applied sciences

MDPI

Article

DARA-Net: Detail-Aware Representation and Aggregation for Lightweight UAV Detection

Luyao Jiang, Xinli Lin *, Xiaoming Chen, Tao Cheng and Jianxun Lu *

College of Urban Transportation and Logistics, Shenzhen Technology University, Shenzhen 518118, China; 2410264027@stumail.sztu.edu.cn (L.J.); chengtao@sztu.edu.cn (T.C.)
* Correspondence: d23092110047@cityu.edu.mo (X.L.); lujianxun@sztu.edu.cn (J.L.)

Abstract

Detecting unmanned aerial vehicles (UAVs) in complex low-altitude scenes remains challenging due to limited target details and background interference. To address these challenges, this paper proposes DARA-Net, a lightweight UAV detection framework that coordinates detail preservation, efficient feature extraction, and multiscale aggregation. Built upon YOLOv11, DARA-Net integrates adaptive window reduction (AWR) to retain informative local features during downsampling, a separable aggregation block (SAB) to improve feature representation with fewer parameters, and a directional weighted pyramid (DWP) to enhance cross-scale feature fusion. Experiments on a visible-light UAV dataset containing 25,941 images demonstrate that DARA-Net achieves 97.0% precision, 91.1% recall, 95.1% mAP@0.5, and 62.0% mAP@0.5:0.95. Compared with the YOLOv11 baseline, recall improves by 2.6 percentage points, while the parameter count decreases by approximately 30.6% and model size decreases by 29.1%, reaching 1.79 M parameters and 3.9 MB, respectively. Compared with YOLOv26n, DARA-Net achieves a higher mAP@0.5:0.95 (62.0% versus 60.77%) with approximately 24.6% fewer parameters and a 28.2% smaller model file, while maintaining comparable mAP@0.5. Additional validation on an external UAV dataset further supports its competitive detection performance. These results highlight the value of DARA-Net as a compact UAV detector that prioritizes localization performance and reduced parameter and storage requirements.

Keywords: UAV detection; compact neural networks; adaptive downsampling; depthwise separable convolution; directional feature aggregation

1. Introduction

Visual surveillance of low-altitude airspace requires detectors that distinguish UAVs from structured backgrounds and other airborne objects. UAV risk assessment, flight control, and collision avoidance provide an operational context for this requirement [1–4]. The task considered here is to localize UAVs in visible-light images. References to a small target footprint concern its image representation, rather than a physical UAV size or weight category. Low contrast, changing orientation, and limited pixel support are challenges identified in related research [5,6]; their occurrence should not be assumed to characterize every image in a detection dataset.

Missed detections and false alarms have different consequences for a monitoring system. Increasing precision alone does not establish that a detector recovers more targets, and reducing the number of parameters does not necessarily reduce inference latency. A useful evaluation must therefore consider precision, recall, localization accuracy, and model

Check for updates

Academic Editor: Krzysztof Zboiński

Received: 15 September 2026
Revised: 20 October 2026
Accepted: 22 October 2026
Published: 26 October 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.

cost together. This study focuses on the accuracy–cost relationship of the detector, rather than optimizing or demonstrating the response time of a complete monitoring system.

In low-altitude monitoring, UAVs may occupy only a small region of an image, making their appearance difficult to distinguish from surrounding structures or other airborne objects. Under such conditions, missed detections can reduce the completeness of visual observations, while false positives may introduce unnecessary monitoring alerts. Improvements in recall therefore indicate that a greater proportion of annotated targets can be recovered, whereas higher mAP reflects improvements in detection performance across confidence levels and localization criteria. However, these accuracy gains must be considered alongside model complexity, particularly when computational and storage resources are limited. A detector that improves target recovery while reducing parameter and storage requirements can provide a more compact perception component for subsequent monitoring systems.

Compact detection networks allocate their capacity across spatial reduction, feature transformation, and cross-scale fusion. Spatial reduction determines which responses remain available at lower resolutions; backbone transformations construct discriminative features; and the neck combines features with different spatial and semantic content. Modifying one operation may reduce model cost while weakening another aspect of detection. This motivates examining the combined network and its component configurations under a common evaluation protocol.

These requirements also motivate considering the interaction between successive feature-processing stages. Aggressive spatial reduction may weaken responses associated with small targets before deeper representations are constructed. Conversely, preserving additional information without controlling feature-transformation costs may increase network complexity. Multiscale fusion provides another opportunity to combine spatial detail with semantic information, but its effectiveness depends on the features retained by earlier stages. Consequently, improvements to individual components do not necessarily produce equivalent gains when integrated into a complete detector. DARA-Net addresses this design problem by coordinating content-dependent reduction, lightweight feature transformation, and directional cross-scale aggregation, with the aim of preserving useful detection information while maintaining a compact network.

Recent UAV detectors have explored efficient backbones, shallow-feature learning, and changes across multiple modules [7–9]. Complementary work has refined feature pyramids and directional processing for small-object localization [10–12]. Motion-guided detection provides a further approach when successive video frames are available [13]. These methods differ in their tasks, input information, and computational requirements. The question addressed here is whether coordinating content-dependent spatial reduction, separable feature transformation, and directional multiscale fusion can improve UAV detection while reducing the parameter and storage requirements of a YOLOv11 baseline.Although recent compact detectors such as YOLOv26n provide strong detection performance, UAV detection applications may differ in their requirements for localization accuracy, model size, parameter count, and inference efficiency. The availability of an efficient detector therefore does not eliminate the need to explore alternative architectural trade-offs. DARA-Net investigates this design space by coordinating spatial reduction, lightweight feature representation, and directional multiscale aggregation, with the objective of improving localization performance while maintaining a compact parameter and storage footprint.

We introduce **DARA-Net**, a network for **detail-aware representation and aggregation** in UAV detection. It combines adaptive window reduction (AWR), a separable aggregation block (SAB), and a directional weighted pyramid (DWP). The contribution is the adaptation

and evaluation of these established design principles within one detector; the underlying convolution, residual connection, and feature-pyramid operations are not claimed as new primitives. The three component contributions are as follows:

1. AWR adapts two-branch feature reduction with log-domain modulation. Used alone, it reduces parameters from 2.58 M to 2.20 M and increases recall from 88.5% to 89.2%, while mAP@0.5 decreases from 94.0% to 93.8%. Its contribution is therefore an observed trade-off between compactness and detection metrics.
2. SAB adapts branched aggregation with separable residual transformations. Used alone, it reduces parameters to 2.40 M and increases precision from 96.7% to 97.8% and mAP@0.5:0.95 from 60.8% to 61.5%.
3. DWP combines trainable cross-scale fusion with pinwheel-shaped convolution. Used alone, it reduces parameters to 2.36 M, retains 94.0% mAP@0.5, and increases recall from 88.5% to 89.0%. In the full configuration with AWR and SAB, recall reaches 91.1% and mAP@0.5 reaches 95.1% with 1.79 M parameters.

These component differences are conditional on the configurations compared and are not additive contributions to the final score. Relative to YOLOv12 in Table 1, the full model increases precision by 1.5 percentage points and recall by 2.6 percentage points while using fewer parameters. The motivation for the design is this joint accuracy–cost trade-off, rather than the precision increment in isolation. The following experiments evaluate this accuracy–compactness trade-off through detector comparisons, component ablations, additional dataset validation, and measured inference latency, while distinguishing the reported findings from broader generalization and deployment claims.

2. Related Work

2.1. UAV Observation and Detection

UAV perception spans different sensing arrangements and output requirements. Radar-assisted detection [14], visible-light UAV-to-UAV observation [6], and object recognition in UAV aerial imagery [15] address different tasks. In particular, using a UAV-mounted camera to detect objects on the ground differs from detecting the UAV itself. Reviews of UAV safety and security place these perception tasks within broader monitoring and operational systems [16,17]. The present task is image-based localization of UAVs using visible-light observations from low-altitude scenes.

Illumination and target geometry have also motivated specialized representations. MSF-GhostNet examines efficient drone detection under low light [18], while LEANet and IFD-YOLO address infrared UAV imagery [19,20]. GLANet combines global and local information for rotated-object detection [21]. These studies motivate attention to local appearance, orientation, and scale, but their sensing modalities and annotation schemes must be considered when interpreting comparisons.

Temporal methods provide another perspective on small-target representations. SiamUF uses dense U-shaped features for near-infrared tracking, and SiamDiff incorporates diffusion into scale-aware tracking [22,23].

2.2. Efficient Feature Transformation

Reducing detector cost requires choices about both the operators and the information paths surrounding them. Shallow and multiscale learning supports the compact design of YOLO-LSM [8]; LT-DETR combines lightweight detection with knowledge distillation [24]. Efficient anti-UAV networks also explore architectural changes across several modules [7,9]. Their common design concern is how to retain useful representations as model capacity is reduced.

Two established components are particularly relevant to the backbone developed here. The LAE design combines local adaptation with efficient downsampling [25]. VoVGSCSP provides a branched aggregation structure within the GSConv-based slim-neck design [26]. AWR adopts the former's two-branch reduction pattern with log-domain modulation; SAB adapts the latter's aggregation pattern using separable residual transformations. Their combination places content-dependent reduction and economical feature learning at successive stages of the backbone.

2.3. Cross-Scale and Directional Aggregation

Feature pyramids connect fine spatial structure with higher-level semantics. R-AFPN uses residual asymptotic aggregation for aerial small-object detection [10], whereas Tri-FRNet organizes fusion along three paths [12]. MD-DRIFPN explores dilated multidirectional processing for small drones [11]. Edge-oriented architectures similarly emphasize efficient interaction among feature scales [27,28].

Weighted bidirectional fusion in EfficientDet assigns trainable contributions to incoming feature maps [29]. Pinwheel-shaped convolution introduces directional branches through asymmetric padding and elongated kernels [30]. DWP combines these two principles in a neck that exchanges information across scales and processes orientation-dependent local patterns. This arrangement complements the backbone's reduction and aggregation operations within one detector.

3. Detail-Aware Representation and Aggregation

3.1. Network Organization

DARA-Net follows a single-stage detection layout with a feature-extraction backbone, a multiscale neck, and prediction heads derived from YOLOv11 [31]. Throughout this paper, baseline denotes the unmodified YOLOv11 comparator reported in Table 1, with 2.58 M parameters and 6.3 GFLOPs at the stated input resolution. AWR and SAB modify backbone processing, while DWP modifies feature aggregation. Figure 1 shows the backbone and neck connections. AWR reduces spatial resolution, SAB transforms features through separable residual paths, and DWP supplies fused representations to the prediction heads.

For a 640×640 input, the feature outputs denoted by P3, P4, and P5 have spatial sizes 80×80, 40×40, and 20×20, respectively, corresponding to strides of 8, 16, and 32. These are feature maps rather than final detections. The prediction heads use them to estimate bounding boxes and UAV confidence scores, from which the detection pipeline produces the displayed image-level predictions. One image supplies the input for a forward prediction; this does not imply that one frame guarantees a correct detection or that the full pipeline operates in real time.

The design assigns a specific function to each stage. Spatial reduction combines candidate responses into a lower-resolution representation. The backbone maintains parallel paths for transformed and directly projected information. The neck then exchanges features across resolutions using trainable fusion coefficients. The following subsections define these operations separately so that their roles in the full detector can be distinguished. Symbols are defined within each module: X denotes its input feature tensor and Y its output, so their dimensions can differ between modules. Tensor subscripts identify elements; operator subscripts such as 1×1 and 3×3 identify spatial kernel sizes. The AWR, SAB, and W fusion blocks in Figure 1 correspond to Equations (1)–(9), (10)–(17) and (18)–(22), respectively.

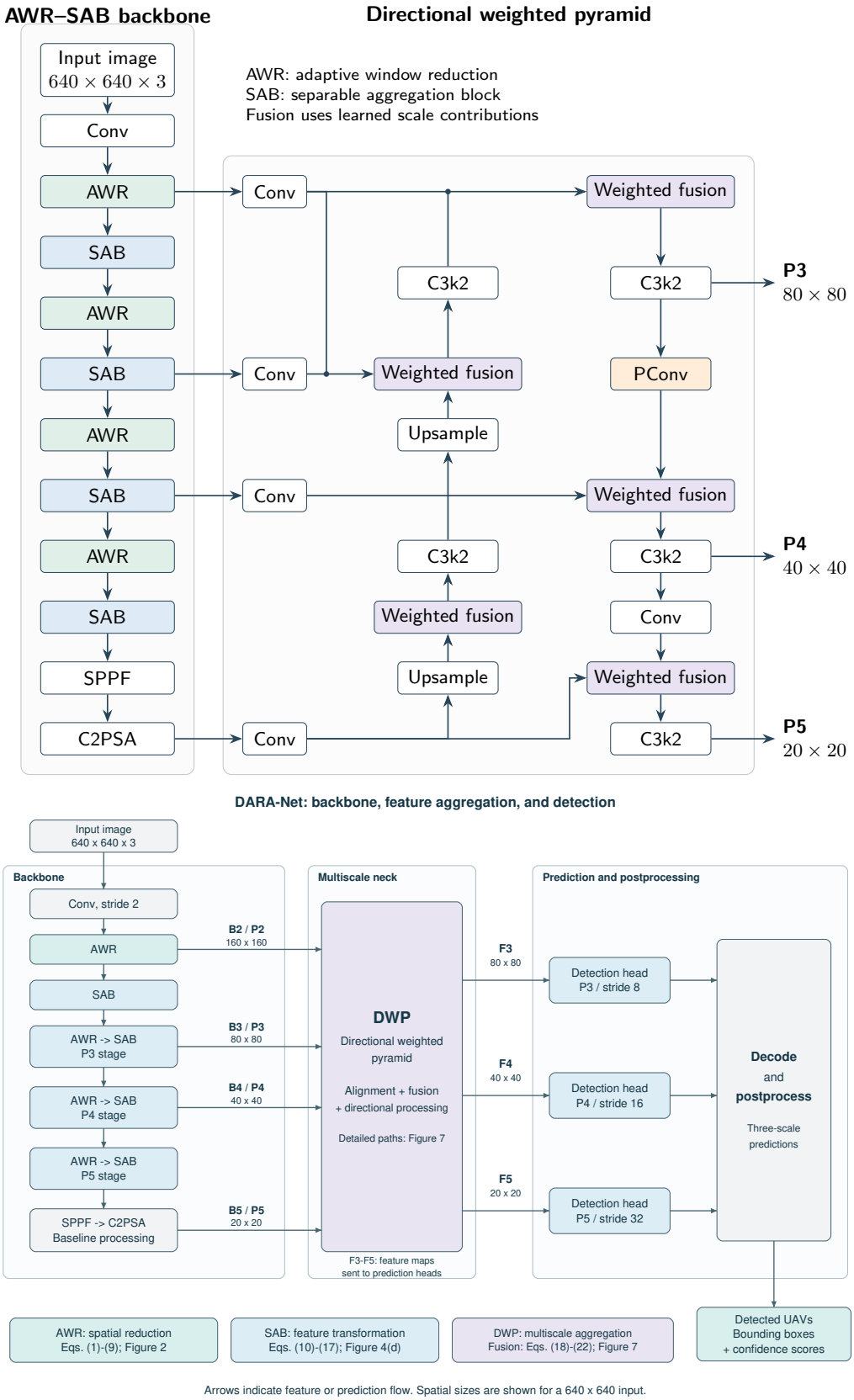


Figure 1. Backbone and neck organization of DARA-Net. AWR reduces spatial resolution, SAB transforms backbone features, and DWP supplies the P3, P4, and P5 feature maps to the prediction heads described in the text. Arrows indicate feature flow; C3k2, SPPF, and C2PSA are retained baseline units.

3.2. Adaptive Window Reduction

AWR builds on the local adaptive extraction structure of LAE [25]. It couples a coefficient branch with a feature branch, so that spatial reduction depends on local image content. The default configuration uses LogSoftmax coefficients to modulate four feature candidates at each reduced location. Figure 2 shows the coefficient and feature paths, including the Softmax mapping used for comparison.

For a batch of feature maps, write

$$X \in \mathbb{R}^{B\times C\times 2H\times 2W}, \tag{1}$$

Here , B is the batch size, C is the channel count, and $2H \times 2W$ is the input spatial resolution; the output resolution is $H \times W$. The indices $b \in \{0,\ldots,B-1\}$ and $c \in \{0,\ldots,C-1\}$ identify a batch element and a channel, while $h \in \{0,\ldots,H-1\}$ and $w \in \{0,\ldots,W-1\}$ index output spatial positions. The notation $\mathrm{Conv}_{k_h\times k_w}$ denotes a convolutional block with the indicated spatial kernel dimensions. In this coefficient branch, the 1×1 block performs channel projection after a 3×3 average-pooling operation with stride 1 and padding 1, preserving spatial dimensions:

$$A = \mathrm{Conv}_{1\times 1}\big(\mathrm{AvgPool}_{3\times 3}(X)\big). \tag{2}$$

In the window notation of Equation (3), A is the response before rearrangement, with the same shape as X, and $\tilde{A} \in \mathbb{R}^{B\times C\times H\times W\times 4}$ stores four candidate responses at each output position. The tilde marks rearrangement, not a separate learned transform. The stated indexing assembles a 2×2 window:

$$\tilde{A}_{b,c,h,w,k} = A_{b,c,2h+\delta_1(k),2w+\delta_2(k)}, \qquad k \in \{1,2,3,4\}, \tag{3}$$

The index k selects one of four candidate positions. The pair $(\delta_1(k),\delta_2(k))$ specifies its row and column offsets in the stated window indexing, in the following order:

$$(\delta_1(k),\delta_2(k)) \in \{(0,0),(0,1),(1,0),(1,1)\}. \tag{4}$$

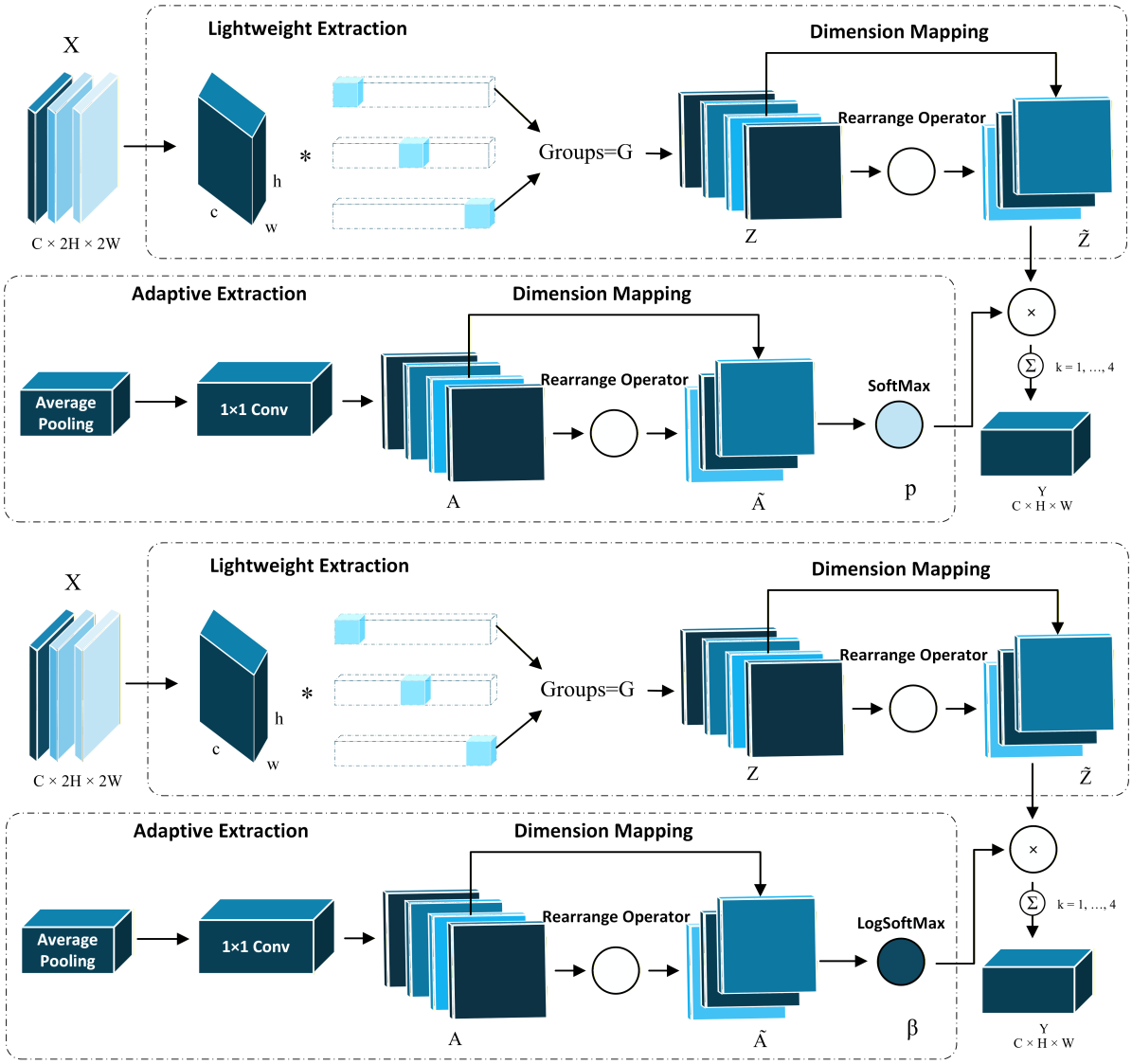
In Equation (4), candidates $k = 1,2,3,4$ use the listed offsets in order: upper left, upper right, lower left, and lower right. Thus, δ_1 is the row offset and δ_2 is the column offset. Applying LogSoftmax along this candidate dimension gives the modulation coefficients:

$$\beta_{b,c,h,w,k} = \log \frac{\exp(\tilde{A}_{b,c,h,w,k})}{\sum_{t=1}^{4}\exp(\tilde{A}_{b,c,h,w,t})}. \tag{5}$$

In Equation (5), t is a summation index over the same four candidates, exp denotes the exponential function, and log denotes the natural logarithm. The batch, channel, and spatial indices are held fixed during this calculation. Their log-domain representation satisfies

$$\beta_{b,c,h,w,k} \le 0, \qquad \sum_{k=1}^{4}\exp(\beta_{b,c,h,w,k}) = 1. \tag{6}$$

These identities follow from the definition of LogSoftmax; they are not additional assumptions. Defining $p_k = \exp(\beta_{b,c,h,w,k})$ at a fixed (b,c,h,w) gives $p_k > 0$ and $\sum_k p_k = 1$, so $\beta_{b,c,h,w,k} = \log p_k \le 0$. The coefficients used in the feature reduction are the log-domain values β, not the probabilities p_k. In particular, the β values do not sum to one, and the reduction is not a convex weighted average. Whether this modulation improves detection is an empirical question examined by the coefficient-mapping comparison.



Shapes omit batch B. X, A: C × 2H × 2W; Z: 4C × H × W; Y: C × H × W.
Ã, Z̃, p, β: C × H × W × 4. Top p: Softmax comparison; bottom β: DARA-Net (LogSoftmax).
×: element-wise product; Σ: sum over k = 1, …, 4. G: group count. Both branches take X as input.
Eqs.: A (2); Ã (3)-(4); β (5)-(6); Z (7); Z̃ (8); lower-panel Y (9).
Schematic edge labels c, h, w denote C, 2H, 2W.
AvgPool: 3×3, stride 1, pad 1. Grouped Conv: 3×3, stride 2.

Figure 2. Two-branch local reduction underlying AWR. The upper diagram shows Softmax modulation, and the lower diagram shows the LogSoftmax configuration used in DARA-Net. Each coefficient path modulates the corresponding grouped-convolution features.

The parallel feature branch applies a grouped convolution with stride two:

$$Z = \mathrm{Conv}_{3\times3}^{s=2,\,g=G}(X), \quad Z \in \mathbb{R}^{B\times4C\times H\times W}, \tag{7}$$

The group count is denoted by G; the superscripts $s = 2$ and $g = G$ specify the convolution stride and number of groups, respectively. The feature branch produces 4C channels at half the input height and width. This stride s is an operator setting. Rearranging these channels assigns four feature candidates to each output channel and spatial position; $\tilde{Z}$ denotes the rearranged form of Z:

$$\tilde{Z} \in \mathbb{R}^{B\times C\times H\times W\times4}. \tag{8}$$

Both β and $\tilde{Z}$ have shape $B \times C \times H \times W \times 4$. The candidate index k pairs each coefficient with one feature candidate. The two branches meet in a weighted reduction:

$$Y_{b,c,h,w} = \sum_{k=1}^{4} \beta_{b,c,h,w,k} \, \tilde{Z}_{b,c,h,w,k}. \tag{9}$$

Figure 2 separates candidate generation from coefficient computation. The pooling and projection branch corresponds to Equation (2); its rearrangement corresponds to Equations (3) and (4). In the lower diagram, the LogSoftmax node produces β according to Equation (5), with the properties in Equation (6). The grouped-convolution branch and its rearrangement produce Z and $\tilde{Z}$ in Equations (7) and (8). The multiplication node pairs their elements with the coefficients; Equation (9) also includes the subsequent sum over k. This summation removes the candidate axis and gives $Y \in \mathbb{R}^{B \times C \times H \times W}$. The upper diagram is the Softmax comparison, so the log-domain properties apply to the lower diagram. Alternative coefficient mappings are evaluated in Section 4.5 with the feature branch held fixed.

3.3. Separable Aggregation Block

SAB adapts the branched topology of VoVGSCSP [26] to local feature transformation with depthwise separable convolutions. One branch performs repeated residual transformations; the other carries a direct projection to the final aggregation. This arrangement combines processed local structure with a shorter information path. The reference aggregation topology appears in Figure 3.

For this block, let $X \in \mathbb{R}^{B \times C_1 \times H \times W}$ be the input tensor, where C_1 and C_2 denote the input and output channel counts, respectively. Let e be the expansion factor controlling the intermediate channel width (not the base of the exponential function), and let $\lfloor \cdot \rfloor$ denote rounding down to an integer. The hidden width is

$$C' = \lfloor e \cdot C_2 \rfloor. \tag{10}$$

Two independently parameterized 1×1 convolutions form the intermediate features:

$$U = \mathrm{Conv}_{1\times1}(X), \qquad V = \mathrm{Conv}_{1\times1}(X), \tag{11}$$

The two projections have separate learnable parameters despite sharing the notation $\mathrm{Conv}_{1\times1}$: U enters the transformed branch and V enters the parallel branch. Repeated occurrences of Conv specify an operator type, not shared weights. Both preserve $H \times W$ and produce C' channels, so $U, V \in \mathbb{R}^{B \times C' \times H \times W}$. The main-path feature U passes through n separable residual bottlenecks (SRBs), where n is the number of repeated blocks and $\mathcal{F}_{\mathrm{SRB}}^{(n)}$ denotes their sequential composition:

$$U_n = \mathcal{F}_{\mathrm{SRB}}^{(n)}(U), \tag{12}$$

The superscript (n) in Equation (12) denotes sequential application of n blocks, not an element-wise power or a requirement to share their parameters. An SRB contains a transformed path and a projected shortcut. Its input feature x is transformed into z by successive separable convolutions, while a 1×1 projection produces the shortcut feature s. Here, lower-case x, z, and s are feature tensors local to the bottleneck; s denotes the shortcut feature, whereas the superscript $s = 2$ in Equation (7) denotes convolution stride. The paths

use compatible channel and spatial dimensions so that their outputs can be added element-wise. The transformed path is

$$z = \mathrm{DSConv}_{3\times3}\big(\mathrm{DSConv}_{1\times1}(x)\big), \tag{13}$$

The shortcut supplies

$$s = \mathrm{Conv}_{1\times1}(x). \tag{14}$$

Their residual combination defines the bottleneck output:

$$\mathrm{SRB}(x) = z + s. \tag{15}$$

Within this block, the separable operator is

$$\mathrm{DSConv}(x) = \mathrm{SiLU}\big(\mathrm{BN}(\mathrm{PW}(\mathrm{DW}(x)))\big), \tag{16}$$

The nesting in Equation (16) specifies the order DW, PW, BN, then SiLU. The 1×1 or 3×3 subscript in Equation (13) specifies the depthwise kernel within the corresponding separable operator. Here, DW denotes a depthwise convolution that filters channels independently, PW denotes a 1×1 pointwise convolution that mixes channels, and BN denotes batch normalization. The SiLU activation is $\mathrm{SiLU}(t) = t\sigma(t)$, with $\sigma(t) = 1/(1 + \exp(-t))$. These operations separate spatial filtering from channel mixing. In the reference topology of Figure 3, the two projected branches correspond to U and V, and the repeated bottleneck path corresponds to the transformation from U to U_n. SAB retains this branched topology while using the separable residual transformation defined above. Finally, Concat joins V and U_n along the channel dimension to produce $2C'$ channels, and the output projection maps them to C_2 channels:

$$Y = \mathrm{Conv}_{1\times1}\big(\mathrm{Concat}(V, U_n)\big). \tag{17}$$

The correspondence between Figures 3 and 4 and the equations is topological: the two input projections give U and V in Equation (11), the repeated main branch gives U_n in Equation (12), and the concatenation and output projection implement Equation (17). Figure 3 depicts the reference GS-based operators. In configuration C of Figure 4, the separable main path, projected shortcut, and addition correspond to Equations (13)–(16); these SAB equations do not redefine the GS operators in the reference diagram. The resulting SAB preserves two information routes until the output projection. Its separable transformations concentrate spatial processing within the residual branch while the parallel projection remains available for aggregation. SAB replaces the baseline C3k2 units in the backbone.

Figure 4 describes a placement study rather than three simultaneous parts of the final network. Configuration A replaces the main-branch input projection, configuration B replaces the parallel projection, and configuration C replaces the two internal bottleneck operators. Only configuration C is used in DARA-Net. Configurations A and B are retained here to define the alternatives compared in Section 4.6; they should not be interpreted as additional modules of the final architecture.

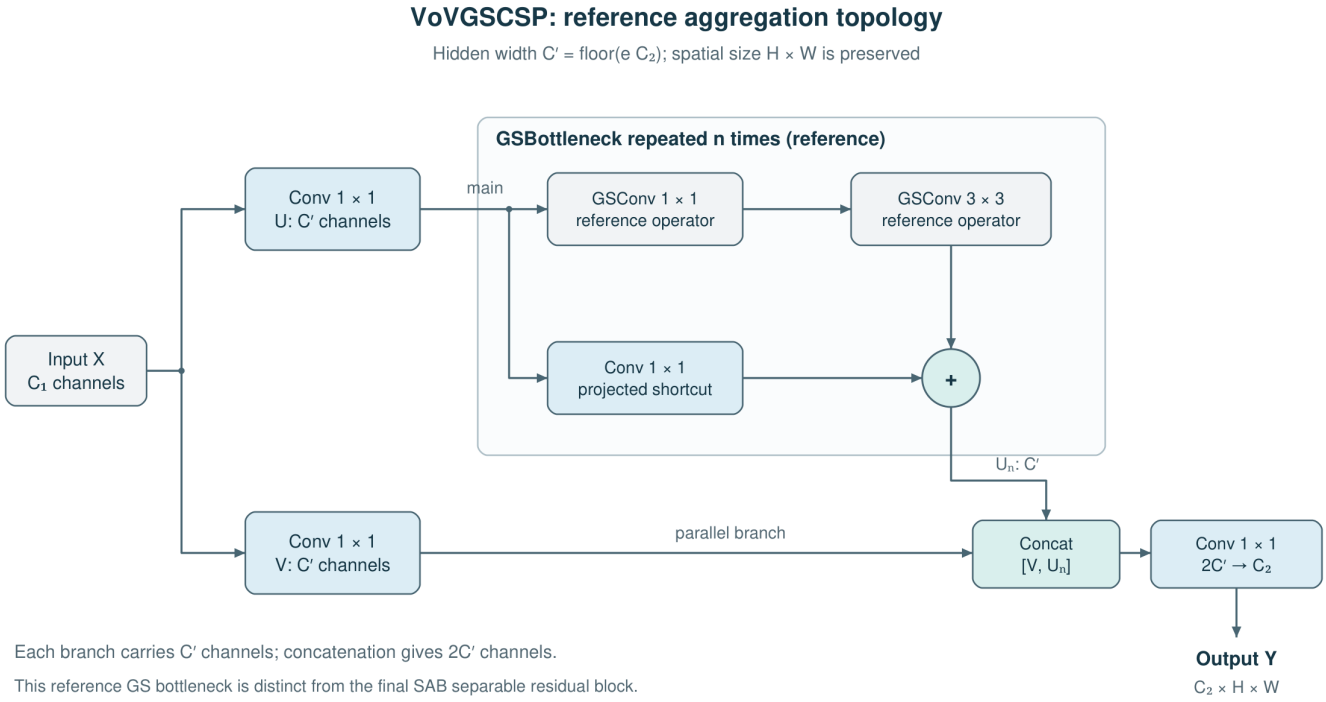


Figure 3. VoVGSCSP reference topology used to construct SAB. A transformed branch and a parallel projection meet through channel concatenation and an output convolution.

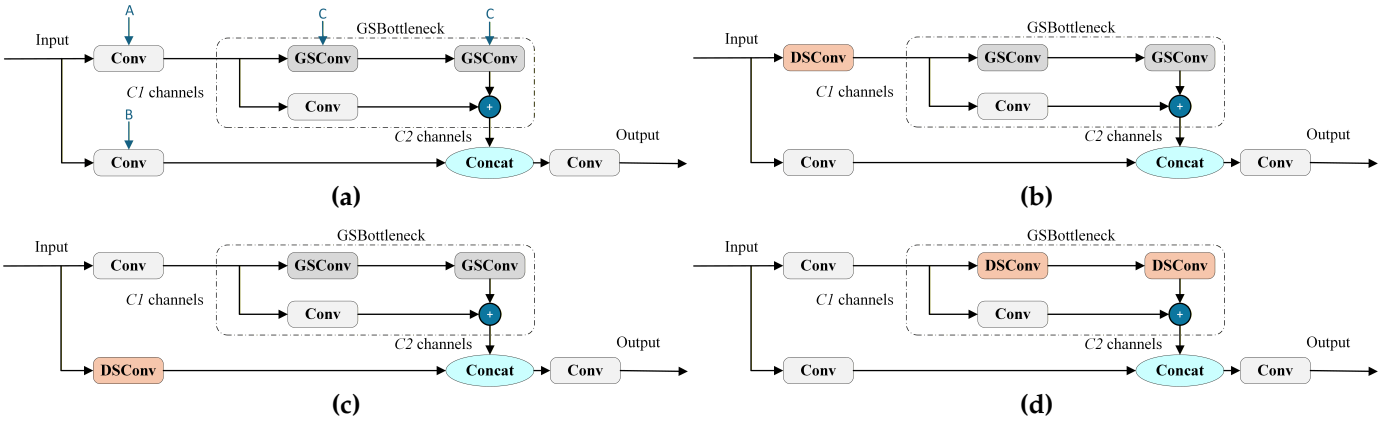


Figure 4. Separable-operator placement in the SAB design: (**a**) candidate positions in the reference block; (**b**) main-branch projection, configuration A; (**c**) parallel projection, configuration B; (**d**) internal bottleneck operators, configuration C. Configuration C is used in DARA-Net.

3.4. Directional Weighted Pyramid

DWP combines trainable feature weighting with directional spatial processing. Its bidirectional paths carry semantic context toward finer resolutions and return aggregated local information toward coarser resolutions. The fusion principle follows BiFPN [29]; Figure 5 contrasts this pattern with the FPN and PANet arrangements.

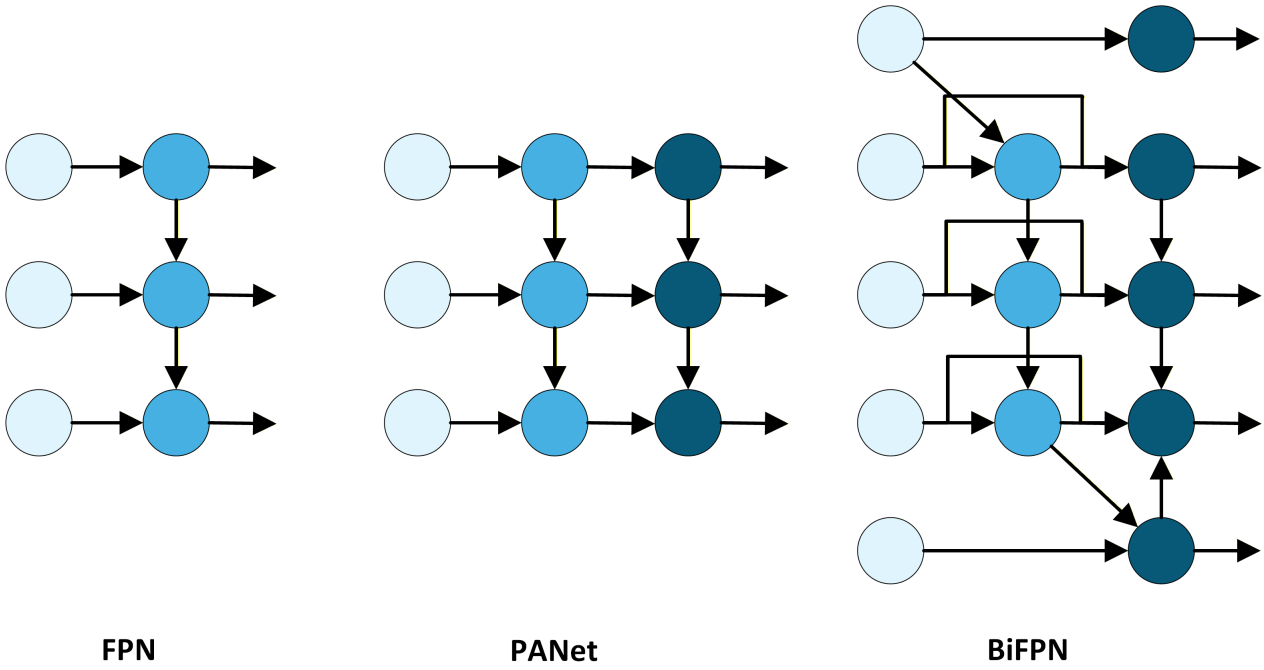


Figure 5. Reference feature-pyramid pathways for FPN, PANet, and BiFPN. Rows represent feature levels, while light, medium, and dark blue distinguish input nodes and successive aggregation stages rather than different spatial resolutions. Arrows indicate the schematic feature connections.

At a fusion node, N denotes the number of incoming feature maps. Each node has its own input count and trainable scalar parameters. Each input X_i has the same batch, channel, height, and width dimensions after spatial resizing and channel projection as needed. The aligned input maps are

$$\{X_1, X_2, \ldots, X_N\}, \tag{18}$$

Here $i \in \{1, \ldots, N\}$ indexes incoming feature maps, not individual pixels or image samples. There is one trainable scalar for each input:

$$w_1, w_2, \ldots, w_N. \tag{19}$$

The coefficient denominator uses the swish transform

$$\text{swish}(w_j) = w_j \cdot \sigma(w_j), \tag{20}$$

where σ is the sigmoid function defined above and j indexes all incoming maps in the denominator. The notation w_i denotes a trainable scalar; it is distinct from the spatial index w in the AWR equations. The coefficient assigned to input i is

$$\hat{w}_i = \frac{w_i}{\sum_{j=1}^{N} \text{swish}(w_j) + \varepsilon}, \qquad \varepsilon = 10^{-4}. \tag{21}$$

The hat distinguishes the effective fusion coefficient $\hat{w}_i$ from the raw trainable parameter w_i, and ε is the additive constant in the denominator. Each scalar coefficient is broadcast over the batch, channel, and spatial dimensions of its input tensor. The node output is then

$$Y = \sum_{i=1}^{N} \hat{w}_i X_i. \tag{22}$$

This parameterization learns both the relative contributions of the incoming maps and their overall scale. The numerator contains the raw scalar w_i, while the denominator sums swish-transformed scalars and includes $\varepsilon = 10^{-4}$. Thus, the coefficients are not constrained to be nonnegative or to sum to one. The expression specifies the weighting used here and should not be interpreted as a probability normalization or as identical to the nonnegative fusion weights in the reference BiFPN formulation. The additive constant alone does not provide a general guarantee that the denominator remains separated from zero for arbitrary learned weights.

Directional processing is provided by pinwheel-shaped convolution (PConv) [30]. Here, k denotes the spatial length of the elongated kernel, rather than the candidate index used in the AWR equations. In Figure 6, each padding tuple lists the left, right, top, and bottom padding amounts, in that order; C_m denotes the channel count of one branch, and concatenating four branches gives $4C_m$ channels. Four asymmetric padding patterns create branches processed with $1 \times k$ or $k \times 1$ kernels. Concatenation collects the branch outputs along the channel dimension, and a 2×2 convolution fuses them. Figure 6 details this operation.

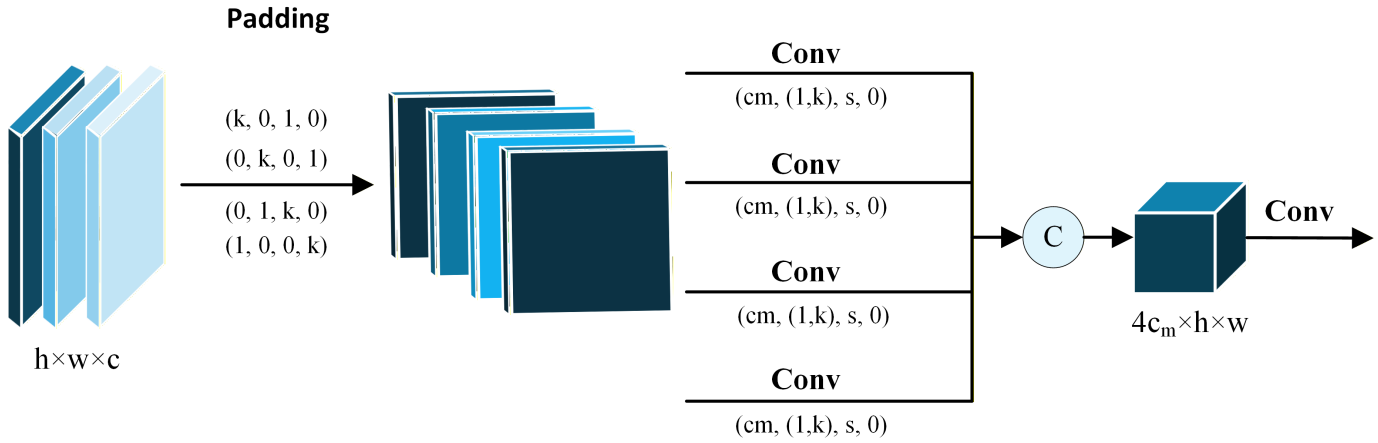


Figure 6. Directional processing with pinwheel-shaped convolution. Asymmetric padding defines four branches whose outputs are concatenated and projected to a common feature representation.

The DWP layout in Figure 7 embeds this directional operator within the bidirectional aggregation path. For each W node, its incoming arrows carry the X_i of Equation (18), Equations (19)–(21) define the scalar weights, and its outgoing arrow carries Y in Equation (22). The P node corresponds to the directional operator in Figure 6; the U and projection nodes align feature sizes before fusion. The diagram letters W, U, C, and P name network operations; they are distinct from tensor width W, the SAB feature U, channel count C, and precision P. Figure 5 provides reference pyramid topologies, rather than the specific weight formula used by DWP. Trainable fusion regulates the contribution of each scale, while the pinwheel branches encode local patterns along different orientations. This neck connects the AWR–SAB backbone to three prediction scales, completing the detail-aware representation and aggregation pipeline.

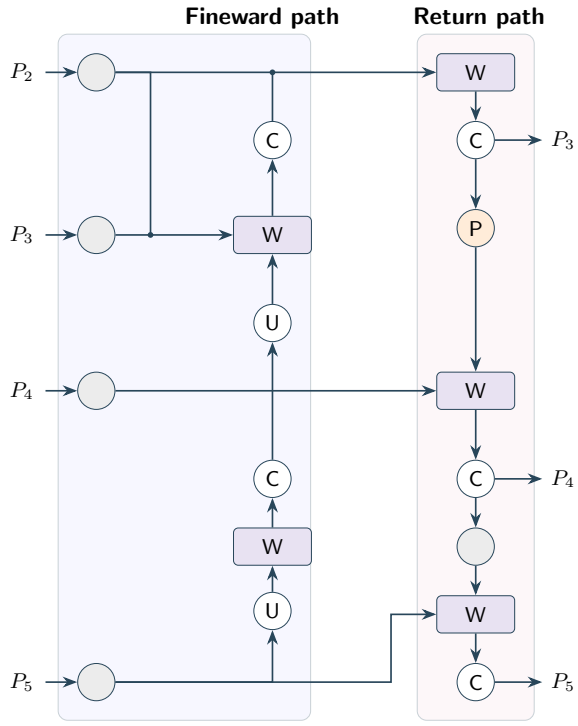


Figure 7. DWP aggregation paths. W denotes the weighted fusion node, U denotes upsampling, C denotes a C3k2 processing block, and P denotes PConv. Gray projection nodes align incoming backbone features. Light-purple and light-red backgrounds denote the fineward path and return path, respectively.

4. Experimental Evaluation

4.1. Data and Training Protocol

The evaluation used an assembled visible-light dataset containing 25,941 images with a single UAV category. Assembled refers to combining public imagery and locally captured scenes under a shared axis-aligned bounding-box annotation scheme. It does not denote a separate simulation benchmark. The training, validation, and test subsets contained 18,158, 5188, and 2595 images, respectively, corresponding to a 7:2:1 partition. The test subset included 2719 annotated UAV instances. Scene coverage included farmland, urban structures, clear sky, sea backgrounds, fog, nighttime observations, and multiple simultaneous targets. The aggregate test metrics below characterize the assembled test subset, rather than separate metrics for individual scene groups.

Training and model selection used the training and validation subsets. The quantitative comparisons reported below were evaluated on the test subset after model selection. Each model received 640×640 inputs and was trained for 100 epochs with a batch size of 64. Stochastic gradient descent used an initial learning rate of 0.01 and a final learning-rate factor of 0.01. Momentum was 0.937, weight decay was 0.0005, and warm-up lasted 3.0 epochs.

Training ran on an NVIDIA RTX 4090 with 24 GB of GPU memory and an Intel Xeon Gold 6430 providing 16 vCPUs. The host had 120 GB of RAM. The software environment comprised Python 3.10.8, PyTorch 2.1.2, CUDA 12.1, and Ultralytics 8.3.107. Test inference used an NVIDIA RTX 5060 Ti with Python 3.10.18, PyTorch 2.7.0, CUDA 12.8, and Ultralytics 8.3.107. Training settings, test inputs, and evaluation code were shared across the compared configurations.

The additional dataset validation runs below were conducted separately on an NVIDIA GeForce RTX 4060 with 8 GB of memory, Python 3.10.19, and PyTorch 2.7.1 with CUDA 11.8. DARA-Net and YOLOv11 used an Ultralytics 8.3.107 codebase, with the original custom

operators retained for DARA-Net; YOLOv26n used Ultralytics 8.4.174. DARA-Net's layer connections, state-tensor shapes, and forward output with shared weights were checked against the historical implementation before training. Historical weights were not used to initialize these additional runs.

The additional RGB models were initialized from scratch at an input size of 640 × 640, with physical batch size of 8, nominal batch size of 64, four data-loader workers, and early stopping disabled. SGD used a learning rate of 0.01, a final learning-rate factor of 0.01, momentum of 0.937, weight decay of 0.0005, and three warm-up epochs.

For additional validation, two public UAV datasets were processed using fixed local splits. The XML-format drone-dataset-uav collection, associated with the visible-imagery UAV detection study by Samadzadegan et al. [32], contains 1097 images and 1194 UAV annotations. The collection was divided locally into 768 training, 219 validation, and 110 test images. Video-derived images were grouped by source, while the remaining images were assigned using random seed 101, with near-duplicate candidates kept within the same split. All RGB detectors used identical partitions and annotations.

For ARD100, the original 65 training and 35 test videos were retained. Using seed 101, 13 training videos were assigned to validation and the remaining 52 to training, yielding 28,008 and 99,232 selected frames, respectively. Frames from the same video were kept within one partition. All detectors used identical center frames and annotations, while YOLOMG additionally used motion maps generated from neighboring frames.

4.2. Accuracy and Model Cost

Detection accuracy was characterized by precision (P), recall (R), and mean average precision (mAP). Precision measures the fraction of positive predictions assigned to true targets:

$$P = \frac{TP}{TP + FP}, \tag{23}$$

Recall measures the fraction of annotated targets recovered by the detector:

$$R = \frac{TP}{TP + FN}, \tag{24}$$

A true positive (TP) is a prediction matched to an annotated UAV under the class and localization criteria of the evaluation. A false positive (FP) is an unmatched or incorrect prediction, and a false negative (FN) is an annotated UAV without a matched detection. A ground-truth object can contribute at most one true-positive match at a given evaluation threshold; duplicate unmatched detections contribute false positives. Precision and recall depend on the confidence operating point as well as the localization criterion. The counts TP, FP, and FN are accumulated over the evaluated set; no primed variants are used in Equations (23) and (24). These equations give dimensionless ratios, which are expressed as percentages in the result tables. They measure detection accuracy, rather than computational efficiency.

For a predicted box B_p and a ground-truth box B_g, intersection-over-union is defined as $\mathrm{IoU}(B_p, B_g) = |B_p \cap B_g| / |B_p \cup B_g|$, where $|\cdot|$ denotes area. At a fixed IoU threshold τ, average precision (AP) summarizes the precision–recall curve obtained by ordering detections by confidence; it is computed numerically from an interpolated precision–recall curve by the evaluator. AP therefore summarizes performance across confidence levels, rather than the precision at one confidence threshold. For K classes, $\mathrm{mAP}_\tau = K^{-1} \sum_{c=1}^{K} \mathrm{AP}_{c,\tau}$. The metric mAP@0.5 uses $\tau = 0.5$, whereas mAP@0.5:0.95 averages mAP over the ten thresholds $0.50, 0.55, \ldots, 0.95$. Since the dataset contains one UAV class ($K = 1$), the class average at each IoU threshold equals AP for that class. These definitions apply to a labeled evaluation set; a confidence score displayed on one image is neither precision nor AP.

Model cost was recorded through parameter count in millions, serialized model size in megabytes, and computational complexity in GFLOPs. These quantities describe network capacity, storage, and floating-point operation count, respectively. They complement detection accuracy but are not measurements of latency or throughput.

4.3. Detector-Level Comparison

Table 1 compares DARA-Net with seven detector configurations under the reported evaluation protocol. YOLOv11 serves as the architectural baseline. The remaining comparators are YOLOv26n [33], YOLOv12, YOLOv10n, YOLOv9t, YOLOv8-world, and YOLOv6.

Table 1. Detection accuracy and model cost for the seven comparator detectors and DARA-Net. All accuracy values use the test subset and are percentages.

Model	P (%)	R (%)	mAP@0.5 (%)	mAP@ 0.5:0.95 (%)	Size (MB)	Params (M)	GFLOPs
YOLOv26n	96.67	91.76	95.15	60.77	5.43	2.375	5.3
YOLOv12	95.5	88.5	93.3	60.2	5.5	2.56	6.3
YOLOv11	96.7	88.5	94.0	60.8	5.5	2.58	6.3
YOLOv10n	94.6	88.5	93.7	60.6	5.8	2.69	8.2
YOLOv9t	96.4	88.5	93.3	60.4	4.7	1.97	7.6
YOLOv8-world	96.2	89.1	93.9	60.5	8.4	4.05	12.8
YOLOv6	96.3	88.3	93.3	60.3	8.7	4.23	11.8
DARA-Net	97.0	91.1	95.1	62.0	3.9	1.79	6.1

As shown in Table 1, DARA-Net achieves the highest precision and mAP@0.5:0.95 among the evaluated detectors, while maintaining the smallest parameter count and model size. Compared with the YOLOv11 baseline, precision increases from 96.7% to 97.0%, recall from 88.5% to 91.1%, and mAP@0.5 from 94.0% to 95.1%. The corresponding improvements are 0.3, 2.6, and 1.1 percentage points, respectively. Meanwhile, mAP@0.5:0.95 increases by 1.2 percentage points, reaching 62.0%. Relative to YOLOv12, DARA-Net improves precision and recall by 1.5 and 2.6 percentage points, respectively, while both mAP measures increase by 1.8 percentage points.

The comparison with YOLOv26n provides a further assessment of detection accuracy and model complexity. YOLOv26n achieves slightly higher recall (91.76% versus 91.1%) and mAP@0.5 (95.15% versus 95.1%), with a lower computational cost of 5.3 GFLOPs. In contrast, DARA-Net achieves higher precision (97.0% versus 96.67%) and mAP@0.5:0.95 (62.0% versus 60.77%). DARA-Net also reduces the parameter count from 2.375 M to 1.79 M and model size from 5.43 MB to 3.9 MB, corresponding to reductions of approximately 24.6% and 28.2%, respectively. These results suggest that DARA-Net offers a more compact model with competitive detection accuracy, particularly when localization performance across multiple IoU thresholds is considered.

In terms of model efficiency, DARA-Net reduces the parameter count from 2.58 M to 1.79 M and model size from 5.5 MB to 3.9 MB compared with YOLOv11, corresponding to reductions of approximately 30.6% and 29.1%, respectively. Computational complexity also decreases from 6.3 to 6.1 GFLOPs. Although YOLOv26n requires fewer floating-point operations, DARA-Net maintains lower parameter and storage requirements while achieving the highest mAP@0.5:0.95 among the compared models.

4.4. Contribution of the Three Components

The component experiment evaluated every combination of AWR, SAB, and DWP using the YOLOv11 baseline defined in Section 3.1. In Table 2, Model1, Model2, and Model3 denote the baseline with AWR, SAB, and DWP alone, respectively. Model4 uses AWR + SAB, Model5

uses AWR + DWP, and Model6 uses SAB + DWP. DARA-Net includes all three components. Wherever AWR is present it uses LogSoftmax modulation, and wherever SAB is present, it uses the internal bottleneck placement designated as configuration C. Thus, the Model1–Model6 labels identify component combinations, whereas A/B/C in the placement study identify positions of separable operators.

The full combination produced the highest recall and both mAP values among the ablated configurations. It also had the smallest model size and parameter count. Relative to the strongest paired mAP@0.5 result, adding the third component increased this metric from 94.7% to 95.1%. The highest paired recall was 89.7%, compared with 91.1% for the complete network. These results support using the three operations together.

Table 2. Component combinations for DARA-Net. Checkmarks indicate included components. AWR uses LogSoftmax, and SAB uses configuration C. Accuracy values are percentages on the test subset.

Model	AWR	SAB	DWP	P (%)	R (%)	mAP@0.5 (%)	mAP@ 0.5:0.95 (%)	Size (MB)	Params (M)	GFLOPs
Baseline				96.7	88.5	94.0	60.8	5.5	2.58	6.3
Model1	✓			95.7	89.2	93.8	60.5	4.7	2.20	6.1
Model2		✓		97.8	88.8	94.4	61.5	5.1	2.40	6.4
Model3			✓	96.2	89.0	94.0	61.0	5.1	2.36	6.3
Model4	✓	✓		97.7	89.3	94.7	61.5	4.4	2.02	6.2
Model5	✓		✓	97.1	88.0	94.0	61.6	4.3	1.97	6.0
Model6		✓	✓	96.8	89.7	94.4	61.3	4.7	2.18	6.5
DARA-Net	✓	✓	✓	97.0	91.1	95.1	62.0	3.9	1.79	6.1

The individual configurations show different trade-offs. AWR provides the largest standalone parameter reduction, from 2.58 M to 2.20 M, while recall rises to 89.2%. However, its precision and both mAP values decrease relative to the baseline. SAB yields the highest standalone mAP@0.5:0.95 at 61.5%, and DWP retains 94.0% mAP@0.5 with 2.36 M parameters. Thus, no claim that every component improves every accuracy metric is supported by these results.

Model2 (+SAB) has higher precision than the full model, at 97.8% versus 97.0%. The full model instead has higher recall (91.1% versus 88.8%), mAP@0.5 (95.1% versus 94.4%), and mAP@0.5:0.95 (62.0% versus 61.5%), with fewer parameters. These results favor the complete model when target recovery and compactness are considered jointly; they do not imply a precision advantage over Model2. Because precision and recall depend on the confidence operating point, the aggregate values alone do not identify a unique mechanism for this difference.

The paired configurations further distinguish the design choices. AWR with SAB reaches 94.7% mAP@0.5, and SAB with DWP reaches 89.7% recall. The AWR–DWP pair uses 6.0 GFLOPs, the lowest operation count in this ablation, compared with 6.1 GFLOPs for DARA-Net. The complete model combines the highest recall and mAP values with the lowest parameter count and storage requirement. These configuration-level differences support evaluating the components together, but their gains cannot be added as independent contributions or taken as proof of a general interaction mechanism across datasets.

4.5. Choice of Local Modulation

The coefficient-mapping comparison held the AWR feature branch fixed and varied its coefficient mapping. Softmax supplied positive normalized coefficients, and LogSoftmax supplied log-probability coefficients. Bi-SiLU denotes the SiLU-based variant evaluated in this study. Table 3 reports each configuration with the baseline backbone and neck; SAB and DWP are not included in these three variants.

As shown in Table 3, all three AWR variants have identical model costs, with 2.20 M parameters, 4.7 MB of storage, and 6.1 GFLOPs. However, their detection performance differs. LogSoftmax achieves the highest recall (89.2%) and mAP@0.5 (93.8%) among the three mappings, whereas Bi-SiLU obtains higher precision (97.0%) and mAP@0.5:0.95 (60.7%). For UAV detection, missed targets are an important concern, particularly when targets occupy limited image regions or appear against complex backgrounds. Therefore, LogSoftmax was selected for DARA-Net because it provides the most favorable recall-oriented trade-off among the evaluated AWR variants without increasing model complexity. Although its standalone mAP@0.5 remains slightly below the baseline, the complete DARA-Net achieves higher recall and mAP than the baseline when AWR is combined with SAB and DWP.

Table 3. Local coefficient mappings with the same AWR feature branch. Each variant changes the downsampling component of the baseline. Accuracy values are percentages on the test subset.

No.	Model	Type	P (%)	R (%)	mAP@0.5 (%)	mAP@ 0.5:0.95 (%)	Size (MB)	Params (M)	GFLOPs
1	YOLOv11	–	96.7	88.5	94.0	60.8	5.5	2.58	6.3
2	+AWR	Softmax	96.6	88.6	93.7	60.2	4.7	2.20	6.1
3	+AWR	LogSoftmax	95.7	89.2	93.8	60.5	4.7	2.20	6.1
4	+AWR	Bi-SiLU	97.0	88.0	93.7	60.7	4.7	2.20	6.1

4.6. Placement of Separable Operators

The SAB placement study changed where separable operators entered the reference aggregation block. Configurations A, B, and C correspond to the positions marked in Figure 4. Their test results are shown in Table 4.

Table 4. SAB operator placements corresponding to Figure 4. Accuracy values are percentages on the test subset.

Model	Position	P (%)	R (%)	mAP@0.5 (%)	mAP@ 0.5:0.95 (%)	Size (MB)	Params (M)	GFLOPs
Baseline	–	96.7	88.5	94.0	60.8	5.5	2.58	6.3
SAB-A	A	96.1	89.0	94.3	61.0	5.8	2.71	6.9
SAB-B	B	96.6	90.0	94.7	61.1	5.8	2.71	6.9
SAB-C	C	97.8	88.8	94.4	61.5	5.1	2.40	6.4

Configuration C required 2.40 M parameters and 6.4 GFLOPs, compared with 2.71 M parameters and 6.9 GFLOPs for configurations A and B. It achieved 97.8% precision and 61.5% mAP@0.5:0.95, the highest values among the three placements. Configuration B achieved the highest recall and mAP@0.5, at 90.0% and 94.7%, respectively. Configuration C was adopted for its combination of compactness, precision, and localization performance across the IoU thresholds.

4.7. Optimization Traces and Detection Examples

Figure 8 shows training losses and validation metrics over 100 epochs for the YOLOv11 baseline and DARA-Net. It shows a rapid initial change followed by more gradual changes in the later epochs. At the plotted scale, several validation curves appear similar. This figure documents optimization behavior and should be distinguished from the final test-set comparison in Table 1, where DARA-Net has higher recall and mAP.

Figure 9 presents qualitative detection examples of DARA-Net under four representative observation conditions: (a) urban and textured backgrounds, (b) sea and sky backgrounds, (c) foggy and multi-target scenes, and (d) dimly lit and nighttime scenes.

Each row contains two original image–prediction pairs, illustrating the model's detection behavior under different visual conditions.

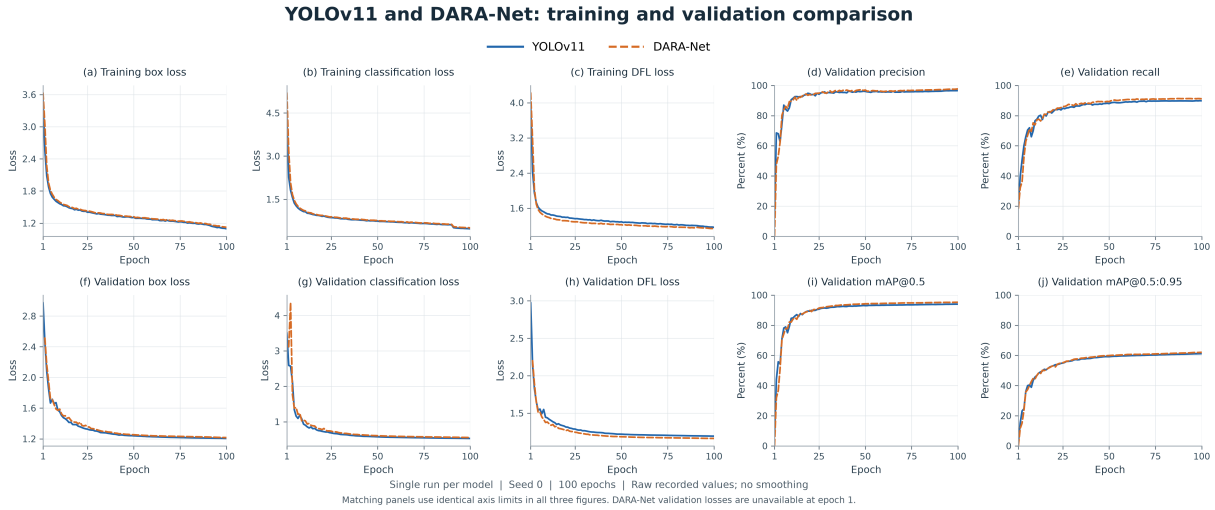


Figure 8. Training loss and validation performance curves of YOLOv11 and DARA-Net over 100 epochs. The curves illustrate the optimization behavior of the two models under the same training protocol.

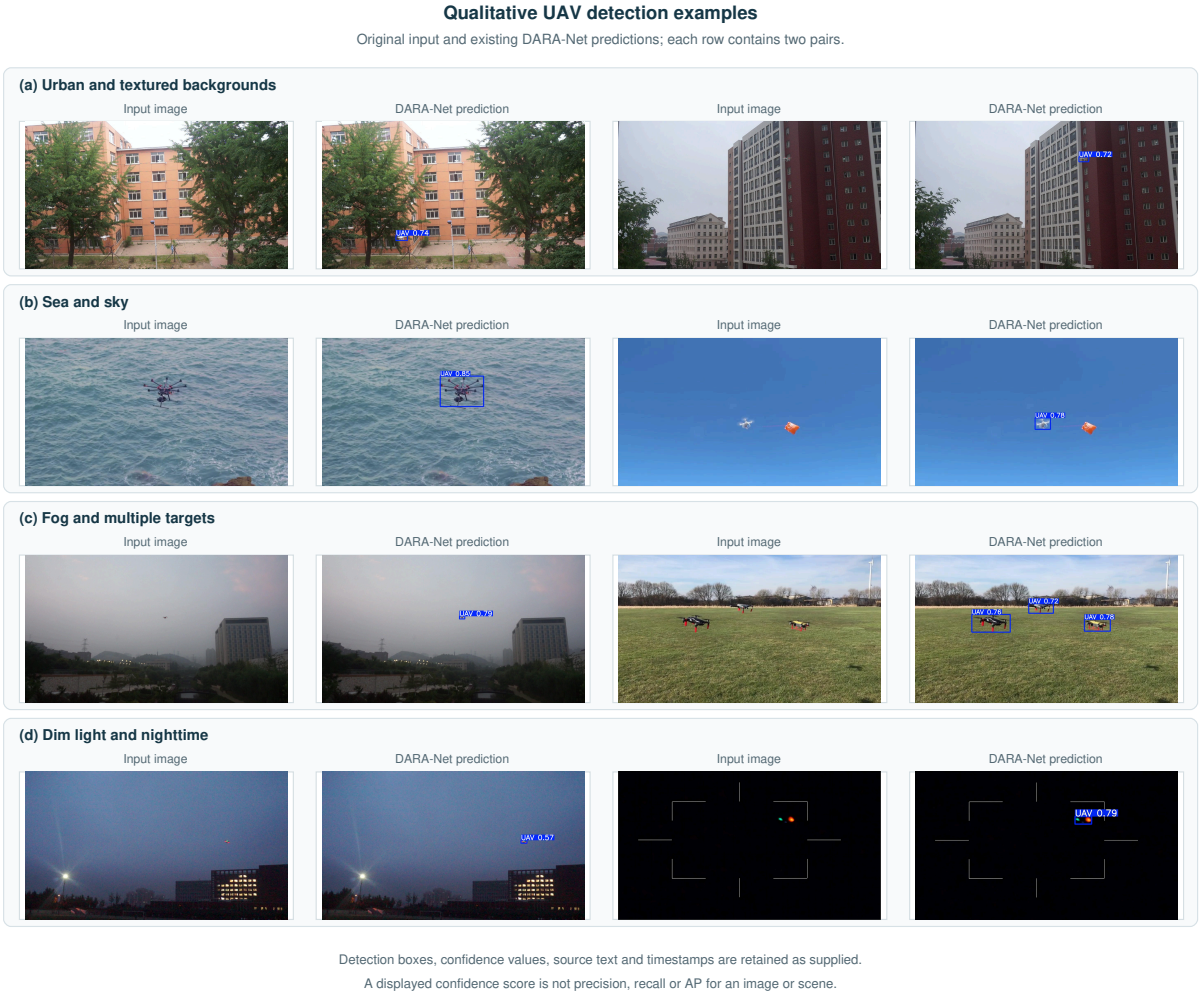


Figure 9. Qualitative detection results of DARA-Net under four representative scene conditions: (**a**) urban and textured backgrounds, (**b**) sea and sky, (**c**) foggy and multi-target scenes, and (**d**) dimly lit and nighttime scenes. Each row contains two original image–prediction pairs, with detection labels indicating the predicted class and confidence score.

The displayed labels indicate the predicted UAV class and its confidence score. These examples provide qualitative illustrations rather than scene-specific quantitative evaluations. The confidence scores should not be interpreted as precision, recall, or average precision. The images do not establish real-time acquisition or inference performance, and the acquisition-system text and timestamps visible in the nighttime examples are unrelated to the measured inference latency.

4.8. Additional Cross-Dataset Validation

To examine the performance of DARA-Net beyond the assembled UAV dataset, additional validation experiments were conducted on the XML-format drone-dataset-uav collection and ARD100. These experiments were performed under separate training configurations and are reported independently of the original detector comparison.

On the XML-format drone dataset, DARA-Net, YOLOv26n, and YOLOv11 were trained for 100 epochs using the configuration described in Section 4.1. Table 5 presents their validation results.

DARA-Net achieves the highest recall and both mAP measures among the three models on the XML validation split. Compared with YOLOv11, recall increases by 2.32 percentage points, while mAP@0.5 and mAP@0.5:0.95 increase by 3.73 and 5.10 percentage points, respectively. Relative to YOLOv26n, DARA-Net improves recall by 3.12 percentage points and mAP@0.5:0.95 by 4.17 percentage points, with comparable precision. These results support the effectiveness of DARA-Net under this additional validation setting.

Table 5. Validation results on the fixed local split of the XML-format drone dataset. All accuracy values are percentages.

Model	P	R	mAP@0.5	mAP@0.5:0.95
DARA-Net	79.32	65.86	76.42	45.79
YOLOv26n	79.37	62.74	73.23	41.62
YOLOv11	78.17	63.54	72.69	40.69

Further experiments were conducted on ARD100 to examine detection performance in video-derived UAV imagery. DARA-Net, YOLOv26n, and YOLOMG-640 were evaluated using the validation records obtained immediately after completing epoch 15 of their respective 100-epoch training schedules. All experiments used random seed 101 and the same center-frame validation split, comprising 28,008 selected frames from 13 held-out training videos. YOLOMG-640 additionally used motion maps generated from neighboring frames. The resulting interim validation metrics are presented in Table 6.

Table 6. Interim ARD100 validation results recorded after epoch 15 of the planned 100-epoch training schedules. YOLOMG-640 uses additional motion information.

Model	Epochs	Input	P	R	mAP@0.5	mAP@0.5:0.95
DARA-Net	15/100	RGB	64.09	25.34	29.17	15.63
YOLOv26n	15/100	RGB	63.04	31.38	33.97	17.27
YOLOMG-640	15/100	RGB + motion	67.57	55.35	62.19	30.73

At epoch 15, DARA-Net achieves slightly higher precision than YOLOv26n (64.09% versus 63.04%), but lower recall (25.34% versus 31.38%) and mAP@0.5 (29.17% versus 33.97%). YOLOMG-640 obtains the highest values across all four accuracy measures. However, its use of both appearance and motion information differs from the RGB-only inputs of DARA-Net and YOLOv26n.

Table 7 summarizes the recorded model costs for the additional experiments. DARA-Net has the smallest parameter count and model size among the evaluated configurations, while YOLOv26n requires fewer GFLOPs. YOLOMG-640 uses a larger computational budget associated with its dual-input architecture. These measurements describe model complexity rather than inference latency, and the reported GFLOPs exclude preprocessing and motion-map generation.

Table 7. Recorded model costs for the additional validation experiments. ARD100 values correspond to epoch 15. GFLOPs exclude preprocessing and postprocessing; cost measurements from different experimental protocols should not be treated as directly interchangeable.

Dataset	Model	Size (MB)	Params (M)	GFLOPs
XML	DARA-Net	3.96	1.788	6.1
XML	YOLOv26n	5.43	2.375	5.3
XML	YOLOv11	5.50	2.582	6.3
ARD100 (epoch 15)	DARA-Net	3.93	1.788	6.0
ARD100 (epoch 15)	YOLOv26n	5.39	2.375	5.3
ARD100 (epoch 15)	YOLOMG-640	7.09	2.036	19.4

4.9. Inference Latency Analysis

To evaluate the inference efficiency of DARA-Net, latency measurements were conducted on an NVIDIA GeForce RTX 4060 GPU using FP32 inference, a batch size of 1, and 640 × 640 letterbox inputs. Each model was warmed up for 100 iterations before processing 1097 images from the XML-format drone dataset. Forward-pass latency and complete inference-pipeline latency were measured separately. The pipeline measurements include preprocessing, network inference, and postprocessing. Table 8 reports the measured mean latency and the 95th-percentile latency (p95).

Table 8. Inference latency comparison on an NVIDIA RTX 4060 GPU using FP32 precision, batch size 1, and 640 × 640 inputs. All values are in milliseconds per image. The configurations prefixed with "+" represent individual or combined modifications to the YOLOv11 baseline.

Model	Forward Mean	Forward p95	Preprocess Mean	Postprocess Mean	Pipeline Mean	Pipeline p95
YOLOv11	4.966	7.131	1.399	0.877	7.266	9.283
+AWR	6.186	8.544	1.389	0.869	8.402	10.683
+SAB	5.434	7.719	1.393	0.861	7.632	9.633
+DWP	6.625	9.643	1.387	0.875	8.785	11.184
+AWR + SAB	6.583	9.300	1.396	0.874	8.771	11.067
+AWR + DWP	7.702	10.623	1.393	0.864	9.980	12.533
+SAB + DWP	7.043	10.050	1.397	0.869	9.186	11.197
DARA-Net	8.128	11.137	1.391	0.874	10.229	12.574
YOLOv26n	6.281	10.258	1.420	0.959	8.564	12.052

As shown in Table 8, DARA-Net achieves an average forward-pass latency of 8.128 ms and an average pipeline latency of 10.229 ms per image. Compared with the YOLOv11 baseline, whose corresponding latencies are 4.966 ms and 7.266 ms, DARA-Net exhibits a 40.8% increase in average pipeline latency. YOLOv26n records an average pipeline latency of 8.564 ms, which is also lower than that of DARA-Net. Thus, the reduction in parameter count and model size achieved by DARA-Net does not translate into faster inference under the evaluated hardware configuration.

The component-level measurements provide further insight into the latency overhead. Relative to YOLOv11, introducing AWR alone increases the average pipeline latency from 7.266 ms to 8.402 ms, while SAB and DWP increase it to 7.632 ms and 8.785 ms,

respectively. Among the individual components, DWP introduces the largest measured latency increase, followed by AWR, whereas SAB has a comparatively smaller effect. The combined configurations exhibit additional latency, with the complete DARA-Net requiring 10.229 ms per image.

Preprocessing and postprocessing times remain relatively similar across the evaluated configurations, indicating that most of the observed pipeline-latency differences arise during network execution.

5. Discussion

The experimental results indicate that DARA-Net provides a favorable balance between detection accuracy and model compactness on the assembled UAV dataset. Compared with the YOLOv11 baseline, the proposed model improves recall and both mAP measures while reducing parameter count and model size. These gains are achieved by combining adaptive window reduction (AWR), separable aggregation blocks (SAB), and a directional weighted pyramid (DWP) within a single detection architecture. The ablation results further show that the full configuration achieves higher recall and mAP than the individual and paired configurations, although the SAB-only model attains higher precision. This suggests that the three components offer complementary contributions to the overall accuracy–cost trade-off rather than independently improving every evaluation metric.

The comparison with YOLOv26n further clarifies the contribution and intended positioning of DARA-Net. Although the two models achieve comparable mAP@0.5 values (95.1% versus 95.15%), their accuracy–cost characteristics differ. DARA-Net achieves a higher mAP@0.5:0.95 (62.0% versus 60.77%) while requiring approximately 24.6% fewer parameters and 28.2% less model storage. These results indicate that the proposed architecture offers a favorable trade-off for UAV detection applications in which model compactness and localization performance are important considerations. In contrast, YOLOv26n achieves slightly higher recall, lower computational complexity, and faster inference on the evaluated GPU. Therefore, DARA-Net is not presented as universally superior to YOLOv26n, but as an alternative architecture with a different balance of detection accuracy, model size, and execution cost. Beyond the numerical comparison, its contribution lies in the coordinated design and ablation-based evaluation of adaptive spatial reduction, separable feature aggregation, and directional multiscale fusion within a compact detection framework.

The qualitative examples in Figure 9 illustrate DARA-Net's detection behavior under different observation conditions, including urban backgrounds, open sky, foggy scenes, and nighttime imagery. These examples complement the aggregate test-set evaluation by presenting individual detection results under varied visual conditions. However, they do not establish quantitative improvements for each scene category.

To further investigate the remaining detection errors, the prediction results of YOLOv11 and DARA-Net were manually examined against the ground-truth annotations. The reviewed cases were grouped into three categories: missed detection, false detection, and duplicate detection. The corresponding statistics are summarized in Table 9.

Table 9. Manually classified detection errors for YOLOv11 and DARA-Net. The counts describe the reviewed cases in the three reported categories rather than an exhaustive decomposition of all evaluation errors.

Error Category	YOLOv11	DARA-Net
Missed Detection	218	169
False Detection	61	53
Duplicate Detection	13	14

As shown in Table 9, the number of manually identified missed detections decreases from 218 for YOLOv11 to 169 for DARA-Net, corresponding to 49 fewer cases. False detections also decrease from 61 to 53, whereas duplicate detections remain nearly unchanged, with 13 cases for YOLOv11 and 14 for DARA-Net. These findings suggest that the proposed model is more effective in recovering UAV targets and reducing incorrect detections in the examined cases.

Figure 10 presents representative detection errors of YOLOv11 and DARA-Net using identical input images and ground-truth annotations. In Figure 10a, neither model detects the annotated UAV against a background containing dense vegetation, illustrating the difficulty of recovering visually ambiguous targets. In Figure 10b, both models generate predictions that do not correctly match the annotated UAV. These unmatched predictions constitute false positives, while the failure to recover the ground-truth target also results in a missed detection. The DARA-Net prediction has a relatively low displayed confidence score of 0.21, indicating uncertainty in this individual prediction rather than a measurement of dataset-level recall. In Figure 10c, YOLOv11 produces a single detection, whereas DARA-Net generates two overlapping predictions associated with the same UAV, illustrating a remaining limitation in duplicate-prediction suppression. These examples complement the manually reviewed error statistics in Table 9 and highlight opportunities to improve target discrimination, localization, and prediction consistency under challenging visual conditions.

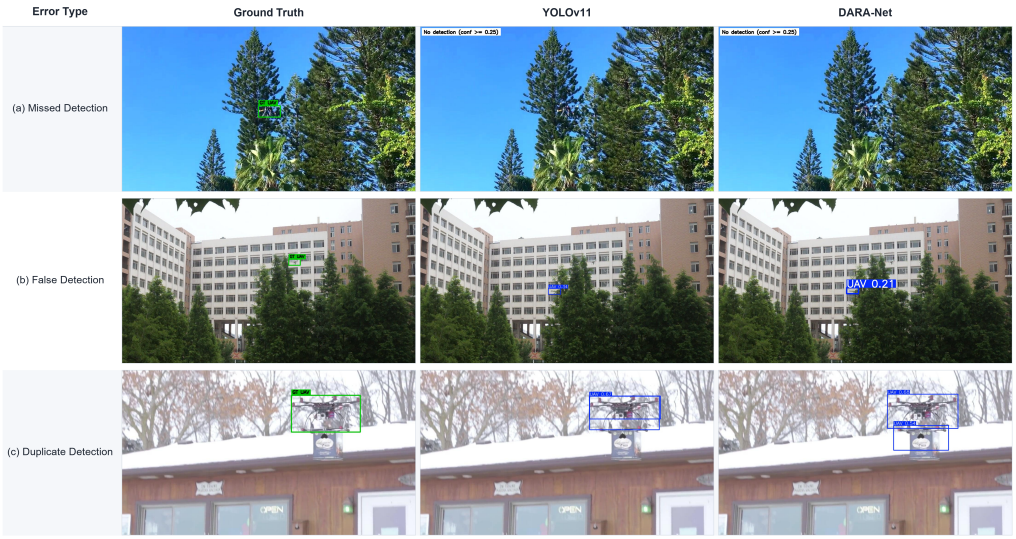


Figure 10. Representative detection errors of YOLOv11 and DARA-Net on identical input images. Rows illustrate (**a**) missed detection of an annotated UAV by both models, (**b**) false-positive predictions of the ground-truth UAV, and (**c**) duplicate predictions produced by DARA-Net. Green boxes indicate ground-truth annotations, while blue boxes represent model predictions with their confidence scores. These examples illustrate individual failure cases rather than the overall frequency of detection errors.

Additional validation on the XML-format drone dataset offers evidence beyond the original assembled dataset. Under the completed 100-epoch training configuration, DARA-Net achieves higher recall and both mAP measures than YOLOv26n and YOLOv11, while maintaining a smaller parameter count and model size. The ARD100 experiments provide additional insight into the behavior of DARA-Net during training on video-derived UAV imagery. At epoch 15, DARA-Net achieves slightly higher precision than YOLOv26n (64.09% versus 63.04%) but lower recall (25.34% versus 31.38%) and mAP@0.5 (29.17% versus 33.97%). YOLOMG-640 achieves the highest accuracy measures among the three

configurations. Although all models were evaluated after 15 training epochs using the same center-frame validation split, their input modalities, batch sizes, and evaluation pipelines differ. In particular, YOLOMG-640 incorporates additional motion information unavailable to the RGB-only detectors. Therefore, these results represent an interim comparison rather than a definitive assessment of fully trained models.

Despite these encouraging results, several limitations should be acknowledged. First, although additional validation on public UAV datasets provides evidence beyond the assembled dataset, independent test-set evaluation remains necessary to further assess the generalization of DARA-Net. Second, while the proposed architecture reduces parameter count and model size, its inference latency on the NVIDIA RTX 4060 is higher than that of the YOLOv11 baseline, indicating a trade-off between model compactness and execution efficiency. Third, although DARA-Net reduces missed detections in the examined cases, challenges remain in detecting visually ambiguous targets and suppressing duplicate predictions. These limitations motivate further investigation into cross-dataset generalization, hardware-aware optimization, and robust detection under challenging visual conditions.

Overall, the results support DARA-Net as a compact architecture that improves target recovery and localization accuracy over its YOLOv11 baseline on the primary test subset. Its principal contribution lies in coordinating local feature reduction, lightweight representation learning, and multiscale aggregation to achieve a useful accuracy–compactness trade-off. The additional evaluations also clarify the conditions under which this advantage has been observed and the further evidence required to assess its broader applicability.

6. Conclusions

This study presents DARA-Net, a compact detection framework designed to improve visible-light UAV detection through coordinated feature reduction, representation, and multiscale aggregation. Built on YOLOv11, the proposed architecture integrates adaptive window reduction, separable feature transformation, and directional weighted aggregation to balance detection accuracy and model complexity.

On the assembled UAV test subset, DARA-Net achieves 97.0% precision, 91.1% recall, 95.1% mAP@0.5, and 62.0% mAP@0.5:0.95, with 1.79 M parameters and a model size of 3.9 MB. Compared with the YOLOv11 baseline, recall improves by 2.6 percentage points, while the parameter count decreases by approximately 30.6%. Compared with YOLOv26n, DARA-Net achieves a 1.23-percentage-point improvement in mAP@0.5:0.95 with approximately 24.6% fewer parameters and a 28.2% smaller model file. These findings highlight its value as a compact alternative for UAV detection tasks that prioritize localization performance and model storage efficiency. The ablation experiments support the effectiveness of the complete configuration, while qualitative examples illustrate the model's detection behavior under different observation conditions.

Additional validation on the XML-format drone dataset demonstrates competitive performance compared with YOLOv26n and YOLOv11, while the interim ARD100 results show that the relative performance varies under different data and training conditions. Together, these results indicate that DARA-Net provides a useful accuracy–compactness trade-off for the evaluated UAV detection tasks.

Latency measurements further show that the reduction in parameter count and model size does not translate into faster inference on the evaluated GPU. Future work will focus on improving execution efficiency, evaluating deployment performance on resource-constrained hardware, and extending validation to additional UAV datasets, particularly under challenging visual conditions.

Author Contributions: Conceptualization, T.C.; methodology, L.J.; software, L.J.; validation, L.J.; formal analysis, L.J. and X.L.; investigation, L.J., X.L. and X.C.; data curation, L.J.; visualization,

L.J.; writing—original draft preparation, L.J.; writing—review and editing, X.L., X.C., T.C. and J.L.; supervision, X.L. and T.C.; project administration, T.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Guangdong University Engineering Technology Research Center for Precision Components of Intelligent Terminal of Transportation Tools under Grant 2021GCZX002; the Shenzhen Science and Technology Program through the project "Large-Scale and High-Density Low-Altitude Aircraft Autonomous Learning Conflict Detection and Multi-Dimensional Resolution Technology" under Grant KJZD20240903095712016; and the Guangdong Provincial Key Laboratory of Low-Altitude Intelligent Perception and Navigation Safety under Grant 2026B1201010001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The visible-light UAV dataset combines public imagery with locally captured scenes using a shared axis-aligned bounding-box annotation scheme. The assembled dataset is accessible at https://github.com/star-jly/UAV-dataset (accessed on 20 October 2026). The implementation code for the proposed DARA-Net architectural modifications has been made publicly available at https://github.com/star-jly/DARA_Net (accessed on 20 October 2026).

Acknowledgments: OpenAI's ChatGPT(GPT-5.5 and GPT-6.0) and Codex were used for language editing, LaTeX formatting, and manuscript-to-code and manuscript-to-results consistency checks, including wording revisions in the Method and Experiments sections. These tools did not generate experimental data.

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

References

1. Zheng, K.; Li, C.; Yang, L.; Pan, X.; Zhou, S. A framework of data-driven real-time risks prediction method on UAV flight safety. In *Equipment Intelligent Operation and Maintenance*; CRC Press: Boca Raton, FL, USA, 2025; pp. 758–765. [CrossRef]
2. Bai, Z.; Shi, J.; Li, Z.; Li, M.; Chen, K.C. Rule-guided DRL for UAV-assisted wireless sensor networks with no-fly zones safety. *IEEE Trans. Cogn. Commun. Netw.* **2025**, *11*, 1268–1280. [CrossRef]
3. Fu, Y.; Wang, B.; Zhao, H.; Zhou, M.; Li, N.; Gao, Z. Adaptive safety attitude control of a hybrid VTOL UAV under transition flight subject to multiple faults and uncertainties. *Aerosp. Sci. Technol.* **2025**, *163*, 110284. [CrossRef]
4. Khargharia, H.S.; Ouali, A.; Shakya, S.; Ahmad, S. Collision avoidance in UAV swarms: A learning-centric perspective on collaborative intelligence. *Neurocomputing* **2026**, *663*, 132020. [CrossRef]
5. Li, D.; Cai, H.; Wang, T.; Zhao, J.; Hua, Y.; Li, T.; Liu, J.; Song, S. Analysis of low-altitude UAV small target detection and snapshot spectral imaging applications (Invited). *Acta Opt. Sin.* **2025**, *45*, 1720021. [CrossRef]
6. Zuo, G.; Zhou, K.; Wang, Q. UAV-to-UAV small target detection method based on deep learning in complex scenes. *IEEE Sens. J.* **2025**, *25*, 3806–3820. [CrossRef]
7. Gao, Y.; Xin, Y.; Yang, H.; Wang, Y. A lightweight anti-unmanned aerial vehicle detection method based on improved YOLOv11. *Drones* **2025**, *9*, 11. [CrossRef]
8. Wu, C.; Cai, C.; Xiao, F.; Wang, J.; Guo, Y.; Ma, L. YOLO-LSM: A lightweight UAV target detection algorithm based on shallow and multiscale information learning. *Information* **2025**, *16*, 393. [CrossRef]
9. Chen, Z.; Fan, K.; Ye, J.; Xu, Z.; Wei, Y. A lightweight multi-module collaborative optimization framework for detecting small unmanned aerial vehicles in anti-unmanned aerial vehicle systems. *Drones* **2026**, *10*, 20. [CrossRef]
10. Chen, Z.; Ma, Y.; Gong, Z.A.; 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. [CrossRef] [PubMed]
11. Luan, H.; Xu, S.; Xu, D.; Liu, L.; Guo, M.; Huang, S. MD-DRIFPN: Dilated multi-directional FPN for small drone object detection. *J. Supercomput.* **2025**, *81*, 1646. [CrossRef]
12. Fan, R.; Nan, W.; Jiang, A.; Meng, H.; Wang, M.; Wang, X.; Dang, J.; Luo, X. Tri-FRNet: A three-path feature fusion network for high-precision small object detection. *Eng. Res. Express* **2025**, *7*, 045221. [CrossRef]
13. Guo, H.; Lin, X.; Zhao, S. YOLOMG: Vision-based drone-to-drone detection with appearance and pixel-level motion fusion. *arXiv* **2025**, arXiv:2503.07115. [CrossRef]

14. Shi, Z.; Lei, Z. Small UAV target detection algorithm using the YOLOv8n-RFL based on radar detection technology. *Sensors* **2025**, *25*, 5140. [CrossRef] [PubMed]
15. Fan, R.; Jiao, R.; Nan, W.; Meng, H.; Jiang, A.; Yang, X.; Zhao, Z.; Dang, J.; Wang, Z.; Tian, Y. ; et al. CS-YOLO: A small object detection model based on YOLO for UAV aerial photography. *Signal Process. Image Commun.* **2026**, *142*, 117460. [CrossRef]
16. Yang, Z.; Zhang, Y.; Zeng, J.; Yang, Y.; Jia, Y.; Song, H.; Lv, T.; Sun, Q.; An, J. AI-driven safety and security for UAVs: From machine learning to large language models. *Drones* **2025**, *9*, 392. [CrossRef]
17. Asghari, O.; Ivaki, N.; Madeira, H. UAV operations safety assessment: A systematic literature review. *ACM Comput. Surv.* **2025**, *57*, 1–37. [CrossRef]
18. Misbah, M.; Khan, M.U.; Kaleem, Z.; Muqaibel, A.; Alam, M.Z.; Liu, R.; Yuen, C. MSF-GhostNet: Computationally efficient YOLO for detecting drones in low-light conditions. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2025**, *18*, 3840–3851. [CrossRef]
19. Hu, X.; Wang, Y.; Chai, C.; Han, S. LEANet: Lightweight and efficient UAV infrared target detection network based on multi-scale attention. *Meas. Sci. Technol.* **2025**, *36*, 125402. Correction in *Meas. Sci. Technol.* **2026**, *37*, 029501. [CrossRef]
20. Li, F.; Lv, X.; Zhao, M.; Wu, W. IFD-YOLO: A lightweight infrared sensor-based detector for small UAV targets. *Sensors* **2025**, *25*, 7449. [CrossRef] [PubMed]
21. Xu, J.; Deng, L.; Zhou, T. GLANet: Global–local adaptive network for efficient rotated object detection. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 5655914. [CrossRef]
22. Qian, K.; Shen, J.; Wang, S.; Wu, Y.; Lu, G. SiamUF: SiamCar based small UAV tracker using dense U-shape deep features in near infrared videos. *Opt. Lasers Eng.* **2025**, *186*, 108825. [CrossRef]
23. Zhang, H.; Kuang, Y.; Wang, J.; Jin, L.; Xu, C.; Meng, Y.; Huang, B. SiamDiff: A diffusion-driven Siamese network for scale-aware anti-UAV tracking. *Remote Sens.* **2026**, *18*, 18. [CrossRef]
24. Han, Z.; Jia, D.; Zhang, L. LT-DETR: Lightweight UAV object detection and dual knowledge distillation for remote sensing scenarios. *Meas. Sci. Technol.* **2025**, *36*, 036005. [CrossRef]
25. Yu, Z.; Guan, Q.; Yang, J.; Yang, Z.; Zhou, Q.; Chen, Y.; Chen, F. LSM-YOLO: A compact and effective ROI detector for medical detection. In *Neural Information Processing*; Communications in Computer and Information Science; Springer: Singapore, 2025; Volume 2287, pp. 30–44. [CrossRef]
26. Li, H.; Li, J.; Wei, H.; Liu, Z.; Zhan, Z.; Ren, Q. Slim-neck by GSConv: A lightweight-design for real-time detector architectures. *J. Real.-Time Image Process.* **2024**, *21*, 62. [CrossRef]
27. Di, X.; Cui, K.; Wang, R.F. Toward efficient UAV-based small object detection: A lightweight network with enhanced feature fusion. *Remote Sens.* **2025**, *17*, 2235. [CrossRef]
28. Su, C.; Deng, X.; Xue, D. EfficientDet-EdgeUAV: A multi-scale fusion architecture for target detection in UAV imagery with edge computing optimization. *Internet Technol. Lett.* **2025**, *8*, e70118. [CrossRef]
29. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–18 June 2020; pp. 10781–10790. [CrossRef]
30. Yang, J.; Liu, S.; Wu, J.; Su, X.; Hai, N.; Huang, X. Pinwheel-shaped convolution and scale-based dynamic loss for infrared small target detection. In Proceedings of the Proc. AAAI Conf. Artif. Intell., Philadelphia, PA, USA, 25 February–4 March 2025; Volume 39, pp. 9202–9210. [CrossRef]
31. Khanam, R.; Hussain, M. YOLOv11: An overview of the key architectural enhancements. *arXiv* **2024**, arXiv:2410.17725. [CrossRef]
32. Samadzadegan, F.; Dadrass Javan, F.; Ashtari Mahini, F.; Gholamshahi, M. Detection and recognition of drones based on a deep convolutional neural network using visible imagery. *Aerospace* **2022**, *9*, 31. [CrossRef]
33. Jocher, G.; Qiu, J.; Liu, M.; Lyu, S.; Akyon, F.C.; Kalfaoglu, M.E. Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models. *arXiv* **2026**, arXiv:2606.03748.

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.