agronomy

MDPI

Article

A Deep-Learning-Based Detection Method for Small Target Tomato Pests in Insect Traps

Song Wang [1,†], Daqing Chen [1,*,†], Jianxia Xiang [1] and Cong Zhang [2]

[1] School of Mathematics and Computer, Wuhan Polytechnic University, Wuhan 430048, China; wnsg2001@163.com (S.W.); xjx990620@163.com (J.X.)
[2] School of Electrical and Electronic Engineering, Wuhan Polytechnic University, Wuhan 430023, China; hb_wh_zc@163.com
* Correspondence: chendaqing524@163.com
† These authors contributed equally to this work.

Abstract: In a greenhouse environment where tomatoes are grown, pests in yellow sticky traps need to be detected in order to control the pest population. However, tomato pests typically found on yellow sticky traps are small in size and lack distinct visual features, making it difficult for convolutional networks to extract sufficient contextual information, thereby rendering the tasks of localization and classification exceptionally challenging. In this work, an improved approach based on the advanced object detection model You Only Look Once version 7-tiny (YOLOv7-tiny) is introduced, aiming to enhance the accuracy of detecting small tomato pests while maintaining computational complexity. Firstly, a context information extraction block (CIE) based on a Transformer encoder is proposed, and this block aims to capture global context, explore potential relationships between features, and emphasize important characteristics. Secondly, an Tiny-ELAN fusion network is introduced, which enhanced the feature fusion ability of the network. Thirdly, the feature fusion part takes the P2 feature layer into account and adds a P2 small target detection head. Finally, the SCYLLA-IoU (SIoU) loss function is introduced, and its components are redefined to incorporate direction information, which enhances the model's learning ability and convergence performance. Experimental results show that our method can accurately detect three insects: whitefly (WF), macrolophus (MR), and nesidiocoris (NC) in the yellow sticky trap images of tomato crops. Compared with Faster R-CNN, SSD, YOLOv3-tiny, YOLOv5s, YOLOv7-tiny, YOLOv7, YOLOv7-x, YOLOv8n, YOLOv8s, YOLOv10n, and RT-DETR, the mean average precision of our method increased by 3.14%, 11.8%, 4.7%, 4.7%, 4.4%, 3.5%, 2.9%, 4.6%, 4.4%, 4.2%, and 4.2%, respectively.

Keywords: pest detection; YOLOv7; Transformer encoder; computer vision; deep learning

Citation: Wang, S.; Chen, D.; Xiang, J.; Zhang, C. A Deep-Learning-Based Detection Method for Small Target Tomato Pests in Insect Traps. *Agronomy* **2024**, *14*, 2887. https://doi.org/10.3390/agronomy14122887

Academic Editor: Christos Athanassiou

Received: 27 October 2024
Revised: 27 November 2024
Accepted: 29 November 2024
Published: 3 December 2024

1. Introduction

In recent years, with economic development and the implementation of various policies to encourage agricultural science and technology, China's greenhouse vegetable technology has rapidly advanced. This progress has led to continuous improvements in the quality and yield of vegetables, with tomatoes being one of the most widely cultivated crops [1]. As a significant global fruit, tomatoes hold an important position in China's national economy [2]. Tomatoes are frequently subject to pest infestations, with whiteflies being a particularly prominent threat. These pests feed on the sap of tomato plants, inducing leaf yellowing and wilting, thereby impeding photosynthesis. This, in turn, restricts the growth of tomato fruits, leading to smaller sizes and, ultimately, substantial reductions in yield. Studies have indicated that in tomato fields severely affected by whiteflies, the yield loss can range from 20% to 50%, and in extreme cases, it can even reach 100% [3]. However, pest control remains a major challenge for farmers [4], making it crucial to find effective strategies to ensure high tomato yields. Due to pests' attraction to specific wavelengths of

light, many methods have been developed to control and monitor them, including the use of yellow sticky traps [5].

Yellow sticky traps play a vital role in integrated pest management (IPM). They can be used alone to reduce pest populations and to monitor pest density [6,7]. Additionally, they can be combined with biological control strategies to manage pest populations effectively [8]. Identifying and counting pests on these traps is essential for effective pest management. During the implementation of biological control, the yellow sticky trap may simultaneously capture tomato pests, such as whitefly (WF), and the predators of tomato pests, like macrolophus (MR) and nesidiocoris (NC). The effectiveness of the current biological control can be analyzed by counting the densities of the pest and their predators. Therefore, improving the accuracy of insect detection in yellow sticky trap images of tomato crops can better assist farmers in managing pests in greenhouse environments.

In recent years, deep-learning-based models have continuously achieved breakthroughs in visual tasks such as image classification and object detection. Since 2012, convolutional neural networks (CNNs) have become the dominant models for visual tasks. Currently, many advanced image classification methods are based on CNNs, such as VGGNet, ResNet, and DenseNet. Another type of image classification method is based on the Transformer. The Transformer network, which was originally widely used in natural language processing tasks [9], has become a highly favored architecture. A particularly notable advancement is the introduction of the Vision Transformer (ViT) [10] in image classification tasks. It divides the image into multiple patches and utilizes the multi-head attention mechanism to capture the intricate dependencies within the image, thereby achieving image classification. To some extent, object detection can be regarded as a downstream task of image classification. As a hot topic in computer vision, object detection algorithms can be divided into anchor-based object detection models and anchor-free object detection models. The representative algorithms of the anchor-based object detection models are the YOLO (You Only Look Once) series [11,12] and the R-CNN (Regions with CNN Features) series [13–15]. The advantages of the YOLO series methods lie in their fast detection speed and high efficiency, enabling real-time object detection. However, due to the relatively slow detection speed, the application of the R-CNN series is limited in some scenarios with high real-time requirements. The representative algorithm of the anchor-free object detection models is the Transformer-based RT-DETR [16]. The disadvantage of such methods is that they may require a large amount of computational resources and data to achieve the best performance. In summary, we choose the YOLO series methods with a relatively high detection speed as the foundation for further research. By comparing the performance of different versions of the YOLO methods, such as YOLOv3-tiny, YOLOv5s, YOLOv7-tiny, and YOLOv8n. In subsequent comparative experiments, we find that YOLOv7-tiny has the best detection accuracy and its inference speed is lower than that of YOLOv7 and YOLOv7x. Therefore, YOLOv7-tiny is determined as the benchmark model.

Despite the advances in deep learning and object detection algorithms, detecting small pests on yellow sticky traps poses significant challenges. The methods based on convolutional neural network like YOLOv7, which have demonstrated outstanding performance in various object detection tasks, face difficulties when applied to images where the targets are small and visually indistinct. The image from the yellow sticky traps dataset [17] shown in Figure 1 is a magnified portion of high-definition photograph taken by a camera of a yellow sticky trap. Tomato pests are marked with rectangular boxes, while other irrelevant insects are unmarked. We observe that tomato pests exhibit few visual features due to their small size, and they are also subject to interference from other irrelevant categories in the environment. These factors present obstacles for the detection task.

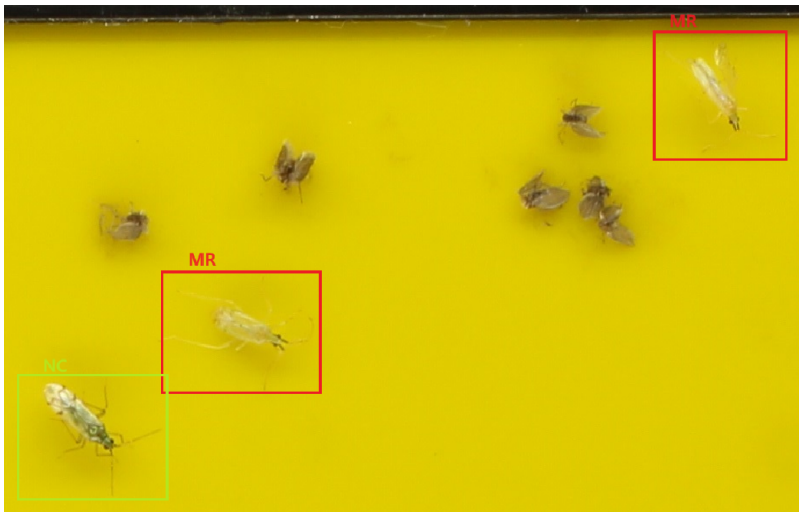


Figure 1. This is an enlarged image of yellow sticky trap. The yellow sticky trap contains both the insects that need to be detected for the biological control of tomato pests, such as the macrolophus genus (MR) and nesidiocoris (NC) which are marked with rectangles, and the unmarked irrelevant insects and debris.

In summary, during the biological control process of tomatoes, the challenges faced in detecting insects on the yellow sticky trap can be summarized as follows: Firstly, the limited image resolution and small size of the targets lead to fewer visual features, making it difficult for the network to extract key information and increasing the susceptibility to interference from the background and other insects. Secondly, when the feature visibility is restricted, for example, under low-light conditions or when pests have similar colors to the background, it becomes extremely difficult to distinguish pests from non-pests based on visual features. For example, as shown in Figure 1, the color of MR insects can be easily confused with the yellow background, and it is difficult to accurately identify them relying only on limited visual features. Thirdly, sometimes when several insects to be detected are close to each other or even overlapping, the network may misrecognize these clustered insects as irrelevant objects, resulting in missed detections.

To overcome the above challenges, we propose an improved YOLOv7-tiny method. In view of the difficulty that the colors of insects themselves are easily confused with the environment, we attempt to combine the object detection method based on deep convolutional neural networks with the Transformer encoder to address the locality limitation of convolutional networks. More specifically, taking YOLOv7-tiny as the foundation, we explore an alternative method CIE based on the Transformer encoder rather than deep convolutional networks. This method aims to effectively capture contextual information and identify significant features, thereby reducing the susceptibility to interference from the background and other insects. To address the challenges of the small size of the insects to be detected, the difficulty in extracting their features, and the defect that the traditional feature pyramid network may lose some information during the process of fusing features of non-adjacent layers, we propose a Tiny-ELAN fusion network based on the Gather-and-Distribute mechanism with a P2 detection head to enhance the feature fusion ability of non-adjacent layers. Moreover, the newly added P2 detection head can effectively improve the detection ability of small target insects. In addition, inspired by Shi et al. [18], who argued that considering the directional mismatch between ground truth and predicted boxes in the regression framework can greatly improve the accurate detection of small-sized objects, we integrate the SIoU loss function into the enhanced YOLOv7-tiny to improve the model performance during training.

The experimental results show that the proposed method outperforms advanced object detection methods such as YOLOv5-tiny, YOLOv5s, YOLOv7-tiny, YOLOv8s, YOLOv10n,

and RT-DETR on a public yellow sticky traps dataset [17], achieving the highest mean average precision (mAP) metric.

The main contributions of this paper are:

(1) We propose a Tiny-ELAN fusion network based on the Gather-and-Distribute mechanism with a P2 detection head to enhance the feature fusion ability of non-adjacent layers.

(2) A context information extraction block based on the Transformer encoder is proposed to enhance the network's ability to extract global semantic information and focus on more important features while maintaining computational complexity.

(3) Introducing the SIoU loss function and redefining components with direction information in the bounding box loss function facilitates easier regression of predicted boxes and enhances the model's convergence ability.

2. Related Work

2.1. Object Detection Methods Based on Machine Learning and Image Processing

For pest identification and detection tasks, before the advent of traditional machine learning, manual identification methods were predominantly used, relying on agricultural experts for identification. This method is relatively time-consuming and labor-intensive, and the accuracy of detection depends on the professional level of the inspector. Moreover, an increasing number of researchers have begun exploring the use of machine learning methods for pest detection.

Pest detection based on traditional computer vision can be mainly divided into two stages: feature extraction and classification [19]. Feature extraction utilizes traditional image processing methods to extract features from pest images. Subsequently, machine learning methods are employed to train classifiers using these extracted features, enabling them to distinguish between different types of pests. The accuracy of this method largely depends on the effectiveness of the feature extraction process. Commonly used morphological features in pest identification include area and perimeter. Researchers have also proposed roundness and eccentricity [20], which are more distinctive, for pest classification. The importance of various morphological features varies across different pest identification tasks.

In the feature extraction stage, representative algorithms include scale-invariant feature transform (SIFT) [21], histogram of oriented gradients (HOG) [22], and Haar-like features [23]. In the classification stage, support vector machine (SVM) [24] is introduced into pest classification tasks due to its strong nonlinear classification ability.

Pest detection techniques based on traditional computer vision and machine learning methods typically require researchers to manually design features. However, due to the diversity of objects, lighting conditions, and backgrounds in images, it is challenging to accurately describe various objects using manually designed feature algorithms. Moreover, in pest detection tasks, traditional object detection methods primarily focus on pest identification and often do not include precise pest localization. In summary, traditional object detection methods have certain limitations.

2.2. Object Detection Methods Based on Deep Learning

In recent years, with continuous improvements in computer hardware performance and the ongoing development of deep learning, object detection methods based on deep learning have been rapidly updated and iterated. Object detection methods utilizing convolutional neural networks typically employ deeper architectures compared to traditional methods based on machine learning, enabling them to learn complex features without manual intervention. Broadly speaking, object detection methods based on deep learning can be categorized into two groups: one-stage detection models and two-stage detection models [25].

Representative object detection algorithms based on two stages include R-CNN [13], Fast R-CNN [14], and Faster R-CNN [15]. These algorithms are divided into two stages: candidate region proposal and classification. R-CNN uses the Selective Search method

to pre-extract candidate regions, then applies convolutional neural networks to extract features and classify them, achieving high accuracy but with a slower processing speed. Fast R-CNN introduced the ROI layer, which significantly improved detection speed by pooling corresponding areas in the feature map for subsequent classification and bounding box regression. Faster R-CNN further enhanced efficiency by incorporating Region Proposal Networks (RPNs) to generate candidate regions, reducing computational load and optimizing detection speed.

The two-stage object detection algorithm is known for its high accuracy but is limited by slower speeds. In contrast, one-stage algorithms, like YOLO [26] introduced in 2015, prioritize speed with similar accuracy. SSD [27], also from 2015, improved real-time detection by combining YOLO's grid-based division with Faster R-CNN's anchor strategy. YOLO has since evolved through versions like YOLOv3, YOLOv4 [28], YOLOv7, and YOLOv8. Improved YOLO models are now used in various fields. In autonomous driving, there is a need for efficient real-time detection systems. Traditional models, with their high computational costs, are unsuitable for edge devices. To address this, researchers developed YOLOv8-Lite [29], a lightweight model using FastDet, TFPN pyramid, and CBAM attention mechanisms, which has demonstrated higher accuracy and robustness on NEXET and KITTI datasets. In agriculture, detecting immature apples is crucial for monitoring and yield prediction. An improved YOLOv3 [30] using CSPDarknet53, CIoU regression, and Mosaic algorithm has significantly enhanced the detection of occluded fruits, aiding agricultural automation. The success of deep learning in general object detection has led researchers to apply these methods to agricultural pest detection as well.

2.3. Pest Detection Methods Based on Deep Learning

As traditional machine learning algorithms for pest detection fall short, an increasing number of researchers have turned to deep learning methods, which have excelled in target detection across various fields.

In the realm of pest detection, prior to model training, techniques such as finding suitable anchors and employing data augmentation play pivotal roles in enhancing performance. Mamdouh et al. [31] proposed an olive fruit fly detection framework using YOLOv4, involving re-clustering for new anchors and data augmentation. The experiments on the DIRT dataset demonstrate a precision of 0.84, a recall of 0.97, an F1 score of 0.9, and a mean average precision (mAP) of 96.68%. However, training takes 52.64 h with a 245 MB weight file, unsuitable for embedded devices. In 2021, Lyu et al. [32] proposed a feature fusion SSD model for grain pests. It uses a top–down module to fuse conv4 and conv5 features, K-means clustering for refined bounding boxes, and five data augmentation strategies, yielding 9990 images. Experimental results show 1.44% mAP improvement over SSD, maintaining FPS.

Enhancing feature fusion and refining loss functions are crucial steps in improving pest detection accuracy. Amrani et al. [33] introduced YOLO-AFF, utilizing Adaptive Feature Fusion in YOLOv3 to improve feature integration across different scales. This approach resulted in a 10.28% increase in mAP on the Pest24 dataset, albeit with a minor reduction in FPS. Additionally, Wen et al. [34] proposed Pest-YOLO based on YOLOv4, employing focal loss to mitigate issues arising from uneven distribution in pest datasets. Similarly, Li et al. [35] developed SAFFPest using VFNet, integrating deformable convolutions and attention mechanisms to enhance feature extraction for diverse shapes of rice pests, achieving a 2.9% mAP improvement over VFNet on the IP102 dataset.

Researchers have also explored lightweight approaches to pest detection. Dong et al. [36] enhanced the YOLO-pest model by replacing its YOLOv4 backbone with MobileNetv3 and utilizing the Lite Fusion Feature Pyramid Network. This adaptation resulted in a 2.4% decrease in mAP on the Croppest12 dataset, but achieved a notable 17.2% increase in FPS, while reducing the model parameters by around 200 million. Similarly, Ali et al. [37] improved the Faster R-CNN method with Faster-PestNet, utilizing MobileNet to boost detection speed, achieving an mAP of 82.43% on the IP102 dataset, indicating strong

generalization capabilities across diverse datasets. Additionally, Yang et al. [38] proposed a lightweight rice pest detection method based on YOLOv8, incorporating a fully connected bottleneck Transformer module to enhance detection performance by focusing on local features during downsampling of large images. To address the imbalanced distribution in the IP102 dataset, FastGAN was employed to generate the GPest14 dataset, resulting in a 1.1% mAP increase over YOLOv8n, with a reduction in model weight by 0.5 MB. In 2023, Zhang et al. [39] introduced the C3M-YOLO method based on YOLOv5, aiming to enhance crop pest detection speed. This approach replaced the original C3 module with a lightweight C3M module and integrated a GAM attention mechanism in the Neck part of the network, demonstrating a 1% mAP increase and a 5% FPS improvement compared to YOLOv5 on the IP102 dataset.

Detection of pests in yellow sticky traps presents unique challenges due to the smaller size of the objects compared to typical environments. To enhance model performance under these conditions, Gonccalves et al. [40] proposed incorporating focal loss into the loss function, which improves the model's ability to learn from hard samples. Additionally, they emphasized dataset enhancement and uniform resizing of image sizes to optimize data processing. Meanwhile, Nieuwenhuizen et al. [41] applied the Faster R-CNN method specifically for pest detection in yellow sticky traps. Given the small size of pests in these environments, they utilized a method akin to that suggested by [42], dividing original high-definition images into patches for training. However, this approach can be cumbersome and can lead to overlap between adjacent patches, resulting in redundant feature extraction during model training. As a remedy, adjusting the model's input size to better accommodate the detection of small objects is recommended to streamline the process and enhance overall efficiency. Furthermore, the paper proposes MD-YOLO [43], targeting the challenge of detecting small lepidopteran pests on sticky insect boards. The method enhances YOLO by incorporating DenseNet blocks and an Adaptive Attention Module (AAM) to improve feature extraction and a dual-path feature fusion network for better spatial information integration. These innovations result in higher detection accuracy, with an mAP of 86.2% and an F1 score of 79.1%. In addition, to further enhance the feature representation of small agricultural pests, a method called Cross-Stage Feature Fusion [44] was designed, which significantly improves the representation of small pests during the feature fusion stage. Similarly, the Multi-Level Spatial Pyramid Pooling (MTSPPF) module [45] has been proposed, which efficiently integrates convolution, normalization, and activation to capture multi-scale features, thereby enhancing the extraction of contextual information.

2.4. Methods for Small Object Detection

Previous research has introduced a variety of methods to address the challenges faced in small object detection. For example, a proposed method [46] utilizes the Convolutional Block Attention Module (CBAM) to enhance the model's ability to extract features of small objects. However, this research does not take into account the inherent locality issue of convolution operations and whether the default anchors are suitable for detecting targets. Starting from the raw data, ref. [47] proposed an algorithm that can restore blurry, small faces into high-resolution faces with fine details. Based on this, the performance of small face detection has been significantly improved, but it also leads to the loss of contextual data. To address the challenge of detecting small infrared targets, ref. [48] proposed a deep convolutional network MNET based on a feature fusion strategy, which can effectively integrate semantic information from different levels. Although this method performs excellently in infrared small target detection, excessively increasing the network depth may result in the loss of critical edge details of small targets [49]. Ref. [50] proposed a MobileNet-SSD model with FPN based on the SSD model. This model improves the network's detection performance for small-sized garbage by introducing FPN and focal loss. However, this method does not consider the limitation that the FPN structure may cause information loss when fusing features of non-adjacent layers.

In summary, these methods have several points worthy of further discussion: First, a limitation problem is brought about by convolution operations, which may lead to the network's inability to explore long-distance semantic information. Second, there is an information loss problem of non-adjacent feature layers caused by the FPN structure.

3. Materials and Methods

3.1. YOLOv7-Tiny Model

YOLOv7 is a recently proposed one-stage object detection model that delivers outstanding performance, offering higher precision and faster speed compared to YOLOv5. Structurally, YOLOv7 extensively utilizes ELAN and MP structures, setting it apart from previous models in the YOLO series, and incorporates model re-parameterization into its architecture. YOLOv7-tiny is the lightweight version of YOLOv7. Furthermore, the structure of YOLOv7-tiny can be divided into three components: backbone, neck, and head. The backbone is primarily responsible for extracting multi-level features, the neck integrates high-level and low-level features, and the head outputs the final detection results. From a structural perspective, YOLOv7 comprises several fundamental components, including CBL, MP, Tiny-ELAN, and Tiny-SPP. The structures of Tiny-ELAN and Tiny-SPP are shown in Figure 2. The feature maps P3, P4, and P5 are generated by the backbone and act as inputs to the neck. After feature fusion, the head performs the predictions.

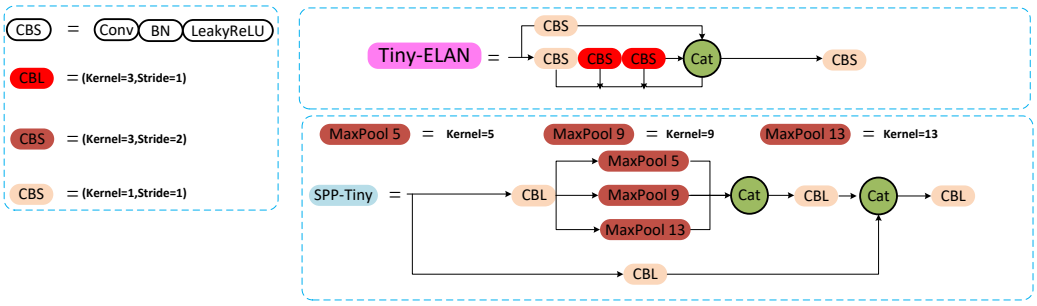


Figure 2. The structures of Tiny-ELAN and Tiny-SPP.

It is important to note that, although the SPPCSPC module proposed by YOLOv7 outperforms the SPPF structure used in YOLOv5 in standard detection tasks, such as those involving the COCO dataset, the deeper convolutional layers of the SPPCSPC module result in the progressive compression and filtering of feature information. Furthermore, in YOLOv7-tiny, the Tiny-SPP structure is used to achieve this; while this layered feature extraction method can capture complex high-level features, it also tends to weaken or lose some detailed information, especially critical features of small objects. Consequently, when dealing with small targets, deep convolutional networks may struggle to extract sufficient and distinctive features, thereby affecting detection accuracy. Our subsequent work will also build upon this foundation.

Additionally, the YOLOv7-tiny network is fundamentally a convolution-based neural network model. However, convolution operations are inherently local, extracting features only within a fixed receptive field. This locality restricts the network's ability to capture global contextual information and long-range dependencies. In complex scenes, small objects often face interference from background clutter and occlusions, making accurate detection difficult when relying solely on local features. Consequently, it is imperative to explore alternative methods to enhance the network's capability to capture global contextual information.

3.2. Improved YOLOv7-Tiny Model Structure

The structure of the improved YOLOv7 is shown in Figure 3 and consists of four parts: the backbone and the CIE (context information extraction) block, neck, and head. Among them, the backbone part remains the same as that of YOLOv7-tiny and is used

to extract features. The CIE is used to extract contextual information. The neck part is implemented through the Tiny-ELAN feature fusion network. Meanwhile, the neck part fuses the features of four parts to ensure the fusion of feature information at various levels. The head part enhances the detection ability of small targets by adding a P2 detection head. Additionally, introducing the SIoU loss function and redefining components with direction information in the bounding box loss function facilitates easier regression of predicted boxes and enhances the model's convergence ability. Our proposed improvement measures and their roles and functions are shown in Table 1.

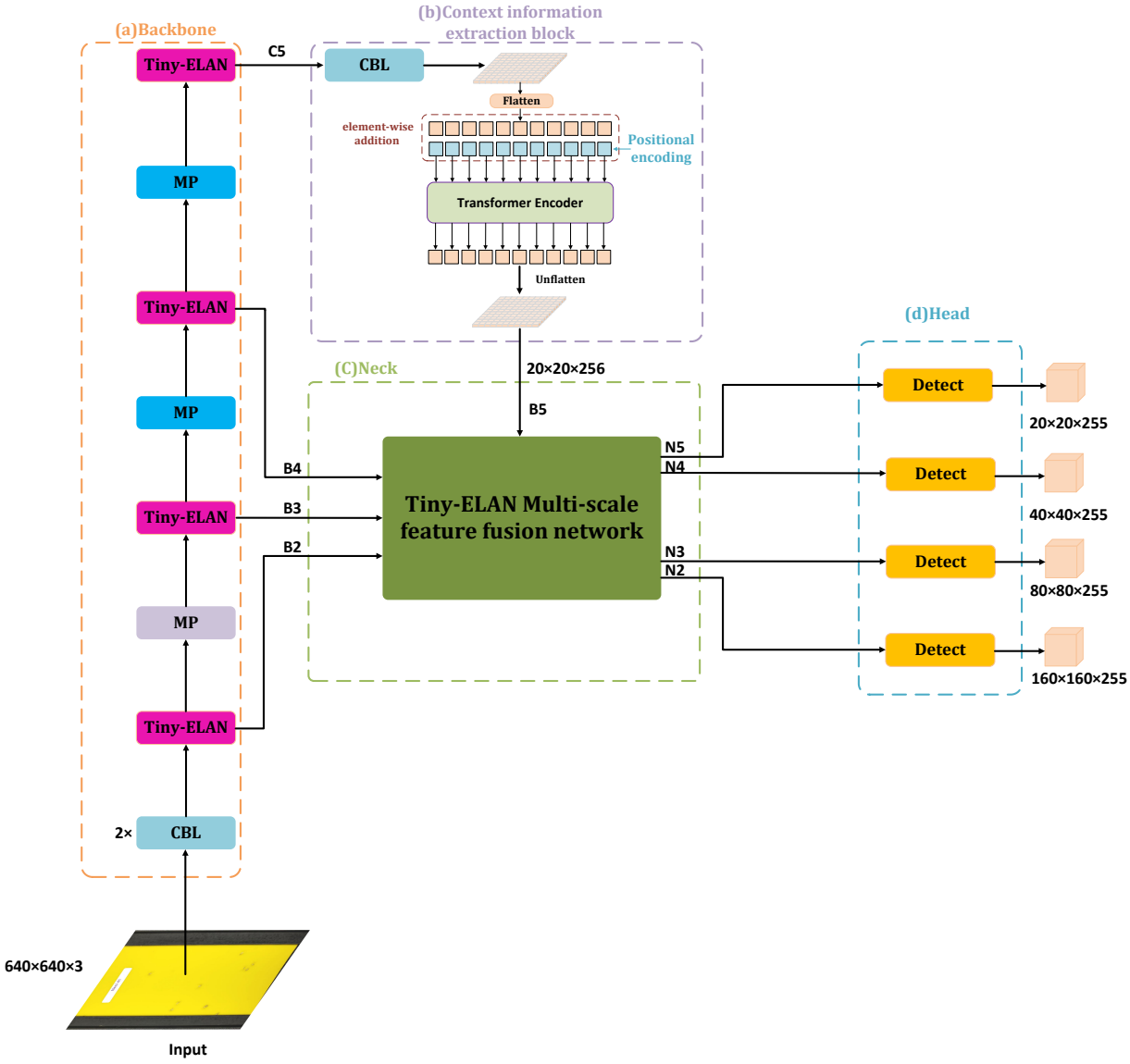


Figure 3. This is structure of the improved YOLOv7-tiny model.

Table 1. Our proposed improvement measures and their roles and functions.

Measures for Improvement	Respective Roles
Tiny-ELAN feature fusion network	Reduce the information loss during the fusion of non-adjacent feature layers
CIE	Extract global semantic information
P2 detection head	Enhance the detection ability for small targets
SIoU loss function	Enhance the model's convergence ability

3.3. Tiny-ELAN Fusion Network

Feature information loss represents a prevalent issue that numerous detection algorithms encounter. Within detection algorithms, distinct branches are tasked with detecting objects of varying scales. At the outset, their feature maps contain a substantial amount of high-level semantic information. Nevertheless, during the process of propagation or fusion, this semantic information gets lost. This implies that the spatial information within the feature maps obtained by the detection heads is incomplete, ultimately exerting an impact on the performance of the algorithms.

A considerable number of algorithms within the YOLO series resort to the traditional FPN-PAN network structure for the purpose of fusing features of different scales, thereby enhancing the accuracy of detection algorithms. The traditional FPN-PAN network is illustrated in Figure 4. This particular structure incorporates multiple branches that are designed to carry out multi-scale feature fusion. However, it is only capable of fully integrating the information sourced from adjacent layers. When it comes to non-adjacent layers, the acquisition of relevant information can only be achieved indirectly and recursively.

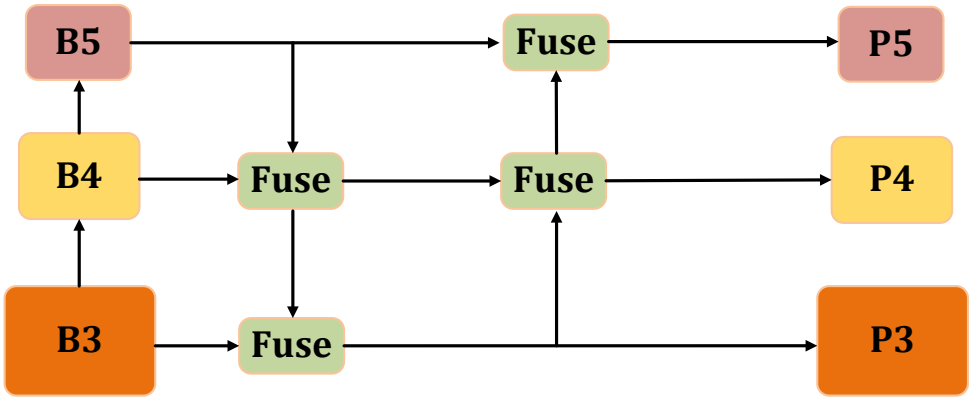


Figure 4. This is the structure of the traditional FPN-PAN network.

For instance, in the case where B3 requires the information of the B4 layer, it is feasible to directly fuse the feature information of the B4 layer by conducting a downsampling operation on the B4 layer. Nevertheless, if B3 intends to obtain the feature information of the B5 layer, it has no alternative but to initially fuse the information of the B4 feature layer and the B5 feature layer to generate an intermediate layer. Subsequently, the feature fusion between this intermediate layer and the B3 feature layer needs to be performed. Evidently, this approach to feature fusion is prone to result in substantial information loss.

Moreover, the traditional FPN-PAN network does not utilize the B2 feature layer in the process of feature fusion, which results in the absence of the P2 detection head in the head part. Furthermore, the P2 detection head (with a size of 160 × 160) can enhance the detection ability of small targets. In addition, the absence of the P2 detection head weakens the network's ability to detect small targets.

In order to prevent the loss of feature information caused by the traditional FPN-PAN structure, a feature fusion mechanism based on the Gather-and-Distribute mechanism [51] was put forward by Huawei. Drawing on this Gather-and-Distribute strategy, our method has proposed a Tiny-ELAN feature fusion network based on the same strategy. Figure 5 presents the Tiny-ELAN fusion work that adopts the Gather-and-Distribute strategy.

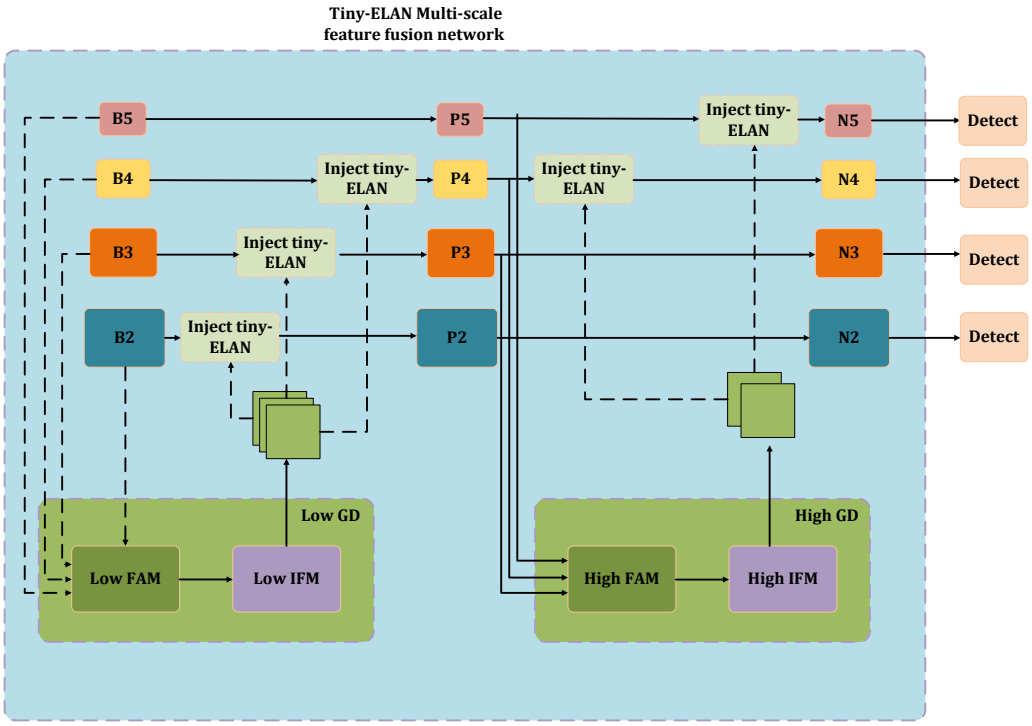


Figure 5. This is the structure of Tiny-ELAN feature fusion network.

More specifically, the Tiny-ELAN feature fusion network using the Gather-and-Distribute mechanism consists of the Feature Alignment Module (FAM), the Information Fusion Module (IFM), and the Tiny-ELAN Injection Module (Inject-Tiny-ELAN). Firstly, the FAM undertakes the task of gathering feature maps of different scales from the backbone and then aligns these feature maps through upsampling or downsampling. Subsequently, the IFM fuses the aligned features to generate global features, which are then divided into two parts for the purpose of distributing them to features of other scales in a targeted manner. Finally, the Inject-Tiny-ELAN module distributes the global features by employing a method similar to the self-attention mechanism, enhancing the detection capabilities of each branch through simple attention operations.

To address the detection of targets of different sizes, the model has specifically designed two branches, namely the Low Gather-and-Distribute (Low-GD) and the High Gather-and-Distribute (High-GD). The structures of Low-GD and High-GD are shown in Figure 6 and Figure 7, respectively. The Low-GD concentrates on extracting and fusing large-sized feature maps, while the High-GD deals with small-sized feature maps, thereby ensuring that the model can effectively conduct target detection at different scales. Moreover, the implementation of the P2 detection head has strengthened the detection ability of small targets.

The size of the B4 feature map is taken as the reference size by the Low-FAM module. As shown in Table 2, these are the parameters of Low-FAM. B2 and B3 utilize the Average Pooling Layer to downsample their feature maps to the size of B4. Moreover, the B5 feature map is scaled up to the size of B4 through bilinear interpolation and then concatenated in the channel dimension. The Low-IFM module is designed to include convolution, the Structural Re-parameterization RepConv Block, and split operations. The RepConv Block takes the output processed by the Low-FAM module as input and splits the output into three outputs in the channel dimension to facilitate the fusion of information at different levels. The formula is expressed as follows:

$$F_{align} = LowFAM([B2, B3, B4, B5]) \quad (1)$$

$$F_{fuse} = RepConv(F_{align}) \quad (2)$$

$$P4_{Inject}, P3_{Inject}, P2_{Inject} = Split(F_{fuse}) \quad (3)$$

The High-FAM module uses the size of the P5 feature map as the basic size. In this process, P3 and P4 make use of the AvgPool layer to downsample their feature maps to the P5 size and then perform splicing on them in the channel dimension. The High-IFM design involves multiple self-attention, MLP, and Split operations. The specific formulas are listed as follows:

$$F_{align} = HignFAM([P3, P4, P5]) \quad (4)$$

$$F_{fuse} = Transformer(F_{align}) \quad (5)$$

$$N4_{Inject}, N5_{Inject} = Split(Conv(F_{fuse})) \quad (6)$$

Table 2. The parameters of the Low-FAM structure.

Layer Index	Layer	Input Index	Output Size	Output Channel
0	Avgpool	[B2, B3]	[40 × 40, 40 × 40]	[64, 128]
1	Bilinear	B5	40 × 40	256
2	Cat	[0, 1, B4]	40 × 40	704

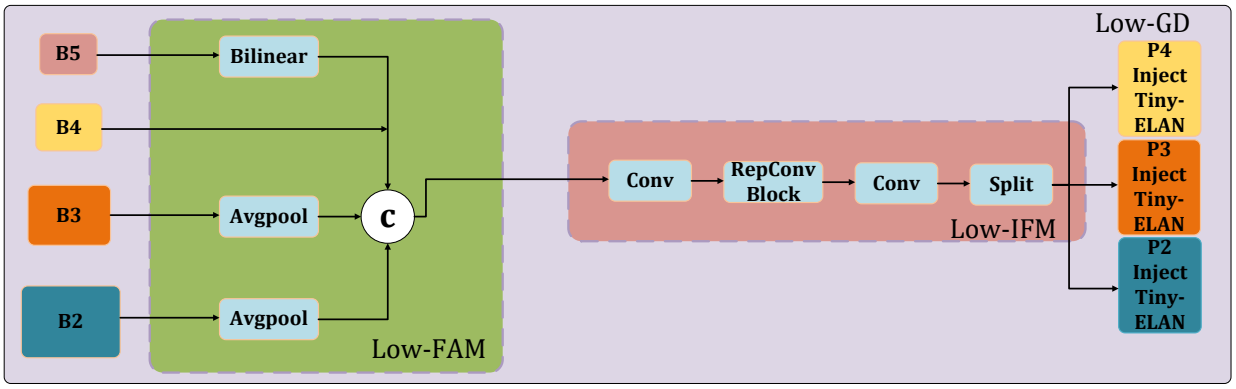


Figure 6. This is the structure of Low-GD.

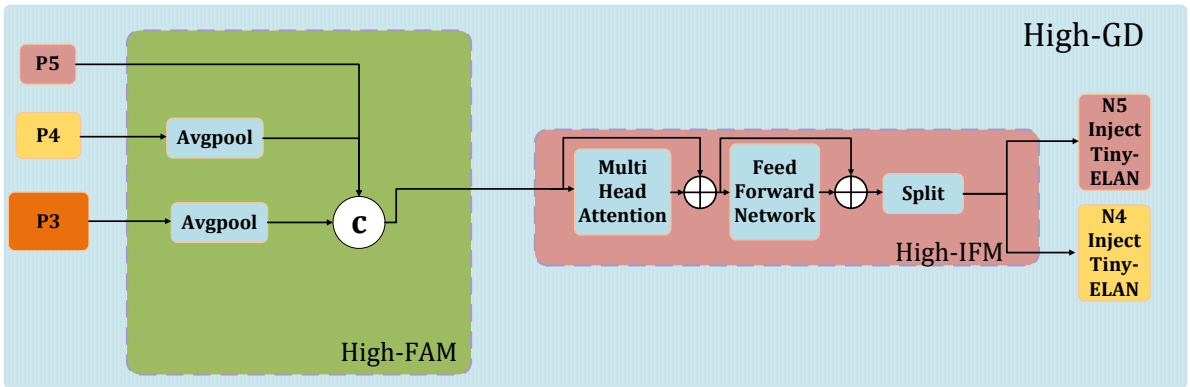


Figure 7. This is the structure of High-GD.

The network structure diagram of the Inject Tiny-ELAN module is shown in Figure 8. It is divided into High-Inject Tiny-ELAN and Low-Inject Tiny-ELAN. In this structure, X_{Global} refers to the global information generated by the Information Fusion Module (IFM module) in Figure 8, which is denoted as F_{inj} in the subsequent mathematical formulas. Meanwhile, the local information of different levels is represented by X_{Local}, and is denoted as F_{local} in the subsequent mathematical formulas.

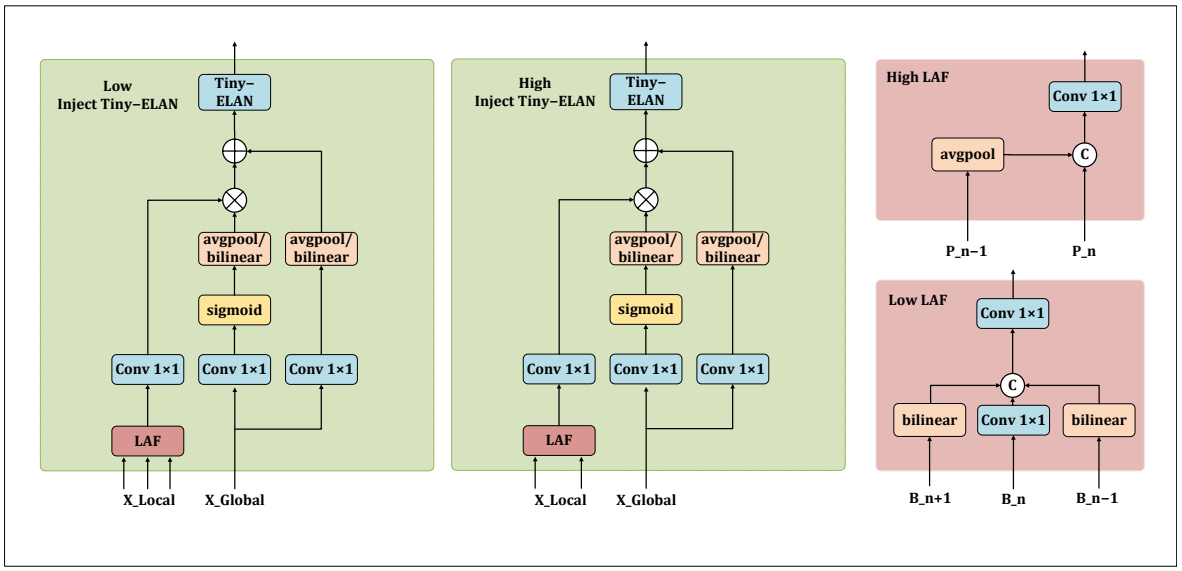


Figure 8. This is structure of Inject Tiny-ELAN.

We perform calculations using two different convolutional operations (Convs) with F_{inj}, obtaining $F_{\text{global\_embed}}$ and F_{act}. $F_{\text{local\_embed}}$ is calculated from F_{local} using a Conv operation. The fused feature F_{out} is computed through an attention mechanism. Because there are size differences between F_{local} and F_{global}, we use average pooling or bilinear interpolation to resize $F_{\text{global\_embed}}$ and F_{act} to match the size of F_{inj} for proper alignment. Finally, after each attention-based fusion process, we add a Tiny-ELAN to further extract and fuse the information. In Low-Inject Tiny-ELAN, since F_{local} is equal to B_i, the formula is presented as follows:

$$F_{\text{global\_act\_}P_i} = \text{resize}(\text{Sigmoid}(\text{Conv}_{\text{act}}(P_i\_Inject))), \tag{7}$$

$$F_{\text{global\_embed\_}P_i} = \text{resize}(\text{Conv}_{\text{global\_embed\_}P_i}(P_i\_Inject)), \tag{8}$$

$$F_{\text{att\_fuse\_}P_i} = \text{Conv}_{\text{local\_embed\_}P_i}(B_i) * F_{\text{inj\_act\_}P_i} + F_{\text{global\_embed\_}P_i}, \tag{9}$$

$$P_i = \text{Tiny-ELAN}(F_{\text{att\_fuse\_}P_i}). \tag{10}$$

The High-Inject Tiny-ELAN is identical to that in Low-Inject Tiny-ELAN. In the high-stage, F_{local} equals P_i; thus, the formula is as follows:

$$F_{\text{global\_act\_}N_i} = \text{resize}(\text{Sigmoid}(\text{Conv}_{\text{act}}(N_i\_Inject))), \tag{11}$$

$$F_{\text{global\_embed\_}N_i} = \text{resize}(\text{Conv}_{\text{global\_embed\_}N_i}(N_i\_Inject)), \tag{12}$$

$$F_{\text{att\_fuse\_}N_i} = \text{Conv}_{\text{local\_embed\_}N_i}(P_i) * F_{\text{inj\_act\_}N_i} + F_{\text{global\_embed\_}N_i}, \tag{13}$$

$$N_i = \text{Tiny-ELAN}(F_{\text{att\_fuse\_}N_i}). \tag{14}$$

By means of Low-Level Attention Fusion (Low LAF) and High-Level Attention Fusion (High LAF), the receptive field of the Inject-Tiny-ELAN module can be further expanded; thus, the ability of this module in feature representation will be enhanced accordingly.

Specifically, various operations such as convolution, sampling, and interpolation need to be performed on the local information and the global information. Then, the different information processed by these operations will be spliced together in the channel dimension. After completing the above steps, the corresponding calculations will be carried out via the Tiny-ELAN module, and thus the final output of the information injection module can be obtained.

All in all, we propose a Tiny-ELAN fusion network which is based on the Gather-and-Distribute mechanism and equipped with a P2 detection head, aiming to strengthen the feature fusion capacity of non-adjacent layers.

3.4. Context Information Extraction (CIE) Block Based on Transformer Encoder

For the task of recognizing small-sized pests, obtaining rich contextual and global information will be beneficial in mitigating confusion between different pest species and minimizing interference from complex backgrounds. However, the locality of convolution operations and excessive pooling operations challenges for further capturing rich contextual information and detecting small objects.

The locality of convolution operations: For deep convolutional networks, the convolution operation is inherently a local operation [52], capable of extracting features within a fixed receptive field. This locality limits the ability of convolutional networks to capture global contextual information and long-range dependencies. In complex scenarios, small objects are often affected by background, occlusion, and other factors, making it difficult to achieve accurate detection solely based on local features. Therefore, the inherent limitations of deep convolutional networks in this regard also affect the effectiveness of small object detection. Conversely, by dividing the feature map generated by the convolutional neural network into multiple patches and processing them with a Transformer, it is possible to effectively leverage different patches to capture global contextual information via multi-head self-attention.

Detail loss due to pooling operations: In the Tiny-SPP module used in YOLOv7-tiny, excessive pooling operations can significantly hinder the detection of small objects. Pooling reduces the spatial dimensions of feature maps, which, while beneficial for computational efficiency, can lead to the loss of critical fine-grained details [53]. For small object detection, these fine-grained details are crucial as they provide the necessary information to distinguish and accurately locate the objects. As a result, an over-reliance on pooling operations may strip away essential features, thereby reducing the model's ability to detect small targets effectively.

Recognizing the limitations of convolutional networks and inspired by Vision Transformer (ViT), we propose a context information extraction structure based on the Transformer encoder to replace the original Tiny-SPP module in YOLOv7-tiny, opting to abandon deeper convolutional layers and pooling operations in favor of incorporating a Transformer encoder to obtain more comprehensive global and contextual information. The context information extraction block is straightforward in design: it embeds the Transformer encoder after the CBL module to achieve global self-attention across a 2D feature map.

We intend to provide an overview of the context information extraction block before detailing the multi-head attention calculation in the Transformer encoder. The context information extraction block primarily consists of the CBL module and a Transformer encoder based on multi-head attention. Figure 9 illustrates the context information extraction block and the multi-head attention module, along with the shapes of the data at each stage.

In the context information extraction block, the feature map C5, output by the backbone, is processed by the CBL module to adjust the number of channels, resulting in the S1 feature map. The specific location of CBL in the context information extraction module is shown in part (a) of Figure 9. The detailed parameters of CIE are presented in Table 3. The motivation for adding the CBL module before the Transformer encoder is to reduce the computational load for the subsequent multi-head attention operations. The CBL module consists of convolution with kernel size 1 and stride 1, followed by batch normalization and SiLU activation.

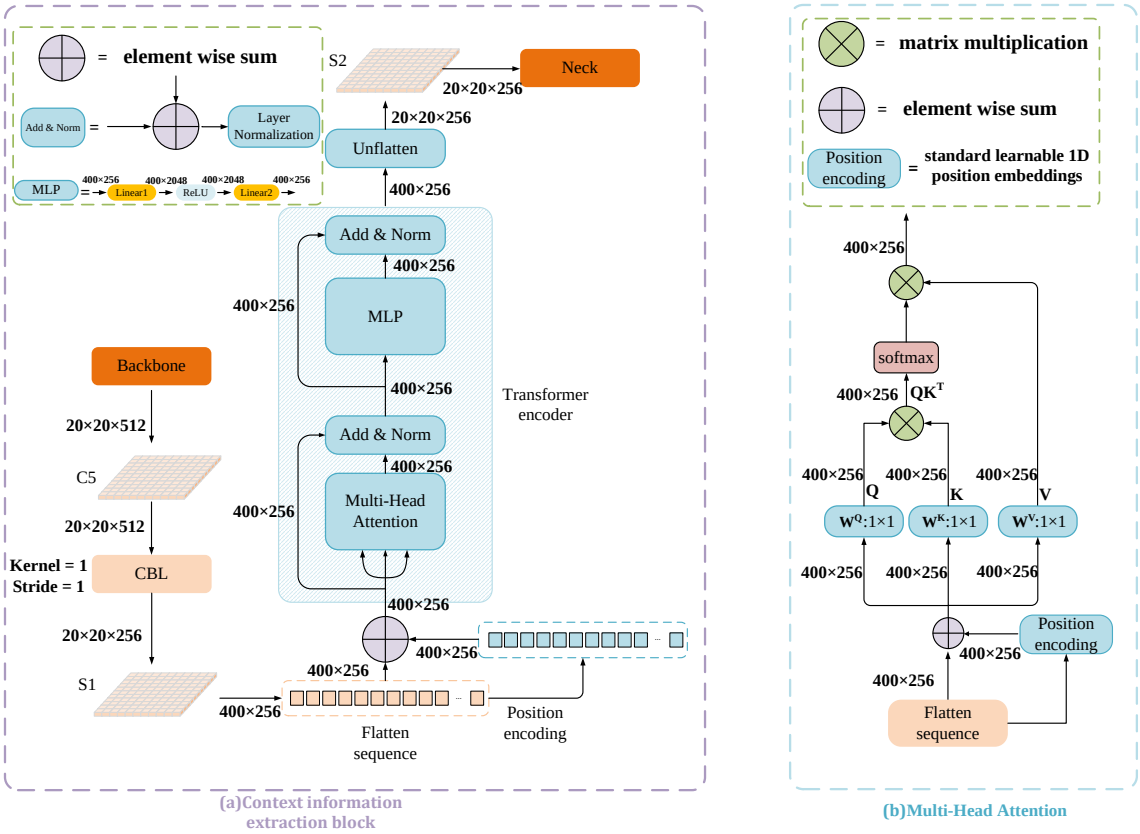


Figure 9. This illustrates the (**a**) context information extraction block and (**b**) multi-head attention. In (**a**), learnable 1D positional embeddings, element-wise addition, and layer normalization are used. In (**b**), 8 heads are employed for multi-head attention, but only one is shown. Symbols $\oplus$ and $\otimes$ denote element-wise sum and matrix multiplication, respectively, with 1×1 indicating pointwise convolution.

Table 3. The parameters of the CIE structure.

Layer Index	Layer	Input Index	Output Size	Output Channel
0	CBL	-	20×20	256
1	Reshape	0	400×256	1
2	Position encoding	1	400×256	1
3	Cat	[1, 2]	400×256	1
4	Transformer encoder	3	400×256	1
5	Unflatten	4	20×20	256

From part (a) of Figure 9, we can see that the S1 feature map, after being flattened, does not inherently contain positional information. To enable the subsequent Transformer encoder to better understand the positional information of the sequence elements, we introduced positional encoding and added it to the input flattened sequence. Position encoding for the flattened sequence is implemented using standard learnable 1D position embeddings with a linear layer to retain positional information. The standard 1D position encoding formula can be expressed as,

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \tag{15}$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \tag{16}$$

where $\mathrm{PE}(pos, 2i)$ represents the encoding at position pos in the $2i$-th dimension. $\mathrm{PE}(pos, 2i+1)$ represents the encoding at position pos in the $2i+1$-th dimension. pos is the position index, with a range of $0, 1, 2, \ldots, L-1$. i is the dimension index, with a range of $0, 1, 2, \ldots, \frac{d_{model}}{2} - 1$. d_{model} is the embedding dimension. In this work, d_{model} is 256 and the sequence length is 400.

A self-attention mechanism can capture the long-range dependency relationships in sequences, which is an important advantage compared to convolutional neural networks (CNNs). For example, in a serialized picture, an insect may have a strong semantic association with a certain surrounding environment, and the self-attention mechanism in Transformer encoder can capture this relationship very well. In the Transformer encoder, positional encoding and multi-head attention are crucial for capturing long-range semantic relationships and contextual information. Therefore, we will provide a detailed discussion of these components in the following sections. To more comprehensively capture the semantic relationships and dependencies between different positions in the input sequence, multi-head attention is designed to process the sequence. Multi-head attention computes multiple attention heads in parallel, allowing each head to independently learn and focus on different parts of the sequence. This approach helps in understanding the relationships between features and determining their relative importance. The schematic diagram of multi-head attention is shown in Figure 9. It is worth noting that, for simplicity, only a single attention head is depicted in the diagram. However, in practice, we use 8 heads. Subsequently, we will provide a detailed explanation of how the flattened sequence is processed in multi-head attention.

For a given input sequence $X \in \mathbb{R}^{n \times d_{\text{model}}}$ (where n is the sequence length and d_{model} is the dimension of the model), we compute linear projections to obtain queries (Q), keys (K), and values (V) for each head. In the self-attention mechanism, Q (Query), K (Key), and V (Value) are of great importance. Query (the Query vector) is like an inquiring tool, which is used to interact with the Key vectors of other positions to find out the relevant parts related to itself. For example, in text processing, it can explore the associations among words in a sentence. Key (the Key vector) is calculated by matching with the Query vector to generate attention weights. It has the same dimension as the Query and measures the relevance between the information of different positions and the current position. Value (the Value vector) contains the actual feature information of each position. The attention weights are weighted and summed with the Value vector to update the representation of the current position, thus integrating the relevant semantics of other positions and enabling the model to handle sequence information better. The calculation process of Q, K, and V is as follows:

$$Q_h = XW_h^Q, \quad K_h = XW_h^K, \tag{17}$$

$$V_h = XW_h^V \tag{18}$$

where $W_h^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_h^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_h^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$. h represents the number of heads in multi-head attention. They are all learnable parameters.

For each head, we compute the attention scores using the scaled dot–product attention mechanism.

$$\mathrm{Attention}(Q_h, K_h, V_h) = \mathrm{softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d_k}}\right) V_h \tag{19}$$

The outputs from each head are concatenated to form the final output.

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h) W^O \tag{20}$$

where $\mathrm{head}_i = \mathrm{Attention}(Q_i, K_i, V_i)$, $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Combining all the steps, the multi-head attention is computed as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{Attention}_1, \ldots, \text{Attention}_h)W^O$$

Multi-head attention, through parallel computation, enables better capture of semantic relationships across different positions, thereby enhancing model performance.

In the output of multi-head attention, an add operation is performed by adding the sub-layer input to the sub-layer output, followed by layer normalization (Norm). This approach helps mitigate the vanishing gradient problem in deep neural networks and accelerates the training process. In the Transformer encoder, the add and Norm operations can be represented as follows:

$$Add(x_i, y_i) = x_i + y_i \tag{21}$$

where x_i and y_i are the vectors to be added, and z_i is the result of the addition.

$$\text{Norm}(z_i) = \frac{z_i - \mu}{\sigma} \cdot \gamma + \beta \tag{22}$$

where μ is the mean of z_i,

$$\mu = \frac{1}{n}\sum_{j=1}^{n} z_{ij}, \tag{23}$$

σ is the standard deviation of z_i,

$$\sigma = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(z_{ij} - \mu)^2}, \tag{24}$$

γ and β are trainable scale and shift parameters.

Notably, the shape of the Transformer encoder's output is the same as the input sequence's shape, both being 400 × 256. Subsequently, the S2 feature map, which has completed the extraction of contextual information, is derived from the unflattened output of the Transformer encoder. Its shape matches that of the S1 feature map, both being 20 × 20 × 256.

3.5. Loss Function

Similar to YOLOv5, the loss function of YOLOv7 also includes three parts: the bounding box loss function, the objectness loss function, and the classification loss function. The total loss is the sum of the losses of each part multiplied by their respective weights. Binary cross-entropy loss is used to calculate the objectness loss and classification loss. The loss function used in the the bounding box loss function is CIoU loss function. CIoU loss takes into account three factors: overlap area, distance and aspect ratio, as illustrated in Figure 10. In Figure 10, rectangle B represents the predicted box, and rectangle B^{GT} represents the ground truth box; d represents the distance between the center of the predicted box and the center of the ground truth box; c represents the diagonal length of the smallest enclosing rectangle that covers both the prediction box B and the ground truth box B^{GT}. The CIoU loss can be can be defined as,

$$Loss_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v, \tag{25}$$

where b and b^{gt} represent the center point of the prediction box and the center point of the ground truth box, respectively. Where ρ represents the Euclidean distance and c represents the diagonal length of the minimum rectangle containing the true box and the predicted box. αv is used to describe the consistency of the aspect ratio between the prediction box and the ground truth box. v is defined as,

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctam\frac{w}{h})^2, \tag{26}$$

where w^{gt} and h^{gt} are the width and height of the ground truth box, respectively, and w and h are the width and height of the prediction box, respectively. α is used to control the proportion of aspect ratio in CIoU loss, which is defined as

$$\alpha = \begin{cases} 0, & if \quad IoU < 0.5, \\ \dfrac{v}{(1 - IoU) + v}, & if \quad IoU \geq 0.5. \end{cases} \tag{27}$$

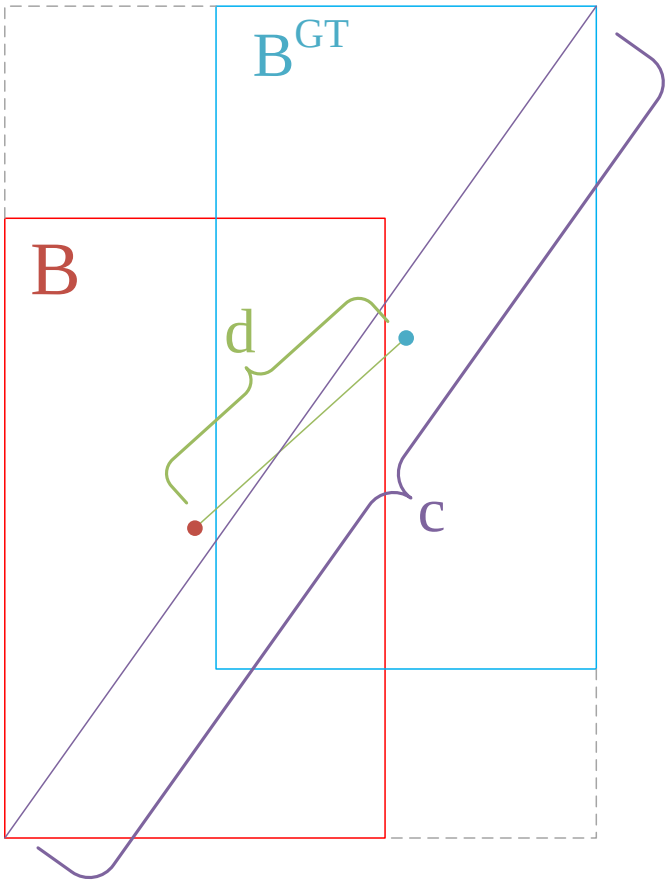


Figure 10. The schematic diagram of CIoU. Rectangle B represents the predicted box, and rectangle B^{GT} represents the ground truth box; d represents the distance between the center of the predicted box and the center of the ground truth box; c represents the diagonal length of the smallest enclosing rectangle that covers both the prediction box and the ground truth box.

Although CIoU takes into account overlapping area, distance, and aspect ratio, it does not take into account the direction of mismatch between the real box and the prediction box, which will affect the convergence speed during training and finally obtain a model with unsatisfactory performance [54]. In order to make the proposed method better for small pest detection, we replace the CIoU loss function with SIoU. The SIoU loss function takes into account four aspects: angle, distance, shape, and overlapping area. The diagram illustrating SIoU is depicted in Figure 11, where rectangle B represents the predicted box, and rectangle B^{GT} represents the ground truth box. σ is the diagonal formed by the connection of the two center points of the prediction box B and the ground truth box B^{GT}. c_h and c_w represent the height and width of the diagonal rectangle. α and β represent the angles between diagonal σ to c_w and c_h, respectively. W and H are the width and height

of the minimum circumscribed rectangle of the ground truth box and the prediction box, respectively.

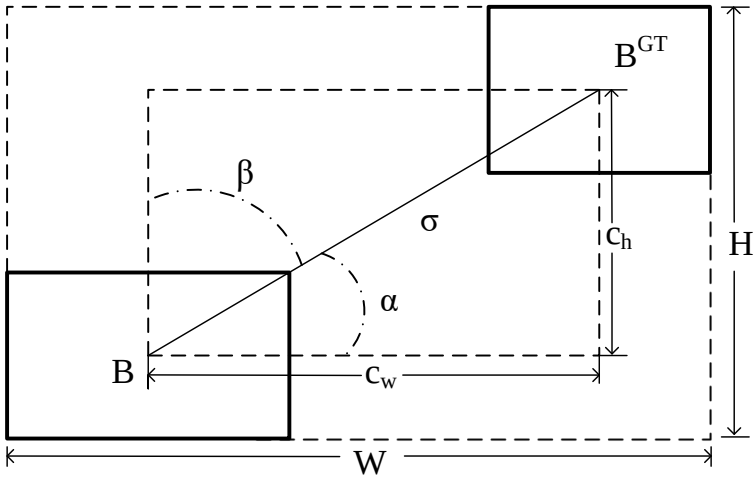


Figure 11. The schematic diagram of the SIoU. Rectangle B represents the predicted box, and rectangle B^{GT} represents the ground truth box. σ is the diagonal formed by the connection of the two center points of the prediction box B and the ground truth box B^{GT}. c_h and c_w represent the height and width of the diagonal rectangle. α and β represent the angles between diagonal σ to c_w and c_h, respectively. W and H are the width and height of the minimum circumscribed rectangle of the ground truth box and the prediction box, respectively.

3.5.1. Angle Cost

In order to make the prediction box converge to the ground truth box faster, SIoU loss considers the angle between the prediction box and the ground truth box, and takes the prediction box as far as possible to the X axis or Y axis, and then further closer to the ground truth box. So angle cost is proposed, which is defined as follows:

$$\Lambda = 1 - 2 \times sin^2(arcsin(x) - \frac{\pi}{4}), \tag{28}$$

where

$$x = \frac{c_h}{\sigma} = sin(\alpha), \tag{29}$$

$$\sigma = \sqrt{(b_{c_x}^{gt} - b_{c_x})^2 + (b_{c_y}^{gt} - b_{c_y})^2}, \tag{30}$$

$$c_h = max(b_{c_y}^{gt}, b_{c_y}) - min(b_{c_y}^{gt}, b_{c_y}). \tag{31}$$

Furthermore, σ is the diagonal formed by the connection of the two center points of the prediction box B and the ground truth box B^{GT}. c_h represents the height of the diagonal rectangle. α represent the diagonal angle to c_w. $b_{c_x}^{gt}$ and b_{c_x} represent the abscissa of the center points of the prediction box B and the real box B^{GT}, respectively. $b_{c_y}^{gt}$ and b_{c_y} represent the ordinates of the center points of the prediction box B and the real box B^{GT}, respectively.

3.5.2. Distance Cost

$$\Delta = \sum_{t=x,y} (1 - e^{-\gamma \rho_t}) \tag{32}$$

$$\rho_x = (\frac{b_{c_x}^{gt} - b_{c_x}}{W})^2 \tag{33}$$

$$\rho_y = (\frac{b_{c_y}^{gt} - b_{c_y}}{H})^2 \tag{34}$$

$$\gamma = 2 - \Lambda \tag{35}$$

where W and H are the width and height of the minimum circumscribed rectangle of the ground truth box and the prediction box, respectively. As $\alpha \to 0$, the influence of the distance cost decreases significantly. Conversely, as $\alpha \to \frac{\pi}{4}$, the contribution of distance cost increases. The challenge intensifies with larger angles. Therefore, γ assigns higher importance to distance values as the angle increases.

3.5.3. Shape Cost

$$\Omega = \sum_{t=w,h} (1 - e^{-\omega_t})^\theta \tag{36}$$

$$\omega_w = \frac{|w - w^{gt}|}{max(w, w^{gt})} \tag{37}$$

$$\omega_h = \frac{|h - h^{gt}|}{max(h, h^{gt})} \tag{38}$$

where w and h are the width and height of the prediction box, and w^{gt} and h^{gt} are the width and height of the ground truth box, respectively. θ is the weight parameter controlling the shape loss, when it is set to 1, shape differences dominate the regression, potentially hindering consideration of other factors.

3.5.4. IoU Cost

$$IoU = \frac{B \bigcap B^{gt}}{B \bigcup B^{gt}} \tag{39}$$

3.5.5. SIoU Loss

$$L_{box} = 1 - IoU + \frac{\Delta + \Omega}{2} \tag{40}$$

4. Experimental Results and Analysis

4.1. Dataset

The dataset used for the experiments in this paper is from [17] and contains 284 images taken by Scoutbox at a resolution of 5184 × 3456. The original dataset contained a total of 7413 labeled insects in yellow sticky traps, belonging to three classes, whitefly (WF), macrolophus (MR), and nesidiocoris (NC). Later, [55] found that some of the insects in the dataset were not labeled, so the dataset was corrected, and the revised dataset contained 8114 labeled insects. Therefore, this paper uses a corrected version. For follow-up experiments, we randomly divided the dataset into a training set, a validation set, and a test set at a ratio of 8:1:1. The training set contains 228 images, the validation set contains 28 images, and the test set contains 28 images. Table 4 shows the distribution of the divided datasets. Before conducting the training, we segmented all the high-resolution pictures. Each picture was divided into 3 × 3 smaller pictures, as shown in Figure 12. Eventually, a total of 2556 pictures were obtained, with 2052 pictures in the training set, 252 pictures in the validation set, and 252 pictures in the test set. This processing method is conducive to the subsequent training.

Table 4. Dataset distribution of annotations by species.

Categories	Number of Instances		
	Training Set	Validation Set	Test Set
MR	1250	195	174
NC	518	94	76
WF	4838	527	442
Total	6606	816	692

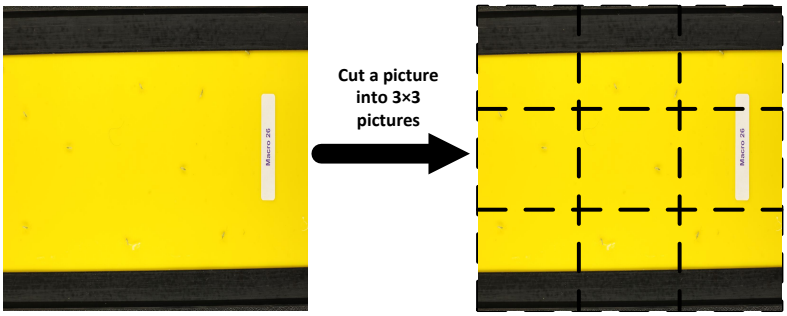


Figure 12. Each picture was divided into 3 × 3 smaller pictures.

4.2. Environment and Parameters

Table 5 shows the environmental parameters of this experiment. The operating system is Windows 11, the CPU is AMD Ryzen 5 5500, the GPU is NVIDIA GeForce RTX 3060, the RAM is 24 GB, the CUDA version is 11.8, and the Python version is 3.8.

Table 5. Experimental environment configuration.

Categories	Configuration Parameter
Operating System	Windows 11
CPU	AMD Ryzen 5 5500
GPU	NVIDIA GeForce RTX 3060
RAM	24 GB
CUDA	11.8
Python	3.8

Table 6 shows the configuration parameters for this experiment. The batch size is set to 4. lr0 is the initial learning rate, set to 0.0025, and lrf is the coefficient that controls the final learning rate, set to 0.1. Since the pretraining weights are not used in this experiment, we set the epoch to 500 to ensure that the network can converge sufficiently. At the beginning of the training, the warm-up method was used to improve the effect of model training, and the warm-up epoch was set to 3.0. During training, the optimizer uses SGD and momentum is set to 0.937.

Table 6. Experimental parameter settings.

Parameter	Value
Bacthsize	4
Lr0	0.0025
Lrf	0.1
Epoch	500
Optimizer	SGD
Momentum	0.937
Warm-up epochs	3.0

4.3. Evaluation

In order to evaluate the performance of our method, mean average precision (mAP), and frame per second (FPS) were used as evaluation indicators for this experiment.

In order to calculate mAP, the precision and recall need to be calculated first. *Precision* represents the proportion of samples that are actually positive among the samples that are predicted to be positive. Its calculation formula can be expressed as,

$$Precision = \frac{TP}{TP + FP} \tag{41}$$

where TP is the abbreviation of true positive, which means that the classifier prediction result is a positive sample, and the actual sample is also a positive sample (that is, the number of positive samples that are correctly identified). FP is false positive, which means that the classifier prediction result is a positive sample, but the actual result is a negative sample (that is, the number of negative samples that are false positives).

Recall represents the proportion of samples predicted as positive samples to actual positive samples. The calculation formula can be expressed as,

$$Recall = \frac{TP}{TP + FN} \tag{42}$$

where FN is the abbreviation of false negative, which means that the classifier prediction result is a negative sample, and the actual result is a positive sample (that is, the number of missed positive samples).

To evaluate the effectiveness of a model, two indicators, precision and recall, must be considered simultaneously. If both precision and recall are relatively high, it can be proven that the model has better performance. However, as the threshold changes, precision and recall usually wax and wane. When precision increases, recall decreases, and vice versa. Therefore, the precision–recall curve (PR curve) is proposed, which can reflect the precision and recall of the model at the same time. The abscissa and ordinate of each point on the PR curve reflect precision and recall under different thresholds, respectively. The area enclosed by the PR curve and the two coordinate axes is called average precision (AP). Its calculation formula can be expressed as,

$$AP = \int_0^1 p(r)dr \tag{43}$$

After calculating the AP of each class, the average is mAP. Its calculation formula can be expressed as,

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N} \tag{44}$$

where N represents the total number of classes.

In addition, in order to evaluate the detection speed of different models, this experiment uses FPS as an indicator to evaluate model speed. FPS indicates how many images the model can process per second. Its calculation formula can be expressed as,

$$FPS = \frac{1}{t} \tag{45}$$

where t represents the time it takes for the model to detect an image.

4.4. Ablation Experiments

To verify the effectiveness of each improvement in the proposed method, we conducted ablation experiments on the dataset. The training set and the validation set were used for model training, while the test set was used to evaluate the performance of the model. In the ablation experiments, we implemented three methods: Method 1 used YOLO-tiny combined with SIoU; Method 2 combined the SIoU loss function and the CIE module on the basis of YOLOv7-tiny; Method 3 introduced the Tiny-ELAN feature fusion structure of the

FPN (Feature Pyramid Network) based on Gather-and-Distribute on the basis of Method 2. The method proposed in this paper introduced the P2 detection head on the basis of Method 3. In the same experimental environment, we compared the performance of YOLOv7-tiny, Method 1, Method 2, Method 3, and the method we proposed. The experimental results are shown in Table 7.

According to Table 7, compared with YOLOv7-tiny, Method 1 increased the mAP metric by 0.2% and reduced the inference time by 2.8 milliseconds. This indicates that the introduction of SIoU has improved the detection accuracy. Compared with Method 1, Method 2 improved the mAP metric by 2.5% and the recall rate by 8%, but the inference time increased by 2 milliseconds. This shows that, after introducing the CIE module, the ability to extract the contextual information of the features in the C5 layer is enhanced, thus improving the detection accuracy. Compared with Method 2, Method 3 improved the recall rate metric by 2%, increased the mAP by 1.1%, and the inference time increased by 0.2 milliseconds, while increasing 3.4 GFLOPs. This indicates that the Tiny-ELAN feature fusion network strengthens the network's feature fusion ability and further improves the accuracy. These results demonstrate that Method 1, Method 2, Method 3, and our method are all superior to the YOLOv7-tiny method in terms of detection accuracy.

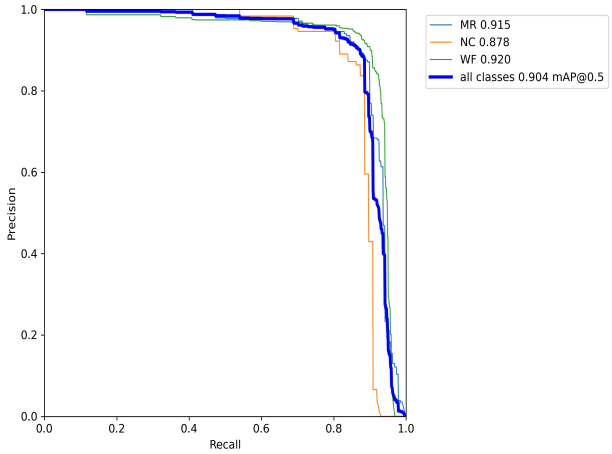
It is worth noting that, compared with Method 3, the recall of our method has decreased by 0.3%. We believe this might be due to the fact that after adding the P2 detection head, the judgment criteria of the model for positive and negative samples of small targets have been changed. Compared with Method 3, our method is more likely to classify some difficult-to-distinguish small target samples as negative samples in order to reduce false positives and improve precision. As a result, the probability of misjudging the real positive small target samples has increased, leading to a decline in recall. For example, for some small targets with blurred boundaries, the P2 detection head may, due to its greater focus on the internal detailed features of small targets, classify those small targets with incomplete boundary parts as background (negative samples), thus causing the recall to decrease.

Regarding the number of parameters, since Method 1 did not change the model structure compared with YOLOv7-tiny, its number of parameters is the same as that of YOLOv7-tiny. Similarly, the GFLOPs also remained unchanged. In Method 2, by replacing the Tiny-SPP structure with the CIE module, the number of parameters was reduced by 0.4 M, and the GFLOPs parameter was reduced by 0.3 GFLOPs. In Method 3, after replacing the original PANet structure with the Tiny-ELAN-based feature fusion network, the number of parameters increased by 3.4 M compared with Method 2, and at the same time, the GFLOPs also increased by 3.4 GFLOPs. In our proposed method, compared with Method 3, the features of the P2 feature layer are further fused in the Tiny-ELAN feature fusion network, which makes the GFLOPs increase by 1.4 GFLOPs.

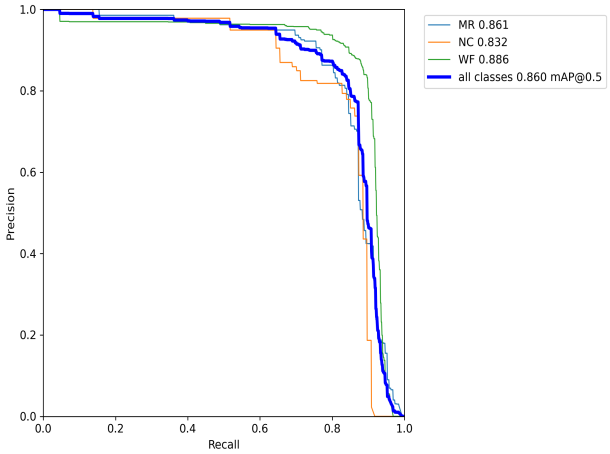
Table 7. Results of ablation experiments. ✓ represents the application of this method, while - represents the non-application of this method.

Methods	SIoU	CIE	GD-FPN	P2	Recall	mAP	Infer Time	Parameters	GFLOPs
YOLOv7-tiny	-	-	-	-	0.839	0.86	12.8 ms	6 M	13
Method 1	✓	-	-	-	0.769	0.862	10 ms	6 M	13
Method 2	✓	✓	-	-	0.849	0.887	12 ms	5.6 M	12.7
Method 3	✓	✓	✓	-	0.869	0.898	12.2 ms	9 M	16.1
Our method	✓	✓	✓	✓	0.866	**0.904**	13.3 ms	9 M	17.5

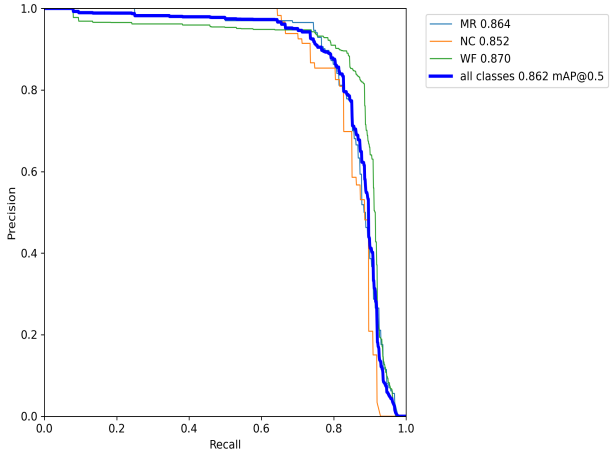
The P-R curves drawn by different methods are shown in Figure 13. The larger the area enclosed by the P-R curve and the two coordinate axes, the better the detection accuracy of the method. We can find that our method has the largest area enclosed by the P-R curve and the coordinate axes.



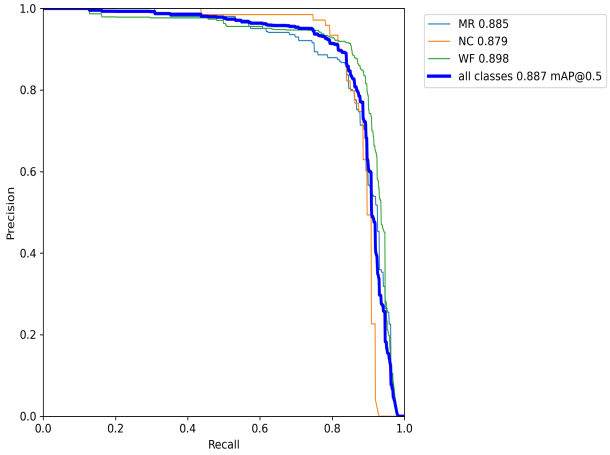
(**a**) Our method.



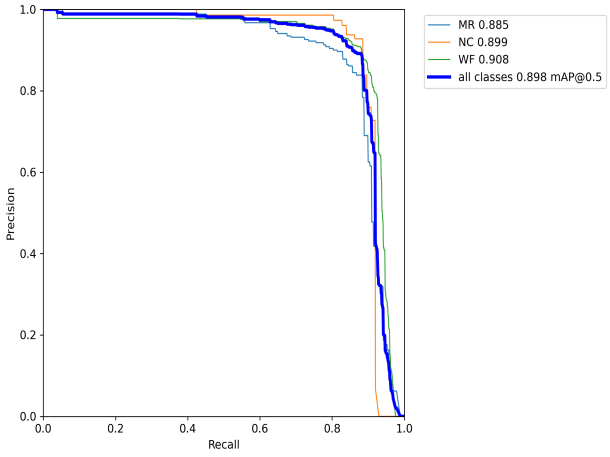
(**b**) YOLOv7-tiny



(**c**) Method 1



(**d**) Method 2

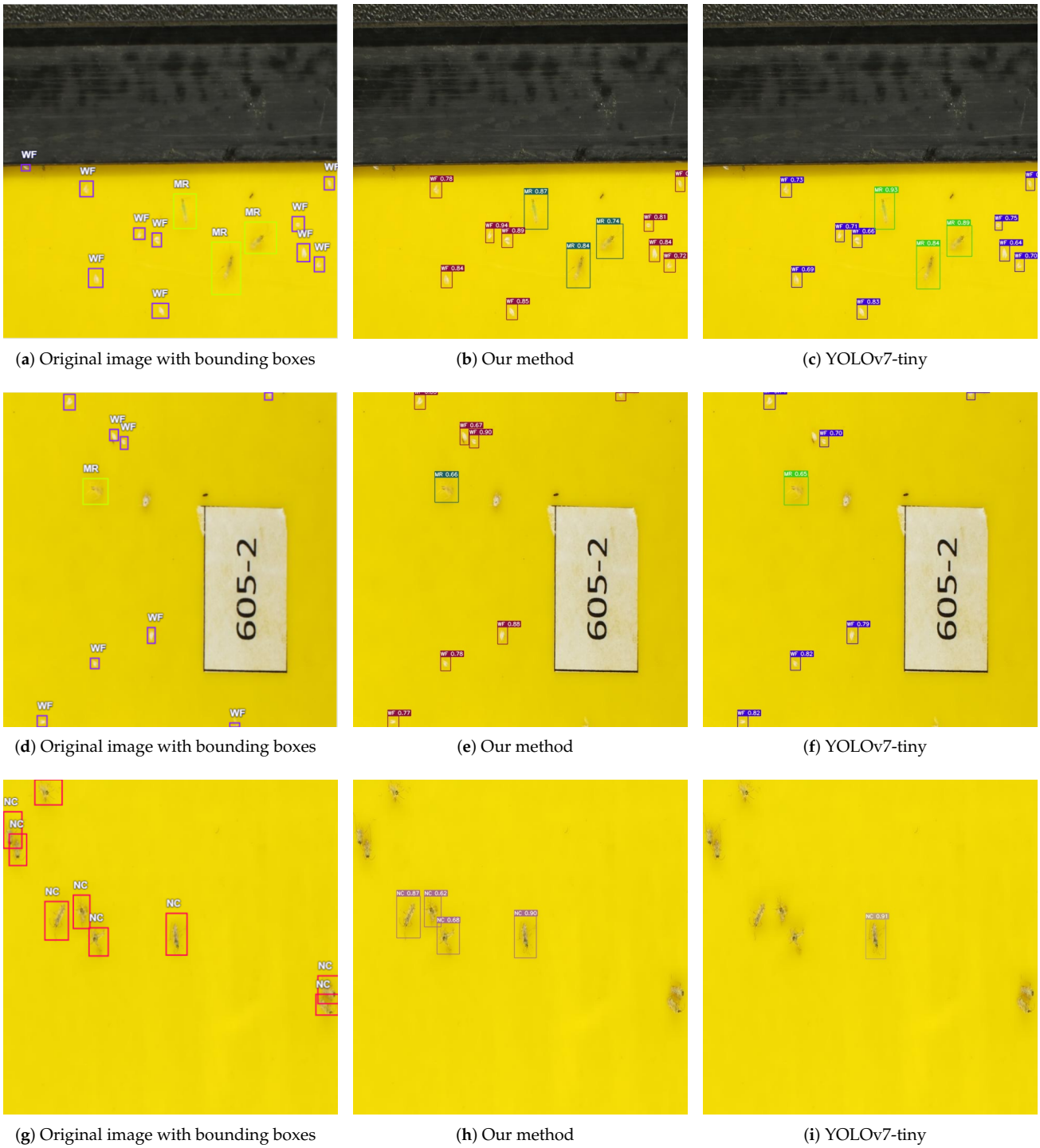


(**e**) Method 3

Figure 13. P-R curves of different methods.

The comparison of the detection results of different methods is shown in Figure 14. Figure 14a,d,g are the original pictures with labeled bounding boxes. Figure 14b,e,h are the detection results of our method, while Figure 14c,f,i are the detection results of the YOLOv7-tiny method. By comparing Figure 14b,c, it is evident that the confidence level of

the prediction boxes in Figure 14b is higher than that in Figure 14c. Looking at Figure 14d–f, it is clear that one WF is missed in the prediction results shown in Figure 14f compared to Figure 14e. Observing Figure 14g–i, it is obvious that several NCs are missed in the prediction results shown in Figure 14i compared to Figure 14h.



(**a**) Original image with bounding boxes (**b**) Our method (**c**) YOLOv7-tiny

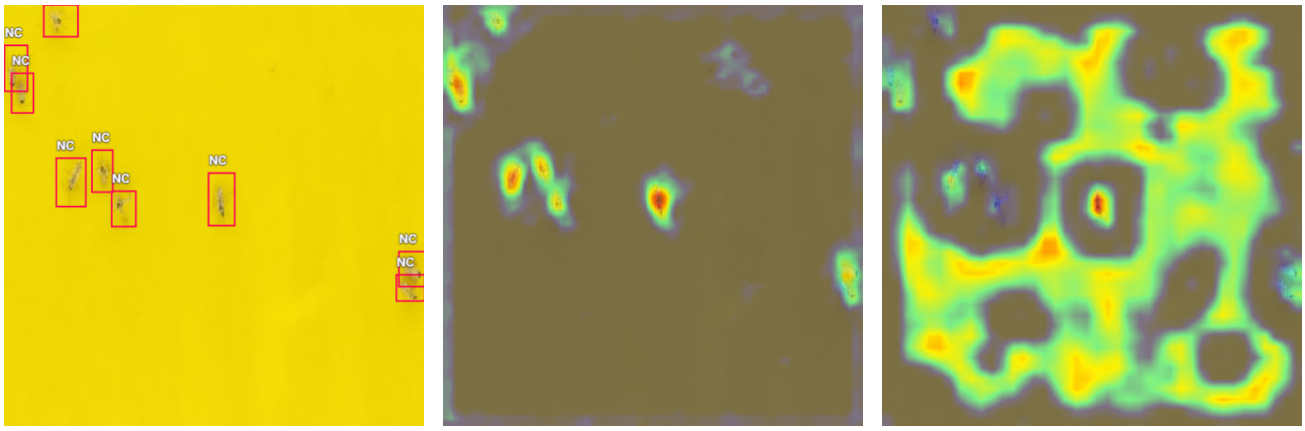
(**d**) Original image with bounding boxes (**e**) Our method (**f**) YOLOv7-tiny

(**g**) Original image with bounding boxes (**h**) Our method (**i**) YOLOv7-tiny

Figure 14. Comparison of the detection results between our proposed method and YOLOv7-tiny.

To better demonstrate that our method has a stronger focus on small-sized insects than YOLOv7-tiny, we generated Grad-CAM maps [56] for both YOLOv7-tiny and our method. The Grad-CAM maps are shown in Figure 15. In the Grad-CAM maps, (a) represents original image with bounding boxes, (b) represents the map generated using our method,

and (c) represents the map generated by YOLOv7-tiny method. In the Grad-CAM images, different colors within a region indicate different degrees of the network's interest in that region. Blue, green, yellow, and red, respectively, represent the increasing levels of the network's attention to the region. The redder the color of a region is, the higher the network's attention to that region is, and the stronger its ability to extract features from that region is. During the detection process, we expect the network to focus more on the insects themselves rather than the background. By comparing these two images (a) and (b), we can observe that the red regions in (b) are mainly concentrated near the target insects, while the red regions in (c) are widely distributed around the environment. This shows that the YOLOv7-tiny method is affected by background interference and still maintains some attention on the background, whereas our method has eliminated the influence of background interference.



(**a**) Original image with bounding boxes. (**b**) Our method. (**c**) YOLOv7-tiny.

Figure 15. Comparison of the Grad-CAM maps between our proposed method and YOLOv7-tiny.

It is noteworthy that, compared with YOLOv7-tiny, our method shows an improvement in mAP. Although this is accompanied by an increase in inference time and the number of parameters (with an increment of 3 M in the number of parameters and a certain degree of rise in inference time), we believe that these costs are worthwhile. Especially considering the current development status of the embedded hardware environment, it can already tolerate such a relatively small increase in the number of parameters and inference time. Moreover, the improvement in mAP will bring significant positive effects on the accuracy of object detection, thus having more advantages in overall performance and being able to better meet the requirements in practical applications and enhance the effectiveness and reliability of the system.

4.5. Comparative Experiments

To evaluate the performance of our proposed method, we conducted comparative experiments on the dataset with advanced models such as Faster R-CNN, SSD, YOLOv3-tiny, YOLOv5s, YOLOv7-tiny, YOLOv7, YOLOv7x, YOLOv8n, YOLOv8s, YOLOv10n, and RT-DETR. Among them, YOLOv3-tiny, YOLOv5s, YOLOv7-tiny, YOLOv7, YOLOv7x, YOLOv8n, and YOLOv8s are different versions of the YOLO series, while RT-DETR is a recently proposed high-precision object detection method based on Transformer. SSD, like the YOLO series, is a single-stage object detection algorithm with a relatively good balance between speed and accuracy. Faster R-CNN is a two-stage algorithm that first generates region proposals and then performs classification and refinement. We hope to explore the characteristics of different methods and the advantages of our method through comparative experiments between our method, the models of the YOLO series, and non-YOLO series object detection models. The comparison metrics include average precision (AP), mean average precision (mAP), recall, the number of parameters, inference time, and

floating-point operations (FLOPs). Among them, the AP metric is further subdivided into three categories: MR, NC, and WF. The experimental results are shown in Table 8. In terms of recall rate, compared with other methods, our method is 8.8% higher than Faster R-CNN, 27.5% higher than SSD, 2.2% higher than Yolov3-tiny, 6.8% higher than Yolov5s, 2.7% higher than Yolov7-tiny, 2.9% higher than Yolov7, 8.4% higher than Yolov7x, 9.7% higher than Yolov8n, 4.4% higher than Yolov8s, 6% higher than Yolov10n, and 4.1% higher than RT-DETR. It can be seen that our method has certain advantages in recall rate. In the AP metric of the MR class, our method has reached the highest value of 91.5%. In the AP metric of the NC class, our method has reached 87.8%, which is only slightly lower than that of YOLOv3-tiny and YOLOv8s. In the AP metric of the WF class, our method has reached the highest value of 92%, surpassing the other eleven methods. In the mAP metric, our method has increased by 3.14%, 11.8%, 4.7%, 4.7%, 4.4%, 3.5%, 2.9%, 4.6%, 4.4%, 4.2%, and 4.2%, respectively, compared with Faster R-CNN, SSD, YOLOv3-tiny, YOLOv5s, YOLOv7-tiny, YOLOv7, YOLOv7-x, YOLOv8n, YOLOv8s, YOLOv10n, and RT-DETR, reaching 90.4%. Through the analysis of the results of the above five metrics, it can be seen that our method has certain advantages in detection accuracy compared with the other eleven methods.

Table 8. Comparison of the performance of our method with other advanced methods.

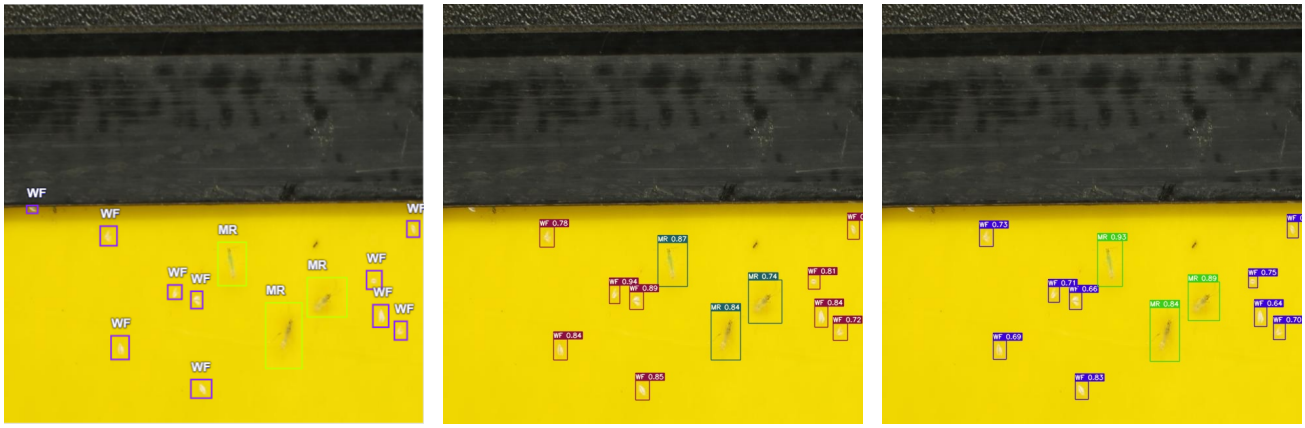
Methods	AP			mAP	Recall	Parameters	Infer Time	GFLOPs
	MR	NC	WF					
Faster R-CNN [15]	0.70	0.802	0.267	0.590	0.778	137 M	81.2 ms	369.8
SSD [27]	0.869	0.861	0.629	0.786	0.591	24 M	11.8 ms	61
Yolov3-tiny [12]	0.848	0.912	0.811	0.857	0.844	12 M	5.2 ms	18.9
Yolov5s	0.862	0.887	0.821	0.857	0.798	9 M	6.9 ms	23.8
Yolov7-tiny [11]	0.867	0.85	0.903	0.86	0.839	6 M	12.8 ms	13
Yolov7 [11]	0.849	0.856	0.901	0.869	0.837	37 M	13.5 ms	103.2
Yolov7x [11]	0.858	0.861	0.906	0.875	0.782	71 M	20.7 ms	188
Yolov8n	0.87	0.842	0.861	0.858	0.769	3 M	4.5 ms	8.1
Yolov8s	0.88	0.892	0.808	0.86	0.822	11 M	7.3 ms	28.4
Yolov10n [57]	0.863	0.86	0.862	0.862	0.806	2 M	4.3 ms	6.5
RT-DETR [16]	0.869	0.828	0.89	0.862	0.825	42 M	23.7 ms	125.6
Our method	**0.915**	0.878	**0.92**	**0.904**	**0.866**	9 M	13.3 ms	17.5

We believe that the advantages in these metrics are attributed to the introduction of the Contextual Information Extraction module (CIE) based on the Transformer encoder, which enhances the network's ability to acquire contextual information, enables it to focus on more important regions, and strengthens the detection of small insects. Meanwhile, the introduction of the Tiny-ELAN feature fusion network with the added P2 detection head has improved the network's feature fusion ability. In addition, the use of the SIoU loss function also helps the network to better detect small insects.

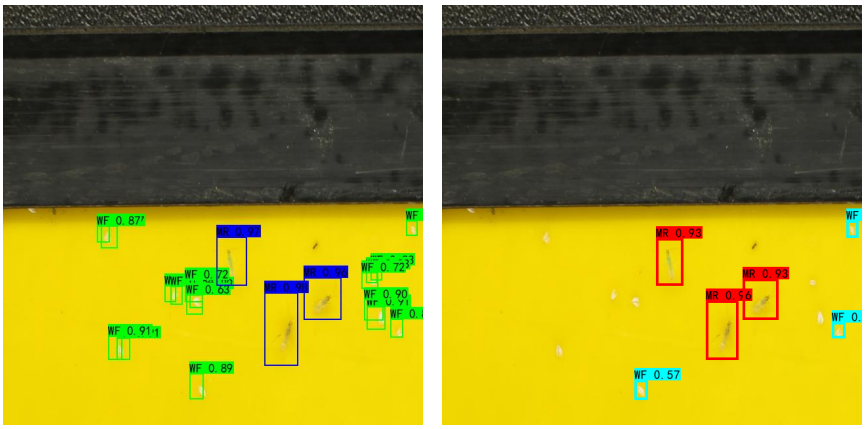
In terms of the number of parameters, YOLOv10n has the fewest parameters, with only 2 M. Our method has 9 M parameters, which is on the lower side among the 12 methods and is comparable to the number of parameters of YOLOv5s. It has 3 M, 2 M, and 33 M fewer parameters than YOLOv3-tiny, YOLOv8s, and RT-DETR, respectively, and 3 M and 7 M more parameters than YOLOv7-tiny and YOLOv10n, respectively. Faster R-CNN has the largest number of parameters, which is 137 M, but its detection performance is not satisfactory. In the three versions of the YOLOv7 series, namely YOLOv7-tiny, YOLOv7, and YOLOv7x, as the number of parameters increases, both the inference time and the mAP also increase successively.

Subsequently, we will deliberate on the detection outcomes of several distinct methods within two specific scenarios: one where insects are prone to being confounded with the environment, and the other where insects are in extremely close proximity. The scenario in which insects are easily confused with the environment is illustrated in Figure 16. The WF insects depicted therein possess a relatively light coloration, rendering them liable to

blend in with the surroundings. It is evident that our proposed method exhibits superior performance. Merely one WF pest is overlooked. The YOLOv7-tiny method also misses one WF pest; nevertheless, the confidence level of this method when detecting WF is somewhat lower than that of our method. This could potentially be attributed to the absence of long-distance semantic information between the insects and the ambient environment in this approach. In the case of the Faster R-CNN method, while WF insects are missed, a substantial number of duplicate bounding boxes are generated. This occurs because the predefined anchor boxes are predominantly square in shape, and the model might endeavor to fit the elongated insects using multiple square anchor boxes, thereby giving rise to numerous redundant boxes. Although this ensures a relatively high recall rate, the superfluous duplicate boxes have a detrimental effect on the mAP. Regarding the SSD algorithm, a significant number of WF pests are undetected. This might be due to its incapacity to effectively capture the semantic information between the insects and their surroundings as well as its deficiency in feature fusion.



(**a**) Original image with bounding boxes. (**b**) Our method. (**c**) YOLOv7-tiny.



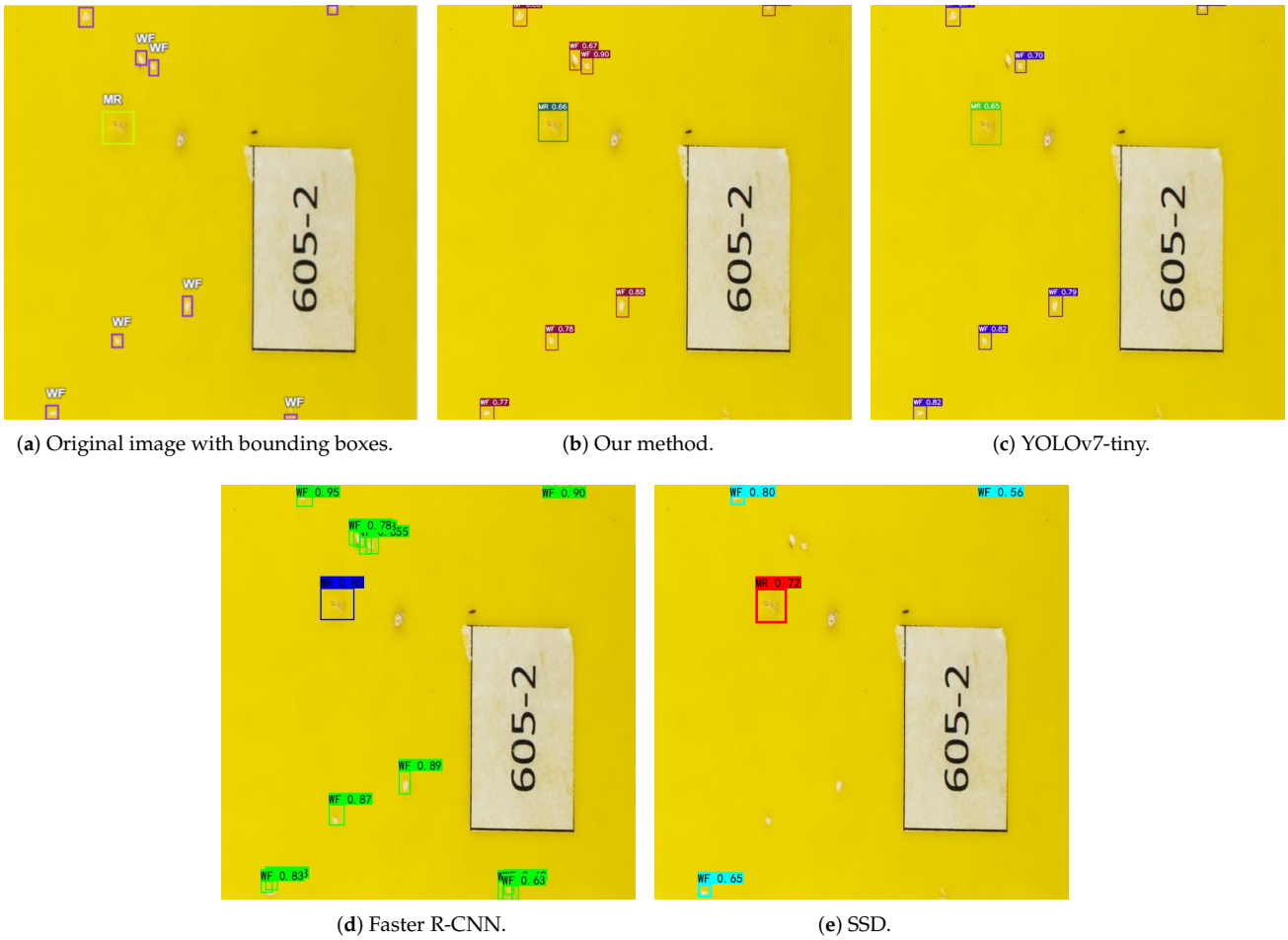
(**d**) Faster R-CNN. (**e**) SSD.

Figure 16. The detection results of various methods when the insects to be detected are easily confused with the environmental colors.

The scenario where the WF pests are clustered in the upper left is presented in Figure 17. It can be observed that our method demonstrates relatively good performance, with only one WF pest being missed. In contrast, the YOLOv7-tiny method misses two WF pests. Moreover, the confidence level of this method in detecting WF is marginally lower than that of our method. This could potentially stem from the fact that it lacks the long-distance semantic information between the insects and the surrounding environment. Similarly, the Faster R-CNN method generates excessive prediction boxes to ensure a certain recall rate.

As for the SSD algorithm, a large number of WF pests go undetected. This might be due to its failure to effectively acquire the semantic information between the insects and their surroundings and its insufficient feature fusion capabilities.

After comprehensive comparison, our method performs best. YOLOv7-tiny is a bit weak when insects blend with the background. Faster R-CNN, because of anchors, is not suitable for detecting diverse forms, generating duplicate boxes and reducing mAP. SSD, due to its network structure, cannot fuse multi-scale features well and lacks semantic extraction, performing poorly in confusing backgrounds and close-distance insect scenarios.



(**a**) Original image with bounding boxes. (**b**) Our method. (**c**) YOLOv7-tiny.

(**d**) Faster R-CNN. (**e**) SSD.

Figure 17. As shown by the two WF pests in the figure, the detection results of various methods when there are insects to be detected that are close to each other in distance.

Our method has great practical value for low-powered devices. With only 9 M parameters, it demands less memory and computational resources, reducing storage and complexity compared to methods like Faster R-CNN. The 13.3 ms inference time is crucial for real-time detection in embedded environments. In applications like surveillance cameras or autonomous robots on resource-constrained platforms, it can quickly analyze visual data and make timely decisions. The combination of low parameters and fast inference makes it suitable for a wide range of low-powered devices, facilitating intelligent system development in Internet of Things (IoT) by enabling local processing and analysis.

In conclusion, the comparison test results indicate that the proposed method has better detection accuracy in the detection of MR, NC, and WF insects than the other five methods. Meanwhile, the parameter scale of the model is also within an acceptable range.

5. Conclusions

Since insects in yellow sticky trap images of tomato crops are usually small, this poses challenges to object detection methods based on deep learning. To solve these problems, we propose an improved YOLOv7-tiny method combined with CIE block. First, in terms of model structure, since the Tiny-SPP structure is based on the CNN network, the locality of the convolution operation limits its ability to acquire global semantic information. To this end, we use the CIE block to replace the original Tiny-SPP structure, which effectively improves the network's acquisition of global semantic information and further focus on more important features, thereby enhancing the feature extraction ability of small-sized insects. Secondly, in terms of loss function, CIoU loss function does not take into account the mismatched direction between the ground truth box and the prediction box, which makes the model unable to obtain satisfactory performance after training. Therefore, we use the SIoU loss function to replace the original CIoU loss function, which is beneficial to improving the model's learning ability for small-sized pests and making the model converge in a satisfactory performance. Finally, we propose a Tiny-ELAN fusion network based on the Gather-and-Distribute mechanism with a P2 detection head to enhance the feature fusion ability of non-adjacent layers. Experimental results show that our method can accurately detect three insects: whitefly (WF), macrolophus (MR), and nesidiocoris (NC) in the yellow sticky trap images of tomato crops. Compared with Faster R-CNN, SSD, YOLOv3-tiny, YOLOv5s, YOLOv7-tiny, YOLOv7, YOLOv7-x, YOLOv8n, YOLOv8s, YOLOv10n, and RT-DETR, the mean average precision of our method increased by 3.14%, 11.8%, 4.7%, 4.7%, 4.4%, 3.5%, 2.9%, 4.6%, 4.4%, 4.2%, and 4.2%, respectively.

6. Future Work

Our method shows good results, but it has limitations. Currently, it requires considerable computational power, restricting its application in resource-limited environments.

For future research, we aim to make our method lightweight. We will use techniques like model pruning and quantization to reduce memory and computational demands, enabling its use in embedded scenarios for real-time pest detection. Furthermore, we will adapt the model to other pest species to provide more comprehensive pest control. Lastly, testing the model in various agricultural environments will enhance its robustness and reliability.

Author Contributions: Conceptualization, S.W., D.C. and C.Z.; methodology, S.W., D.C., J.X. and C.Z.; software, D.C. and J.X.; validation, D.C. and J.X.; investigation, D.C.; resources, C.Z.; data curation, J.X.; writing-original draft preparation, D.C.; writing-review and editing, S.W. and D.C.; visualization, D.C.; supervision, S.W.; project administration, C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The public dataset used in this study are available on github: https://github.com/md-121/yellow-sticky-traps-dataset.

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

References

1. He, X.; Qiao, Y.; Liu, Y.; Dendler, L.; Yin, C.; Martin, F. Environmental impact assessment of organic and conventional tomato production in urban greenhouses of Beijing city, China. *J. Clean. Prod.* **2016**, *134*, 251–258. [CrossRef]
2. Fuentes, A.; Yoon, S.; Kim, S.C.; Park, D.S. A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. *Sensors* **2017**, *17*, 2022. [CrossRef] [PubMed]
3. Arnemann, J.A.; Bevilaqua, J.G.; Bernardi, L.; Rosa, D.O.d.; Encarnaç ao, F.A.d.; Pozebon, H.; Marques, R.P.; Moro, D.; Ribas, D.; Patias, L.S.; et al. Integrated management of tomato whitefly under greenhouse conditions. *J. Agric. Sci.* **2019**, *11*, 443. [CrossRef]
4. Bhadane, G.; Sharma, S.; Nerkar, V.B. Early pest identification in agricultural crops using image processing techniques. *Int. J. Electr. Electron. Comput. Eng.* **2013**, *2*, 77–82.
5. Pinto-Zevallos, D.M.; Vänninen, I. Yellow sticky traps for decision-making in whitefly management: What has been achieved? *Crop Prot.* **2013**, *47*, 74–84. [CrossRef]

6. Heinz, K.M.; Parrella, M.P.; Newman, J.P. Time-efficient use of yellow sticky traps in monitoring insect populations. *J. Econ. Entomol.* **1992**, *85*, 2263–2269. [CrossRef]
7. Cho, J.; Choi, J.; Qiao, M.; Ji, C.; Kim, H.; Uhm, K.; Chon, T. Automatic identification of whiteflies, aphids and thrips in greenhouse based on image analysis. *Red* **2007**, *346*, 244.
8. Yano, E. Control of the greenhouse whitefly, Trialeurodes vaporariorum Westwood (Homoptera: Aleyrodidae) by the integrated use of yellow sticky traps and the parasite Encarsia formosa Gahan (Hymenoptra: Aphelinidae). *Appl. Entomol. Zool.* **1987**, *22*, 159–165. [CrossRef]
9. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
10. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16 × 16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
11. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023, Vancouver, BC, Canada, 18–22 June 2023; pp. 7464–7475.
12. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
13. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
14. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision 2015, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
15. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. [CrossRef]
16. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. Detrs beat yolos on real-time object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2024, Seattle, WA, USA, 17–21 June 2024; pp. 16965–16974.
17. Nieuwenhuizen, A.; Hemming, J.; Janssen, D.; Suh, H.; Bosmans, L.; Sluydts, V.; Brenard, N.; Rodríguez, E.; Tellez, M. Raw data from Yellow Sticky Traps with insects for training of deep learning Convolutional Neural Network for object detection. *Wagening. Univ. Res.* **2019**.
18. Shi, Y.; Li, X.; Chen, M. SC-YOLO: A object detection model for small traffic signs. *IEEE Access* **2023**, *11*, 11500–11510. [CrossRef]
19. Nagar, H.; Sharma, R. A comprehensive survey on pest detection techniques using image processing. In Proceedings of the 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 13–15 May 2020; pp. 43–48.
20. Yang, H.; Liu, W.; Xing, K.; Qiao, J.; Wang, X.; Gao, L.; Shen, Z. Research on insect identification based on pattern recognition technology. In Proceedings of the 2010 Sixth International Conference on Natural Computation, Yantai, China, 10–12 August 2010; Volume 2, pp. 545–548.
21. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
22. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–26 June 2005; Volume 1, pp. 886–893.
23. Lienhart, R.; Maydt, J. An extended set of haar-like features for rapid object detection. In Proceedings of the International Conference on Image Processing 2002, Rochester, NY, USA, 22–25 September 2002; Volume 1, p. I.
24. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
25. Li, W.; Feng, X.S.; Zha, K.; Li, S.; Zhu, H.S. Summary of target detection algorithms. In *Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2021; Volume 1757, p. 012003.
26. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
27. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14; pp. 21–37.
28. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
29. Yang, M.; Fan, X. YOLOv8-Lite: A Lightweight Object Detection Model for Real-time Autonomous Driving Systems. *IECE Trans. Emerg. Top. Artif. Intell.* **2024**, *1*, 1–16. [CrossRef]
30. Huang, Z.; Zhang, P.; Liu, R.; Li, D. An Improved YOLOv3-Based Method for Immature Apple Detection. *IECE Trans. Internet Things* **2023**, *1*, 9–14. [CrossRef]
31. Mamdouh, N.; Khattab, A. YOLO-based deep learning framework for olive fruit fly detection and counting. *IEEE Access* **2021**, *9*, 84252–84262. [CrossRef]
32. Lyu, Z.; Jin, H.; Zhen, T.; Sun, F.; Xu, H. Small object recognition algorithm of grain pests based on ssd feature fusion. *IEEE Access* **2021**, *9*, 43202–43213. [CrossRef]
33. Amrani, A.; Sohel, F.; Diepeveen, D.; Murray, D.; Jones, M.G. Insect detection from imagery using YOLOv3-based adaptive feature fusion convolution network. *Crop Pasture Sci.* **2022**, *74*, 615–627. [CrossRef]

34. Wen, C.; Chen, H.; Ma, Z.; Zhang, T.; Yang, C.; Su, H.; Chen, H. Pest-YOLO: A model for large-scale multi-class dense and tiny pest detection and counting. *Front. Plant Sci.* **2022**, *13*, 973985. [CrossRef] [PubMed]
35. Li, S.; Wang, H.; Zhang, C.; Liu, J. A self-attention feature fusion model for rice pest detection. *IEEE Access* **2022**, *10*, 84063–84077. [CrossRef]
36. Dong, S.; Zhang, J.; Wang, F.; Wang, X. YOLO-pest: A real-time multi-class crop pest detection model. In Proceedings of the International Conference on Computer Application and Information Security (ICCAIS 2021), Wuhan, China, 18–19 December 2021; Volume 12260, pp. 12–18.
37. Ali, F.; Qayyum, H.; Iqbal, M.J. Faster-PestNet: A Lightweight deep learning framework for crop pest detection and classification. *IEEE Access* **2023**, *11*, 104016–104027. [CrossRef]
38. Yang, Y.; Xiao, Y.; Chen, Z.; Tang, D.; Li, Z.; Li, Z. FCBTYOLO: A Lightweight and High-Performance Fine Grain Detection Strategy for Rice Pests. *IEEE Access* **2023**, *11*, 101286–101295. [CrossRef]
39. Zhang, L.; Zhao, C.; Feng, Y.; Li, D. Pests Identification of IP102 by YOLOv5 Embedded with the Novel Lightweight Module. *Agronomy* **2023**, *13*, 1583. [CrossRef]
40. Gonçalves, J.; Silva, E.; Faria, P.; Nogueira, T.; Ferreira, A.; Carlos, C.; Rosado, L. Edge-Compatible Deep Learning Models for Detection of Pest Outbreaks in Viticulture. *Agronomy* **2022**, *12*, 3052. [CrossRef]
41. Nieuwenhuizen, A.; Hemming, J.; Suh, H. Detection and classification of insects on stick-traps in a tomato crop using Faster R-CNN. In Proceedings of the The Netherlands Conference on Computer Vision, Eindhoven, The Netherlands, 26–27 September 2018.
42. Ozge Unel, F.; Ozkalayci, B.O.; Cigla, C. The power of tiling for small object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops 2019, Long Beach, CA, USA, 16–20 June 2019.
43. Tian, Y.; Wang, S.; Li, E.; Yang, G.; Liang, Z.; Tan, M. MD-YOLO: Multi-scale Dense YOLO for small target pest detection. *Comput. Electron. Agric.* **2023**, *213*, 108233. [CrossRef]
44. Tang, Z.; Lu, J.; Chen, Z.; Qi, F.; Zhang, L. Improved Pest-YOLO: Real-time pest detection based on efficient channel attention mechanism and Transformer encoder. *Ecol. Inform.* **2023**, *78*, 102340. [CrossRef]
45. Dong, Q.; Sun, L.; Han, T.; Cai, M.; Gao, C. PestLite: A novel YOLO-based deep learning technique for crop pest detection. *Agriculture* **2024**, *14*, 228. [CrossRef]
46. Deng, J.; Yang, C.; Huang, K.; Lei, L.; Ye, J.; Zeng, W.; Zhang, J.; Lan, Y.; Zhang, Y. Deep-Learning-Based Rice Disease and Insect Pest Detection on a Mobile Phone. *Agronomy* **2023**, *13*, 2139. [CrossRef]
47. Bai, Y.; Zhang, Y.; Ding, M.; Ghanem, B. Finding tiny faces in the wild with generative adversarial network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018, Salt Lake City, UT, USA, 18–22 June 2018; pp. 21–30.
48. Wang, K.; Li, S.; Niu, S.; Zhang, K. Detection of infrared small targets using feature fusion convolutional network. *IEEE Access* **2019**, *7*, 146081–146092. [CrossRef]
49. Ye, R.; Gao, Q.; Qian, Y.; Sun, J.; Li, T. Improved Yolov8 and Sahi Model for the Collaborative Detection of Small Targets at the Micro Scale: A Case Study of Pest Detection in Tea. *Agronomy* **2024**, *14*, 1034. [CrossRef]
50. Meng, J.; Jiang, P.; Wang, J.; Wang, K. A mobilenet-SSD model with FPN for waste detection. *J. Electr. Eng. Technol.* **2022**, *17*, 1425–1431. [CrossRef]
51. Wang, C.; He, W.; Nie, Y.; Guo, J.; Liu, C.; Wang, Y.; Han, K. Gold-YOLO: Efficient object detector via gather-and-distribute mechanism. *Adv. Neural Inf. Process. Syst.* **2024**, *36*
52. Chi, L.; Jiang, B.; Mu, Y. Fast fourier convolution. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 4479–4488.
53. Liu, J.; Zhang, Y.; Cui, J.; Feng, Y.; Pang, L. Fully convolutional multi-scale dense networks for monocular depth estimation. *IET Comput. Vis.* **2019**, *13*, 515–522. [CrossRef]
54. Gevorgyan, Z. Siou loss: More powerful learning for bounding box regression. *arXiv* **2022**, arXiv:2205.12740.
55. Deserno, M.; Briassouli, A. Faster r-CNN and EfficientNet for accurate insect identification in a relabeled yellow sticky traps dataset. In Proceedings of the 2021 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor), Trento-Bolzano, Italy, 3–5 November 2021; pp. 209–214.
56. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision 2017, Venice, Italy, 22–29 October 2017; pp. 618–626.
57. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. Yolov10: Real-time end-to-end object detection. *arXiv* **2024**, arXiv:2405.14458.