilizes localization gradients through discretized localization and uncertainty modeling. GradNorm [42] adjusts task weights according to gradient norms and relative training rates, and PCGrad [43] projects conflicting gradients. YOLOv7 [8] supervises intermediate layers through an auxiliary detection head. CSGD retains the original detection loss and its weights, using a stop-gradient operation at the shared P3 input to block direct backpropagation through the two added semantic paths. It preserves forward semantic information flow and backpropagation along the original feature pyramid paths (Section 3.4).

3. Aerial Small Object Detection with Shared Semantic Modulation and Cross-Scale Gradient Decoupling

3.1. Overall Architecture

Figure 1 shows the overall architecture. Outside the dashed outline, we retain the YOLO11s [44] baseline, which predicts on P3, P4, and P5 with strides of 8, 16, and 32, respectively. The outline encloses the added P2 detection branch and the associated P3 fusion paths. The P2 head predicts on a stride-4 feature map to make use of finer spatial detail.

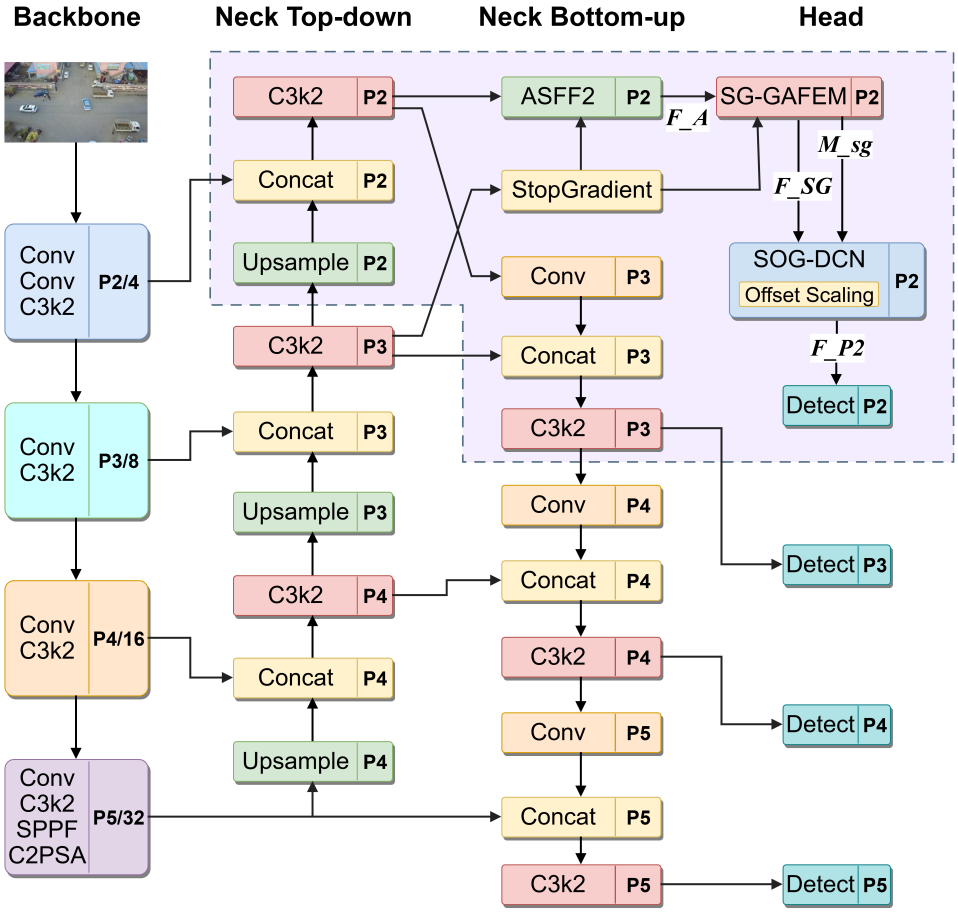


Figure 1. Overall architecture of the proposed model. The light purple region within the dashed outline marks the added P2 detection branch and its associated P3 fusion paths.

Along the P2 branch in Figure 1, the backbone P2 feature and the upsampled P3 feature are first fused to form the P2 detail feature. The detail feature is downsampled for bottom-up fusion at P3 and also feeds ASFF2, a two-input fusion module based on ASFF [30]. ASFF2 aligns the P3 feature to the P2 grid and fuses the two features with spatially varying weights (Section 3.2).

SG-GAFEM generates a shared semantic modulation map from P3 and converts it to a bias for the P2 feature enhancement gate. SOG-DCN takes the enhanced P2 feature and the same map, scales the predicted sampling offsets at each spatial location, and passes its output to the P2 detection head (Section 3.3). During training, CSGD preserves forward semantic information flow from P3 to ASFF2 and SG-GAFEM. It blocks direct gradients from the P2 detection loss back to P3 through these two added semantic paths. The P2 detection loss can still backpropagate to P3 through the path that constructs the P2 detail feature (Section 3.4).

3.2. High-Resolution P2 Branch and Feature Fusion

Small objects occupy few pixels in UAV images, and successive downsampling can weaken their boundary and location cues. We add a stride-4 P2 detection branch to YOLO11s so that the head can use shallow spatial detail on a denser grid. The neck first concatenates the backbone P2 feature with the upsampled P3 feature and forms the P2 detail feature with C3k2. ASFF2 then aligns a separate P3 semantic input to the P2 grid and fuses it with the detail feature. The fusion retains the P2 resolution and learns different combinations of detail and semantic features at each location before gated enhancement and local sampling.

As shown in Figure 1, ASFF2 receives two inputs. The first is the P2 detail feature F_2, the output of C3k2 at the top-down P2 level, with dimensions $C \times H \times W$. The second is the top-down P3 feature after the shared stop-gradient node, $F_3 = \mathrm{SG}(F_3^{\mathrm{raw}})$, with dimensions $2C \times (H/2) \times (W/2)$. The same F_3 provides semantic input to SG-GAFEM (Section 3.4). Here, $C = 64$ and $H \times W$ is the spatial resolution of P2 in the neck. ASFF2 follows ASFF [30] by weighting the relative contributions of P2 detail and P3 semantics at each location.

Each projection, φ_2 or φ_3, consists of a 1×1 convolution, batch normalization, and SiLU, and outputs C channels. The P2 projection retains its spatial dimensions. The P3 projection yields $C \times (H/2) \times (W/2)$ and is aligned to $H \times W$ using bilinear interpolation $U_{H,W}$ with `align_corners=False`. The following expressions give the two aligned features, each of size $C \times H \times W$.

$$\widetilde{F}_2 = \varphi_2(F_2), \qquad \widetilde{F}_3 = U_{H,W}(\varphi_3(F_3))$$

We concatenate the aligned features along the channel dimension, denoted by $[\cdot,\cdot]$, to obtain a $2C \times H \times W$ tensor. A 1×1 convolution $g(\cdot)$ then produces branch scores of size $2 \times H \times W$. At each spatial location, softmax normalizes the scores across the two branches to obtain the weights w_2 and w_3 in Equation (1).

$$[w_2, w_3] = \mathrm{Softmax}(g([\widetilde{F}_2, \widetilde{F}_3])) \tag{1}$$

Each weight map has dimensions $1 \times H \times W$. The weights sum to one at each location and are broadcast across the C channels of their respective features. Equation (2) sums the element-wise weighted features ($\odot$ denotes element-wise multiplication) and combines them with ψ_A, which consists of a 3×3 convolution, batch normalization, and SiLU. The output F_A retains dimensions $C \times H \times W$ and feeds SG-GAFEM.

$$F_A = \psi_A(w_2 \odot \widetilde{F}_2 + w_3 \odot \widetilde{F}_3) \tag{2}$$

3.3. Shared Semantic Modulation

The ASFF2 output retains detail for localization and also contains background texture. The enhancement gate therefore varies spatially. Local sampling also requires a displacement for each sampling point relative to the regular grid. To use a common cross-scale semantic input for both operations, SG-GAFEM generates a shared semantic modulation map from P3 and converts it into a spatial bias for the local enhancement gate. SOG-DCN reuses that map to scale the predicted sampling offsets at each location.

3.3.1. Shared Semantic Map Generation and Gated Feature Enhancement

Figure 2 shows SG-GAFEM. The module adopts the local gating and residual enhancement structure of AFEM [15]. Two independent GhostConv [34] branches produce the candidate feature and the gating input. The lower-right inset shows the GhostConv block. The dashed SiLU blocks apply only to the candidate feature branch.

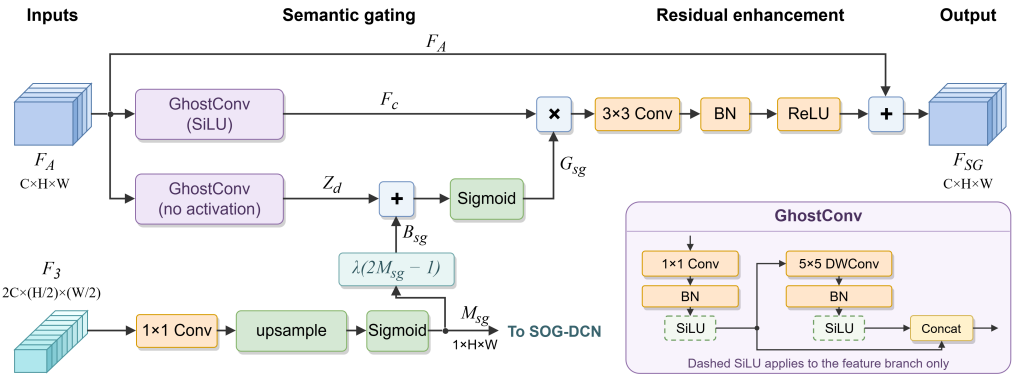


Figure 2. SG-GAFEM architecture with the GhostConv block shown in the inset.

The two GhostConv branches process F_A to generate the candidate feature F_c and the detail-based gating logits Z_d in Equation (3). In each branch, a 1×1 convolution produces primary features, and a 5×5 depthwise convolution generates additional features from them. Both convolutions are followed by batch normalization. The features are concatenated along the channel dimension to yield 64 channels. The candidate branch $\varphi_f(\cdot)$ applies SiLU after both normalization layers, and the gating branch $\varphi_g(\cdot)$ omits both activations. In GhostAFEM, a sigmoid maps the logits to the local gating weights G_0. The element-wise weighted feature passes through $\psi_E(\cdot)$, consisting of a 3×3 convolution, batch normalization, and ReLU, and the result is added to the input (Equation (4)).

$$F_c = \varphi_f(F_A), \qquad Z_d = \varphi_g(F_A) \tag{3}$$

$$\begin{aligned} G_0 &= \sigma(Z_d) \\ F_{\text{GAFEM}} &= F_A + \psi_E(F_c \odot G_0) \end{aligned} \tag{4}$$

SG-GAFEM projects F_3 to a single channel with a 1×1 convolution and aligns it to $H \times W$ by bilinear interpolation with `align_corners=False`. A sigmoid then produces the shared semantic modulation map M_{sg} with dimensions $B \times 1 \times H \times W$, where B is the batch size (Equation (5)).

$$Z_s = U[\text{Conv}_{1\times1}(F_3)], \qquad M_{\text{sg}} = \sigma(Z_s) \tag{5}$$

The map yields a zero-centered semantic bias B_{sg}. The bias is broadcast across channels and added to Z_d. A sigmoid then yields the feature enhancement gate G_{sg} in Equation (6). We set $\lambda = 0.25$, so the bias lies in $(-0.25, 0.25)$.

$$B_{\text{sg}} = \lambda(2M_{\text{sg}} - 1), \qquad G_{\text{sg}} = \sigma(Z_d + B_{\text{sg}}) \tag{6}$$

For a given Z_d, a location with $M_{\text{sg}} > 0.5$ receives a positive bias that raises its gate above the baseline value. A location with $M_{\text{sg}} < 0.5$ receives a negative bias that lowers the gate. The bias is shared across channels at a given location, and each channel retains its own gating logit Z_d. At $M_{\text{sg}} = 0.5$, $B_{\text{sg}} = 0$ and G_{sg} equals the baseline gate in Equation (4).

Equation (7) replaces the baseline gate in Equation (4) with G_{sg} to obtain the enhanced feature F_{SG}.

$$F_{\text{SG}} = F_A + \psi_E(F_c \odot G_{\text{sg}}) \tag{7}$$

The semantic projection weights and bias are initialized to zero, giving $M_{\text{sg}} = 0.5$ and $B_{\text{sg}} = 0$ initially. The enhancement gate therefore starts at the baseline in Equation (4). F_{SG} retains 64 channels and the same spatial dimensions and serves as the sampling input to SOG-DCN. SOG-DCN uses the same M_{sg} to scale the predicted sampling offsets.

The detection loss provides indirect supervision for the map, without separate object saliency supervision.

3.3.2. Semantic Offset Gating and Local Sampling

Figure 3 shows the computation and sampling geometry of SOG-DCN. The module samples the enhanced feature F_{SG}. Its offset and modulation prediction branch produces offsets and modulation scores, and M_{sg} scales the predicted offsets at each location.

The 3×3 convolution on the left of Figure 3 produces a $27 \times H \times W$ tensor. It is split into an $18 \times H \times W$ tensor of offsets Δp and a $9 \times H \times W$ tensor of modulation scores Z_m, corresponding to two offset components and one modulation score for each of the nine sampling points. The sigmoid function σ converts the modulation scores Z_m into modulation weights $m = \sigma(Z_m)$, which are passed directly to DCNv2. Before semantic offset scaling, DCNv2 [36] samples as in Equation (8).

$$y(p_0) = \sum_k w_k \cdot x(p_0 + p_k + \Delta p_k) \cdot m_k \tag{8}$$

Here, $x(\cdot)$ and $y(\cdot)$ denote the input and output features. p_0 is the output position, p_k is the relative coordinate of the kth point in the regular 3×3 sampling grid, and w_k is the corresponding kernel weight. Δp_k and m_k are the predicted offset and modulation weight for that point.

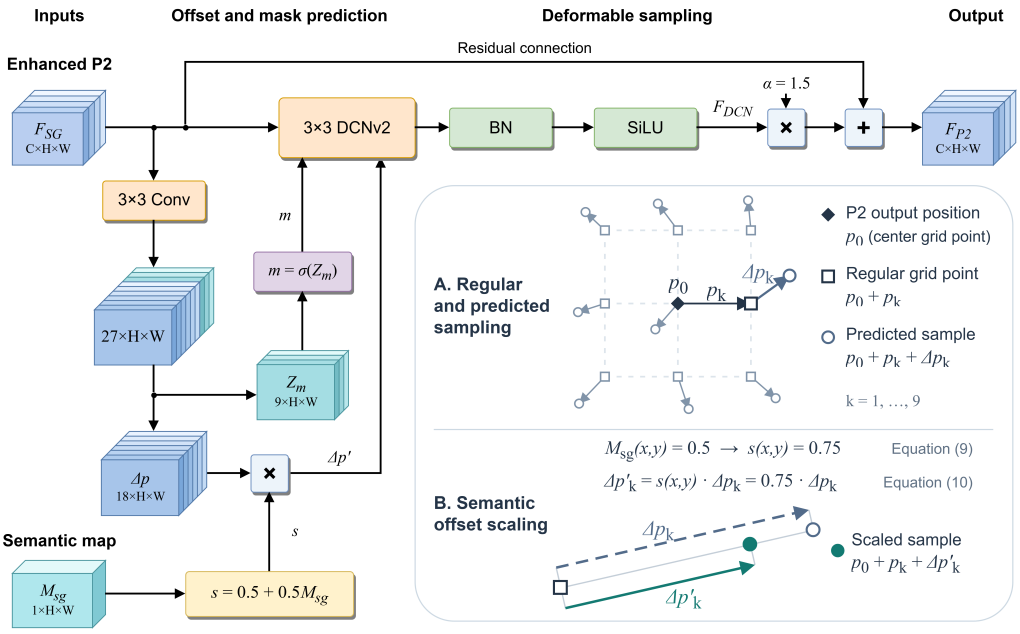


Figure 3. SOG-DCN architecture and sampling geometry.

Figure 3A shows the regular sampling points and their predicted positions for one P2 output location. The nine sampling points on the regular 3×3 grid are labeled by the index $k = 1, \ldots, 9$, which runs from 1 to 9. The dark diamond marks the center p_0, the open squares mark the regular positions $p_0 + p_k$, and the open circles mark the predicted positions $p_0 + p_k + \Delta p_k$. At each spatial location (x, y) on the P2 feature map, Equation (9) converts the semantic modulation value $M_{sg}(x, y)$ into the offset-scaling factor $s(x, y)$.

$$s(x, y) = 0.5 + 0.5 M_{sg}(x, y) \tag{9}$$

Equation (10) uses the same scaling factor for all nine two-dimensional offsets at a P2 output location. The factor $s(x, y)$ is broadcast to 18 offset channels, two per sampling point.

$$\Delta p'_k(x, y) = s(x, y) \cdot \Delta p_k(x, y) \tag{10}$$

Because $M_{sg} \in (0, 1)$, the factor satisfies $s \in (0.5, 1)$. For a given original offset, larger map values yield less contraction. In Figure 3B, $M_{sg}(x, y) = 0.5$ gives $s(x, y) = 0.75$. The scaled point lies between the respective regular and predicted points. The offset retains its direction and has 75% of its original magnitude. The final magnitude depends on both the original offset and the scaling factor. Section 4.5.2 compares scaling rules at inference with the weights of each of three checkpoints held fixed. Position-dependent scaling improves mAP@0.5 by averages of 0.161 and 0.129 percentage points over global constant scaling and per-image mean scaling, respectively.

DCNv2 uses the scaled offsets $\Delta p'$ and the modulation weights m. Batch normalization (BN) and SiLU then yield the sampling branch output F_{DCN}, as shown in Equation (11).

$$F_{DCN} = \mathrm{SiLU}\{\mathrm{BN}[\mathrm{DCNv2}(F_{SG}; \Delta p', m)]\} \tag{11}$$

In the residual path at the top of Figure 3, the sampling branch output is multiplied by the fixed coefficient α and added to F_{SG}. The result feeds the P2 detection head (Equation (12)).

$$F_{P2} = F_{SG} + \alpha F_{DCN}, \qquad \alpha = 1.5 \tag{12}$$

The deformable convolution uses a 3×3 kernel, stride 1, and padding 1. The prediction convolution has zero-initialized weights and bias, yielding initial offsets of zero and modulation weights of 0.5. The initial semantic map value of 0.5 gives a scaling factor of 0.75. Sampling begins on the regular grid because the initial offsets are zero.

The output F_{P2} feeds only the P2 detection head. The bottom-up path uses the P2 detail feature before enhancement.

3.4. Cross-Scale Semantic Gradient Decoupling

The P3 fusion input to ASFF2 and the P3 semantic input to SG-GAFEM add two cross-scale semantic paths to the P2 branch. These paths can carry gradients from the P2 detection loss directly to the parameters upstream of P3. Cross-scale semantic gradient decoupling (CSGD) places a shared stop-gradient node before the two P3 projections. The node preserves forward semantic information flow and blocks direct gradient feedback through the added inputs. Figure 4 shows the node and which paths retain backpropagation.

The shared node uses PyTorch's `detach()`. It takes the top-down P3 feature F_3^{raw} and outputs $F_3 = \mathrm{SG}(F_3^{raw})$ for both projection branches. It preserves the feature values in the forward pass (Equation (13)). In the backward pass, the gradient from the node to its input is zero for any gradient g_{out} reaching F_3 (Equation (14)).

$$\mathrm{SG}_{fwd}(F_3^{raw}) = F_3^{raw} \tag{13}$$

$$\mathrm{SG}_{bwd}(g_{out}) = 0 \tag{14}$$

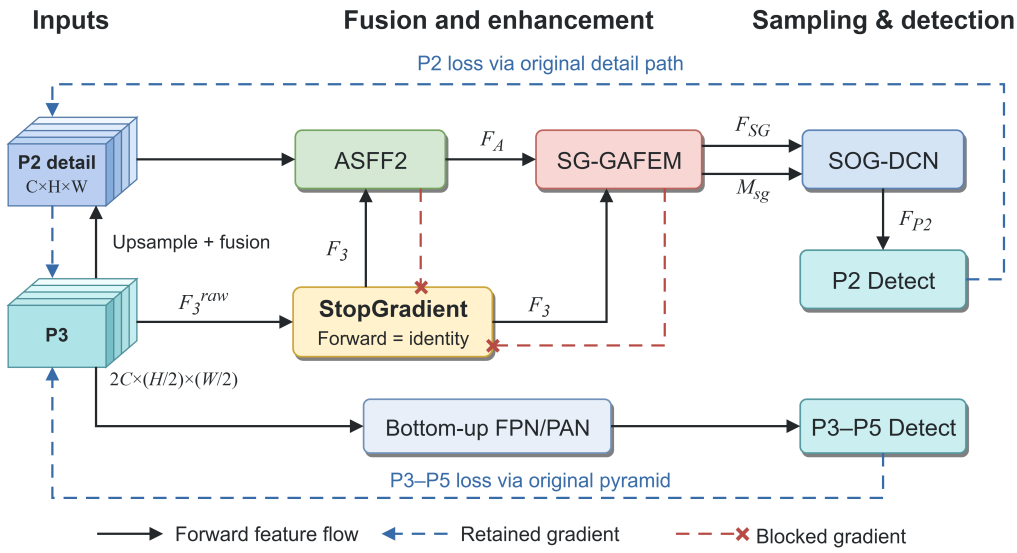


Figure 4. Forward and backward paths in CSGD.

The two dashed red lines in Figure 4 mark the direct gradient paths from the P3 projection in ASFF2 and the semantic projection in SG-GAFEM. Both paths stop at the shared node. Because the node precedes the projections, both projections and their downstream modules still receive gradients from the detection loss. The SG-GAFEM enhancement gate and SOG-DCN offset scaling both use M_{sg}, so gradients from both operations can update the shared semantic projection. The node stops these gradients before they reach P3 through the semantic input.

The dashed blue lines in Figure 4 mark two paths through which gradients can still reach the top-down P3 feature. The P2 detail construction path upsamples the undetached F_3^{raw} and fuses it with the backbone P2 feature. Gradients from the P2 detection loss can therefore still reach P3 through this path. Gradients from the P3–P5 detection losses flow through the bottom-up feature pyramid.

CSGD retains the original detection loss and weights. It sets a backpropagation boundary without assuming that gradient directions conflict or projecting them. It adds no learnable parameters and preserves the forward computation at inference. Section 4.5.1 compares the different gradient path configurations.

4. Experiments and Results

We first describe the datasets, training setup, and metrics. We compare complete detectors on the VisDrone validation set, then retrain the compared models on SOA-PD and evaluate aerial pedestrian detection on its independent test set. We return to the VisDrone validation set for component ablations, gradient path diagnostics, and offset modulation analysis. Finally, we visualize detections in daytime and nighttime aerial examples.

4.1. Datasets and Experimental Setup

We use VisDrone2019-DET [45] for the main experiments and SOA-PD [46] for supplementary validation, following the original splits of both datasets. VisDrone2019-DET was released by the AISKYEYE team at the Machine Learning and Data Mining Laboratory of Tianjin University and contains 10 object categories. The UAV images were captured at different altitudes, viewing angles, and lighting conditions across urban roads, residential areas, and public spaces. They contain many small, densely distributed objects, substantial scale variation, and frequent occlusion (Figure 5). We train on 6471 images, select checkpoints on 548 validation images, and use the validation set for the overall comparisons

and ablations. The longer side of the original validation images ranges from 960 to 1920 pixels, with a median of 1360 pixels. The 1610-image test-dev subset is excluded from training, tuning, and reporting. SOA-PD was released on Zenodo by the KIOS Research and Innovation Centre of Excellence at the University of Cyprus. Its images, captured on campus and during civil defense exercises, are used for single-class pedestrian detection in top-down UAV imagery. We retrain the compared models on 2092 training images, select checkpoints on 523 validation images, and report final results on 521 independent test images.

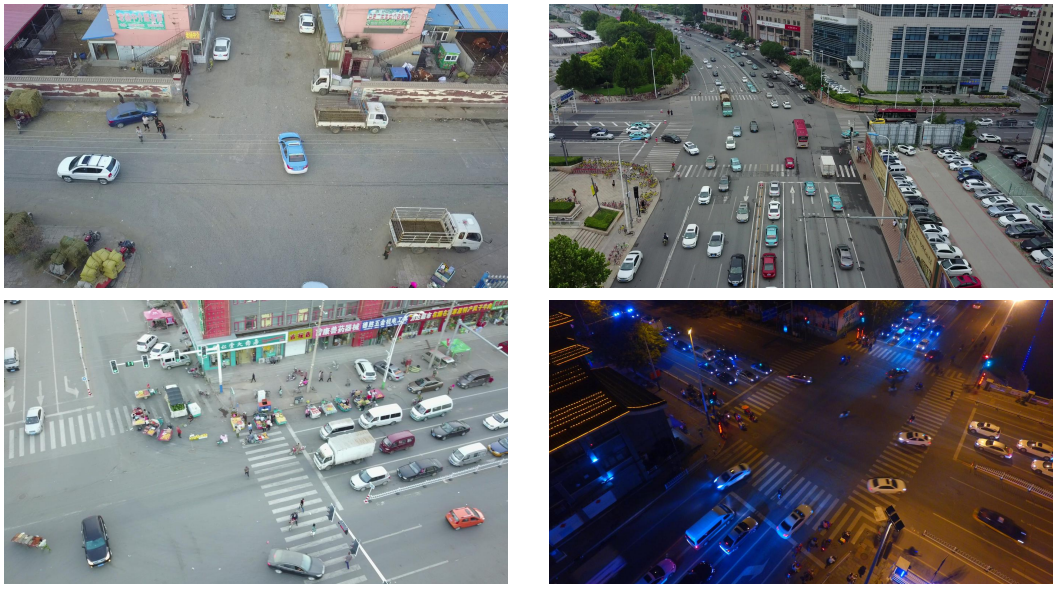


Figure 5. Representative scenes from the VisDrone dataset.

Training on both datasets uses the same Linux server with 4 NVIDIA GeForce RTX 4090 GPUs, each with approximately 24 GiB of memory. The software environment consists of Python 3.10.13, PyTorch 2.4.1, CUDA 12.1, and Ultralytics 8.3.59. The main experiments use 640 × 640 inputs. For the resolution comparison in Section 4.4, YOLO11s, YOLO11s + P2, and the proposed model are each trained and evaluated on VisDrone at 960 × 960, starting from random initialization with seed 0. Both input settings use a global batch size of 16 and a maximum of 300 epochs. In the main 640 × 640 experiments, the proposed model loads pretrained YOLO11s weights. The `auto` optimizer setting selects stochastic gradient descent (SGD), with a learning rate that decreases linearly from 0.01 to a target of 0.0001. The remaining settings follow the Ultralytics defaults.

4.2. Evaluation Metrics

We report overall precision (P), recall (R), mAP@0.5, and mAP@0.5:0.95 following the Ultralytics protocol. P and R are macro-averaged across classes at IoU 0.50, using the confidence threshold at the peak of the smoothed class-averaged F1 curve. The primary detection metric is mAP@0.5 at an IoU threshold of 0.50. GFLOPs are estimated statically with THOP for one forward pass at 640 × 640. The cost of the custom deformable operator is approximate.

Small objects are defined by a ground-truth box area below 32^2 square pixels in the original image. We compute AP_{small} and $Recall_{small}$ for all models using the same custom evaluator. AP_{small} integrates the continuous precision envelope and averages over IoU thresholds from 0.50 to 0.95 in steps of 0.05 and over classes containing small objects. $Recall_{small}$ is the final class-macro-averaged recall at IoU 0.50, with a confidence threshold of 0.001 and a maximum of 300 predictions per image. The evaluator prioritizes matches to

unmatched small ground-truth boxes of the same class and ignores predictions matched to other scales and unmatched predictions outside the small scale.

4.3. Overall Model Comparisons and Repeated Runs

4.3.1. Model Comparison on VisDrone

On VisDrone, both the proposed model and the YOLO11s baseline are trained with seeds 0, 1, and 2. Their mean mAP@0.5 values are 46.672% and 39.143%, with sample standard deviations of 0.120 and 0.087 percentage points, respectively. The proposed model achieves a higher mAP@0.5 for every matched seed, with an average gain of 7.528 percentage points.

Table 1 compares the seed 0 results for the proposed model, five YOLO baselines, two YOLO-based methods, and two RT-DETR variants. AFE-YOLOv8s [16] and ARSOD-YOLOv8n [15] were reimplemented in Ultralytics 8.3.59 following their original papers. The supplementary comparisons use the same VisDrone splits, 640 × 640 inputs, a global batch size of 16, seed 0, and 300 training epochs. For our model, the seed 0 checkpoint gives the highest mAP@0.5 across the three seeds. RT-DETR-l [12] serves as a general-purpose DETR baseline. The RT-DETR++[17] reimplementation uses a ResNet-18vd backbone and implements AU, AD, and CSP-PAC in the P3–P5 encoder.

Table 1. Detection accuracy and computational cost on the VisDrone validation set.

Model	P (%)	R (%)	mAP@0.5 (%)	mAP@0.5:0.95 (%)	AP_{small} (%)	$Recall_{small}$ (%)	GFLOPs
YOLOv5s	50.579	38.198	39.357	23.560	10.938	35.698	24.213
YOLOv8n	46.378	34.863	35.360	20.635	8.449	30.970	8.204
YOLOv8s	51.137	38.162	39.390	23.603	10.404	34.135	28.817
YOLOv10s	51.329	37.658	39.013	23.484	12.258	51.635	25.110
YOLO11s	51.429	38.062	39.123	23.547	10.750	33.936	21.568
AFE-YOLOv8s	52.211	41.483	42.091	25.394	12.388	38.845	32.631
ARSOD-YOLOv8n	47.233	34.991	35.743	20.244	8.636	32.039	28.145
RT-DETR-l	60.292	46.490	46.451	28.309	17.360	50.275	108.032
RT-DETR++ (reimpl.)	57.838	42.957	43.689	26.368	15.445	46.759	60.301
Proposed model	57.243	43.984	46.807	28.826	16.271	47.635	34.258

P and R are measured at the peak of the smoothed mean F1 curve. Section 4.2 defines AP_{small} and $Recall_{small}$. GFLOPs correspond to 640 × 640 inputs.

In Table 1, the proposed model achieves the highest mAP@0.5 of 46.807%, exceeding YOLO11s by 7.684 percentage points and the reimplementations of AFE-YOLOv8s and ARSOD-YOLOv8n by 4.716 and 11.064 percentage points, respectively. Its static computational cost is 34.258 GFLOPs, compared with 32.631 GFLOPs for AFE-YOLOv8s.

RT-DETR++ uses channel gating to modulate cross-scale sampling in its P3–P5 encoder and parallel dilated convolutions in CSP-PAC to aggregate multiscale features. We add a P2 detection branch to YOLO11s and use a shared P3 semantic map to modulate feature enhancement and deformable sampling. The overall comparison reflects differences between complete models. Sections 4.4 and 4.5 examine the individual mechanisms through controlled comparisons.

The proposed model exceeds RT-DETR-l and the RT-DETR++ reimplementation by 0.356 and 3.118 percentage points in mAP@0.5, respectively. The RT-DETR++ reimplementation has 20.555 M parameters, approximately 37.4% fewer than RT-DETR-l, and requires 60.301 GFLOPs. On the same device, RT-DETR-l, the RT-DETR++ reimplementation, and the proposed model achieve model-forward speeds of 24.60, 36.94, and 49.88 FPS, respectively. The proposed model has 9.723 M parameters and requires 34.258 GFLOPs.

Figures 6 and 7 show the class-wise precision–recall (PR) curves for YOLO11s and the proposed model, respectively. The class values in the legends represent AP@0.5, and the

thick blue line shows the average of the class-wise PR curves. Both figures use seed 0 checkpoints and an IoU threshold of 0.50.

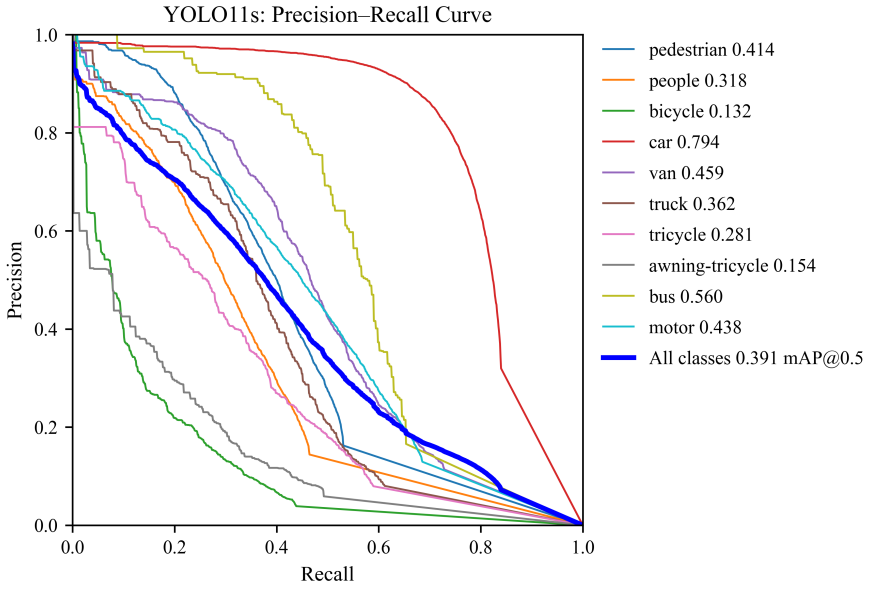


Figure 6. Class-wise precision–recall curves of YOLO11s on VisDrone.

The proposed model achieves AP@0.5 values of 85.2% for cars and 63.8% for buses, compared with 20.3% for bicycles and 17.7% for awning-tricycles. The training set has 10,480 bicycle and 3246 awning-tricycle annotations, considerably fewer than the 144,867 car annotations. Using the original-image area criterion in Section 4.2, small objects account for 81.82% of bicycles and 57.33% of awning-tricycles in the validation set. AP@0.5 remains low for these classes, which combine high proportions of small instances with relatively few training annotations.

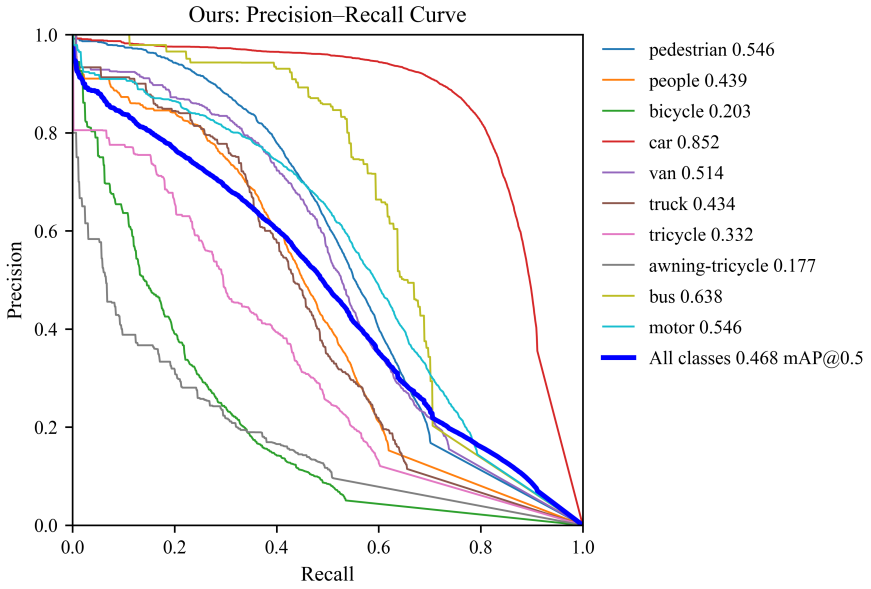


Figure 7. Class-wise precision–recall curves of the proposed model on VisDrone.

4.3.2. Independent Test on SOA-PD

We retrain YOLOv5s, YOLOv8s, YOLOv10s, YOLO11s, RT-DETR-l, and the proposed model on SOA-PD and evaluate them on the independent test set (Table 2). All models

train for 300 epochs with 640 × 640 inputs, a global batch size of 16, and seed 0. RT-DETR-l uses an HGNetv2-L backbone with its corresponding pretrained weights and is trained with AdamW at an initial learning rate of 0.0001, with AMP and the `deterministic` setting disabled. The models have the same training budget and retain their respective optimizers, pretrained weights, and native processing pipelines.

Table 2. Pedestrian detection results on the independent SOA-PD test set.

Model	P (%)	R (%)	mAP@0.5 (%)	mAP@0.5:0.95 (%)	AP_{small} (%)	$Recall_{small}$ (%)
YOLOv5s	70.710	44.584	52.340	24.109	14.414	48.761
YOLOv8s	70.648	44.833	53.178	24.723	14.823	49.343
YOLOv10s	69.747	44.398	52.220	24.563	14.773	48.920
YOLO11s	70.244	43.558	51.772	24.049	14.154	47.670
RT-DETR-l	78.898	70.319	73.182	31.711	28.300	84.781
Proposed model	77.596	70.583	75.738	34.040	27.573	82.006

All results use seed 0. Section 4.2 defines AP_{small} and $Recall_{small}$.

In Table 2, the proposed model achieves 75.738% mAP@0.5, the highest in this comparison. It exceeds YOLO11s by 23.966 percentage points and RT-DETR-l by 2.556 percentage points.

4.4. Component Ablations

The ablations use the fixed settings in Section 3. When SG-GAFEM is enabled, the semantic bias strength is $\lambda = 0.25$. When SOG-DCN is enabled, offset scaling follows $s = 0.5 + 0.5M_{sg}$, with a residual scaling factor of $\alpha = 1.5$. We compare the incremental contributions of component combinations under these settings. Section 4.5.2 further compares position-dependent and uniform scaling rules.

Table 3 summarizes eight ablation configurations on the VisDrone validation set. A is YOLO11s, and B adds only P2. C and D add SG-GAFEM and ASFF2 in succession. E and F decouple one and both paths, respectively. G adds SOG-DCN to D, and H uses both dual-path CSGD and SOG-DCN. The complete model applies ASFF2, SG-GAFEM, and SOG-DCN in that order.

Table 3. Component ablations on the VisDrone validation set.

Config.	P2	ASFF2	SG-GAFEM	CSGD	SOG-DCN	mAP@0.5 (%)	mAP@0.5:0.95 (%)	AP_{small} (%)	$Recall_{small}$ (%)
A	×	×	×	×	×	39.123	23.547	10.750	33.936
B	✓	×	×	×	×	43.808	26.644	14.935	44.776
C	✓	×	✓	×	×	44.724	27.258	15.057	45.368
D	✓	✓	✓	×	×	44.829	27.412	15.556	46.835
E	✓	✓	✓	SG	×	45.089	27.716	15.361	46.252
F	✓	✓	✓	Dual	×	45.857	28.040	16.085	47.476
G	✓	✓	✓	×	✓	46.115	28.247	16.111	47.160
H	✓	✓	✓	Dual	✓	46.807	28.826	16.271	47.635

✓ and × indicate enabled and disabled, respectively. SG blocks direct gradients along P3 → SG-GAFEM. Dual blocks direct gradients along both added semantic paths. G and H differ only in the CSGD setting.

Adding the high-resolution P2 detection branch to YOLO11s (A→B) raises mAP@0.5 from 39.123% to 43.808%, a gain of 4.685 percentage points (Table 3). Adding SG-GAFEM (B→C) and then ASFF2 (C→D) increases mAP@0.5 by 0.916 and 0.105 percentage points, respectively.

Configuration D, which contains P2, SG-GAFEM, and ASFF2, serves as the common reference. Adding dual-path CSGD in F raises mAP@0.5 from 44.829% to 45.857%, a gain of 1.028 percentage points. Adding SOG-DCN in G instead yields 46.115% mAP@0.5,

1.286 percentage points above D. D and F are from earlier ablation runs, and G is from the subsequent comparison of complete architectures.

The complete model (H), which combines dual-path CSGD and SOG-DCN, achieves the highest mAP@0.5 in Table 3 at 46.807%. This is 1.978 percentage points above D and 2.999 above the P2-only configuration B. G and H use the same forward architecture, parameter count, and static computational cost. Enabling dual-path CSGD in this comparison improves mAP@0.5 by 0.692 percentage points.

To distinguish the inference cost of the high-resolution P2 branch from that of the remaining components, we compare configurations A, B, and H from Table 3. Table 4 reports measurements from a single NVIDIA GeForce RTX 3070 at 640 × 640 with batch size 1 and FP32 weights and inputs. We perform three repeats, each with 50 warm-up passes and 500 forward passes timed using CUDA events. FPS measures model-forward speed, with timing limited to network computation.

Table 4. Model size and inference cost of configurations A, B, and H.

Model	Params (M)	GFLOPs	FPS
A (YOLO11s)	9.432	21.568	64.89
B (YOLO11s + P2)	9.577	28.984	56.51
H (proposed model)	9.723	34.258	49.88

P2 adds 7.416 GFLOPs, accounting for 58.4% of the complete model's increase over the baseline. The remaining components add another 5.274 GFLOPs. Configurations A, B, and H achieve model-forward speeds of 64.89, 56.51, and 49.88 FPS, respectively.

To assess the contribution of P2 at a higher input resolution, we train YOLO11s, YOLO11s + P2, and the proposed model at 960 × 960 and evaluate them at the same resolution on the VisDrone validation set (Table 5).

Table 5. Detection results on the VisDrone validation set at an input size of 960 × 960.

Model	Best Epoch	mAP@0.5 (%)	mAP@0.5:0.95 (%)	AP_{small} (%)	$Recall_{small}$ (%)
YOLO11s	207	48.429	29.851	17.309	48.456
YOLO11s + P2	265	51.168	31.964	20.449	55.471
Proposed model	232	53.500	33.577	21.943	58.361

In the 960 × 960 experiments with random initialization, adding only P2 raises mAP@0.5 from 48.429% to 51.168%, a gain of 2.739 percentage points. The complete model reaches 53.500%, which is 2.332 percentage points above the P2-only configuration. Tables 3 and 5 show that mAP@0.5 increases with the addition of P2 and again with the complete model under each training setting.

4.5. Analysis of CSGD and SOG-DCN

4.5.1. CSGD Gradient Path Analysis

We compare four settings on the P2 + ASFF2 + SG-GAFEM architecture with SOG-DCN disabled. No detach retains both paths, ASFF detach blocks direct gradients from ASFF2 to P3, SG detach blocks direct gradients from the SG-GAFEM semantic input to P3, and Dual CSGD blocks direct gradients along both added paths. All four settings use identical forward computation and vary only the backpropagation boundaries at the added semantic inputs. Table 6 shows the effects on detection accuracy.

Table 6. CSGD gradient paths and detection accuracy.

Setting	P3 → ASFF2	P3 → SG-GAFEM	mAP@0.5 (%)	mAP@0.5:0.95 (%)	AP_{small} (%)	$Recall_{small}$ (%)
No detach	✓	✓	44.829	27.412	15.556	46.835
ASFF detach	×	✓	45.704	27.886	16.056	47.622
SG detach	✓	×	45.089	27.716	15.361	46.252
Dual CSGD	×	×	45.857	28.040	16.085	47.476

✓ and × indicate that a gradient path is retained and blocked, respectively. P3 → SG-GAFEM refers to the direct semantic input.

Blocking only the ASFF2 path or only the SG-GAFEM path raises mAP@0.5 by 0.875 or 0.260 percentage points, respectively, relative to No detach. Dual CSGD achieves the highest mAP@0.5 of the four settings at 45.857%, 1.028 percentage points above No detach.

To compare the gradient magnitudes of models trained under different configurations, we use the `best.pt` checkpoints from P2 + SG-GAFEM, No detach, and Dual CSGD. The diagnostic uses the same first 20 fixed batches of the VisDrone validation set, with 4 images per batch and 640 × 640 inputs. For each checkpoint and batch, we perform global label assignment once, then separate the P2 detection loss from the sum of the P3–P5 detection losses. We compute the gradients of each loss with respect to the P2 backbone parameters (layers 0–2) and the top-down P3 parameters (node 16). Figure 8b compares No detach and Dual CSGD by showing the mean L2 norms, across the 20 batches, of the P2 detection loss gradients with respect to these two parameter groups.

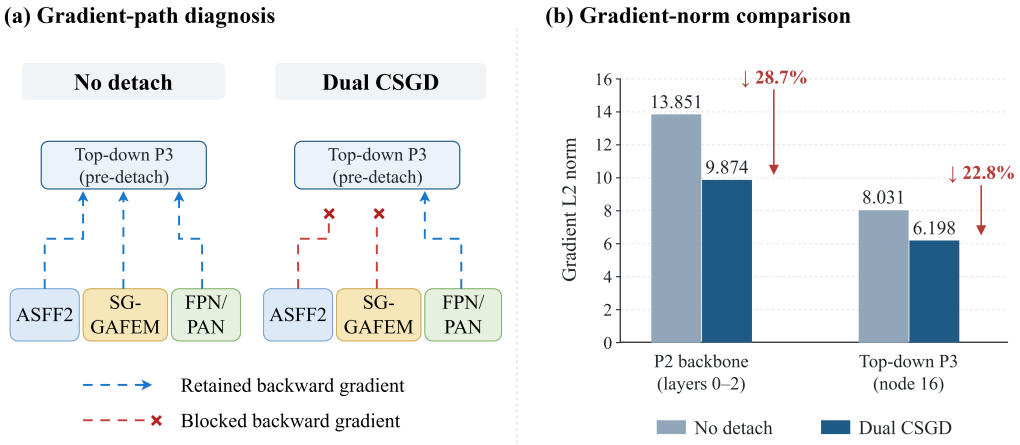


Figure 8. Gradient paths and gradient norms with and without CSGD.

Figure 8a summarizes the gradient paths for the two settings. Section 3.4 details the backpropagation boundaries.

In Figure 8b, Dual CSGD has mean gradient L2 norms that are 28.7% and 22.8% lower for the P2 backbone and the top-down P3 parameters, respectively. These differences describe gradients at independently trained checkpoints. Measuring the immediate effect of toggling CSGD would require holding the checkpoint, inputs, and label assignments fixed.

4.5.2. SOG-DCN Offset Modulation Analysis

We compare the offset distributions for Raw, Constant, and SOG-DCN across all 548 images in the VisDrone validation set using the same seed 0 checkpoint. The rules share the original offsets, modulation weights, and enhanced features, with model weights and the residual factor $\alpha = 1.5$ held fixed. Constant uses the mean semantic offset-scaling factor of configuration H in Table 3, computed on the VisDrone training set. Equation (15) expresses the scaling rules using the notation in Equation (10).

$$\Delta p'_k(x, y) = \beta(x, y) \cdot \Delta p_k(x, y) \tag{15}$$

The nine sampling points at a given P2 output position share the factor $\beta(x, y)$. Raw sets $\beta(x, y) = 1$, Constant sets $\beta(x, y) = 0.5530006257$, and SOG-DCN uses $\beta(x, y) = s(x, y)$ from Equation (9), reducing Equation (15) to Equation (10). The subsequent accuracy comparison also includes per-image mean scaling. This rule averages $s(x, y)$ at the P2 grid centers within the valid image region and applies that mean uniformly to the entire offset map for the image.

We measure offset magnitude as the mean two-dimensional magnitude across the nine sampling points at each position, in P2 feature-grid units. Ground-truth boxes define object regions. The statistics weight object instances equally in object regions and images equally in background regions, with padding excluded.

Table 7 and Figure 9 show regional and spatial distributions of offset magnitudes. Compared with Constant, SOG-DCN reduces mean magnitudes by approximately 3.40%, 4.04%, and 5.11% in small, medium, and large object regions, respectively, and increases the background mean by approximately 0.50%. In Figure 9, the orange box marks a crosswalk and traffic island where SOG-DCN predominantly increases offset magnitudes relative to Constant, and the blue box marks a dense small-object region where it predominantly decreases them. Blue and red in the difference map indicate decreases and increases in magnitude. The upper limit of the shared color scale is the 99th percentile of Raw magnitudes, and the difference scale has symmetric limits set by the 99th percentile of absolute differences. The regional means describe changes in magnitude. Whether sampling points cross object boundaries requires separate verification.

Table 7. Offset magnitudes under different scaling rules with fixed model weights.

Region	Raw	Constant × 0.5530	SOG-DCN	SOG-DCN − Constant
Small objects	2.07056	1.14502	1.10608	−0.03894
Medium objects	2.39915	1.32673	1.27314	−0.05359
Large objects	3.57520	1.97709	1.87599	−0.10110
Background	2.32752	1.28712	1.29360	+0.00648

Offsets are in P2 feature-grid units. Constant uses $c = 0.5530006257$. The averages assign equal weight to each object instance in object regions and to each image in background regions.

We keep the weights fixed at the three checkpoints trained with seeds 0, 1, and 2 and replace the offset-scaling rule only at inference. Position-dependent scaling improves mAP@0.5 by an average of 0.161 and 0.129 percentage points over constant and per-image mean scaling, respectively, with gains over both rules at all three checkpoints.

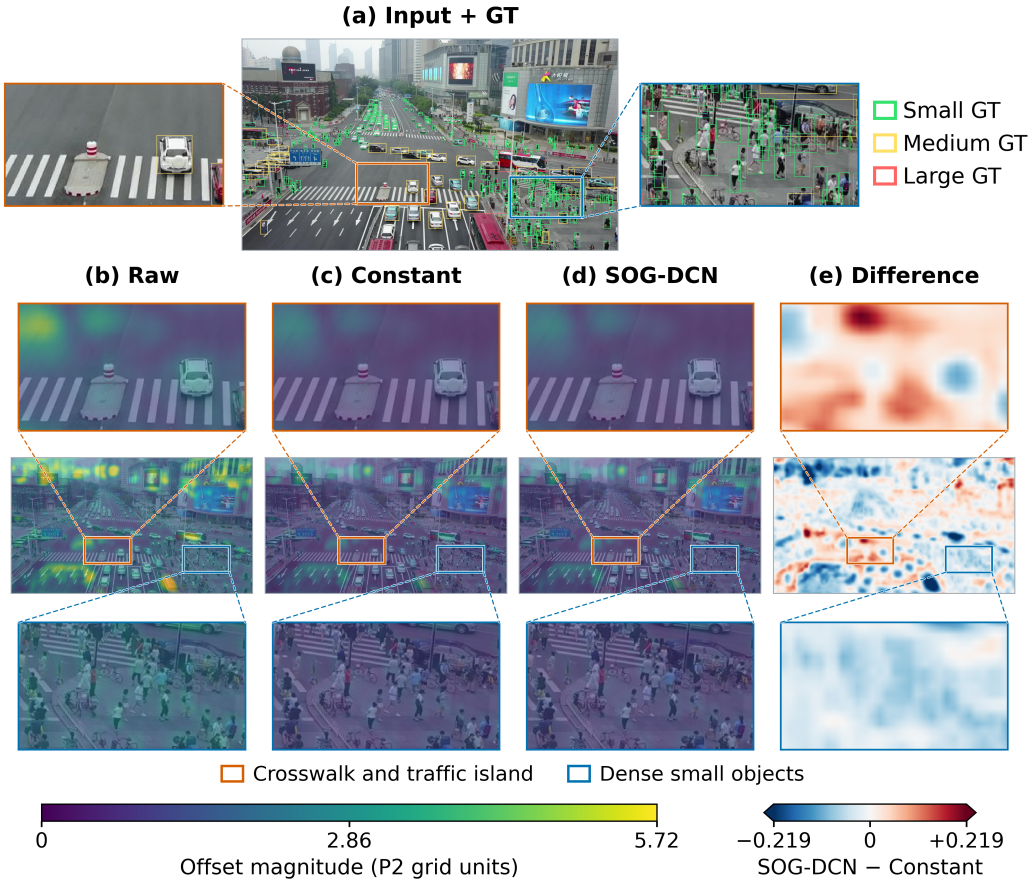


Figure 9. Offset magnitudes under different scaling rules.

4.6. Qualitative Analysis

We select one daytime and one nighttime intersection image from the VisDrone validation set. Figures 10 and 11 use a 2 × 3 layout to show the ground truth, YOLOv8s, YOLO11s, AFE-YOLOv8s, the RT-DETR++ reimplementation, and the proposed model, in that order. The five models use the best checkpoints corresponding to Table 1 and run FP32 inference with 640 × 640 inputs, a confidence threshold of 0.25, and a maximum of 300 detections per image. The YOLO models use non-maximum suppression at an IoU threshold of 0.70, and the RT-DETR++ reimplementation uses native query decoding. Box colors distinguish classes and are consistent across panels for each scene. Blue rectangles in Figure 10 and white rectangles in Figure 11 mark the selected regions. Dashed lines of the same color connect each region to the two upper corners of its enlarged view. Small TP counts correctly detected small objects across the image, and FP counts false positives at all scales. Counts use one-to-one matches of the same class at IoU of at least 0.50. Small objects have an original-image ground-truth box area below 1024 square pixels.

The daytime intersection in Figure 10a contains 111 annotated small objects. In the order of panels (b) through (f), the five models record 27, 31, 35, 79, and 60 Small TP, and 22, 21, 19, 84, and 20 FP. The proposed model detects 29 more small objects than YOLO11s, with one fewer false positive across the image. The RT-DETR++ reimplementation detects the most small objects. The proposed model detects 19 fewer small objects than this reimplementation, with 64 fewer false positives across the image.

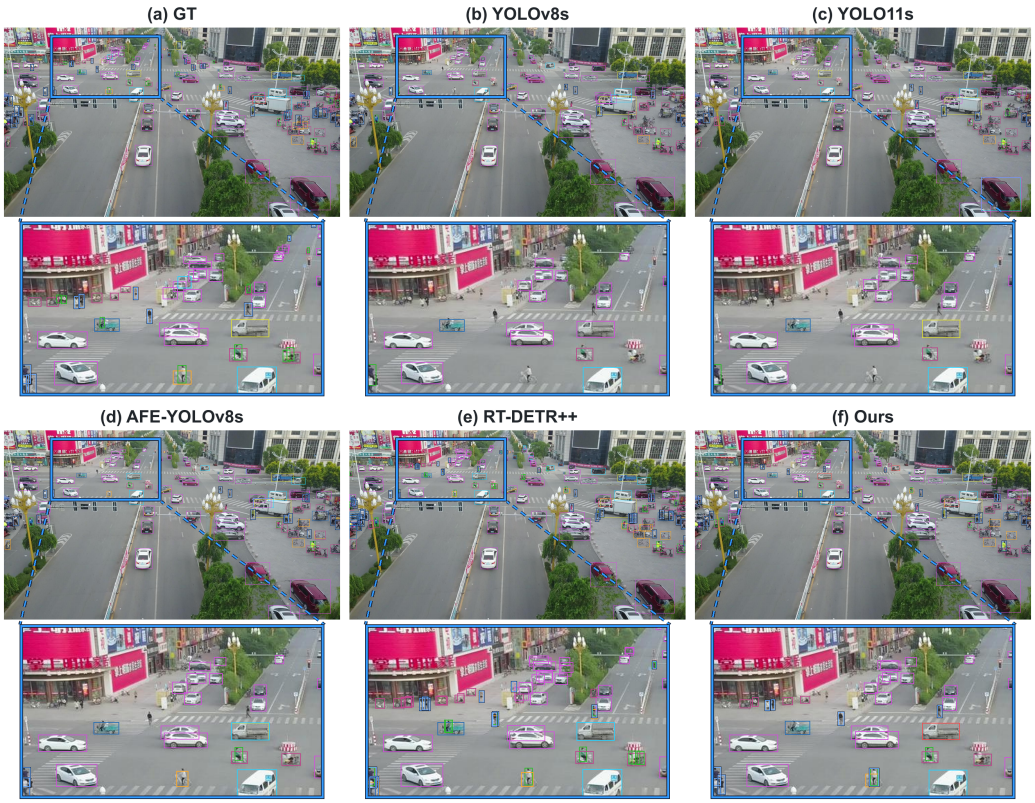


Figure 10. Comparison of detection results in a daytime VisDrone scene.

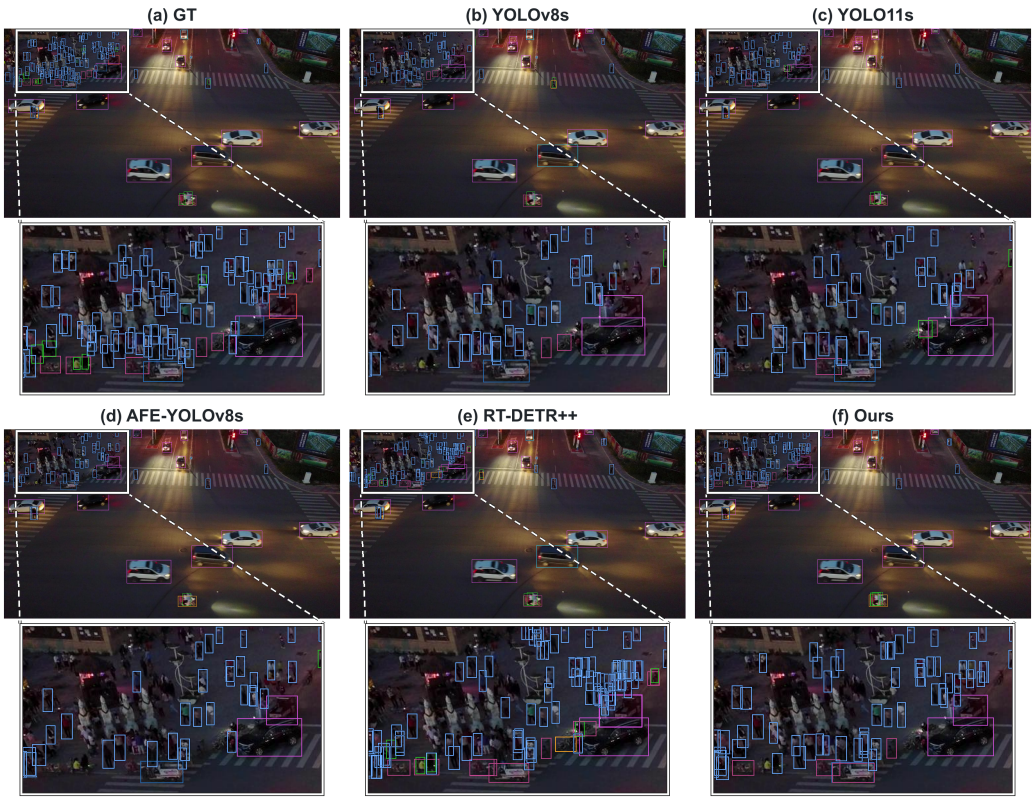


Figure 11. Comparison of detection results in a nighttime VisDrone scene.

The nighttime intersection in Figure 11a has uneven lighting and a dense crowd in the upper left. The image contains 112 annotated small objects. In the order of panels (b) through (f), the five models record 34, 37, 36, 56, and 57 Small TP, and 17, 18, 11, 67,

and 21 FP. The numbers of missed detections across the image are 83, 78, 81, 60, and 58, respectively. The proposed model correctly detects a similar number of small objects to the RT-DETR++ reimplementation, with 46 fewer false positives across the image. Compared with the three YOLO models, it misses fewer objects in this image and produces more false positives.

These differences describe the two examples at a fixed confidence threshold. Section 4.3 compares mAP@0.5 on the full validation set. Sections 4.4 and 4.5 examine the contributions of the components and the detection effects of offset scaling, respectively.

5. Discussion

The proposed model achieves higher mAP@0.5 than YOLO11s at additional computational cost. In the VisDrone comparison, it outperforms RT-DETR-l and the RT-DETR++ reimplementation in both mAP@0.5 and model-forward FPS measured on the same device. In the selected nighttime example, it detects more small objects than YOLOv8s, YOLO11s, and AFE-YOLOv8s, with more false positives across the image. Further evaluation in crowded and low-light scenes could characterize the trade-off between detections and false positives.

The fixed-weight inference comparison on VisDrone shows a modest gain in mAP@0.5 from position-dependent offset scaling. Controlled ablations could directly compare a shared semantic map with separate semantic maps. Sampling-position statistics could establish whether offset scaling changes how often sampling points cross object box boundaries. Our experiments include a 960×960 input comparison on VisDrone. Future work could evaluate higher input resolutions and assess sensitivity to the semantic bias strength and residual scaling factor.

6. Conclusions

Small objects occupy few pixels in UAV images, so downsampling can erase their spatial detail, while background textures in high-resolution features can interfere with recognition. We add a high-resolution P2 detection branch to YOLO11s to preserve spatial detail and use ASFF2 to align P2 and P3 features and fuse shallow detail with deep semantics at each position. To reduce interference from background textures during feature enhancement, SG-GAFEM generates a shared semantic modulation map from P3 and adjusts P2 enhancement strength by location. SOG-DCN reuses the map to scale the predicted deformable sampling offsets at each location, allowing a single semantic signal to modulate both feature enhancement and local sampling. To limit the direct influence of the P2 detection loss on P3 feature learning through the added semantic paths, CSGD blocks direct backpropagation along both paths at the shared P3 input while preserving forward semantic information flow and backpropagation through the original feature pyramid. Experiments on VisDrone and SOA-PD show that the proposed model achieves high detection accuracy compared with several widely used object detectors. Together with the same-device speed measurements on VisDrone, these results indicate that the proposed model balances detection accuracy and inference speed.

Author Contributions: C.G. designed and conducted the experiments, developed the model, analyzed the results, and wrote the manuscript. Z.Z. supervised the research and provided feedback on manuscript revisions. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions of this study are included in the article. Further inquiries can be directed to the corresponding author.

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

References

1. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*; Curran Associates: Red Hook, NY, USA, 2015; Volume 28. Available online: https://papers.nips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html (accessed on 17 October 2026).
2. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot MultiBox detector. In *Computer Vision-ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; Volume 9905, pp. 21–37. https://doi.org/10.1007/978-3-319-46448-0_2.
3. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999–3007. https://doi.org/10.1109/ICCV.2017.324.
4. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788. https://doi.org/10.1109/CVPR.2016.91.
5. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525. https://doi.org/10.1109/CVPR.2017.690.
6. Redmon, J.; Farhadi, A. YOLOv3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767. https://doi.org/10.48550/arXiv.1804.02767.
7. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934. https://doi.org/10.48550/arXiv.2004.10934.
8. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 7464–7475. https://doi.org/10.1109/CVPR52729.2023.00721.
9. Jocher, G.; Chaurasia, A.; Qiu, J. Ultralytics YOLOv8. 2023. Available online: https://github.com/ultralytics/ultralytics (accessed on 17 October 2026).
10. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. YOLOv10: Real-time end-to-end object detection. In *Advances in Neural Information Processing Systems*; Curran Associates: Red Hook, NY, USA, 2024; Volume 37, pp. 107984–108011. https://doi.org/10.52202/079017-3429.
11. Khanam, R.; Hussain, M. YOLOv11: An overview of the key architectural enhancements. *arXiv* **2024**, arXiv:2410.17725. https://doi.org/10.48550/arXiv.2410.17725.
12. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. DETRs beat YOLOs on real-time object detection. In Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024; pp. 16965–16974. https://doi.org/10.1109/CVPR52733.2024.01605.
13. Zhu, X.; Lyu, S.; Wang, X.; Zhao, Q. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), Virtual Conference, 11–17 October 2021; pp. 2778–2788. https://doi.org/10.1109/ICCVW54120.2021.00312.
14. Wang, G.; Chen, Y.; An, P.; Hong, H.; Hu, J.; Huang, T. UAV-YOLOv8: A small-object-detection model based on improved YOLOv8 for UAV aerial photography scenarios. *Sensors***2023**, *23*, 7190. https://doi.org/10.3390/s23167190.
15. Qiu, Y.; Zheng, X.; Hao, X.; Zhang, G.; Lei, T.; Jiang, P. ARSOD-YOLO: Enhancing small target detection for remote sensing images. *Sensors* **2024**, *24*, 7472. https://doi.org/10.3390/s24237472.
16. Wang, S.; Zhang, Z.; Chao, Q.; Yu, T. AFE-YOLOv8: A novel object detection model for unmanned aerial vehicle scenes with adaptive feature enhancement. *Algorithms* **2024**, *17*, 276. https://doi.org/10.3390/a17070276.
17. Yuan, S. RT-DETR++ for UAV Object Detection. *arXiv* **2025**, arXiv:2509.09157. https://doi.org/10.48550/arXiv.2509.09157.
18. Yang, Y.; Guo, F.; Niu, P. UAVDet: A CNN–Mamba hybrid network for efficient small object detection in UAV imagery. *Comput. Vis. Image Underst.* **2026**, 264, 104637. https://doi.org/10.1016/j.cviu.2026.104637.
19. Liu, K.; Fu, Z.; Jin, S.; Chen, Z.; Zhou, F.; Jiang, R.; Chen, Y.; Ye, J. ESOD: Efficient Small Object Detection on High-Resolution Images. *IEEE Trans. Image Process.* **2025**, 34, 183–195. https://doi.org/10.1109/TIP.2024.3501853.
20. Tong, Y.; Ye, H.; Yang, J.; Yang, X. ACD-DETR: Adaptive Cross-Scale Detection Transformer for Small Object Detection in UAV Imagery. *Sensors* **2025**, 25, 5556. https://doi.org/10.3390/s25175556.
21. Chen, Y.; Ye, Z.; Sun, H.; Gong, T.; Xiong, S.; Lu, X. Global–Local Fusion With Semantic Information Guidance for Accurate Small Object Detection in UAV Aerial Images. *IEEE Trans. Geosci. Remote Sens.* **2025**, 63, 4701115. https://doi.org/10.1109/TGRS.2025.3532612.

22. Liu, Z.; Gao, G.; Sun, L.; Fang, Z. HRDNet: High-resolution detection network for small objects. In Proceedings of the 2021 IEEE International Conference on Multimedia and Expo (ICME), Shenzhen, China, 5–9 July 2021; pp. 1–6. https://doi.org/10.1109/ICME51207.2021.9428241.
23. Courtrai, L.; Pham, M.T.; Lefèvre, S. Small object detection in remote sensing images based on super-resolution with auxiliary generative adversarial networks. *Remote Sens.* **2020**, *12*, 3152. https://doi.org/10.3390/rs12193152.
24. Li, C.; Zhao, R.; Wang, Z.; Xu, H.; Zhu, X. RemDet: Rethinking Efficient Model Design for UAV Object Detection. *Proc. AAAI Conf. Artif. Intell.* **2025**, 39, 4643–4651. https://doi.org/10.1609/aaai.v39i5.32490.
25. Yang, C.; Huang, Z.; Wang, N. QueryDet: Cascaded sparse query for accelerating high-resolution small object detection. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–24 June 2022; pp. 13658–13667. https://doi.org/10.1109/CVPR52688.2022.01330.
26. Huang, Y.; Chen, J.; Huang, D. UFPMP-Det: Toward Accurate and Efficient Object Detection on Drone Imagery. *Proc. AAAI Conf. Artif. Intell.* **2022**, 36, 1026–1033. https://doi.org/10.1609/aaai.v36i1.19986.
27. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944. https://doi.org/10.1109/CVPR.2017.106.
28. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8759–8768. https://doi.org/10.1109/CVPR.2018.00913.
29. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and efficient object detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual Conference, 14–19 June 2020; pp. 10778–10787. https://doi.org/10.1109/CVPR42600.2020.01079.
30. Liu, S.; Huang, D.; Wang, Y. Learning spatial fusion for single-shot object detection. *arXiv* **2019**, arXiv:1911.09516. https://doi.org/10.48550/arXiv.1911.09516.
31. Chen, Z.; Ma, Y.; Gong, Z.; Cao, M.; Yang, Y.; Wang, Z.; Wang, T.; Li, J.; Liu, Y. R-AFPN: A residual asymptotic feature pyramid network for UAV aerial photography of small targets. *Sci. Rep.* **2025**, *15*, 16233. https://doi.org/10.1038/s41598-025-00008-3.
32. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141. https://doi.org/10.1109/CVPR.2018.00745.
33. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. CBAM: Convolutional block attention module. In *Computer Vision-ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer: Cham, Switzerland, 2018; Volume 11211, pp. 3–19. https://doi.org/10.1007/978-3-030-01234-2_1.
34. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. GhostNet: More features from cheap operations. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual Conference, 14–19 June 2020; pp. 1577–1586. https://doi.org/10.1109/CVPR42600.2020.00165.
35. Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; Wei, Y. Deformable convolutional networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 764–773. https://doi.org/10.1109/ICCV.2017.89.
36. Zhu, X.; Hu, H.; Lin, S.; Dai, J. Deformable ConvNets v2: More deformable, better results. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 9300–9308. https://doi.org/10.1109/CVPR.2019.00953.
37. Wang, W.; Dai, J.; Chen, Z.; Huang, Z.; Li, Z.; Zhu, X.; Hu, X.; Lu, T.; Lu, L.; Li, H.; et al. InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 14408–14419. https://doi.org/10.1109/CVPR52729.2023.01385.
38. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In Proceedings of the 9th International Conference on Learning Representations (ICLR), Virtual Conference, 3–7 May 2021. https://arxiv.org/abs/2010.04159.
39. Ma, Y.; Xi, C.; Ma, T.; Sun, H.; Lu, H.; Xu, X.; Xu, C. I-YOLOv11n: A lightweight and efficient small target detection framework for UAV aerial images. *Sensors***2025**, *25*, 4857. https://doi.org/10.3390/s25154857.
40. Zhang, C.; Guo, Z. HSAR-DETR: Hierarchical Spatial–Frequency Attention Network for UAV Small Object Detection. *Remote Sens.* **2026**, 18, 2861. https://doi.org/10.3390/rs18172861.
41. Sun, H.; Li, Y.; Yang, L.; Cao, X.; Zhang, B. Uncertainty-aware gradient stabilization for small object detection. In Proceedings of the 2025 IEEE/CVF International Conference on Computer Vision, Honolulu, HI, USA, 19–23 October 2025; pp. 8407–8417. Available online: https://openaccess.thecvf.com/content/ICCV2025/html/Sun_Uncertainty-Aware_Gradient_Stabilization_for_Small_Object_Detection_ICCV_2025_paper.html (accessed on 17 October 2026).

42. Chen, Z.; Badrinarayanan, V.; Lee, C.Y.; Rabinovich, A. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 794–803. Available online: https://proceedings.mlr.press/v80/chen18a.html (accessed on 17 October 2026).
43. Yu, T.; Kumar, S.; Gupta, A.; Levine, S.; Hausman, K.; Finn, C. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*; Curran Associates: Red Hook, NY, USA, 2020; Volume 33, pp. 5824–5836. Available online: https://proceedings.neurips.cc/paper/2020/hash/3fe78a8acf5fda99de95303940a2420c-Abstract.html (accessed on 17 October 2026).
44. Jocher, G.; Qiu, J. Ultralytics YOLO11. 2024. Available online: https://docs.ultralytics.com/models/yolo11/ (accessed on 17 October 2026).
45. Zhu, P.; Wen, L.; Du, D.; Bian, X.; Fan, H.; Hu, Q.; Ling, H. Detection and tracking meet drones challenge. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 7380–7399. https://doi.org/10.1109/TPAMI.2021.3119563.
46. Makrigiorgis, R.; Kyrkou, C.; Kolios, P. *Small Object Aerial Person Detection Dataset, Version 1.0*; Zenodo: Geneva, Switzerland, 2023. https://doi.org/10.5281/zenodo.7740081.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.