Frontiers in Plant Science

TYPE Original Research
PUBLISHED 11 December 2025
DOI 10.3389/fpls.2025.1716703

YOLO-DCPG: a lightweight architecture with dual-channel pooling gated attention for intensive small-target agricultural pest detection

Jingwei Liu[1,2,3], En Yu[1,2,3], Yongke Li[1,2,3]*, Yunjie Zhao[1,2,3]* and Bowen Mao[1,2,3]

[1]College of Computer and Information Engineering, Xinjiang Agricultural University, Urumqi, China, [2]Xinjiang Agricultural Informatization Engineering Technology Research Center, Urumqi, China, [3]Research Center for Intelligent Agriculture, Ministry of Education Engineering, Urumqi, China

Check for updates

OPEN ACCESS

EDITED BY
Chaolong Zhang,
Jinling Institute of Technology, China

REVIEWED BY
Jianlong Wang,
Henan Polytechnic University, China
Qing Dong,
Northeastern University, United States

*CORRESPONDENCE
Yongke Li
lyk@xjau.edu.cn
Yunjie Zhao
zyj@xjau.edu.cn

RECEIVED 30 September 2025
REVISED 08 November 2025
ACCEPTED 21 November 2025
PUBLISHED 11 December 2025

CITATION
Liu J, Yu E, Li Y, Zhao Y and Mao B (2025) YOLO-DCPG: a lightweight architecture with dual-channel pooling gated attention for intensive small-target agricultural pest detection. *Front. Plant Sci.* 16:1716703. doi: 10.3389/fpls.2025.1716703

Accurate and rapid identification of agricultural pests is essential for intelligent pest monitoring. However, existing pest detection models often suffer from high parameter counts and computational complexity, limiting their deployment on edge devices. To address these challenges, this paper proposes a lightweight agricultural pest detection model, YOLO-DCPG, based on YOLOv8n. First, a Dual Channel Pooling Gated Attention (DCPGAttention) module is designed, applying mean and standard deviation pooling to enhance global information capture. A lightweight backbone network, StarNet, is employed for feature extraction, while a feature fusion neck, Small-Neck, is introduced, based on an improved bidirectional feature pyramid network (a-BIFPN) and integrating the efficient GSConv module. This design reduces model parameters and computational cost while maintaining detection accuracy. Furthermore, a scale factor based on Inner-IoU is incorporated into the WIoU loss function, enabling more precise control over auxiliary bounding boxes and improving regression for small pest regions. Experimental results on the Pest24 dataset show that YOLO-DCPG achieves a precision of 80.1%, mAP@50 of 74%, and mAP@50~95 of 47.5%, representing improvements of 4.5%, 0.8%, and 0.9% over the baseline YOLOv8n, respectively. Meanwhile, the number of parameters, GFLOPs, and model size are reduced by 51.2%, 30.1%, and 46.7%, respectively. Finally, YOLO-DCPG is successfully deployed on Raspberry Pi 4B, achieving stable, real-time pest detection, demonstrating its effectiveness and practicality for edge agricultural applications.

KEYWORDS

agricultural pest detection, lightweight, DCPGAttention, StarNet, small-neck

1 Introduction

As one of the largest agricultural countries in the world, agricultural pests have seriously affected the growth of crops in China Sprague (1975). The wide variety of pest species and the difficulty in identifying them have long troubled agricultural practitioners. Traditional pest identification relies primarily on experts, which requires extensive professional knowledge and is time-consuming. Manual identification often fails to provide timely feedback on pest outbreaks, leading to missed opportunities for optimal pest control (Song et al., 2025). Therefore, timely and accurate pest detection is crucial for crop protection.

Traditional Automated pest identification methods mainly rely on image processing and feature engineering combined with machine learning algorithms for pest classification. Ebrahimi et al. (2017) employed Support Vector Machines (SVM) for automatic detection of thrips in strawberry greenhouses. Chodey and Shariff (2023) proposed a method for extracting texture features based on the Gray-Level Co-occurrence Matrix (GLCM). Amrani et al. (2024) proposed a pest detection method based on Bayesian multitask learning and conducted aphid detection experiments on images of various crops. The detection accuracies for aphids in corn, rapeseed, rice, and wheat images were 75.77%, 66.39%, 70.01%, and 59%, respectively. However, these methods were tested on images with simple backgrounds and only a few pest species. They did not fully consider the complexity of real agricultural environments and the diversity of pests. Although these methods show progress in pest detection and have simple structures with high computational efficiency, which is useful for real-time or resource-limited applications, they mainly rely on low-level features such as color and shape for classification. This limits the generalization ability of the models. In addition, most traditional methods focus on datasets with simple backgrounds and few pest types. This makes them less suitable for real-world conditions. Compared to them, deep learning methods use convolutional neural networks (CNNs) to extract global contextual features. This overcomes the limits of traditional methods and greatly improves the accuracy and robustness of agricultural pest detection.

In recent years, with the rapid development of convolutional neural networks (CNNs), agricultural pest detection has become a popular research area. Currently, CNN-based object detection algorithms are mainly divided into two-stage and single-stage methods.

The two-stage methods include Regions with Convolutional Neural Network features (R-CNN) (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), and Faster R-CNN (Ren et al., 2016). These methods typically first generate candidate regions and then perform classification and bounding box regression for each region. For example, Ali et al. (2023)Ali et al. proposed a novel deep learning method called Faster-PestNet. It uses MobileNet to extract sample features and then employs an improved two-stage locator based on Faster R-CNN for pest recognition. The method achieved an accuracy of 82.43% on the IP102 pest dataset. However, it did not consider real agricultural environments or deployment on edge devices. Deepika et al. used Mask R-CNN to identify five types of pests and achieved good detection results. Nevertheless, their study focused on images with simple backgrounds and lacked research on insect detection in complex environments (Deepika and Arthi, 2022). Overall, two-stage methods have advantages in detection accuracy, but they typically incur high computational costs, making them less suitable for real-time applications and deployment on edge devices. The single-stage methods mainly include You Only Look Once (YOLO) (Adarsh et al., 2020; Bochkovskiy et al., 2020; Jocher et al., 2020; Li et al., 2022a; Khanam and Hussain, 2024; Varghese and Sambath, 2024; Wang et al., 2024) and Single Shot MultiBox Detector (SSD) (Liu et al., 2016). They are characterized by fast detection speed, making them suitable for real-time detection tasks. Depending on the research focus, they can be divided into small-pest detection methods and lightweight approaches.

In small-pest detection, certain studies have enhanced accuracy by incorporating feature enhancement modules and optimized loss functions. Dong et al. (2024) developed the PestLite model based on YOLOv5. By introducing a multi-level spatial pyramid pooling module (MTSPPF), the model effectively captures multi-scale features and achieved 90.7% mAP@50 on rice and corn pest detection tasks. Wen et al. (2022)Wen et al. proposed Pest-YOLO, a detection algorithm aimed at counting tiny pests. This method improves small target detection by incorporating an intersection loss function and a merging strategy. On the Pest24 dataset, Pest-YOLO achieved 77.71% recall and 69.59% mAP@50. However, the method only focuses on the number of detected pests and overlooks a comprehensive evaluation of detection accuracy, resulting in a low precision of 46.97%. Additionally, the model has a large number of parameters, which hinders deployment and real-time application on edge devices. Zhang et al. (2025) proposed an improved small-object detection model, FCDM-YOLOv8, for field crop pest detection. The model integrates lightweight modules, context feature enhancement, an additional small-object detection layer, and a dynamic detection head to improve accuracy and recall for dense and small pests. Experiments on VisDrone2019 and COCO2017-small demonstrated that FCDM-YOLOv8 outperforms YOLOv8n and other mainstream detectors. However, the model still shows limited mAP improvement and insufficient recall in complex scenarios. Liu et al. (2025) proposed an RP-DETR model for rice pest detection, which combines Transformers and CNNs. The model introduces a Small Target Improved Pyramid (STIP) in the neck, integrating Cross Stage Partial Networks and an Omni-Kernel Module with large convolution kernels to alleviate the problem of small-object feature loss and improve detection accuracy. However, the model suffers from high computational cost and relatively slow inference speed, which may limit its deployment in large-scale agricultural scenarios. Tang et al. (2023) proposed an enhanced version of Pest-YOLO by incorporating the ECA attention mechanism and a Transformer encoder to improve global feature capture. The model achieved 73.4% mAP@50 on the Pest24 dataset. However, this method did not adequately address model lightweight design and deployment on edge devices.

In terms of lightweight design, some studies have improved YOLO-series models through structural optimization and attention mechanisms. Dong et al. (2025b) proposed a lightweight deep learning model, REDNet, for real-time railway defect detection. By introducing efficient multi-scale feature aggregation and task decomposition modules, REDNet achieves high detection accuracy and fast inference speed, demonstrating the effectiveness of lightweight architectures in industrial inspection tasks. Similarly, Dong et al. (2025a) introduced RSNet, a lightweight and efficient detection framework tailored for small object detection in high-resolution remote sensing images. RSNet integrates compact detection heads and adaptive downsampling modules to enhance small-target sensitivity while maintaining computational efficiency, achieving state-of-the-art performance on DOTA and NWPU VHR-10 datasets. These studies highlight the versatility and effectiveness of lightweight architectures across diverse domains. In the agricultural field (Xue and Wang, 2025) proposed a lightweight detection model, PEW-YOLO. The model optimizes the PP-LCNet backbone with GSConv, introduces a lightweight PGNet backbone, integrates an efficient multi-scale attention mechanism in the neck, and adopts the Wise-IoU loss function. These improvements reduce parameters while enhancing accuracy and speed, enabling real-time detection of citrus pests and diseases. Huang et al. (2025) improved the YOLOv10n architecture using the SPD-Conv convolutional model, reverse residual attention mechanism, and Inner-SIoU loss function. Their model achieved 83.2% mAP@50 on the yellow sticky trap dataset with a computational complexity of 8.8 GFLOPs. Sun et al. (2025) proposed an improved storage pest detection algorithm based on YOLO11n, called PDA-YOLO, which primarily utilizes the PoolFormer_C3K2 module to reduce the model's computational complexity to 6.9 GFLOPs. However, the dataset contains only a single pest species, resulting in limited generalization ability of the model. Yu et al. (2025) proposed a lightweight model, DGS-YOLOv7-Tiny, based on YOLOv7-Tiny and applied it to tomato pest detection. By introducing a global attention mechanism and a fused convolution structure, the model enhances small object detection capability while effectively reducing the number of parameters and computational complexity, thereby enabling efficient and real-time detection in edge computing environments.

Existing pest detection studies have achieved certain advances in accuracy, but they still share common limitations. First, most methods are trained on datasets with relatively simple backgrounds and limited pest species, resulting in poor generalization to complex real-world environments. Second, some approaches focus on improving small-object detection accuracy, which often leads to large model sizes and slow inference, hindering deployment on edge devices. Third, although lightweight methods reduce computational complexity, they often compromise detection accuracy across multiple pest species, making it difficult to achieve a balance between accuracy and model efficiency. To address this, this paper proposes a lightweight agricultural pest detection model named YOLO-DCPG based on YOLOv8n (Varghese and Sambath, 2024). The main idea is to optimize the model structure to reduce network parameters, computational complexity, and model size, while maintaining high detection accuracy. This enables better adaptation to edge device deployment. The main contributions of this work are as follows:

1. This paper proposes a DCPGAttention mechanism that uses mean and standard deviation pooling on input features to improve global information capture.
2. To balance detection accuracy and model lightweight design, the feature extraction backbone of YOLOv8n is replaced with the lighter StarNet. At the same time, a more efficient multi-scale feature fusion structure called Small-Neck is designed. This structure combines an improved a-BiFPN with the lightweight convolution module GSConv, which reduces computational cost while enhancing feature representation.
3. To accelerate model convergence, a scale factor based on Inner-IoU is introduced into the WIoU loss function to control auxiliary bounding boxes. This helps the model focus more on small pest regions and speeds up the regression of small targets, achieving a balance between detection accuracy and inference speed.
4. To further validate the practicality of the proposed lightweight model on edge devices, this paper deploys it on a Raspberry Pi 4B, achieving stable and real-time pest detection.

The structure of this paper is as follows: Section 2 mainly introduces the dataset acquisition and the YOLO-DCPG model architecture, with detailed descriptions of the improvements in each module. Section 3 presents the experimental design, result analysis, and edge device deployment. Section 4 provides the conclusion.

2 Materials and methods

2.1 Dataset introduction

The dataset used in this study is Pest24 (Wang et al., 2020a). It was collected by the Institute of Intelligent Machines, Chinese Academy of Sciences, using a self-developed pest image acquisition device. The dataset contains 25,378 images covering 24 distinct pest categories. Compared with general object detection datasets such as VOC and COCO, the targets in Pest24 are significantly smaller, and the images often contain complex scenes with overlapping and occluded objects. Some sample images are shown in Figure 1 Additionally, Pest24 suffers from severe class imbalance, with large variations in the number of instances per category. For example, the 20th pest category (Anomala corpulenta) has 53,347 instances, while the 18th category (Holotrichia oblita) contains only 108. These characteristics make Pest24 a highly challenging dataset for object detection and place higher demands on model robustness and generalization.

To address the class imbalance problem, this paper adopts a combination of offline and online data augmentation methods. For

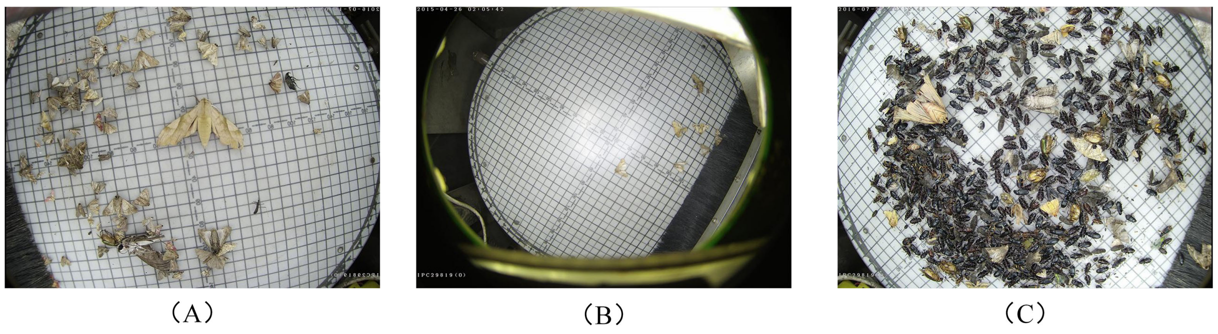

FIGURE 1
Sample images from the Pest24 dataset. **(A)** Images containing multiple pest types. **(B)** Samples captured at different scales. **(C)** Images showing dense pest distribution and occlusion.

offline augmentation, six techniques are applied: horizontal flipping, vertical flipping, random 90-degree rotation, brightness adjustment, horizontal translation, and vertical translation. These are used for pest categories with fewer instances, such as Land tiger, Eight-character tiger, Holotrichia oblita, and Nematode trench. For online augmentation, Mosaic (Bochkovskiy et al., 2020) and Mixup (Zhang et al., 2017) are employed during training to increase dataset diversity and improve model robustness. Mosaic augmentation combines four images into one, effectively enriching background textures and simulating multi-scale and occlusion conditions commonly observed in field pest detection scenarios. Mixup generates virtual samples by blending pairs of images and their corresponding labels, which helps the model learn smoother decision boundaries and reduces overfitting to specific pest categories. Incorporating spatial and illumination diversity through these augmentations improves the model's ability to generalize across complex field conditions. As a result, the total number of images after augmentation reaches 29,680. The dataset is divided into training, validation, and test sets in a 7:2:1 ratio, with 20,776 images for training, 5,936 for validation, and 2,968 for testing.

2.2 YOLO-DCPG object detection model

YOLOv8 (You Only Look Once version 8), proposed by Ultralytics (Varghese and Sambath, 2024), consists of three main components: a backbone for feature extraction, a neck for feature fusion, and a detection head. The neck adopts a Path Aggregation Network (PAN) (Liu et al., 2018), which enhances the propagation of deep semantic features through a bottom-up information flow mechanism, thereby improving multi-scale feature fusion.

To address the deployment challenges of agricultural pest detection on edge devices caused by the large number of parameters in YOLOv8, we propose a lightweight model named YOLO-DCPG, which is built upon YOLOv8n and employs StarNet as the feature extraction backbone. The overall architecture of YOLO-DCPG is illustrated in Figure 2. First, a Dual Channel Pooling Gated Attention mechanism (DCPGAttention) is designed. It replaces the Global Context Embedding module of the Gated Channel Transformation (GCT) (Yang et al., 2020) attention mechanism with dual pooling operations based on mean and standard deviation. This module is integrated at the end of the backbone network to enhance the model's ability to capture global pest information. Second, a lightweight Small-Neck network is designed to replace the neck of YOLOv8. It combines an enhanced bidirectional feature pyramid network (a-BiFPN) with the efficient convolution module GSConv (Li et al., 2022b), enabling effective fusion of features from different levels and better capturing multi-scale pest features. Additionally, in the loss function, a scale factor called ratio based on Inner-IoU (Zhang et al., 2023) is introduced into WIoU (Tong et al., 2023) to control the influence range of auxiliary bounding boxes. This helps the model focus more on small pest regions and accelerates the regression of small targets. The improved loss function enhances detection accuracy while effectively controlling computational complexity, achieving a balance between detection performance and inference efficiency. Through these three improvements, the model significantly reduces both parameter count and computational load.

2.2.1 StarNet feature extraction network

For pest detection tasks, high redundancy exists among feature maps, which affects model efficiency. Meanwhile, practical applications often rely on edge devices with limited computing resources. To reduce computational cost and parameter count, this paper adopts the lightweight and efficient StarNet-S100 (Ma et al., 2024) as the feature extraction backbone. It achieves a good balance between accuracy and computational overhead. The model structure is shown in the Backbone part of Figure 2. StarNet adopts a 4-stage hierarchical architecture, where downsampling is performed using convolutional layers. At each stage, a Star Block is introduced for feature extraction. The core component of the architecture is the Star Operation, an efficient feature transformation method based on element-wise multiplication, which enhances inter-channel modeling capability with extremely low computational cost. However, since the Star Operation inherently lacks spatial receptive fields, its ability to model local image structures such as edges and textures is limited. To address

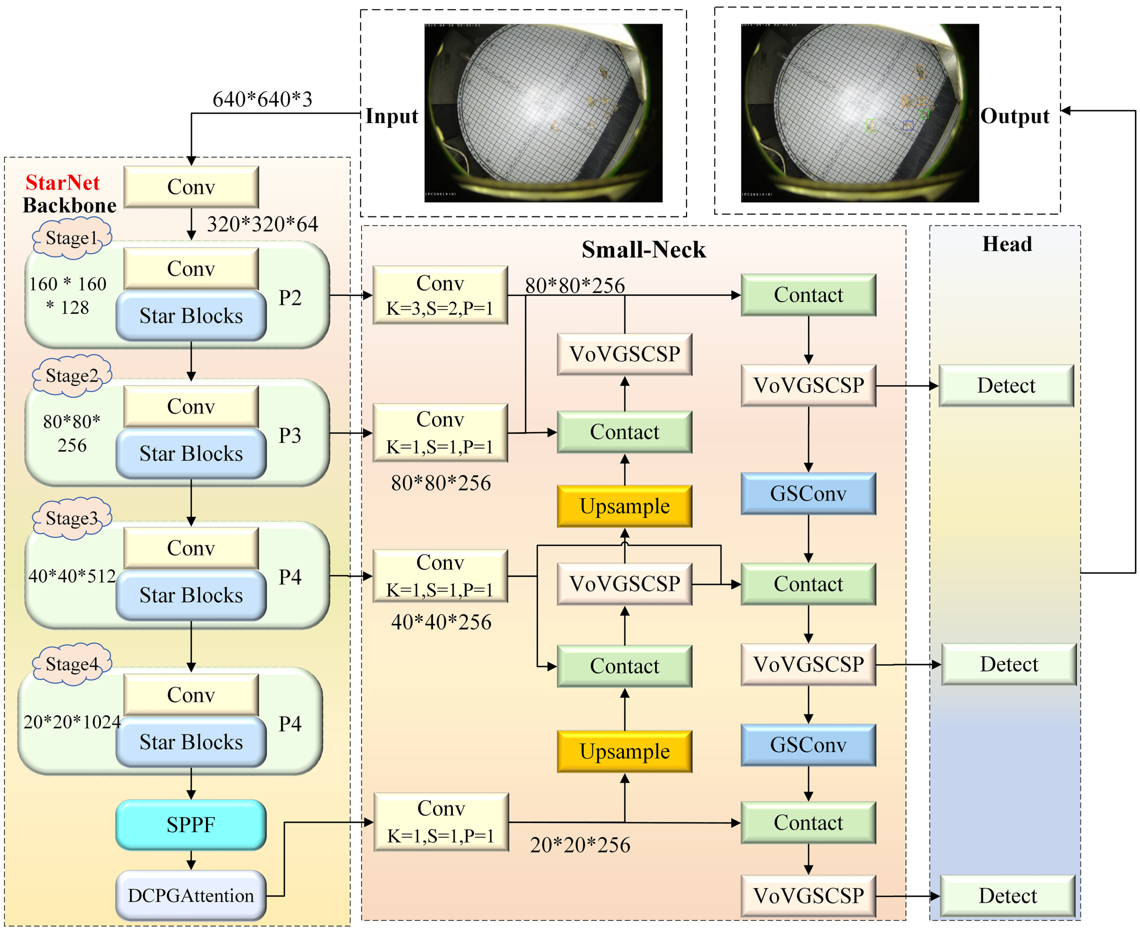

FIGURE 2
Architecture of the YOLO-DCPG model. The backbone employs StarNet-S100 for feature extraction, while the feature fusion network adopts a lightweight Small-Neck structure to enhance feature propagation and reduce parameters. The detection head retains the original YOLOv8 design for object classification and localization.

this issue, StarNet introduces Depthwise Convolution (DWConv) (Chollet, 2017) in each Star Block to enhance spatial modeling capability. In this design, the Star Operation is responsible for inter-channel interaction, while DWConv handles spatial feature extraction. The two components work together to achieve a balance between structural efficiency and representational capacity. In addition, to further improve detection efficiency and deployment performance, Batch Normalization is used after DWConv to replace the original Layer Normalization. This accelerates training convergence and simplifies computation. At the same time, a lightweight ReLU6 activation function is adopted to enhance non-linear modeling capability and reduce numerical instability. The structure of the Star Block is shown in Figure 3.

2.2.2 DCPGAttention module

The structure of the Gated Channel Transformation (GCT) (Yang et al., 2020) module is shown in Figure 4A. It consists of three parts: Global Context Embedding, Channel Normalization, and Gating Adaptation. Although the GCT module effectively introduces a learnable gating mechanism to adaptively adjust channel responses, its Global Context Embedding stage relies on L2-norm–based feature encoding, which enforces fixed mean values across channels. This design limits the ability to distinguish fine-grained feature variations and leads to the loss of informative distributional characteristics.

The Style-based Recalibration Module (SRM, Figure 4B) (Lee et al., 2019) aims to improve channel-wise feature representation by leveraging global statistical information. SRM extracts both the mean and standard deviation of each feature channel as style descriptors, capturing richer distributional characteristics compared to single-statistic methods. These dual statistics provide a more comprehensive global context representation and enable the network to recalibrate channel responses based on overall feature style information. However, the recalibration mechanism of SRM is fixed and input-independent, meaning that the weighting parameters remain constant once trained. As a result, SRM cannot flexibly respond to variations across different samples, which limits the model's dynamic adaptability, generalization, and robustness in complex visual scenes.

To address these limitations, we propose the Dual Channel Pooling Gated Attention (DCPGAttention) module, as illustrated in

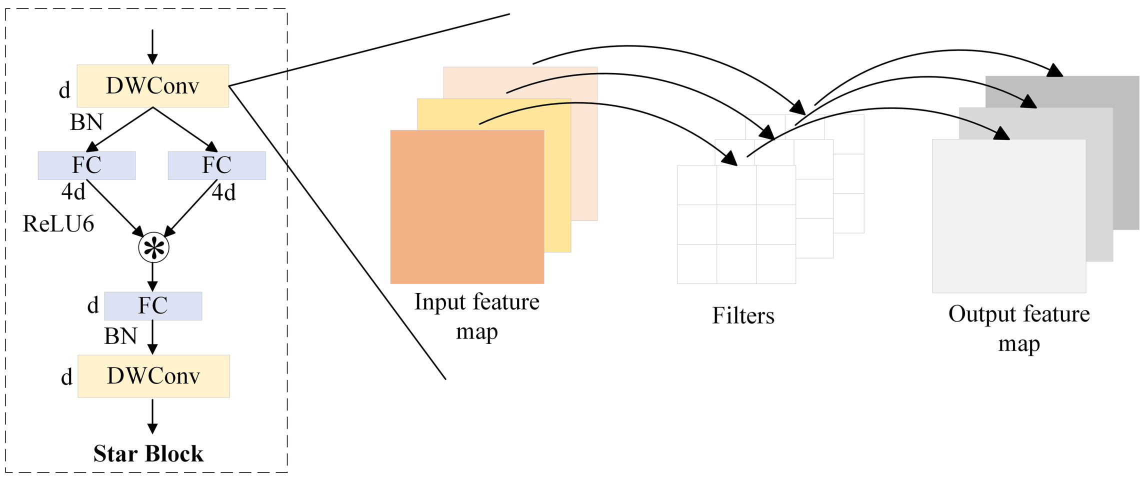

FIGURE 3
Structure of the Star Block. *denotes the star operation.

Figure 4C. DCPGAttention integrates the dual-statistics pooling strategy of SRM with the adaptive gating concept of GCT to achieve both comprehensive and dynamically adjustable global feature modeling. Specifically, it applies mean and standard deviation pooling to extract richer contextual information, while a learnable weighting factor α adaptively balances the contribution of different channels. Compared with the L2-norm–based embedding in GCT, this design provides a more flexible and informative representation of global context. Given an input feature map $x \in R_{n\times c\times h\times w}$, the feature computation is defined as follows (Equations 1–4):

$$\mu_{nc} = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_c \tag{1}$$

$$\sigma_{nc} = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_c - \mu_{nc})^2} \tag{2}$$

$$t_n c = [\mu_{nc}, \sigma_{nc}] \tag{3}$$

$$s_c = \alpha_c(t_{nc} + \xi) \tag{4}$$

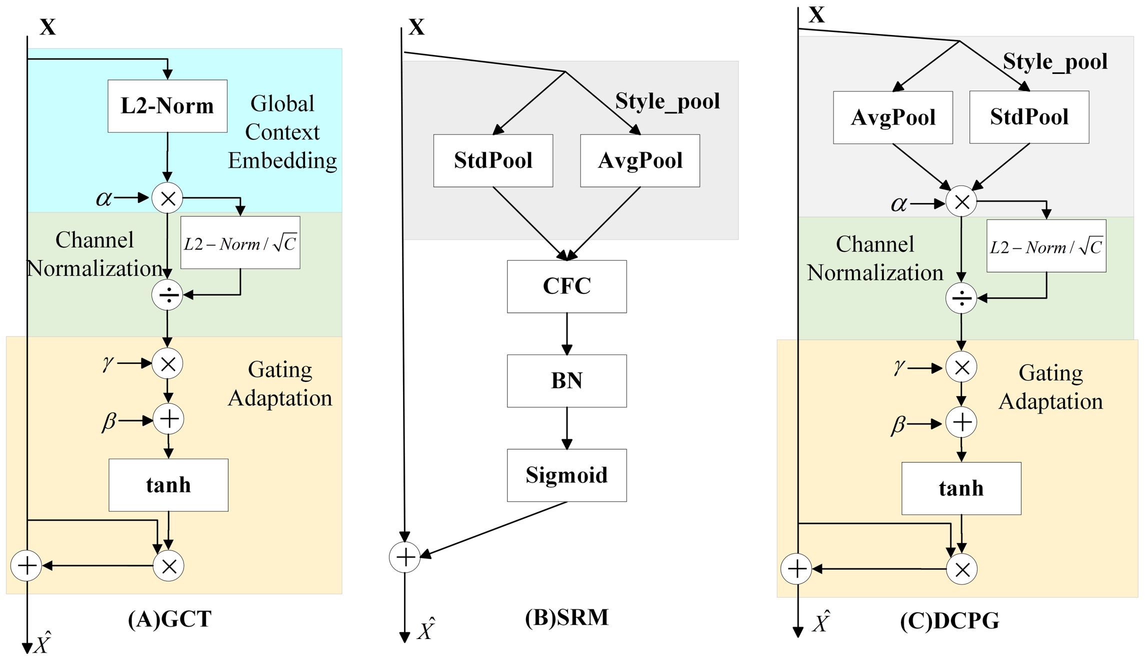

FIGURE 4
Diagram of the DCPG attention mechanism module.

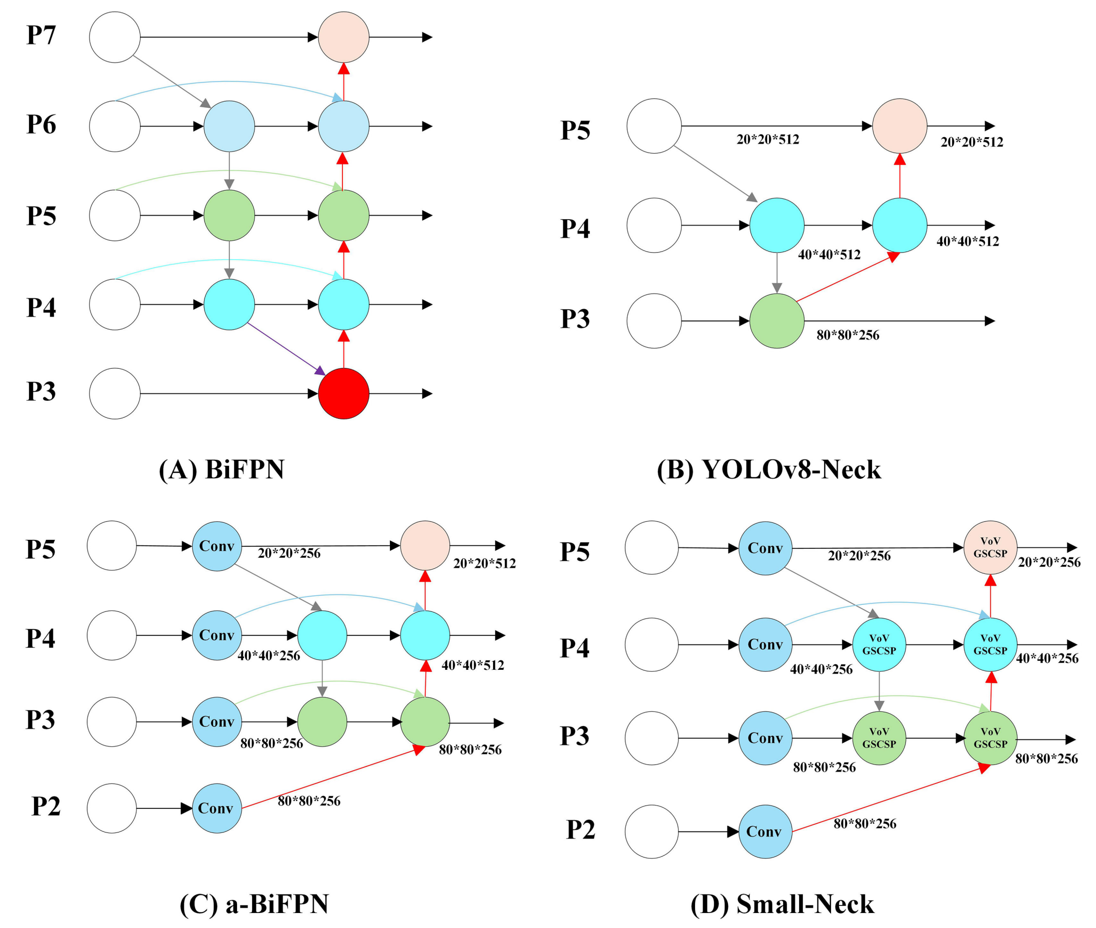

FIGURE 5
Small-neck feature fusion network architecture. **(A)** BiFPN: achieves efficient bidirectional feature fusion for multi-scale information flow. **(B)** Original YOLOv8 feature fusion network: adopts a PANet-style structure to enable both top-down and bottom-up feature aggregation. **(C)** a-BiFPN: introduces an additional P2 shallow feature layer, which preserves finer spatial details crucial for small object detection and enhances small-target perception. **(D)** Small-Neck: builds upon the a-BiFPN design, incorporating the more efficient GSConv convolution module and VoVGSCSP fusion block. Furthermore, channel pruning is applied based on the characteristics of the Pest24 dataset, achieving a balance between lightweight and efficient network design.

μ_{nc} denotes the mean of the channel features, and σ_{nc} represents the standard deviation. x_c is the input feature, where H and W are the spatial dimensions, n is the batch size, c is the number of channels, and t refers to the operation that integrates the dual pooled channel features. $\alpha = \alpha[\alpha_1, \cdots, \alpha_c]$ denotes the learned embedding weight, and ξ is a small constant added for numerical stability, preventing issues during backpropagation near zero.

In the Channel Normalization part, L2-norm is retained. This method normalizes the scale across channels while preserving the relative magnitude between channels. It introduces an attention mechanism that maintains higher activation values for strongly responding channels after normalization, thereby suppressing weakly responding channels and guiding the model to focus more on channels with prominent features. The normalization formula is defined as follows:

$$\widehat{s_c} = \frac{\sqrt{C}s_c}{[(\sum_{c=1}^{C} s_c^2)+\xi]^{\frac{1}{2}}} \tag{5}$$

In Equation 5, ξ is a small constant added for numerical stability to prevent division by zero. The scalar $\sqrt{C}$ is introduced to keep values within a reasonable range and avoid extreme magnitudes.

In the Gating Adaptation stage, the gating mechanism from GCT is retained. This mechanism introduces trainable parameters: a weight γ and a bias β, which control whether channel features are activated. A positive γ emphasizes inter-channel competition, while a negative γ promotes inter-channel cooperation. This dynamic mechanism allows the model to adaptively enhance or suppress channel features across different semantic levels, thereby improving its representational capacity. Gate control weight $\gamma = [\gamma_1, \cdots, \gamma_c]$, Gate control bias $\beta = [\beta_1, \cdots, \beta_c]$, The gate control function is formulated as Equation 6:

$$\widehat{x_c} = x_c[1 + \tanh(\gamma_c \hat{s}_c + \beta_c)] \tag{6}$$

In this equation, x_c represents the original input channel feature. The final channel weight is represented by $1 + \tanh(\gamma_c \hat{s}_c + \beta_c)$.

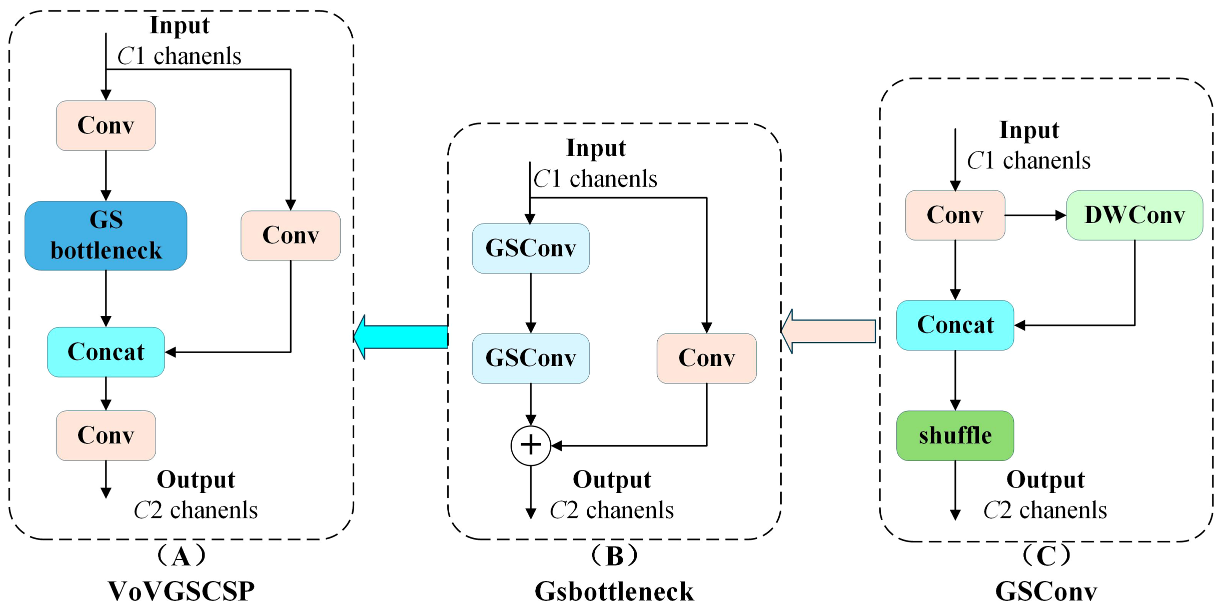

FIGURE 6
GSConv and VoVGSCSP module structure.

2.2.3 Small-neck: a lightweight feature fusion network

In YOLOv8, the feature fusion network adopts a PAN-FPN ((Liu et al., 2018) (Lin et al., 2017) structure similar to that of YOLOv5 (Jocher et al., 2020), as shown in Figure 5B. This design aims to enhance the model's ability to detect objects at different scales through top-down and bottom-up feature fusion. However, this feature fusion network faces several challenges in small pest detection tasks. Multiple downsampling operations in the YOLOv8 neck cause the loss of shallow features that are essential for identifying small pests. Moreover, the fusion process focuses mainly on high-level semantic features, resulting in insufficient utilization of shallow spatial details.

To improve the representation of small-scale features, an enhanced feature fusion module named a-BiFPN (Figure 5C) is designed based on the YOLOv8 neck. Unlike the original PAN-FPN structure, a-BiFPN introduces an additional P2 shallow feature layer, which contains finer spatial information critical for small-object detection. The fusion of P2–P5 layers is realized through bidirectional top-down and bottom-up pathways similar to BiFPN (Figure 5A) (Tan et al., 2020), enabling effective multi-level feature integration. Experimental results on the Pest24 dataset demonstrate that this modification significantly improves the detection accuracy of small pests.

Although a-BiFPN enhances small-object detection performance, its fusion of deep-level features (P4 and P5) introduces redundant computation for the Pest24 dataset, which mainly consists of small pest targets. Considering that the contribution of these deep-level features is relatively limited, the proposed Small-Neck Figure 5D) simplifies the a-BiFPN by applying a dataset-specific channel pruning strategy. Specifically, the output channels of the P3–P5 layers are uniformly reduced to 256 to minimize unnecessary computation while retaining essential shallow and mid-level features for accurate small-object detection. In addition, standard convolutions are replaced by the more efficient GSConv module (Figure 6C) to accelerate inference and enhance real-time performance, while the original C2f module is substituted with VoVGSCSP (Figure 6A) (Li et al., 2022b) to improve feature reuse and gradient propagation. As a result, Small-Neck achieves a favorable balance among detection accuracy, computational complexity, and inference speed.

Considering that input features generated from different layers vary in importance and require an efficient and stable feature fusion strategy, this paper adopts the Fast Normalized Fusion method proposed by Tan et al. (2020). This method optimizes the weight normalization approach to ensure model training stability while accelerating convergence, thereby improving the effectiveness of feature fusion. Taking the fusion of two features at the P4 layer in Small-Neck as an example:

$$P_4^{td} = Conv\left(\frac{w_1 \cdot P_4 in + w_2 \cdot \mathrm{Re}\,size(P_5 in)}{w_1 + w_2 + \xi}\right) \tag{7}$$

$$P_4^{out} = Conv\left(\frac{w_1{\prime} \cdot \mathrm{P}_4^{\mathrm{in}} + \mathrm{w}_2{\prime} \cdot P_4^{td} + w_3{\prime} \cdot \mathrm{Resize}(P_4^{out})}{w_1{\prime} + w_2{\prime} + w_3{\prime} + \xi}\right) \tag{8}$$

In Equations 7, 8, P4td represents the intermediate feature at the P4 layer in the top-down path; P4out represents the output feature at the P4 layer in the bottom-up path. The Resize operation refers to either downsampling or upsampling. The parameter w is learned during training and is used to measure and differentiate the importance of different feature maps in the fusion process for the final output.

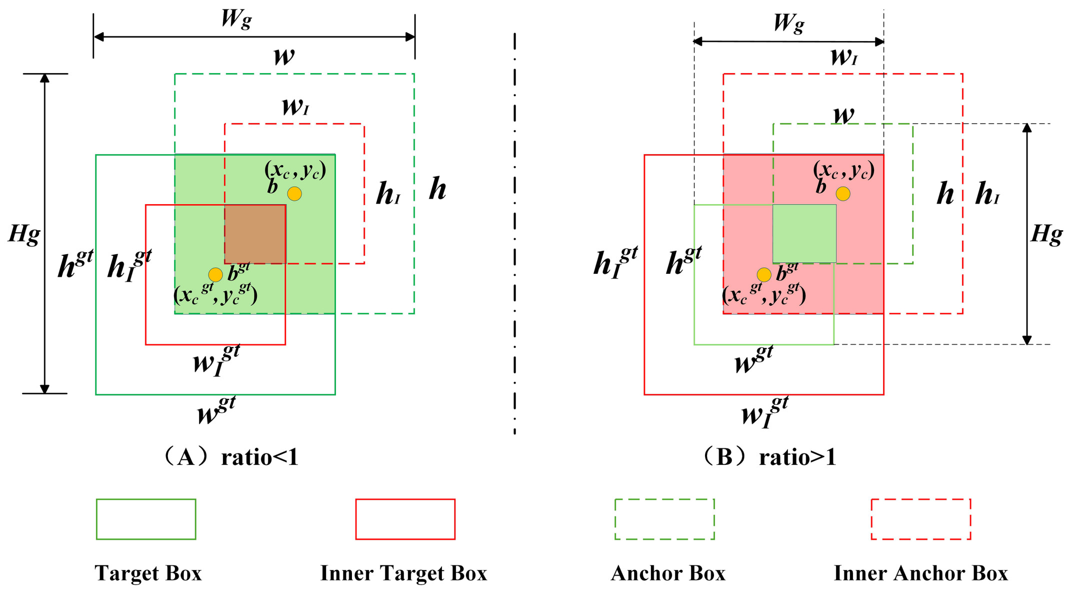

FIGURE 7
Inner-IoU mechanism diagram.

2.2.4 Inner-WIoU loss function

The YOLOv8 model adopts the CIoU (Zheng et al., 2020) loss function for bounding box regression, aiming to optimize the regression by jointly considering the distance between box centers, aspect ratio, and overlap area. However, CIoU lacks adaptive adjustment capability across different detection tasks, resulting in limited generalization. To accelerate bounding box regression for small pests and enhance the model's performance in small pest detection. This paper introduces a scale factor from Inner-IoU (Zhang et al., 2023) into the WIoU (Tong et al., 2023) loss function to control the auxiliary boxes, enabling the model to focus more on small pest regions. Therefore, the Inner-WIoU loss function is proposed in this paper, and its mechanism is illustrated in Figure 7.

In bounding box regression tasks, IoU (Yu et al., 2016) loss accurately describes the matching degree between the predicted box and the ground truth (GT) box, ensuring that the model effectively learns object localization information. As shown in Equation 9, B and Bgt represent the predicted box and the ground truth box, respectively.

$$IoU = \frac{|B \cap \quad B^{gt}|}{|B \cup \quad B^{gt}|} \tag{9}$$

The WIoUv3 loss function introduces a focusing mechanism by constructing a method for computing gradient gain, and it proposes a reasonable strategy for gradient gain allocation. This strategy reduces the competitiveness of high-quality anchor boxes and also mitigates the harmful gradients produced by low-quality anchor boxes. Therefore, WIoUv3 can effectively focus on hard samples anchor boxes, helping the model pay more attention to these challenging samples during training, thereby improving its generalization ability. Specifically, WIoUv3 assigns differentiated gradient gains to different anchor boxes during gradient computation. This ensures that the network focuses more on hard samples while reducing the excessive influence of easy samples on model training. High-quality anchor boxes typically have high IoU with ground truth (GT) boxes and are easier to predict accurately. Therefore, the WIoUv3 loss reduces their influence to prevent them from dominating the training process. For low-quality anchor boxes, WIoUv3 decreases their gradient contributions to avoid negative effects on model training. Through this gradient gain allocation strategy, WIoUv3 significantly improves the model's performance in detecting small pests and handling dense regions, thereby enhancing its generalization ability.

$$L_{IoU} = 1 - IoU \tag{10}$$

$$R_{WIoU} = exp\left(\frac{(x_c - x_c^{gt})^2 + (y_c - y_c^{gt})^2}{(W_g + H_g)^2}\right) \tag{11}$$

$$L_{WIoUv1} = R_{WIoU} L_{IOU} \tag{12}$$

$$L_{WIoUv3} = rL_{WIoUv1}, r = \frac{\beta}{\delta\alpha^{\beta-\delta}}, \beta = \frac{\overset{*}{L_{IoU}}}{\overline{L_{IoU}}} \in [0, +\infty) \tag{13}$$

In Equations 10–13, R_{WIoU} denotes the exponential transformation of the normalized length between the center points. Wg and Hg represent the width and height of the union region between the predicted box and the ground-truth box. $\overset{*}{L}_{IoU}$ is the monotonic focusing factor, $\overline{L_{IoU}}$ is the momentum-based moving average, and β is the non-monotonic focusing factor.

In this study, inspired by Inner-IoU, a scaling factor named ratio is introduced into the WIoUv3 loss to control the auxiliary bounding box. This allows the model to focus more on small pest regions. Based on the original WIoUv3 loss, the proposed Inner-WIoU further enhances attention to small pests, ensuring that the auxiliary box contributes effectively to regression accuracy while

TABLE 1 Hardware and software environment.

Configuration Item	Training device	Edge computing device
CPU	Intel(R) Xeon(R) Gold 5318S CPU @ 2.10GHz	Raspberry Pi 4B
GPU	NVIDIA A100(80GB)	–
CUDA	11.3	–
Deep learning frame	Pytorch 1.11.0	Pytorch 1.11.0
Programming Language	Python 3.9.7	Python 3.9.7
Operating System	Ubuntu 20.04.5 LTS	Ubuntu server 24.04

reducing unnecessary computational cost. The detailed computation process is illustrated in Figure 7, and the mathematical formulation of the loss function and auxiliary box regression is provided in Equations 14–20.

$$b_l^{gt} = x_c^{gt} - \frac{w^{gt} \cdot ratio}{2}, b_r^{gt} = x_c^{gt} + \frac{w^{gt} \cdot ratio}{2} \tag{14}$$

$$b_t^{gt} = y_c^{gt} - \frac{h^{gt} \cdot ratio}{2}, b_b^{gt} = y_c^{gt} + \frac{h^{gt} \cdot ratio}{2} \tag{15}$$

$$b_l = x_c - \frac{w \cdot ratio}{2}, b_r = x_c + \frac{w \cdot ratio}{2} \tag{16}$$

$$b_t = y_c - \frac{h \cdot ratio}{2}, b_b = y_c + \frac{h \cdot ratio}{2} \tag{17}$$

$$\text{inter} = (\min(b_r^{gt}, b_r) - \max(b_l^{gt}, b_l)) \cdot (\min(b_b^{gt}, b_b) - \max(b_t^{gt}, b_t)) \tag{18}$$

$$union = (w^{gt} \cdot h^{gt}) \cdot (ratio)^2 + (w \cdot h) \cdot (ratio)^2 - \text{inter} \tag{19}$$

$$L_{Inner-WIoU} = L_{WIoUv3} + IoU - \frac{\text{inter}}{union} \tag{20}$$

In Equations 14–20, (x_c^{gt}, y_c^{gt}) are the center coordinates of the Target Box and the Inner Target Box. (x_c, y_c) are the center coordinates of the Anchor Box and the Inner Anchor Box. hgt and wgt are the height and width of the target box. h and w are the height and width of the anchor box. *ratio* $\in [0.5, 1.5]$ is the scaling factor; $L_{Inner-WIoU}$ is the final Inner-WIoU loss function.

Based on the study by Zhang et al., the default value of the ratio is set to 0.7 (Zhang et al., 2023). When the ratio is less than 1 (as shown in Figure 7A), the scale of the auxiliary bounding box is smaller than that of the actual bounding box, resulting in a regression effective range smaller than that of the IoU loss. At this time, the absolute gradient of the auxiliary bounding box is greater than that of the IoU loss, accelerating the convergence of high-IoU samples. When the ratio is greater than 1 (as shown in Figure 7B), the larger-scale auxiliary bounding box expands the effective range of regression, providing gains for low-IoU regressions. Since the dataset used in this study contains a large number of small pests, a more detailed investigation of the ratio value was conducted. Specifically, experiments were performed within the range of [1, 1.2], with an interval of 0.05. The results show that when the ratio is set to 1.05, the model achieves the best performance in terms of mean Average Precision (mAP), as well as the highest detection accuracy for small pests. Further analysis reveals that, at this value, the regression range of the auxiliary bounding box is well-balanced. It enables better capture of small-object features while avoiding excessive focus on large targets, thereby improving detection performance for small pests.

3 Results

3.1 Experimental setup

The experiments were conducted using the PyTorch 1.11.0 framework. The detailed environment configurations are listed in Table 1. For training settings, the batch size was set to 32, with a total of 200 epochs. The initial learning rate was 0.01, weight decay was set to 0.0005, and the momentum parameter was set to 0.937. All models were trained from scratch without using any pretrained weights. Stochastic Gradient Descent (SGD) was adopted as the optimizer to improve the model's ability to escape local optima. To enhance the training performance of the model, a combination of the Cosine Annealing learning rate scheduler and the Early Stopping mechanism was employed. The Cosine Annealing strategy dynamically adjusts the learning rate during training, gradually reducing it in later stages to facilitate stable convergence near the optimal solution and avoid local minima. The Early Stopping mechanism monitors the validation loss, and if no significant decrease is observed over 50 consecutive epochs, training is automatically terminated to prevent overfitting and reduce training time. The synergy between these two strategies effectively optimizes the training process and improves the model's generalization performance on the validation set.

3.2 Evaluation metrics

In pest detection tasks, it is essential to balance detection accuracy and model complexity. To comprehensively evaluate the detection performance of the proposed model, this study adopts Precision, Recall, and mean Average Precision (mAP) as the primary evaluation metrics. Meanwhile, to measure model complexity, we report the number of parameters and Giga Floating Point Operations per Second (GFLOPs). In addition, Frames Per Second (FPS) is used to assess the real-time inference performance of the model. The calculation methods of these metrics are provided in Equations 21–26:

$$\Pr ecision = \frac{TP}{TP+FP} \times 100\% \tag{21}$$

$$\operatorname{Re} call = \frac{TP}{TP+FN} \times 100\% \tag{22}$$

In Equations 21, 22: TP(True Positive): the number of pest samples correctly detected by the model. FN(False Negative): the number of actual pest samples that were not detected by the model.

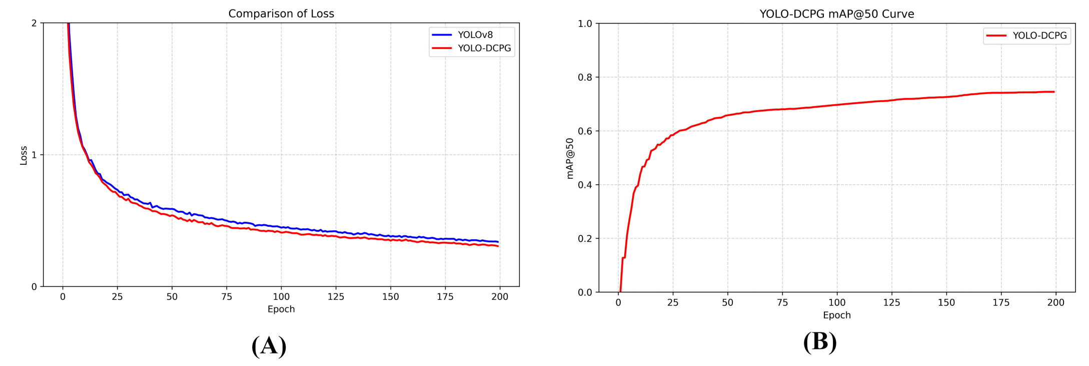

FIGURE 8
The training loss curve variation **(A)** and mAP@50 training variation **(B)**.

TN(True Negative): TN(True Negative): the number of non-pest samples correctly identified as non-pests by the model. FP(False Positive): the number of non-pest samples incorrectly detected as pests by the model.

$$mAP = \frac{1}{n}\sum_{1}^{n} P_i = \frac{1}{n}(P_1 + P_2 + \cdots + P_n) \tag{23}$$

$$mAP@50^{\sim}95 = \frac{1}{c}\sum_{k=1}^{c} mAP@50_k \tag{24}$$

$$mAP@50^{\sim}95 = \frac{1}{10}(mAP@50 + mAP@55 + \cdots mAP95) \tag{25}$$

$$FPS = \frac{1}{T} \tag{26}$$

In Equations 23–26: mAP@0.5 represents the mean Average Precision when the Intersection over Union(IoU) threshold is set to 0.5. mAP@0.5~0.95 refers to the average of mAP values calculated at multiple IoU thresholds ranging from 0.5 to 0.95 with a step size of 0.05. Compared to mAP@0.5, this metric is more stringent and provides a more comprehensive evaluation of the model's detection performance. T denotes the processing time required for a single input image. FPS measures the number of images the model can process per second during inference, reflecting its real-time performance.

3.3 Result and analysis

3.3.1 Training and validation of the YOLO-DCPG algorithm

Figure 8A illustrates the comparison of training loss curves between the baseline YOLOv8 model and the proposed YOLO-DCPG model. At the early stages of training, both models exhibit a rapid decline in training loss, indicating efficient feature learning. YOLO-DCPG demonstrates a noticeably faster convergence rate compared to the baseline, reflecting the overall effectiveness of the proposed improvements. As training progresses, the loss curve of YOLO-DCPG stabilizes earlier and maintains lower fluctuations, suggesting better training stability and stronger generalization capability, which confirms the effectiveness of the overall network optimization strategy.

In Figure 8B, the variation of mAP@50 across epochs illustrates the model's performance evolution during training. The mAP@50 rises sharply in the early phase, indicating a rapid improvement in detection accuracy. As training proceeds, the curve gradually levels off, and by around the 160th epoch, the model's performance approaches its optimal state. Combined with Figure 7A, it can be observed that the proposed YOLO-DCPG not only converges faster but also exhibits better generalization capability, verifying the overall effectiveness of the improvements.

3.3.2 Comparison with state-of-the-art methods

To evaluate the effectiveness of the YOLO-DCPG model proposed in this paper for pest detection tasks, its performance is compared with several mainstream object detection models, including YOLOv3-tiny (Adarsh et al., 2020), YOLOv5n (Jocher et al., 2020), YOLOv6n (Li et al., 2022a), YOLOv8n (Varghese and Sambath, 2024), YOLOv10n (Wang et al., 2024), YOLOv11n (Khanam and Hussain, 2024), YOLOv12n (Tian et al., 2025) and NanoDet-Plus-m-1.5x (Lyu, 2020) as well as Transformer-based (Lv et al., 2024) models such as RT-DETR-n (Zhao et al., 2024) and RT-DETR-resnet18 (Qin et al., 2024). The comparison results are shown in Table 2.

In comparison with lightweight YOLO series object detection models, the YOLO-DCPG model proposed in this paper demonstrates strong overall performance. With a model size of 3.2 MB and a computational cost of 5.7 GFLOPs, It is the most resource-efficient models among all YOLO variants. Although YOLOv3-tiny achieves an mAP@50 that is 0.5% higher than YOLO-DCPG, YOLO-DCPG delivers comparable detection performance with only one-fifth of the resource consumption. In addition, YOLO-DCPG achieves a detection precision of 80.1%, which is higher than YOLOv3-tiny's 76.9%. Although YOLOv5 and YOLOv12 achieve higher inference speeds, YOLO-DCPG demonstrates superior detection performance, with improvements of 1.5% and 2.5% in mAP@50, and 1.8% and 2.4% in mAP@50–95, respectively. YOLOv6n reaches 68.3% for mAP@50 and 42.5% for

TABLE 2 The performance comparison of different object detection models.

Model	Precision (%)	mAP@50 (%)	mAP@50~95 (%)	GFLOPs (G)	Parameters (M)	FPS (Image/s)	Model Size (M)
YOLOv3-tiny	76.9	74.3	48.0	14.5	9.54	121.9	18.3
YOLOv5n	78.7	72.5	45.7	5.7	2.01	92.6	4.1
YOLOv6n	75.3	68.3	42.5	11.6	4.16	101.0	8.2
YOLOv10n	74.9	73.0	46.7	8.4	2.72	90.9	5.5
YOLOv11n	73.0	72.8	46.6	6.5	2.59	84.0	5.2
YOLOv12n	79.5	71.5	45.1	6.5	2.57	102.0	5.3
NanoDet-Plus-m-1.5x	70.3	67.3	40.2	2.97	2.44	117.6	4.7
RT-DETR-n	81.2	74.5	46.1	37.9	15.57	34.1	30.0
RT-DETR-resnet18	77.8	71.5	40.4	58.6	20.2	48.3	42.0
YOLOv8n	75.6	73.2	46.6	8.2	3.02	105.3	6.0
YOLO-DCPG(our)	80.1	74.0	47.5	5.7	1.47	98.0	3.2

mAP@50~95, both clearly lower than YOLO-DCPG. Its computational cost is 11.6 GFLOPs, almost twice that of YOLO-DCPG, and it has 4.16M parameters, which means higher resource usage. YOLOv10n and YOLOv11n achieve mAP@50~95 scores of 46.7% and 46.6%, respectively, which are slightly lower than that of YOLO-DCPG. Their inference speeds are also slower. In addition, both models have larger sizes and higher computational complexity compared to YOLO-DCPG. Therefore, YOLO-DCPG achieves a better balance among accuracy, computational cost, and inference efficiency.

Compared with NanoDet-Plus-m-1.5x, the proposed YOLO-DCPG achieves a 6.7% improvement in mAP@50 and a 7.3% improvement in mAP@50~95. Meanwhile, its inference speed reaches 98 FPS, slightly lower than 117.6 FPS of NanoDet-Plus-m-1.5x.These results demonstrate that YOLO-DCPG achieves a better balance between detection performance and inference speed, offering superior overall efficiency. Compared with the Transformer-based RT-DETR models, YOLO-DCPG also shows clear advantages in lightweight design. RT-DETR-n has 15.57M parameters, 37.9 GFLOPs of computation, and a model size of 30 MB. In contrast, YOLO-DCPG has only 1.47M parameters, 5.7 GFLOPs, and a model size of 3.2 MB. These are about one-tenth of RT-DETR-n. In terms of accuracy, YOLO-DCPG is slightly lower than RT-DETR-n in mAP@50. However, it performs better in mAP@50~95. RT-DETR-resnet18 has even higher computational complexity. But its accuracy is not better. Its mAP@50~95 is only 40.4%. YOLO-DCPG reaches an inference speed of 98 FPS. RT-DETR-n and RT-DETR-resnet18 run at only 34.1 FPS and 48.3 FPS. This shows that YOLO-DCPG is more efficient for edge device deployment.

Compared with the baseline model YOLOv8n, YOLO-DCPG shows better performance with only a slight drop in inference speed. Precision, mAP@50, and mAP@50~95 increase by 4.6%, 0.8%, and 0.9%, respectively. At the same time, the model size, GFLOPs, and number of parameters are reduced by 46.7%, 30.1%, and 51.2%, respectively. This improvement mainly comes from the DCPGAttention module. It helps the model focus more on pest target regions and improves feature discrimination in complex backgrounds. This increases the ability to distinguish between different classes and improves detection accuracy. The model also uses the lightweight StarNet-S100 as the backbone for feature extraction, and a Small-Neck structure for feature fusion. These components help make better use of shallow features, which improves the detection of small pests. Experimental results show that YOLO-DCPG reduces model complexity while maintaining high detection accuracy. It achieves a good balance between precision and lightweight design.

3.3.3 Ablation study

To evaluate the performance of YOLO-DCPG in pest detection tasks, ablation experiments were conducted on the Pest24 dataset. The tested components include StarNet-S100, Small-Neck, DCPGAttention, and Inner-WIoU. The results are shown in Table 3. Compared with the baseline model YOLOv8n, the improved YOLO-DCPG model reduces the number of parameters by 51.2% and lowers the computational cost (GFLOPs) by 46.7%. At the same time, detection performance improves. Precision, mAP@50, and mAP@50~95 increase by 4.5%, 0.8%, and 0.9%, respectively. These results show that the improvements in model structure and loss function help reduce model complexity while maintaining detection performance.

For the feature extraction network, this paper replaces the original YOLOv8n backbone with the StarNet-S100 network. Compared with the original YOLOv8n backbone, StarNet-S100 shows a slight drop in mAP but has a clear advantage in parameter size. In terms of attention mechanism, this paper adds the DCPGAttention module to the bottom of the YOLOv8n backbone. This module combines the control gate from GCT with

TABLE 3 Ablation experiment results.

Model	Precision (%)	Recall (%)	mAP@50 (%)	mAP@50~95 (%)	GFLOPs (G)	Parameters (M)
YOLOv8n	75.6	67.4	73.2	46.6	8.2	3.02
YOLOv8n+DCPGAttention	78.5	69.9	75.0	48.7	8.2	3.02
YOLOv8n+StarNet	80.0	67.4	72.9	46.2	6.9	2.39
YOLOv8n+Small-Neck	79.9	69.4	74.3	47.6	6.6	1.88
YOLOv8n+StarNet+Small-Neck	77.5	68.9	73.5	46.9	5.7	1.47
YOLOv8n+StarNet+Small-Neck+ DCPGAttention	78.1	69.6	73.7	47.5	5.7	1.47
YOLOv8n+StarNet+Small-Neck+ DCPGAttention+Inner-WIoU (YOLO-DCPG)	80.1	69.2	74.0	47.5	5.7	1.47

dual pooling on the mean and standard deviation of the input features. It helps the model capture global information more effectively. After adding this module, the model reaches a Precision of 78.5%, mAP@50 of 75.0%, and mAP@50~95 of 48.7%. These are improvements of 2.9%, 1.8%, and 2.1% over the original YOLOv8n. For feature fusion, this paper simplifies the Neck part of YOLOv8n by replacing it with a lightweight Small-Neck. Compared with YOLOv8n, Small-Neck reduces the number of parameters by 37.7% and GFLOPs by 19.5%. At the same time, it improves performance in Precision, Recall, mAP@50, and mAP@50~95. For the loss function, this paper introduces a scale factor into the WIoU loss. This factor adjusts the size of the auxiliary box. It helps the model focus more on small pest regions and improves the detection of small objects.

To evaluate the combined effect of each module, this paper designs an ablation study with step-by-step integration of the four proposed components. The results are shown in Table 3. The combination of StarNet-S100 and Small-Neck reduces parameters by 51.2% and GFLOPs by 30.1%, while slightly improving accuracy. After adding the DCPGAttention module, mAP@50 increases by 0.2% and mAP@50~95 by 0.6%. Finally, by integrating all modules, the complete YOLO-DCPG model is formed. Without significantly increasing computation, it further improves Precision by 2.0% and mAP@50 by 0.3% compared to the previous version. These results confirm the effectiveness of the combined improvements and their synergy.

3.3.4 Comparison of different lightweight backbone networks

For backbone selection, this paper conducts comparative experiments on StarNet-S50 (Ma et al., 2024), StarNet-S100 (Ma et al., 2024), StarNet-S150 (Ma et al., 2024), MobileNetv4-small (Qin et al., 2024), and FasterNet-t0 (Chen et al., 2023). The goal is to evaluate the performance of different lightweight networks on the pest detection task. The focus is on their impact on model parameters, GFLOPs, FPS, and detection accuracy. The results are shown in Table 4. In terms of detection accuracy, FasterNet-t0 achieved the best performance, with a Precision of 80.6% and an mAP@50 of 73.1%, indicating strong detection capability. However, it has significantly higher GFLOPs and parameter count compared to other networks, and its inference speed is slower. These factors limit its suitability for deployment on edge devices. MobileNetv4-small shows lower detection accuracy and higher computational complexity in the pest detection task, making it less suitable for efficient deployment scenarios. StarNet-S50 has the smallest number of parameters and the fastest inference speed, but its accuracy drops noticeably. StarNet-S150 achieves similar accuracy to StarNet-S100 but requires more computation. In contrast, StarNet-S100 achieves a good balance between performance and efficiency. It maintains low computation (6.9 GFLOPs) and a low parameters(2.39M), while reaching high detection accuracy. Its inference speed reaches 106.4 FPS, making it a well-balanced choice for both accuracy and speed.

TABLE 4 Comparison of different lightweight backbones.

Model	Precision (%)	Recall (%)	mAP@50 (%)	mAP@50~95 (%)	GFLOPs (G)	Parameters (M)	FPS (Image/s)
YOLOv8n	76.9	66.3	71.3	45.0	8.2	3.02	105.3
+StarNet-S50	78.0	66.4	71.0	44.3	5.3	1.89	111.1
+StarNet-S100	80.0	67.4	72.9	46.2	6.9	2.39	106.4
+StarNet-S150	79.0	68.4	72.9	46.6	7.6	2.90	104.2
+MobileNetv4-small	76.0	64.3	68.9	42.9	21.4	5.38	90.1
+FasterNet-t0	80.6	68.3	73.1	46.3	10.7	4.18	92.6

3.3.5 Comparison of attention modules

To verify the effectiveness of the proposed DCPGAttention module in pest detection tasks, this paper uses YOLOv8n as the baseline model. Five attention mechanisms are tested: GCT (Yang et al., 2020), SRM (Lee et al., 2019), SE (Hu et al., 2018), ECA (Wang et al., 2020b), and the proposed DCPGAttention. All other components are kept unchanged to ensure fair comparison.

The experimental results are shown in Table 5. All attention modules improve the detection performance to varying degrees, although they slightly increase GFLOPs and parameters. SE achieves the best Recall and FPS, but its mAP@50 and mAP@50~95 are slightly lower than those of DCPGAttention. GCT and SRM show advantages in Precision and Recall, but their overall mAP scores are relatively low. ECA achieves a similar inference speed to DCPGAttention, but still lags in mAP. Overall, DCPGAttention achieves the best mAP performance, with an mAP@50 of 75.0% and an mAP@50~95 of 48.7%. This suggests that the combination of dual channel pooling and gate control mechanisms effectively captures global features and improves detection performance. Meanwhile, DCPGAttention maintains a comparable parameter and computational cost to other attention modules, and reaches an inference speed of 114.9 FPS, showing good efficiency and real-time capability.

To further illustrate the influence of the DCPGAttention mechanism on feature learning, attention heatmaps were generated for the highly relevant GCT, SRM, and DCPGAttention modules for comparison. As shown in Figure 9, different attention modules guide the model to focus on distinct spatial regions, revealing their varying abilities to capture critical target features. Specifically, the GCT module tends to produce attention over a relatively large area, but some targets are missed and the focus is dispersed, lacking concentration on key regions. The SRM module improves upon this by concentrating attention more effectively, but the focused regions are still limited, and some pests are not effectively captured by the attention mechanism. The DCPGAttention module, while occasionally missing individual pests, demonstrates a more concentrated and comprehensive focus compared to GCT and SRM, effectively capturing both the target body and edges. This indicates that DCPGAttention provides improved attention distribution, highlighting critical features necessary for accurate pest detection more effectively than the other two modules.

3.3.6 Small-neck and channel pruning

Different feature fusion networks have a key impact on the model's detection performance. To verify the advantage of the improved lightweight Small-Neck in small pest detection tasks, ablation experiments were conducted based on the YOLOv8n baseline. The results are shown in Table 6. After introducing the enhanced a-BiFPN, the model's detection precision improved from 75.5% to 80.0%, and mAP@50 increased to 74.4%, with almost no impact on inference speed. Then, integrating GSConv and VoVGSCSP structures further reduced computation and parameters, while maintaining stable accuracy. However, inference speed slightly decreased. Finally, considering that pests in the Pest24 dataset are generally small, we observed redundancy in the feature fusion layers P4 and P5 designed for large objects. Therefore, we pruned the channel numbers of P4 and P5 to 256, forming the final Small-Neck structure. Although Small-Neck caused a slight drop in accuracy, it significantly reduced parameters and computation. Its FPS increased to 108.7, greatly improving deployment efficiency on edge devices and meeting practical application requirements.

3.3.7 Comparison of different IoU loss functions

To evaluate the impact of different regression loss functions on detection performance, this paper uses YOLO-DCPG as the baseline model. The tested losses include CIoU, SIoU (Gevorgyan, 2022), DIoU (Zheng et al., 2020), WIoU v1 (Tong et al., 2023), WIoU v2 (Tong et al., 2023), WIoU v3 (Tong et al., 2023), and Inner-WIoU. All experiments are conducted under the same training settings and dataset. The comparison focuses on differences in localization accuracy and pest detection ability. The results are shown in Table 7. Compared with CIoU, SIoU, DIoU, WIoU v1, WIoU v2, and WIoU v3 improve detection precision but show varying degrees of decline in Recall. Among them, WIoU v3 achieves the best mAP@50 performance.

Based on this, we introduce a scale factor called ratio into the WIoU v3 loss function, using Inner-IoU to control the size of the auxiliary box. This helps the model focus more on small pest regions and effectively improves bounding box regression accuracy. When the ratio is set to 0.7, the model achieves the highest Precision, but the Recall drops significantly. To further optimize this parameter, we conducted a systematic experiment with a step size of 0.05 in the range [1, 1.2]. The results show that

TABLE 5 Comparison of different attention models' performance.

Model	Precision (%)	Recall (%)	mAP@50 (%)	mAP@50~95 (%)	GFLOPs (G)	Parameters (M)	FPS (Image/s)
YOLOv8n	75.5	67.4	73.2	46.6	8.2	3.02	105.3
+GCT	78.8	70.0	74.3	48.1	8.2	3.02	113.6
+ECA	78.5	69.5	74.4	48.3	8.2	3.02	114.9
+SRM	79.0	70.0	74.0	47.8	8.2	3.02	114.9
+SE	78.2	70.1	74.5	48.5	8.2	3.03	116.3
+DCPGAttention	78.5	69.9	75.0	48.7	8.2	3.02	114.9

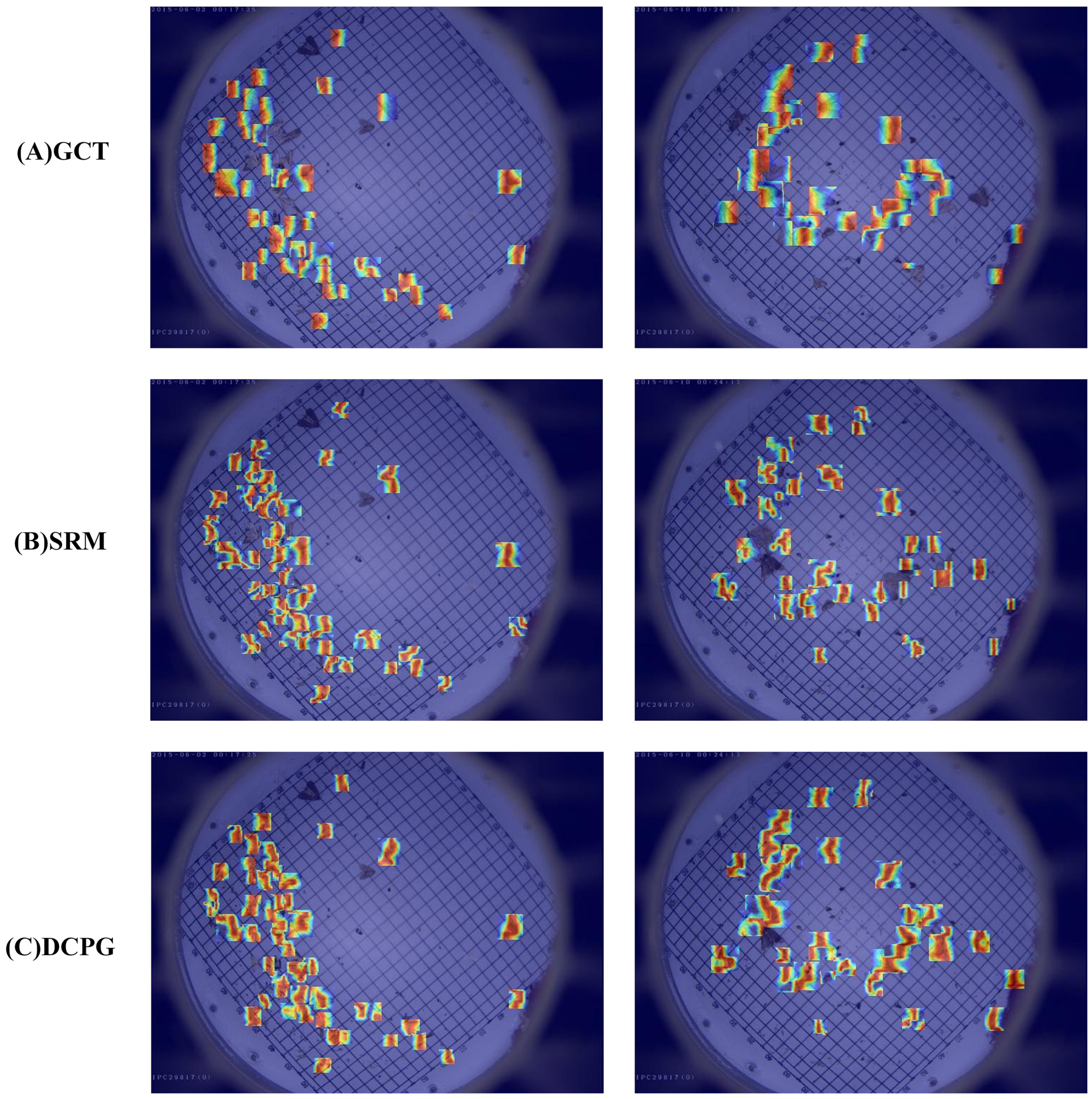

FIGURE 9
Visualization of attention heatmap. Where warmer colors represent higher attention intensity, highlighting the regions most focused by the model during pest detection.

TABLE 6 Comparison of different attention models' performance.

Model	Precision (%)	Recall (%)	mAP@50 (%)	mAP@50~95 (%)	GFLOPs (G)	Parameters (M)	FPS (Image/s)
YOLOv8n	75.5	67.4	73.2	46.6	8.2	3.02	105.3
+a-BiFPN	80.0	70.0	74.4	48.0	7.9	2.85	105.3
+GSConv+VoVGSCSP	76.0	69.7	74.2	47.8	7.4	2.81	104.2
+a-BiFPN+GSConv +VoVGSCSP	80.1	69.9	74.6	47.5	7.0	2.65	101.1
Small-Neck(Channel Pruning)	79.9	69.4	74.3	47.6	6.6	1.88	108.7

TABLE 7 Comparison of different loss functions.

IoU	Precision (%)	Recall (%)	mAP@50 (%)	mAP@50~95 (%)	GFLOPs (G)	Parameters (M)
CIoU	78.1	69.6	73.5	46.9	5.7	1.47
SIoU	78.8	69.0	73.5	47.4	5.7	1.47
DIoU	79.4	69.2	73.6	47.4	5.7	1.47
WIoU v1	79.6	69.0	73.1	47.2	5.7	1.47
WIoU v2	78.3	69.1	73.4	47.3	5.7	1.47
WIoU v3	80.1	69.1	73.8	47.2	5.7	1.47
Inner-WIoU (radio=0.7)	81.0	67.9	73.4	47.2	5.7	1.47
Inner-WIoU (radio=1.2)	74.9	70.0	73.7	47.8	5.7	1.47
Inner-WIoU (radio=1.15)	80.5	69.1	73.6	47.6	5.7	1.47
Inner-WIoU (radio=1.1)	76.4	69.3	73.6	48.0	5.7	1.47
Inner-WIoU (radio=1.05)	80.1	69.2	74.0	47.5	5.7	1.47

setting the ratio to 1.05 leads to the best mAP@50 improvement. It also provides a better balance across multiple metrics, including Precision, Recall, and mAP@50~95, resulting in more stable and effective supervision during training.

3.3.8 Visualization of model predictions

To provide a more comprehensive comparison of detection performance across models, this paper presents visualization results of Original, RT-DETR-resnet18 (Zhao et al., 2024), YOLOv3-tiny (Adarsh et al., 2020), YOLOv11n (Khanam and Hussain, 2024), YOLOv8n (Varghese and Sambath, 2024), and the proposed YOLO-DCPG. The results are shown in Figure 10. From left to right, the number of pests in the sample images gradually increases, making the detection task more challenging. For images with fewer pests, most models perform reasonably well. RT-DETR-resnet18, which has the largest number of parameters, shows the best performance in these cases. However, as pest density increases, performance differences between models become more noticeable. YOLOv11n struggles in high-density scenes, with many missed detections. YOLOv3-tiny and YOLOv8n also show some missed and overlapping boxes when detecting small pests or dense clusters. In contrast, YOLO-DCPG, with structural improvements over YOLOv8n, significantly reduces model parameters and computational cost while maintaining high detection accuracy. Even in high-density, multi-target scenarios, YOLO-DCPG demonstrates strong robustness and generalization capability.

3.3.9 Generalization evaluation on other datasets

To evaluate the generalization capability of the YOLO-DCPG module, experiments were conducted on the RP11 dataset using both the baseline YOLOv8n and the proposed YOLO-DCPG model. RP11 is a real-world dataset containing 4,559 images of 11 rice pest species. The dataset was divided into training, validation, and test sets in a 7:2:1 ratio, resulting in 3,191 training images, 912 validation images, and 456 test images. The detection results of the two models on this dataset are shown in Table 8.

As shown in Table 8, the YOLO-DCPG model achieves a precision of 90.3%, mAP@50 of 87.8%, mAP@75 of 82.8%, and mAP@50~95 of 71.3% on the RP11 dataset. Compared with the baseline YOLOv8n, these correspond to improvements of 1.1%, 2.4%, 3.6%, and 1.1%, respectively. In addition, YOLO-DCPG significantly reduces computational cost and model size, with GFLOPs decreasing from 8.2 to 5.7 and parameters reduced from 3.02M to 1.47M. The results demonstrate that YOLO-DCPG enhances detection performance and efficiency, confirming its effectiveness and broad applicability. To provide a clearer comparison of the models' detection performance, the visualization results are presented in Figure 11.

3.4 Edge deployment and visualization results

3.4.1 Deployment on edge devices

To verify the practical applicability of the proposed model in real agricultural environments, this study deploys the model on an edge computing device based on the Raspberry Pi platform. Specifically, a Raspberry Pi 4B is used as the core hardware. It is equipped with a Hikvision camera, a 4G network module, a voltage regulation circuit, and a custom 3D-printed case to ensure system stability and environmental adaptability. The assembled device is shown in Figure 12.

To meet the limited computational resources of edge devices, this paper adopts the lightweight backbone StarNet-S100 during

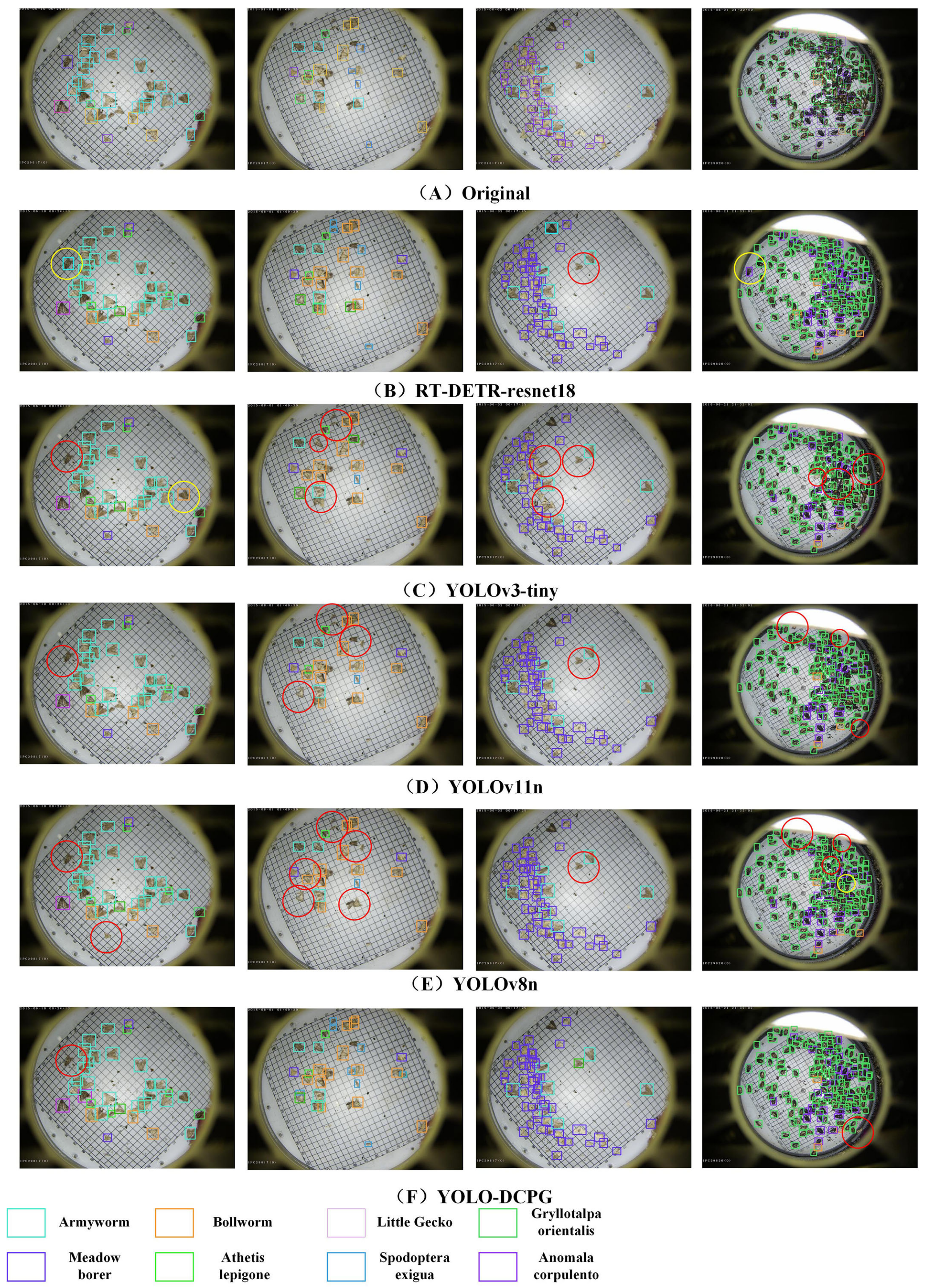

FIGURE 10
Visualization results of different models. The red circles indicate missed detections, while the yellow circles indicate false detections.

TABLE 8 Comparison of different attention models' performance.

Model	Precision (%)	mAP@50 (%)	mAP@75 (%)	mAP@50~95 (%)	GFLOPs (G)	Parameters (M)
YOLOv8n	89.2	85.4	79.2	70.2	8.2	3.02
YOLO-DCPG(our)	90.3	87.8	82.8	71.3	5.7	1.47

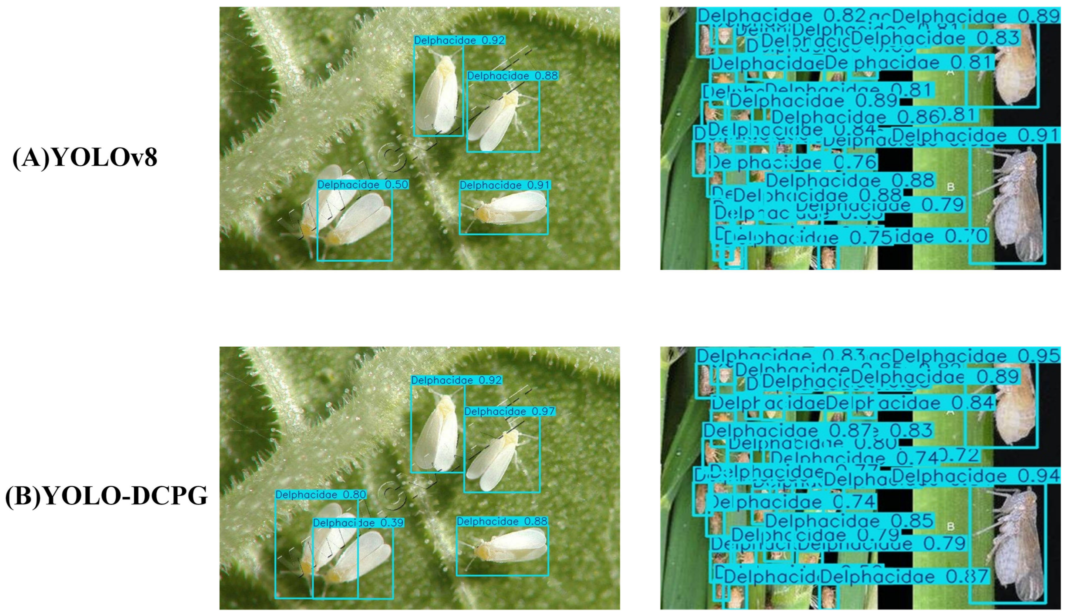

FIGURE 11
Visualization results of different models.

model training. In addition, a Small-Neck feature fusion module combined with channel pruning is designed to effectively reduce the model's parameters and computational complexity. During deployment, the trained model is converted to the ONNX format and integrated into the Raspberry Pi environment. ONNX Runtime is used as the inference engine. It is optimized for ARM architecture and supports multi-threading, as well as fast loading and execution of lightweight models. This setup enables the YOLO-DCPG model to achieve a real-time inference speed of 12.4 FPS on the Raspberry Pi 4B. It ensures efficient and stable operation on edge devices.

3.4.2 Visualization of inference results on edge computing device

To evaluate the detection performance of models on edge devices, this paper compares five lightweight models: YOLOv3-tiny (Adarsh et al., 2020), YOLOv5n (Jocher et al., 2020), YOLOv11n (Khanam and Hussain, 2024), YOLOv8n (Varghese and Sambath, 2024), and YOLO-DCPG. All models were exported to the ONNX format and deployed on a Raspberry Pi 4B for inference testing. The results are shown in Figure 13. The top-left corner of each image displays the current inference frame rate

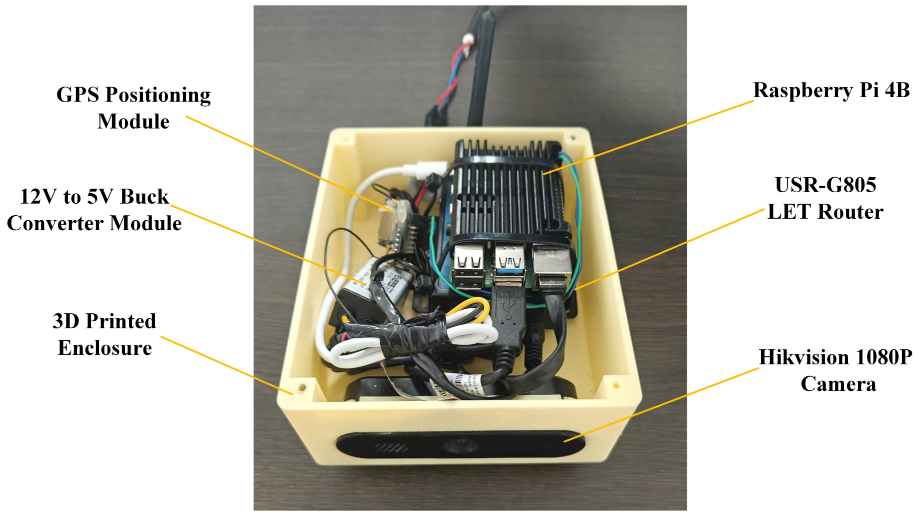

FIGURE 12
Photo of the device.

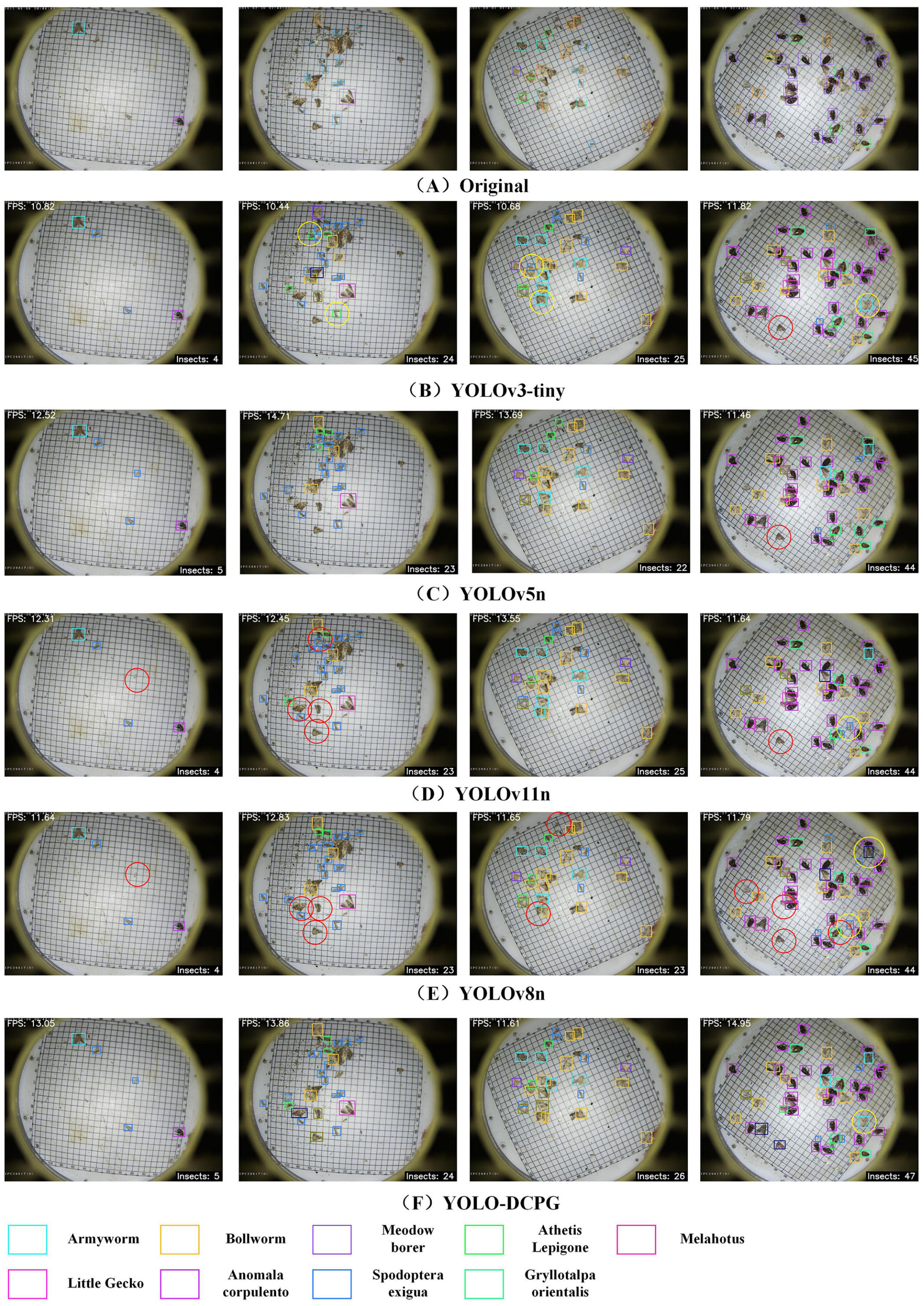

FIGURE 13
Visualization of inference results on raspberry Pi. The red circles indicate missed detections, while the yellow circles indicate false detections.

(FPS), while the bottom-right corner shows the number of detected pests. To further compare inference speed and detection performance, Table 9 summarizes the number of detected pests, average FPS, and ONNX model size for each model.

As shown in Table 9 and Figure 13, all models perform well when detecting simple sample images. However, as the number of pests increases, YOLO-DCPG detects 102 pest samples, while YOLOv5n and YOLOv8n detect only 94, indicating noticeable missed detections. YOLOv3-tiny and YOLOv11n achieve similar results but still fail to identify some targets. In terms of inference speed, YOLO-DCPG reaches 13.4 FPS on the Raspberry Pi, exceeding YOLOv3-tiny, YOLOv5n, YOLOv11n, and YOLOv8n by 2.4, 0.3, 0.9, and 1.4 FPS, respectively. Moreover, its ONNX model size is only 5.5 MB, the smallest among all compared models. YOLO-DCPG provides

TABLE 9 Inference speed and detection performance of different pest detection models on the Raspberry Pi platform.

Model	Number of detected pests	FPS (Image/s)	ONNX model size (M)
YOLOv3-tiny	98	11.0	36.3
YOLOv5n	94	13.1	7.7
YOLOv11n	96	12.5	9.9
YOLOv8n	94	12.0	10.0
YOLO-DCPG(our)	102	13.4	5.5

comparable inference speed and higher detection accuracy on the Raspberry Pi, identifying more pest targets and better satisfying the dual requirements of real-time performance and accuracy in agricultural applications. Therefore, YOLO-DCPG demonstrates higher practical value for edge device deployment.

4 Discussion

To balance high precision pest detection and edge deployment efficiency, this paper proposes a pest detection model based on dual channel pooling gated attention. This attention module uses both average pooling and standard deviation pooling channels to effectively enhance the model's focus on key pest regions. In the feature extraction stage, the lightweight backbone StarNet-S100 is used. It significantly reduces the model size while maintaining good feature representation. This facilitates deployment on edge devices. In the feature fusion stage, a Small-Neck structure is designed. It combines an improved a-BiFPN and GSConv modules to achieve efficient multi-scale feature fusion and transmission. To improve small object detection, the Inner-WIoU loss function is introduced in the regression loss. The scale factor ratio guides the model to better focus on tiny pests, enhancing localization accuracy and speeding up convergence. On the Pest24 tiny pest dataset, YOLO-DCPG achieves a Precision of 80.1%, mAP@50 of 74.0%, and mAP@50~95 of 47.5%. The model's computation is 5.7 GFLOPs with only 1.47 million parameters, and the final model size is limited to 3.2 MB. Compared to the baseline YOLOv8n, the parameters, computation, and model size are reduced by 51.2%, 30.1%, and 46.7%, respectively. This significantly improves the model's suitability for deployment on edge devices. It provides a more efficient and practical solution for agricultural pest monitoring in limited-resource environments.

Although this study has made progress in tiny pest detection, some limitations remain. On one hand, the model still suffers from missed and false detections in complex scenes with high object density or occlusion. On the other hand, the current dataset has limited coverage and cannot fully represent the diversity of pests in real field environments. To improve the model's generalization and practical performance, future work will include continuous collection of real-world images, expansion of the dataset, and incremental annotation and training. In addition, the model will be migrated to high-performance edge devices such as Jetson Nano to further enhance system responsiveness and real-time performance, better meeting the needs of agricultural pest monitoring in real-world applications.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material. Further inquiries can be directed to the corresponding authors.

Author contributions

JL: Conceptualization, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. EY: Formal Analysis, Investigation, Methodology, Resources, Validation, Writing – original draft, Writing – review & editing. YL: Funding acquisition, Investigation, Methodology, Resources, Supervision, Writing – review & editing. YZ: Conceptualization, Formal Analysis, Investigation, Methodology, Resources, Validation, Visualization, Writing – original draft, Writing – review & editing. BM: Project administration, Software, Writing – original draft.

Funding

The author(s) declare that financial support was received for the research and/or publication of this article. This research was funded by the National Science and Technology Major Special Project of New Generation Artificial Intelligence "Demonstration of Integrated Application of Key Technologies in Smart Farm" (No. 2022ZD0115805). The major scientific and technological project of Xinjiang Uygur Autonomous Region: "Research on Key Technologies for Digitalization and Intelligence of Farms" (No. 2022A02011). The National Natural Science Foundation of China (No. 32560419), with support from the open research projects of the Xinjiang Agricultural Informatization Engineering Technology Research Center (No. XJAIEC2025K011).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Generative AI was used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

References

Adarsh, P., Rathi, P., and Kumar, M. (2020). "YOLO v3-Tiny: Object Detection and Recognition using one stage improved model," in *2020 6th international conference on advanced computing and communication systems (ICACCS)*, (Piscataway, New Jersey, USA: IEEE). 687–694. doi: 10.1109/ICACCS48705.2020.9074315

Ali, F., Qayyum, H., and Iqbal, M. J. (2023). Faster-PestNet: A Lightweight deep learning framework for crop pest detection and classification. *IEEE Access* 11, 104016–104027. doi: 10.1109/ACCESS.2023.3317506

Amrani, A., Diepeveen, D., Murray, D., Jones, M. G., and Sohel, F. (2024). Multi-task learning model for agricultural pest detection from crop-plant imagery: A Bayesian approach. *Comput. Electron. Agric.* 218, 108719. doi: 10.1016/j.compag.2024.108719

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. doi: 10.48550/arXiv.2004.10934. arXiv preprint arXiv:2004.10934.

Chen, J., Kao, S.-H., He, H., Zhuo, W., Wen, S., Lee, C.-H., et al. (2023). "Run, don't walk: chasing higher FLOPS for faster neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (Vancouver, BC, Canada: IEEE) 12021–12031. doi: 10.1109/CVPR52729.2023.01157

Chodey, M. D., and Shariff, C. N. (2023). Pest detection via hybrid classification model with fuzzy C-means segmentation and proposed texture feature. *Biomed. Signal Process. Control* 84, 104710. doi: 10.1016/j.bspc.2023.104710

Chollet, F. (2017). "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition* (Piscataway: IEEE) 1251–1258. doi: 10.1109/CVPR.2017.195

Deepika, P., and Arthi, B. (2022). Prediction of plant pest detection using improved mask FRCNN in cloud environment. *Measure: Sensors* 24, 100549. doi: 10.1016/j.measen.2022.100549

Dong, Q., Han, T., Wu, G., Qiao, B., and Sun, L. (2025a). RSNet: compact-align detection head embedded lightweight network for small object detection in remote sensing. *Remote Sens.* 17, 1965. doi: 10.3390/rs17121965

Dong, Q., Han, T., Wu, G., Sun, L., Huang, M., and Zhang, F. (2025b). Industrial device-aided data collection for real-time rail defect detection via a lightweight network. *Eng. Appl. Artif. Intell.* 161, 112102. doi: 10.1016/j.engappai.2025.112102

Dong, Q., Sun, L., Han, T., Cai, M., and Gao, C. (2024). PestLite: A novel YOLO-based deep learning technique for crop pest detection. *Agriculture* 14, 228. doi: 10.3390/agriculture14020228

Ebrahimi, M., Khoshtaghaza, M. H., Minaei, S., and Jamshidi, B. (2017). Vision-based pest detection based on SVM classification method. *Comput. Electron. Agric.* 137, 52–58. doi: 10.1016/j.compag.2017.03.016

Gevorgyan, Z. (2022). SIoU loss: More powerful learning for bounding box regression. doi: 10.48550/arXiv.2205.12740. arXiv preprint arXiv:2205.12740.

Girshick, R. (2015). "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision* (Piscataway: IEEE) 1440–1448. doi: 10.1109/ICCV.2015.169

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition* (Piscataway: IEEE) 580–587. doi: 10.1109/CVPR.2014.81

Hu, J., Shen, L., and Sun, G. (2018). "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition* (Piscataway: IEEE) 7132–7141. doi: 10.1109/CVPR.2018.00745

Huang, Y., Liu, Z., Zhao, H., Tang, C., Liu, B., Li, Z., et al. (2025). YOLO-YSTs: an improved YOLOv10n-based method for real-time field pest detection. *Agronomy* 15, 575. doi: 10.3390/agronomy15030575

Jocher, G., Stoken, A., Borovec, J., Changyu, L., Hogan, A., Diaconu, L., et al. (2020). *ultralytics/yolov5: v3. 0* (Zenodo). doi: 10.5281/zenodo.3983579

Khanam, R., and Hussain, M. (2024). Yolov11: An overview of the key architectural enhancements. doi: 10.48550/arXiv.2410.17725. arXiv preprint arXiv:2410.17725.

Lee, H., Kim, H.-E., and Nam, H. (2019). "Srm: A style-based recalibration module for convolutional neural networks," in *Proceedings of the IEEE/CVF International conference on computer vision* (Seoul, Korea: IEEE) 1854–1862. doi: 10.1109/ICCV.2019.00194

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., et al. (2022a). YOLOv6: A single-stage object detection framework for industrial applications. doi: 10.48550/arXiv.2209.02976. arXiv preprint arXiv:2209.02976.

Li, H., Li, J., Wei, H., Liu, Z., Zhan, Z., and Ren, Q. (2022b). Slim-neck by GSConv: A better design paradigm of detector architectures for autonomous vehicles. doi: 10.48550/arXiv.2206.02424. arXiv preprint arXiv:2206.02424 10.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition* (Piscataway: IEEE) 2117–2125. doi: 10.1109/CVPR.2017.106

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., et al. (2016). "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference* (Amsterdam, The Netherlands). 21–37. doi: 10.1007/978-3-319-46448-0_2

Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. (2018). "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition* (Piscataway: IEEE) 8759–8768. doi: 10.1109/CVPR.2018.00913

Liu, J., Zhou, C., Zhu, Y., Yang, B., Liu, G., and Xiong, Y. (2025). RicePest-DETR: A transformer-based model for accurately identifying small rice pest by end-to-end detection mechanism. *Comput. Electron. Agric.* 235, 110373. doi: 10.1016/j.compag.2025.110373

Lv, W., Zhao, Y., Chang, Q., Huang, K., Wang, G., and Liu, Y. (2024). Rt-detrv2: Improved baseline with bag-of-freebies for real-time detection transformer. doi: 10.48550/arXiv.2407.17140. arXiv preprint arXiv:2407.17140.

Lyu, R. (2020). Super fast and lightweight anchor-free object detection model. *Real-time Mobile Devices*. Available online at: https://github.com/RangiLyu/nanodet (Accessed October 27, 2025).

Ma, X., Dai, X., Bai, Y., Wang, Y., and Fu, Y. (2024). "Rewrite the stars," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Piscataway: IEEE) 5694–5703.

Qin, D., Leichner, C., Delakis, M., Fornoni, M., Luo, S., Yang, F., et al. (2024). "MobileNetV4: universal models for the mobile ecosystem," in *European Conference on Computer Vision*. 78–96 (Cham, Switzerland: Springer). doi: 10.1007/978-3-031-73661-2_5

Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 1137–1149. doi: 10.1109/TPAMI.2016.2577031

Song, J., Cheng, K., Chen, F., and Hua, X. (2025). RDW-YOLO: A deep learning framework for scalable agricultural pest monitoring and control. *Insects* 16, 545. doi: 10.3390/insects16050545

Sprague, G. (1975). Agriculture in China. *Science* 188, 549–555. doi: 10.1126/science.188.4188.549

Sun, F., Guan, Z., Lyu, Z., and Liu, S. (2025). High-precision stored-grain insect pest detection method based on PDA-YOLO. *Insects* 16, 610. doi: 10.3390/insects16060610

Tan, M., Pang, R., and Le, Q. V. (2020). "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (Piscataway: IEEE) 10781–10790. doi: 10.1109/CVPR42600.2020.01079

Tang, Z., Lu, J., Chen, Z., Qi, F., and Zhang, L. (2023). Improved Pest-YOLO: Real-time pest detection based on efficient channel attention mechanism and transformer encoder. *Ecol. Inf.* 78, 102340. doi: 10.1016/j.ecoinf.2023.102340

Tian, Y., Ye, Q., and Doermann, D. (2025). Yolov12: Attention-centric real-time object detectors. doi: 10.48550/arXiv.2502.12524. arXiv preprint arXiv:2502.12524.

Tong, Z., Chen, Y., Xu, Z., and Yu, R. (2023). Wise-IoU: bounding box regression loss with dynamic focusing mechanism. doi: 10.48550/arXiv.2301.10051. arXiv preprint arXiv:2301.10051.

Varghese, R., and Sambath, M. (2024). "Yolov8: A novel object detection algorithm with enhanced performance and robustness," in *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)* (Chennai, India: IEEE) 1–6. doi: 10.1109/ADICS58448.2024.10533619

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., and Han, J. (2024). Yolov10: Real-time end-to-end object detection. *Adv. Neural Inf. Process. Syst.* 37, 107984–108011. doi: 10.48550/arXiv.2405.14458

Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W., and Hu, Q. (2020b). "ECA-Net: Efficient channel attention for deep convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (Piscataway: IEEE) 11534–11542. doi: 10.1109/CVPR42600.2020.01155

Wang, Q.-J., Zhang, S.-Y., Dong, S.-F., Zhang, G.-C., Yang, J., Li, R., et al. (2020a). Pest24: A large-scale very small object data set of agricultural pests for multi-target detection. *Comput. Electron. Agric.* 175, 105585. doi: 10.1016/j.compag.2020.105585

Wen, C., Chen, H., Ma, Z., Zhang, T., Yang, C., Su, H., et al. (2022). Pest-YOLO: A model for large-scale multi-class dense and tiny pest detection and counting. *Front. Plant Sci.* 13. doi: 10.3389/fpls.2022.973985

Xue, R., and Wang, L. (2025). Research on lightweight citrus leaf pest and disease detection based on PEW-YOLO. *Processes* 13, 1365. doi: 10.3390/pr13051365

Yang, Z., Zhu, L., Wu, Y., and Yang, Y. (2020). "Gated channel transformation for visual recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (Piscataway: IEEE) 11794–11803. doi: 10.1109/CVPR42600.2020.01181

Yu, J., Jiang, Y., Wang, Z., Cao, Z., and Huang, T. (2016). "Unitbox: An advanced object detection network," in *Proceedings of the 24th ACM international conference on Multimedia* (Amsterdam, The Netherlands) 516–520. doi: 10.1145/2964284.2967274

Yu, P., Zong, B., Geng, X., Yan, H., Liu, B., Chen, C., et al. (2025). DGS-Yolov7-Tiny: a lightweight pest and disease target detection model suitable for edge computing environments. *Sci. Rep.* 15, 29818. doi: 10.1038/s41598-025-13410-8

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. doi: 10.48550/arXiv.1710.09412. arXiv preprint arXiv:1710.09412.

Zhang, X.a., Qiao, H., Xi, L., Ma, W., Xu, X., Hui, X., et al. (2025). Decting small objects for field crop pests using improved YOLOv8. *Trans. Chin. Soc. Agric. Eng.* 41, 225–233. doi: 10.11975/j.issn.1002-6819.202502116

Zhang, H., Xu, C., and Zhang, S. (2023). Inner-iou: more effective intersection over union loss with auxiliary bounding box. doi: 10.48550/arXiv.2311.02877. arXiv preprint arXiv:2311.02877.

Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., et al. (2024). "Detrs beat yolos on real-time object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (Piscataway: IEEE) 16965–16974. doi: 10.48550/arXiv.2304.08069

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., and Ren, D. (2020). "Distance-IoU loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI conference on artificial intelligence* (New York, USA) 12993–13000. doi: 10.1609/aaai.v34i07.6999