

## Article

# Enhancing Pest Detection in Agriculture: A Multi-Scale Feature Fusion Approach with YOLOv3

He Zhang, Xiaochen Liu, Chenguang Wang, Jun Tang, Chong Shen \* and Jun Liu \*

State Key Laboratory of Extreme Environment Optoelectronic Dynamic Measurement Technology and Instrument, School of Instrument and Electronics, North University of China, Taiyuan 030051, China

\* Correspondence: shenchong@nuc.edu (C.S.); liuj@nuc.edu.cn (J.L.)

## Abstract

The stable production of crops such as corn, wheat, soybeans, and canola is increasingly threatened by widespread pest infestations. Conventional manual pest surveys are hampered by low operational efficiency, subjective assessment bias, and delayed feedback, thereby impeding their ability to satisfy the demands of precision agriculture. To address these challenges, we propose an intelligent pest detection framework based on EfficientNet and Feature Pyramid Network (FPN) for fast and accurate field pest identification. EfficientNet is adopted as the lightweight attention-embedded backbone to extract hierarchical features, and multi-scale detection plus hierarchical FPN fusion are integrated to improve recognition performance for tiny, inconspicuous pests. The experimental results on 37 common pest species in field crops showed that the proposed model achieves a mean average precision at Intersection-over-Union (IoU) threshold 0.5 (mAP@0.5) of 98.89%, 1.57% average recognition error rate, and with an average inference time of merely 0.048 s per image, balancing outstanding detection accuracy and real-time performance. Furthermore, this approach delivers a lightweight, reliable, and automated monitoring solution for field pest surveillance, thereby facilitating data-driven, precise pest management and advancing the practice of sustainable, green precision agriculture.

**Keywords:** precision agriculture; YOLOv3; pest detection; multi-scale feature fusion; deep learning

## 1. Introduction

Sustainable production of staple cereal (e.g., wheat and maize) and oilseed crops (e.g., soybean and canola) underpins global agricultural stability and catalyzes the advancement of precision and ecologically sustainable farming systems [1,2]. However, pest infestation remains the predominant biotic constraint on crop productivity, accounting for global yield losses ranging from 10% to 28%, with even more severe losses reported in developing countries [3]. Additionally, pests are indirectly responsible for major losses in few plants, such as cankers and shoot blights in nuts, since many insects serve as a vector for pathogens [4]. Beyond significant yield reduction, pest infestation often compels farmers to increase pesticide application, hinders the development of green and sustainable agriculture [5,6]. These adverse effects further disrupt routine field management, reduce crop productivity, and hinder the implementation of precision agriculture, forming a core bottleneck limiting the stability and sustainability of agricultural production systems. With the global transition toward digital and precision agriculture, mitigating pest-induced hazards is essential for advancing agronomic development and optimizing crop cultivation



Academic Editor: Jayme Garcia Arnal Barbedo

Received: 15 May 2026

Revised: 30 June 2026

Accepted: 2 July 2026

Published: 18 August 2026

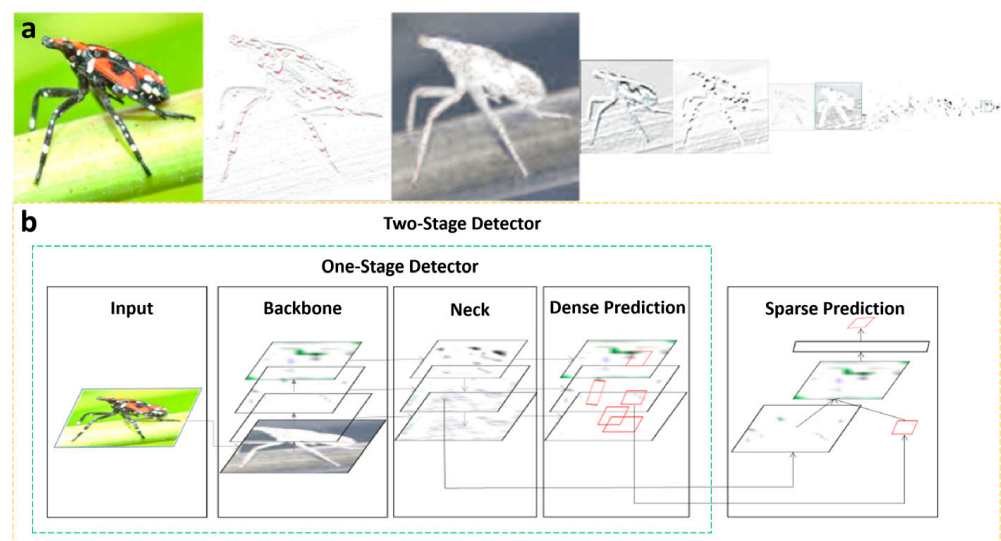
**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

and management [7]. Traditional pest monitoring relies heavily on manual field surveys, which are characterized by low operational efficiency, delayed feedback loops, and limited spatial coverage [8]. In the absence of robust, reliable monitoring systems, large-scale pest outbreaks can emerge readily—compromising agroecosystem integrity and significantly reducing crop productivity [9].

In recent years, advances in deep learning and computer vision have built a new intelligent agricultural risk management paradigm [10]. Vision-based pest detection has become a mainstream research direction, outperforming manual surveys in field practicality [11]. Convolutional neural network (CNN)-based pest identification methods automatically extract hierarchical visual features to enable fast, high-accuracy pest classification [12,13]. The hierarchical features extracted by the VGG network are shown in Figure 1a. Compared with traditional methods, deep learning detection techniques have superior performance in real-time capability, scalability and accuracy, relying on standardized unified frameworks. As illustrated in Figure 1b, mainstream object detectors adopt a four-stage structure: input, backbone, neck and head (Figure 1b). These detectors are broadly categorized into single-stage (dense prediction) and two-stage (sparse prediction) paradigms. Each paradigm further subdivides into anchor-based and anchor-free approaches. Representative dense-prediction models include the anchor-based YOLO and RetinaNet, as well as the anchor-free CenterNet and FCOS. In contrast, sparse-prediction models encompass the anchor-based Faster R-CNN and the anchor-free RepPoints. Among these algorithms, the YOLO series models have attracted considerable attention due to their excellent balance between detection speed and accuracy [14,15].



**Figure 1.** Schematic diagram of convolutional neural network feature extraction and object detection structure. (a) Schematic diagram of feature extraction in convolutional neural networks; (b) Schematic diagram of the target detection structure.

Although YOLO-based architectures have demonstrated robust performance in agricultural pest monitoring, several critical methodological limitations persist in practical field-scale crop pest detection tasks. First, the original Darknet-53 backbone of YOLOv3 lacks attention modules, failing to capture subtle features of tiny pests in complex field backgrounds [16,17], leading to misclassification of similar pest species and frequent missed detections [18]. Second, the primitive simple upsampling feature fusion module of original YOLOv3 fails to construct complete top-down multi-scale feature transmission, weakening the model's ability to distinguish small-size pests against cluttered leaf [19,20]. Existing feature fusion frameworks rarely construct bidirectional feature circulation, restricting

miniature pest recognition [21,22]. Third, most detection models adopt generic anchor boxes pre-trained on public natural image datasets. The size distribution of these anchor boxes may not match miniature field pests, leading to frequent missed localization of small pest individuals. Dataset-adaptive anchor clustering effectively alleviates this issue [23,24]. Fourth, existing YOLO variants struggle to balance detection precision and computational overhead, forming an unaddressed trade-off dilemma [25]. Deep heavy backbones deliver favorable recognition accuracy but incur excessive inference latency, which cannot be deployed on low-power portable edge devices for field real-time monitoring [26]. While lightweight YOLO-tiny models cut computational cost by simplifying network layers, yet the compressed feature extraction capacity drastically reduces detection accuracy for small and overlapping pests [27]. Furthermore, few datasets have been specially annotated for wheat, corn, soybean and canola pests, restricting the targeted optimization and benchmark evaluation of detection algorithms for dominant field crops [28,29]. To address the above shortcomings, this study proposes an optimized YOLOv3-based intelligent pest detection framework with three targeted optimizations: replacing Darknet-53 with lightweight attention-equipped EfficientNet as the feature extraction backbone, introducing a complete Feature Pyramid Network (FPN) for comprehensive multi-scale feature fusion, and comparing the COCO anchor boxes with the anchor boxes that are adaptively generated from the dataset through K-means clustering, thereby selecting the scale features that are more suitable for field pests. Additionally, a manually annotated multi-class pest dataset covering four major field crops is constructed to validate the performance of the optimized detection model. The proposed model demonstrates enhanced pest detection performance and delivers a computationally efficient, lightweight detection solution, thereby providing technical support for intelligent and precision monitoring of crop pests in agricultural fields.

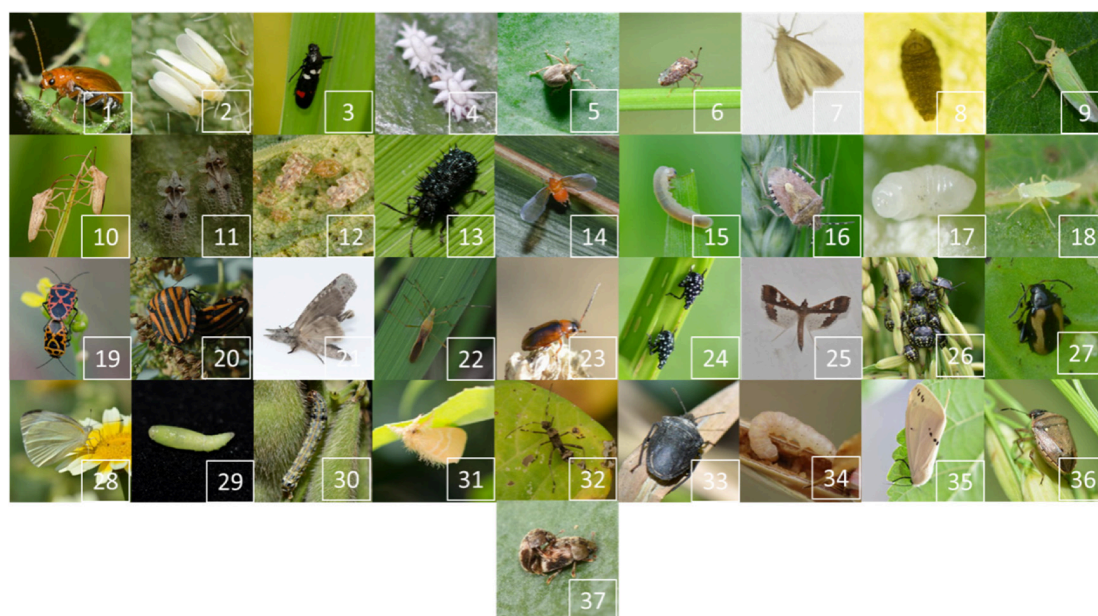
## 2. Material and Methods

### 2.1. Dataset

The experimental image data in this paper is sourced from the Anhui Academy of Agricultural Sciences [30,31], covering 4285 images of 37 types of pests found in several common field crops, including corn, soybean, wheat, and canola. All images were captured by the use of digital cameras (e.g., Canon, Nikon, Tokyo, Japan, and mobile devices). Among them, 3162 images (73.8% of the entire dataset) contained only a single pest individual; the remaining 1123 images (26.2% of the total) contained two or more overlapping pest instances. Furthermore, Following the standard pixel-area-based threshold definition used in YOLO-based detection tasks, we classified annotated pest objects into three size categories and computed their respective proportions: small targets (pixel area  $< 32^2$ ) constitute 47.2%, medium targets ( $32^2 \leq \text{pixel area} < 96^2$ ) account for 39.6%, and large targets (pixel area  $\geq 96^2$ ) represent only 13.2%. The predominance of small-scale pests further underscores the necessity of our multi-scale feature fusion architecture. To eliminate the potential negative effects of illumination variability, all sample images were preprocessed with uniform illumination settings in crop field situations, as was done by Wen and Guyer (2012) [32]. The various types of pest examples included in the dataset are shown in Figure 2, and the names and quantities of each type of pest are listed in Table 1.

### 2.2. Data Processing

To train the model, it is necessary to label the pests in the dataset first. LabelImg is a tool for marking the position and name of the target in the image in deep learning. It is written in Python 3.12 and uses PyQt as its graphical interface. During the annotation process, the annotations are saved as XML files in the PASCAL VOC format. The images of the labeled pests and the generated XML files after labeling are shown in Figure 3.



**Figure 2.** Sample images from the pest dataset, along with their corresponding identifiers, which match those listed in Table 1.

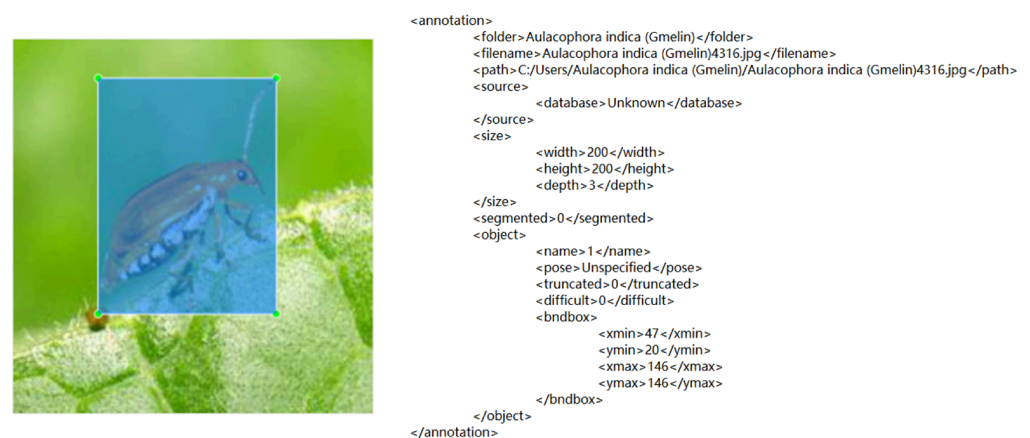
**Table 1.** Pest infestation dataset: image names and quantities.

| NO. | Species Name                              | Quantity | NO. | Species Name                                | Quantity |
|-----|-------------------------------------------|----------|-----|---------------------------------------------|----------|
| 1   | <i>Aulacophora indica</i>                 | 79       | 20  | <i>Graphosoma rubrolineata</i> (Westwood)   | 126      |
| 2   | <i>Bemisia tabaci</i> (Gennadius)         | 147      | 21  | <i>Iscadia inexacta</i> (Walker, 1858)      | 79       |
| 3   | <i>Callitettix versicolor</i> (Fabricius) | 156      | 22  | <i>Leptocorisa acuta</i> (Thunberg)         | 133      |
| 4   | <i>Ceroplastes ceriferus</i> (Anderson)   | 100      | 23  | <i>Luperomorpha suturalis</i> Chen          | 101      |
| 5   | <i>Ceutorhynchus asper</i> Roelofs        | 146      | 24  | <i>Lycorma delicatula</i> (White)           | 92       |
| 6   | <i>Chauliops fallax</i> Scott             | 68       | 25  | <i>Maruca testulalis</i> Grye               | 73       |
| 7   | <i>Chilo suppressalis</i> (Walker)        | 93       | 26  | <i>Nezara viridula</i> (Linnaeus)           | 175      |
| 8   | <i>Chromatomyia horticola</i> (Goureau)   | 114      | 27  | <i>Phyllotreta striolata</i> (Fabricius)    | 187      |
| 9   | <i>Cicadella viridis</i> (Linnaeus)       | 138      | 28  | <i>Pieris rapae</i> (Linnaeus)              | 71       |
| 10  | <i>Cletus punctiger</i> (Dallas)          | 169      | 29  | <i>Plutella xylostella</i> (Linnaeus)       | 68       |
| 11  | <i>Corythucha ciliata</i> (Say)           | 90       | 30  | <i>Porthesia taiwana</i> Shiraki            | 141      |
| 12  | <i>Corythucha marmorata</i> (Uhler)       | 98       | 31  | <i>Riptortus pedestris</i> (Fabricius)      | 110      |
| 13  | <i>Di cladispa armigera</i> (Olivier)     | 150      | 32  | <i>Scotinophara lurida</i> (Burmeister)     | 117      |
| 14  | <i>Diostrombus politus</i> Uhler          | 238      | 33  | <i>Sesamia inferens</i> (Walker)            | 126      |
| 15  | <i>Dolerus tritici</i> Chu                | 88       | 34  | <i>Spilosoma obliqua</i> (Walker)           | 66       |
| 16  | <i>Dolycoris baccarum</i> (Linnaeus)      | 87       | 35  | <i>Spodoptera litura</i> (Fabricius)        | 130      |
| 17  | <i>Dryocosmus kuriphilus</i> Yasumatsu    | 50       | 36  | <i>Stollia ventralis</i> (Westwood)         | 72       |
| 18  | <i>Empoasca flavescens</i> (Fabricius)    | 133      | 37  | <i>Strongyloides variegatus</i> (Fairmaire) | 135      |
| 19  | <i>Eurydema dominulus</i> (Scopoli)       | 150      |     | In total                                    | 4285     |

Secondly, the dataset constitutes the foundational input for training deep neural network models, and its quality critically determines the performance and generalizability of the resulting models. Prior to model construction, it is essential to design an input data processing strategy aligned with the chosen network architecture commonly referred to as dataset preprocessing. This preprocessing stage typically comprises two key components: image rescaling and image augmentation.



All pest images in this dataset have an original resolution of  $200 \times 200$  pixels. However, the YOLOv3 network model expects input images of size  $416 \times 416$  pixels by default. Thus, all images are first resized to this dimension. This upsampling operation inevitably introduces interpolation artifacts, which smooth fine high-frequency spatial gradients and attenuate subtle edge-texture features, potentially degrading the model's capability in feature extraction and precise localization of tiny pest targets. To mitigate the adverse effects of such gradient distortion, we implement a suite of coordinated optimization strategies: (i) the multi-scale feature fusion architecture inherent to YOLO-series networks preserves delicate local gradient information through shallow feature branches, specifically enhancing sensitivity to small objects; (ii) dataset-adaptive for anchor box generation ensures anchor dimensions closely match the actual pixel scales of miniature pests, thereby reducing localization bias arising from blurred target boundaries; and (iii) L2 regularization and gradient accumulation configured via the hyperparameters jointly stabilize gradient updates that would otherwise fluctuate due to resizing-induced artifacts.



**Figure 3.** Sample images and annotated pest files.

A unified baseline augmentation pipeline is implemented identically across all four models to guarantee fair cross-architecture comparison. All augmentation operations are applied on-the-fly during training rather than generating a static offline expanded dataset; the raw dataset retains a fixed total of 4285 original images, with unique random distortions generated dynamically for every training iteration. The shared augmentation suite includes standardized geometric and photometric transformations, with full parameter constraints defined as follows: (1) Isotropic random scaling: Each image and its corresponding bounding boxes are scaled synchronously with a randomly sampled scaling factor ranging from 0.5 to 1.5, which avoids extreme shrinkage or magnification that would obscure pest morphological features; (2) Random translation: Horizontal and vertical translation offsets are restricted within  $\pm 15\%$  of the total image width and height to prevent partial cropping of pest bodies. (3) Horizontal flipping and color jittering: Random horizontal mirroring is performed with a 50% probability, while mild brightness, contrast and saturation jitter are applied to enrich photometric diversity without distorting pest texture identities. Meanwhile, to prevent morphological distortion that could lead to inter-species pest misidentification, strict amplitude constraints are imposed on all geometric transformations. Concurrently, shallow feature extraction branches retain fine-grained texture gradients, while COCO anchor boxes are more suitable for the natural size distribution of the target pests in this research dataset. These measures collectively enhancing the model's ability to discriminate visually similar pest individuals, even under augmentation-induced distortions. Since all YOLO variants are bounding-box-based object detectors, ground-truth bounding box coordinates are transformed in strict synchrony with each augmentation

operation, thereby preserving consistent spatial alignment between input images and their corresponding annotation labels.

Beyond the shared baseline preprocessing, YOLOv4 and YOLOv4-tiny incorporate an exclusive additional data augmentation technique named Mosaic augmentation, a refined variant derived from CutMix [33], while Mosaic augmentation is not activated during the training of YOLOv3 and YOLOv3-tiny. The detailed operating procedure of Mosaic augmentation is as follows. Firstly, the Mosaic data augmentation technique combines four different input images (each with its corresponding bounding box) into a single composite image. Following this integration, a new image and its corresponding bounding boxes are generated and subsequently fed into the neural network for training and optimization. Specifically, each of the four constituent images is first independently subjected to a series of geometric and photometric transformations, including horizontal flipping, isotropic scaling, and color-space jittering. The transformed images are then spatially arranged in the top-left, bottom-left, bottom-right, and top-right quadrants of the output canvas, respectively. Finally, fixed subregions from each quadrant are extracted via matrix indexing and seamlessly stitched together to form the final mosaic image, whose merged bounding boxes are fed into the network for training and optimization.

### 2.3. The Framework Structure of the YOLO Model

YOLOv3 is developed based on YOLOv1 and YOLOv2. As shown in Figure 4, it is a fully convolutional network with a total of 252 layers, using Darknet53 as the backbone feature extraction network [34]. To enhance the detection ability for small objects, the network adopts the methods of feature map upsampling and multi-scale feature fusion, conducting detection on feature maps of multiple scales. The input is the image of pests. The backbone feature extraction network adopts the 52 convolutional layers of Darknet53 after removing the last fully connected layer, which consists of 5 groups of ResNet residual blocks. Each group of residual blocks is repeated 1, 2, 8, 8, and 4 times, respectively. During each repetition process, a  $1 \times 1$  convolution is executed first, followed by a  $3 \times 3$  convolution, and finally the filter is halved and then restored. YOLO takes the entire insect damage image as the input of the network. After the backbone network extracts the feature map of a certain size ( $S \times S$ ), the entire insect damage image is divided into  $S \times S$  grids. Each grid predicts three bounding boxes. If the center of a certain insect damage target falls within a certain grid, then this grid is responsible for predicting this insect damage target. The output layer outputs the position of the bounding box and the category to which the insect damage target belongs. Each bounding box predicts four positions ( $x, y, w, h$ ) and one confidence, totaling five values. Each grid also predicts one category information, with a total of  $C$  categories (there are 37 types of insect damage in the dataset of this article). For one anchor box, the total number of predicted channels equals  $5 + 37 = 42$ ; since each grid predicts three boxes, the total channel count per grid is  $3 \times 42 = 126$ , and ultimately the output tensor of each detection branch takes the shape of  $S \times S \times 126$ , where  $S \times S$  stands for the spatial grid partition of the feature map, channels 1–4 correspond to the bounding box regression parameters ( $x, y, w, h$ ) of the first anchor box, channel 5 represents the objectness confidence score of the first anchor box, channels 6–42 denote the class probability distribution of the first anchor box, and this identical channel layout is repeated sequentially for the second and third anchor boxes within every grid cell.

YOLOv3-tiny is a compressed version of the YOLOv3 network, which has removed the residual layers and only uses two YOLO output layers of different scales. The network has a total of 23 layers, including 13 convolutional layers, 6 pooling layers, 1 routing layer, 1 upsampling layer, and 2 output layers. Except for the convolutional layer before the YOLO output layer, which uses a linear activation function, each convolutional layer

is followed by a BN layer, and each convolutional layer is followed by a convolutional layer with a leaky rectified linear unit (Leaky ReLU) activation function [35]. The output dimensions of the two output layers are  $13 \times 13$  and  $26 \times 26$ , respectively. Each grid predicts three bounding boxes. There are a total of 37 categories in this dataset. The final output layer dimensions are  $13 \times 13 \times 126$  and  $26 \times 26 \times 126$ . Figure 5 is a schematic diagram of the structure of YOLOv3-tiny.

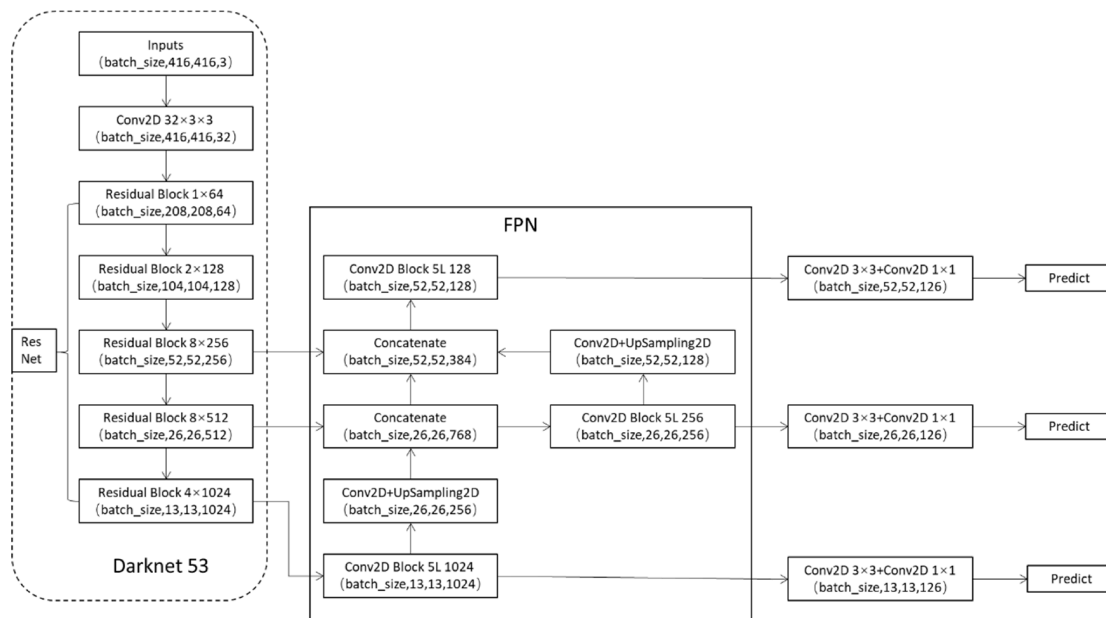


Figure 4. Schematic diagram of the YOLOv3 structure.

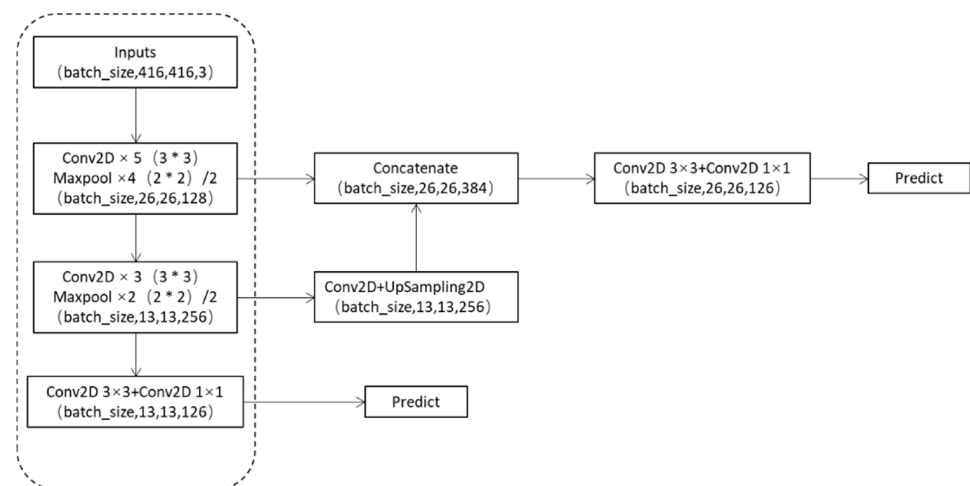


Figure 5. YOLOv3-tiny architecture diagram.

YOLOv4 comprises a total of 162 layers and builds upon the YOLOv3 object detection architecture, integrating several state-of-the-art optimization techniques recently developed in the field of CNNs. Key improvements are introduced across multiple components, including data preprocessing, backbone network design, training methodology, activation functions, and loss function formulation. Notably, YOLOv4 demonstrates superior detection performance over YOLOv3 for occluded objects [36]. Figure 6 is a schematic diagram of the structure of YOLOv4.

YOLOv4-tiny is a lightweight variant of YOLOv4, comprising a total of 38 layers. It employs CSPDarknet53-tiny as its backbone network for feature extraction and adopts the leaky ReLU activation function throughout. Unlike its full-scale counterpart, YOLOv4-tiny

performs classification and regression prediction using only two feature pyramid levels. Figure 7 presents the architectural diagram of YOLOv4-tiny.

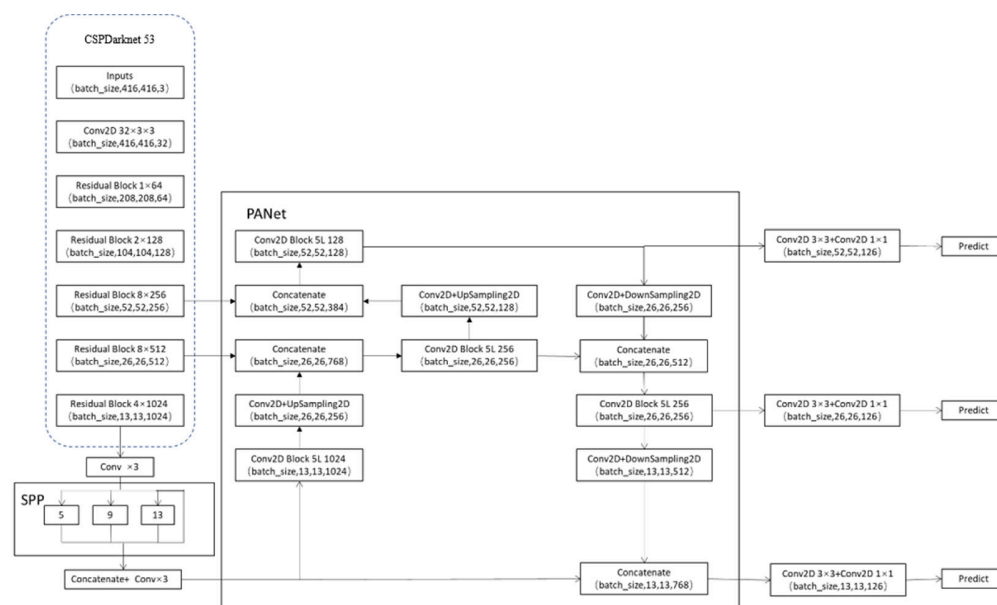


Figure 6. YOLOv4 architecture diagram.

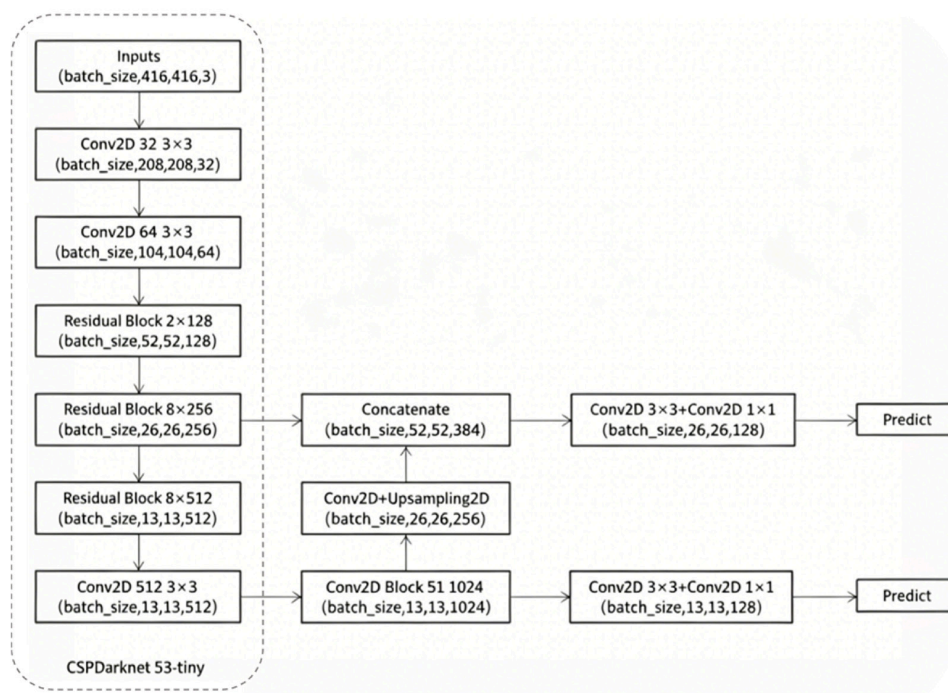


Figure 7. YOLOv4-tiny architecture diagram.

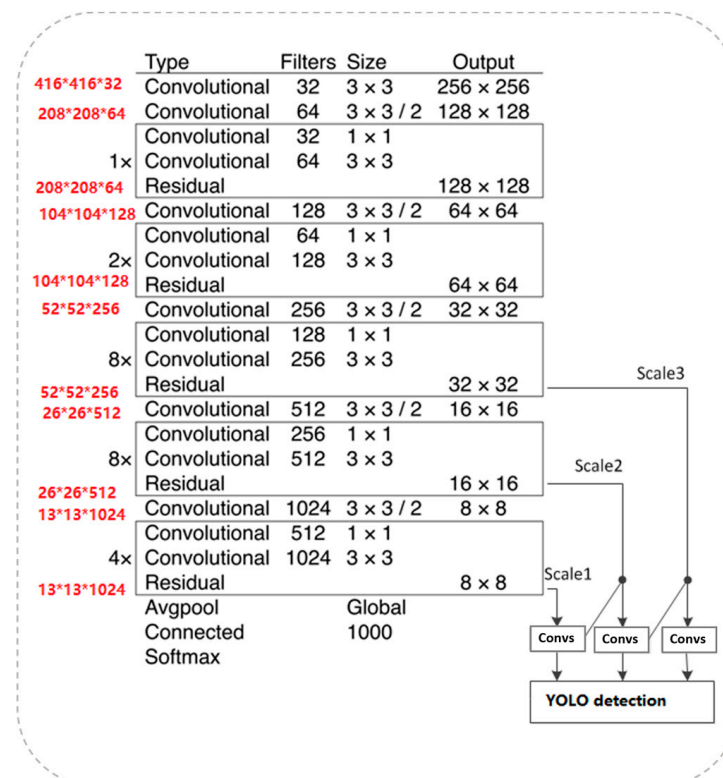
### 2.3.1. Darknet53

YOLOv3 is a fully convolutional neural network, where the primary feature extraction module employs only the first 52 layers of the Darknet53 backbone. The Darknet53 architecture comprises five groups of ResNet-style residual blocks, with the number of repetitions per group being 1, 2, 8, 8, 4, respectively. Table 2 presents a detailed layer-wise breakdown of the Darknet53 structure. The architecture diagram as shown in Figure 8.



**Table 2.** Structure diagram of Darknet53.

| No. of Repetitions | Convolutional Layer | Filter | Convolution Kernel Size | Feature Map Size |
|--------------------|---------------------|--------|-------------------------|------------------|
| 1×                 | Convolutional       | 32     | 3 × 3                   | 416 × 416        |
|                    | Convolutional       | 64     | 3 × 3/2                 | 208 × 208        |
|                    | Convolutional       | 32     | 1 × 1                   | 208 × 208        |
|                    | Convolutional       | 64     | 3 × 3                   |                  |
|                    | Residual            |        |                         |                  |
| 2×                 | Convolutional       | 128    | 3 × 3/2                 | 104 × 104        |
|                    | Convolutional       | 64     | 1 × 1                   | 104 × 104        |
|                    | Convolutional       | 128    | 3 × 3                   |                  |
|                    | Residual            |        |                         |                  |
|                    | Convolutional       | 256    | 3 × 3/2                 | 52 × 52          |
| 8×                 | Convolutional       | 128    | 1 × 1                   | 52 × 52          |
|                    | Convolutional       | 256    | 3 × 3                   |                  |
|                    | Residual            |        |                         |                  |
|                    | Convolutional       | 512    | 3 × 3/2                 | 26 × 26          |
|                    | Convolutional       | 256    | 1 × 1                   |                  |
|                    | Convolutional       | 512    | 3 × 3                   |                  |
| 8×                 | Residual            |        |                         | 26 × 26          |
|                    | Convolutional       | 1024   | 3 × 3/2                 |                  |
|                    | Convolutional       | 512    | 1 × 1                   |                  |
|                    | Convolutional       | 1024   | 3 × 3                   |                  |
|                    | Residual            |        |                         |                  |
| 4×                 | Convolutional       | 1024   | 3 × 3/2                 | 13 × 13          |
|                    | Convolutional       | 512    | 1 × 1                   |                  |
|                    | Convolutional       | 1024   | 3 × 3                   |                  |
|                    | Residual            |        |                         |                  |
|                    | Residual            |        |                         |                  |
|                    | Avgpool             |        | Global                  |                  |
|                    | Connected           |        | 1000                    |                  |
|                    | SoftMax             |        |                         |                  |

**Figure 8.** Darknet53 architecture diagram.

Nevertheless, the rigid residual stacking design of vanilla Darknet53 restricts its feature extraction efficiency and lightweight deployment capacity, which provides a clear rationale for evaluating alternative backbone architectures to boost detection performance. Current approaches to enhancing network performance primarily fall into three categories: widening the network, deepening the network, and increasing input resolution. In practice, only one of these dimensions is typically modified to improve network architecture. Simultaneously scaling two or three of these dimensions often necessitates laborious manual hyperparameter tuning to achieve a favorable trade-off between accuracy and computational efficiency.

### 2.3.2. Efficientnet

In the EfficientNet structure, the authors proposed that there should be an intrinsic connection among these three methods. Therefore, they introduced the compound model scaling algorithm, which uniformly optimizes the network width, depth, and resolution by using a set of fixed scaling coefficients. This approach achieves comparable classification accuracy to existing networks while significantly reducing the number of parameters and computational cost of the model [37]. The process of obtaining the optimal parameters of the three dimensions is as follows: First, the expression of the classification network is obtained as shown in Equation (1), where  $N$  represents the classification network,  $X$  represents the input,  $i$  represents the stage,  $F_i$  represents the basic network layer, and  $L_i$  represents the number of repetitions of  $F_i$  in the  $i$ -th stage,  $H_i, W_i, C_i$  represent the height, width and number of channels of the image, respectively. To quantitatively balance the three scaling dimensions, the compound scaling algorithm defines explicit optimization objectives and computational constraints via the mathematical formulas listed below. The problem that the compound model scaling algorithm aims to solve is to maximize the accuracy of the network in Equation (3) under the constraints of model computational cost and parameter limitations as stipulated in Equation (2). The network depth  $d$ , network width  $w$ , and resolution  $r$  in Equation (4) are the parameters to be optimized.

$$N = \odot_{i=1}^s F_i^{L_i} \left( X_{\langle H_i, W_i, C_i \rangle} \right) \quad (1)$$

$$\text{Memory}(N) \leq \text{target\_memory} \quad (2)$$

$$\max_{d,w,r} \text{Accuracy}(N(d, w, r)) \quad (3)$$

$$N(d, w, r) = \odot_{i=1}^s \hat{F}_i^{d \hat{L}_i} \left( X_{\langle r \cdot \hat{H}_i, r \cdot \hat{W}_i, r \cdot \hat{C}_i \rangle} \right) \quad (4)$$

To find the optimal values of the three parameters, the paper introduces the  $\Phi$  parameter and uses  $\Phi$  to represent the three parameters to be optimized, as shown in formulas (5)–(7). To reduce the computational load of grid search, the base is restricted by Formulas (8) and (9). Additionally, the base restriction makes the FLOPS calculation more convenient. The Efficientnet-B0 benchmark network designed through network structure search is shown in Table 3.

$$\text{depth} : d = \alpha^{\varnothing} \quad (5)$$

$$\text{width} : w = \beta^{\varnothing} \quad (6)$$

$$\text{resolution} = \gamma^{\varnothing} \quad (7)$$

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \quad (8)$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1 \quad (9)$$

**Table 3.** Efficientnet-B0 baseline network.

| Stage<br>$i$ | Operator<br>$\hat{F}_i$          | Resolution<br>$\hat{H}_i \times \hat{W}_i$ | Channels<br>$\hat{C}_i$ | Layers<br>$\hat{L}_i$ |
|--------------|----------------------------------|--------------------------------------------|-------------------------|-----------------------|
| 1            | Conv3 $\times$ 3                 | $224 \times 224$                           | 32                      | 1                     |
| 2            | MBConv1, k $3 \times 3$          | $112 \times 112$                           | 16                      | 1                     |
| 3            | MBConv6, k $3 \times 3$          | $112 \times 112$                           | 24                      | 2                     |
| 4            | MBConv6, k $5 \times 5$          | $56 \times 56$                             | 40                      | 2                     |
| 5            | MBConv6, k $3 \times 3$          | $28 \times 28$                             | 80                      | 3                     |
| 6            | MBConv6, k $5 \times 5$          | $28 \times 28$                             | 112                     | 3                     |
| 7            | MBConv6, k $5 \times 5$          | $14 \times 14$                             | 192                     | 4                     |
| 8            | MBConv6, k $3 \times 3$          | $7 \times 7$                               | 320                     | 1                     |
| 9            | Conv $1 \times 1$ & Pooling & FC | $7 \times 7$                               | 1280                    | 1                     |

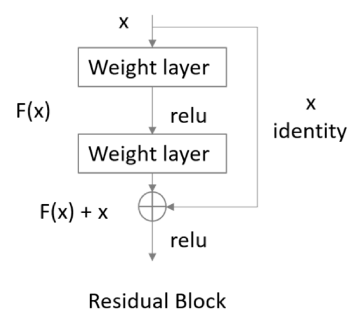
Compared with Darknet, the EfficientNet architecture features fewer layers and parameters. In this experiment, both Darknet and EfficientNet are employed as backbone networks for feature extraction in YOLOv3. The optimal backbone is determined by comparatively evaluating their respective object detection performance.

### 2.3.3. ResNet

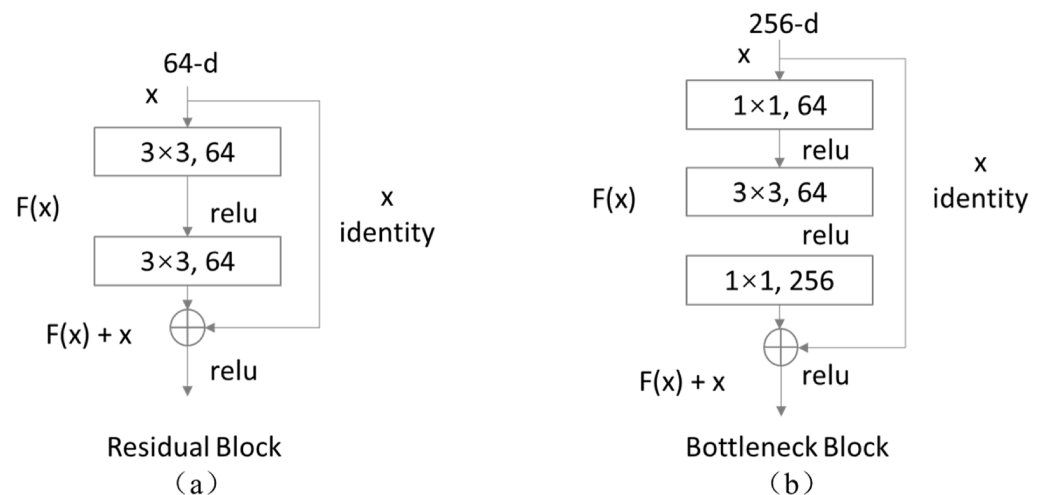
As the depth of the network increases in deep learning, the problem of gradient dispersion becomes more and more serious, making it difficult for the network to converge or even impossible to converge. Currently, this issue can be addressed through methods such as network initialization standardization and data standardization (batch normalization). In addition, the deepening of the network may also be accompanied by a decline in the accuracy of the training set. To address this issue, He et al. introduced the ResNet model—a network structure with cross-layer connections [38]. ResNet constructs residual learning modules upon the VGG architecture by introducing skip connections that bridge shallow and deep layers, thereby enabling training errors from deeper layers to be directly backpropagated to shallower ones. This design effectively mitigates the gradient vanishing problem inherent in excessively deep networks, facilitating the successful training of substantially deeper models. Although VGG19 allocates a substantial portion of its parameters to fully connected layers, ResNet replaces these layers with global average pooling, thereby markedly reducing the total number of trainable parameters.

The residual block structure is shown in Figure 9 and can be expressed by Formula (10) (where  $W_i$  represents convolution). The residual block can be divided into two branches or two mappings: (1) The identity mapping refers to the rightward curved line in Figure 8, which is the mapping of  $x$  itself; (2) The residual mapping refers to the  $F(x)$  part in Figure 8, and its value is  $y - x$ .

$$y = F(x, \{W_i\}) + x \quad (10)$$

**Figure 9.** Residual block structure diagram.

For residual networks of different depths, the authors proposed two types of residual blocks. Figure 10a illustrates the basic residual block, in which the input passes through two successive  $3 \times 3$  convolutions, each with 64 output channels, yielding a 64-channel feature map; the resulting output is then element-wise added to the original input. This block design is primarily employed in relatively shallow architectures, such as ResNet-34. Figure 10b depicts the bottleneck residual block, specifically devised for deeper networks to alleviate computational and memory overhead by reducing dimensionality. Here, the input first undergoes a  $1 \times 1$  convolution projecting it to 64 channels, followed by a  $3 \times 3$  convolution maintaining 64 channels, and finally a  $1 \times 1$  convolution expanding the representation to 256 channels; the resulting 256-channel output is then summed element-wise with the input (after appropriate dimension matching, e.g., via projection). Table 4 summarizes the architectural configurations of residual networks across different depths.



**Figure 10.** Residual block structure diagram. (a) the basic residual block; (b) the bottleneck block.

**Table 4.** Detailed information of residual network structures with different layers.

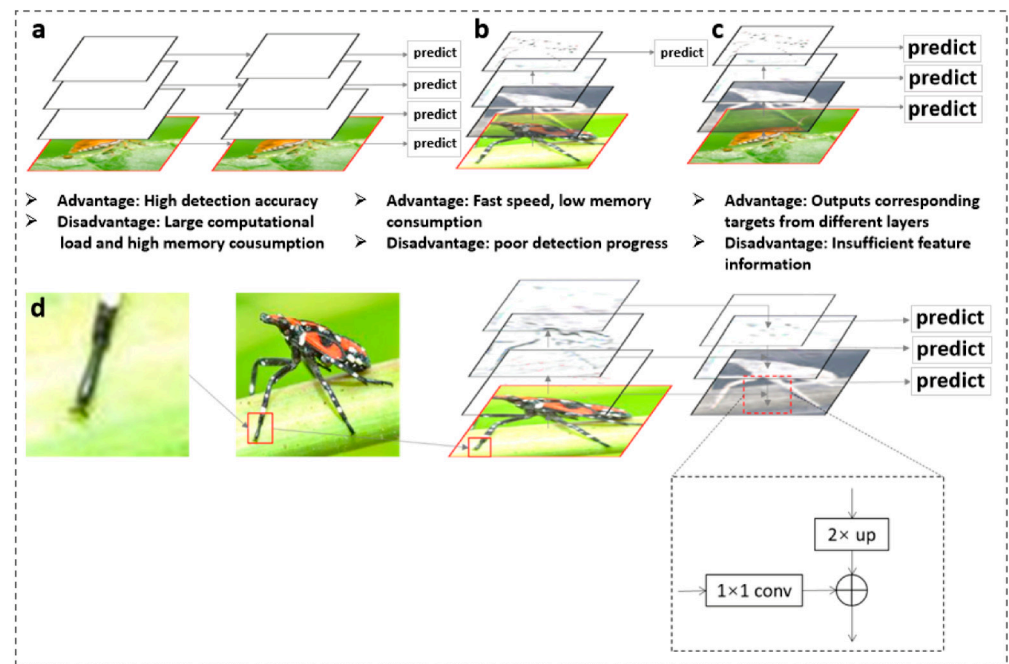
| Layer Name | Output Size      | 18-Layers                                            | 34-Layers                                            | 50-Layers                                                             | 101-Layers                                                             | 152-Layers                                                             |
|------------|------------------|------------------------------------------------------|------------------------------------------------------|-----------------------------------------------------------------------|------------------------------------------------------------------------|------------------------------------------------------------------------|
| Conv1      | $112 \times 112$ | $7 \times 7, 64, \text{stride}2$                     |                                                      |                                                                       |                                                                        |                                                                        |
|            |                  | $3 \times 3, \text{max pool}, \text{stride}2$        |                                                      |                                                                       |                                                                        |                                                                        |
| Conv2_x    | $56 \times 56$   | $3 \times 3, 64$<br>$\times 2$<br>$3 \times 3, 64$   | $3 \times 3, 64$<br>$\times 2$<br>$3 \times 3, 64$   | $1 \times 1, 64$<br>$3 \times 3, 64 \times 3$<br>$1 \times 1, 256$    | $1 \times 1, 64$<br>$3 \times 3, 64 \times 3$<br>$1 \times 1, 256$     | $1 \times 1, 64$<br>$3 \times 3, 64 \times 3$<br>$1 \times 1, 256$     |
| Conv3_x    | $28 \times 28$   | $3 \times 3, 128$<br>$\times 2$<br>$3 \times 3, 128$ | $3 \times 3, 128$<br>$\times 4$<br>$3 \times 3, 128$ | $1 \times 1, 128$<br>$3 \times 3, 128 \times 4$<br>$1 \times 1, 512$  | $1 \times 1, 128$<br>$3 \times 3, 128 \times 4$<br>$1 \times 1, 512$   | $1 \times 1, 128$<br>$3 \times 3, 128 \times 8$<br>$1 \times 1, 512$   |
| Conv4_x    | $14 \times 14$   | $3 \times 3, 256$<br>$\times 2$<br>$3 \times 3, 256$ | $3 \times 3, 256$<br>$\times 6$<br>$3 \times 3, 256$ | $1 \times 1, 256$<br>$3 \times 3, 256 \times 6$<br>$1 \times 1, 1024$ | $1 \times 1, 256$<br>$3 \times 3, 256 \times 23$<br>$1 \times 1, 1024$ | $1 \times 1, 256$<br>$3 \times 3, 256 \times 36$<br>$1 \times 1, 1024$ |
| Conv5_x    | $7 \times 7$     | $3 \times 3, 512$<br>$\times 2$<br>$3 \times 3, 512$ | $3 \times 3, 512$<br>$\times 3$<br>$3 \times 3, 512$ | $1 \times 1, 512$<br>$3 \times 3, 512 \times 3$<br>$1 \times 1, 2048$ | $1 \times 1, 512$<br>$3 \times 3, 512 \times 3$<br>$1 \times 1, 2048$  | $1 \times 1, 512$<br>$3 \times 3, 512 \times 3$<br>$1 \times 1, 2048$  |
|            | $1 \times 1$     | Average pool, 1000 d fc, softmax                     |                                                      |                                                                       |                                                                        |                                                                        |
| FLOPS      |                  | $1.0 \times 10^9$                                    | $3.6 \times 10^9$                                    | $3.8 \times 10^9$                                                     | $7.6 \times 10^9$                                                      | $11.3 \times 10^9$                                                     |

#### 2.3.4. FPN

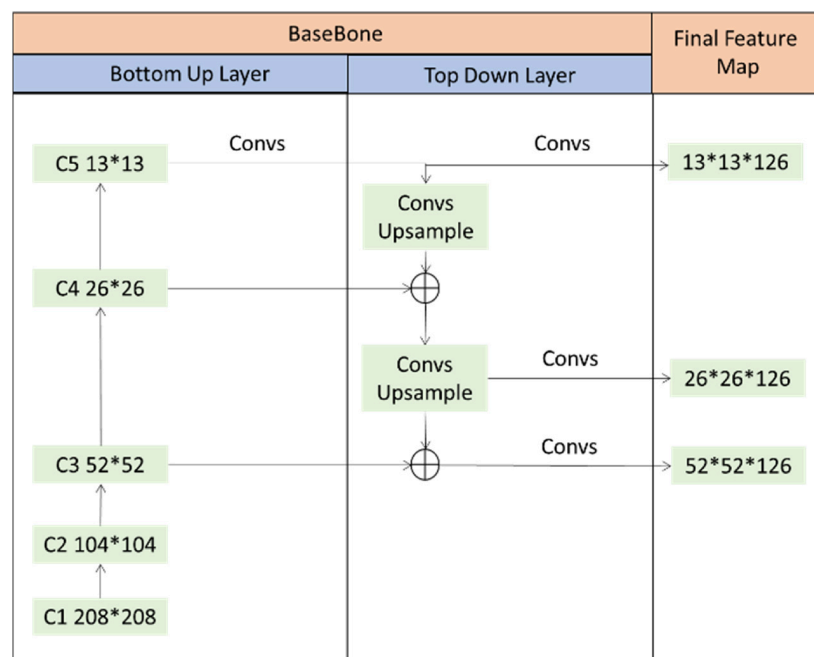
In object detection, whether applied to static images or video sequences, the scale of the target object often exhibits significant variation. Robustly detecting objects across diverse



scales constitutes a fundamental challenge in modern object detection systems. To achieve accurate localization of targets spanning multiple scales, a widely adopted strategy is to construct a feature pyramid that explicitly encodes multi-scale representations [39]. The basic schematic diagram of the featurized image pyramid is shown in Figure 11a. Firstly, an image pyramid is constructed, and then different features are extracted from each layer of the image pyramid for corresponding position prediction. Although this method can achieve relatively high detection accuracy, its disadvantages such as large computational load and high memory requirements often become the performance bottleneck of the entire algorithm. Therefore, this algorithm is rarely used. Subsequently, Single-scale feature maps for object detection were first adopted by Krizhevsky et al. to balance detection speed and computational overhead [40] (as shown in Figure 11b), which are obtained by inputting the original image into a convolutional neural network and using convolution and pooling operations to extract. This results in feature maps at different spatial resolutions. Deeper network layers exhibit lower spatial resolution but capture increasingly abstract, high-level semantic representations, thereby enhancing the accuracy of target detection. Therefore, feature maps from the final convolutional layer are employed for prediction [41]. This approach offers advantages in terms of computational efficiency and low memory footprint. However, a key limitation is its exclusive reliance on high-level, semantically rich features extracted by the deepest layer, while disregarding fine-grained spatial details preserved in earlier, shallower layers. In practice, natural images often contain multiple objects of varying scales. For relatively simple or small-scale objects, detection can be effectively achieved using shallow-layer features; in contrast, detecting complex or large-scale objects typically requires integrating deeper-layer semantic representations. For the reasons mentioned above, as shown in Figure 11c, Liu et al. proposed a pyramidal feature hierarchy structure [42]. First, the original image undergoes deep convolutional processing to extract multi-level feature maps; regression predictions are then performed independently on each of these feature maps. A key advantage of this approach is its ability to generate target predictions at multiple network layers without requiring all layers to be traversed for each prediction, thereby improving detection efficiency and potentially enhancing overall detection performance. A notable limitation, however, is that the extracted features lack sufficient robustness and fail to comprehensively capture target-related information. To address the aforementioned limitations, Lin et al. introduced Feature Pyramid Networks (FPN) [43] (Figure 11d), the FPN architecture comprises a top-down pathway with lateral connections. Specifically, an input image is first processed through a deep convolutional backbone to generate multi-scale feature maps. Shallow (i.e., high-resolution but low-level) feature maps are then projected into lower-dimensional spaces via  $1 \times 1$  convolutions to align their channel dimensions with those of deeper (i.e., lower-resolution but semantically richer) feature maps. Next, the deeper-level feature maps are upsampled via methods such as nearest-neighbor or bilinear interpolation to align their spatial dimensions with those of the corresponding shallower-level feature maps. Finally, the upsampled deeper features are combined with the projected shallow features via element-wise addition to produce fused feature representations. This design enables the network to retain both strong semantic information from deep layers and fine-grained spatial details from shallow layers, thereby yielding more robust and discriminative multi-scale feature representations for object detection. Through this hierarchical fusion mechanism, FPN constructs a semantically enhanced feature pyramid. As shown in Figure 12, it is a schematic diagram of the feature extraction and fusion process of pest images in YOLOv3.



**Figure 11.** Schematic diagram of different scale transformation solutions in object detection. (a–c) Schematic diagram of scale transformation solutions in object detection; (d) Schematic illustration of the Feature Pyramid Network (FPN) architecture.



**Figure 12.** Schematic diagram of feature extraction and fusion process of YOLOv3.

#### 2.4. Evaluation Indicators for Test Results

The targets are generally divided into two categories: positive examples (P) and negative examples (N). Model evaluation metrics are typically represented by four parameters: TP (True Positive), FP (False Positive), FN (False Negative), and TN (True Negative), as shown in Table 5. Here, TP refers to the number of positive examples correctly identified as positive, FN refers to the number of positive examples wrongly identified as negative, FP refers to the number of negative examples wrongly identified as positive, and TN refers to the number of negative examples correctly identified as negative.

**Table 5.** Model evaluation indicators.

|          | <b>P</b> | <b>N</b> |
|----------|----------|----------|
| <b>P</b> | TP       | FN       |
| <b>N</b> | FP       | TN       |

Accuracy refers to the proportion of correct judgments among all judgments, where correct judgments include both positive cases judged as positive and negative cases judged as negative, totaling TP + TN. The accuracy A can be calculated using Formula (11). Precision is relative to the prediction results. It represents the proportion of the number of positive samples predicted as positive among all samples predicted as positive. The samples predicted as positive include TP which are the positive samples correctly predicted as positive and FP which are the negative samples wrongly predicted as positive. The precision P can be calculated by Formula (12). Recall refers to the proportion of all positive samples that are correctly predicted among all positive samples. Among them, there are TP positive samples predicted as positive, and the total number of positive samples includes TP positive samples correctly predicted as positive and FN positive samples wrongly predicted as negative. Therefore, the total number of positive samples is TP + FN. The recall rate R can be calculated by Formula (13).

$$A = \frac{TP + TN}{TP + TN + FN + FP} \quad (11)$$

$$P = \frac{TP}{TP + FP} \quad (12)$$

$$R = \frac{TP}{TP + FN} \quad (13)$$

#### 2.4.1. Average Precision

The Average Precision (AP) metric can comprehensively express the classifier performance of an image segmentation model, being a combined representation of accuracy and recall. It has been widely applied in the evaluation of object detection methods [44,45]. Typically, during the evaluation process, a set of overlapping IoU thresholds is employed, and the resulting metrics are averaged across these thresholds to mitigate bias. Here, overlapping threshold denotes the IoU between the predicted bounding box of a single instance and its corresponding ground-truth bounding box. For instance, AP<sup>50%</sup> represents the AP of instances with an IoU value greater than 50%. Average precision is the integral of precision over recall, and its calculation formula is shown in Equation (14), where P is precision and R is recall. Generally, the higher the AP value, the better the performance of the classifier.

$$AP = \int_0^1 P(R) dR \quad (14)$$

#### 2.4.2. Mean Average Precision

The AP value corresponding to each category of image can be calculated. The Mean Average Precision (mAP) refers to the arithmetic mean of the AP values of all categories that the classifier can recognize. Its calculation formula is shown in Formula (15).

$$mAP = \frac{\sum_{c=1}^C AP(c)}{C} \quad (15)$$

### 2.4.3. Frame per Second

In addition to classification and localization accuracy, inference speed constitutes a critical performance metric for object detection algorithms. For mobile pest identification applications, real-time capability is essential, making computational efficiency a decisive factor in algorithm selection. The standard metric for quantifying speed is frames per second (FPS), defined as the number of images processed per second. Alternatively, per-image inference latency provides a complementary and often more granular measure of speed; lower latency corresponds to higher throughput. Conventionally, an FPS rate exceeding 20 is regarded as sufficient to satisfy the requirements of real-time object detection. In this work, per-image inference latency is adopted as the primary speed evaluation metric.

### 2.4.4. Mean Loss Rate

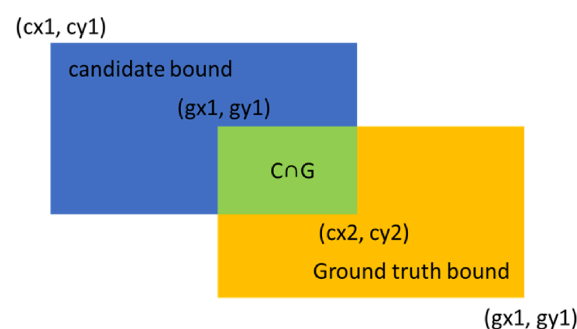
Whether it is object detection or segmentation, the objects are classified, that is, which category the detected objects belong to. In this paper, the proportion of the number of misclassified images of each type of pest in the test dataset to the total number of test images of this type is taken as the category error rate (Loss Rate, LR), and the average recognition error rate (mean Loss Rate, mLR) is obtained by averaging the error rates of each category. The calculation formula is shown in Formula (16), where  $i$  takes an integer in the range of [1, 37] to represent 37 types of pests,  $n = 37$  indicates a total of 37 types of pests, and  $LR(i)$  represents the error rate of each type of pest.

$$mLR = \frac{\sum_{i=1}^{37} LR(i)}{n} \quad (16)$$

### 2.4.5. Detection Accuracy

Intersection-over-Union (IoU) is a standard used in object detection to measure the accuracy of the location of corresponding objects in a specific dataset. IoU represents the ratio of the intersection to the union between the generated candidate bounding box and the original ground truth bounding box, as shown in Figure 13. The calculation formula is shown in Equation (17), where  $area(C)$  represents the area of the candidate box and  $area(G)$  represents the area of the ground truth box. The localization accuracy can be measured by comparing the mean average precision under different IoU conditions. The article measures the localization accuracy by comparing the mean average precision obtained by different models under IoU conditions of 0.5, 0.6, and 0.75, respectively.

$$IoU = \frac{area(C) \cap area(G)}{area(C) \cup area(G)} \quad (17)$$



**Figure 13.** Venn diagram of intersection and union.

### 2.5. Experimental Design

The original pest image dataset comprises a total of 4285 images. For the experiments, the dataset was partitioned into training, validation, and test sets in an 8:1:1 ratio. During model training, the data augmentation techniques described in the preceding chapters were applied to improve the model's robustness and generalization capability. To ensure methodological consistency and fair comparison across all models evaluated in this section, identical hyperparameter configurations were adopted for all experiments. A physical batch sample size of 4 was uniformly adopted for all training procedures, which was determined by the hardware VRAM limitation of our experimental device: the multi-scale detection framework matching  $416 \times 416$  high-resolution pest images brings heavy memory overhead, and batch sizes greater than 4 will directly cause out-of-memory failures and interrupt training. To offset the gradient noise potentially induced by small-batch Adam optimization, we adopted multiple coordinated stabilization approaches supported by the hyperparameters listed in Table 6. L2 regularization with a coefficient of 0.0005 limits drastic parameter fluctuations, and a two-stage training pipeline (35 training rounds for partial frozen layers at a learning rate of 0.001, followed by another 35 rounds of full-network fine-tuning at a lower learning rate of 0.0001) achieves progressive and stable parameter updating. We additionally introduced gradient accumulation with a step of 2 as an auxiliary training scheme, which aggregates gradients over two forward passes to reach an effective batch size of 8 and further reduce random gradient oscillation. The experimental framework consists of three complementary components: (i) optimization of the YOLOv3 backbone network, (ii) refinement of anchor box configurations, and (iii) comparative evaluation of alternative network architectures.

**Table 6.** Details of experimental hyperparameters.

| Hyperparameter                                               | Parameter Value |
|--------------------------------------------------------------|-----------------|
| Batch sample size                                            | 4               |
| Gradient accumulation steps                                  | 2               |
| Number of training rounds for frozen layer and partial layer | 35              |
| Learning rate for training of frozen layer and partial layer | 0.001           |
| Number of training rounds for all layers                     | 35              |
| Learning rate for training of all layers                     | 0.0001          |
| L2 regularization                                            | 0.0005          |
| Optimization algorithm                                       | Adam            |

## 3. Results and Discussion

### 3.1. Optimization of Anchor Boxes

Anchor box generation constitutes a critical step in object detection pipelines. Since the aspect ratios of objects vary across different image datasets, anchor box design must be tailored to the specific dataset employed in the actual experiment. To this end, we apply the K-means clustering algorithm to the bounding box annotations stored in the XML files generated during manual image labeling. This capability enables empirical investigation of the aspect ratio and scale distributions of the ground-truth bounding boxes in the experimental dataset. Leveraging the statistical properties of this distribution, we then determine the optimal anchor box sizes and aspect ratios.

The COCO dataset employs nine default anchor boxes, specified as follows: [(10, 13), (16, 30), (33, 23), (30, 61), (62, 45), (59, 119), (116, 90), (156, 198), (373, 326)]. In contrast, the anchor boxes derived via K-means clustering on the same dataset are: [(54, 47), (126, 156),



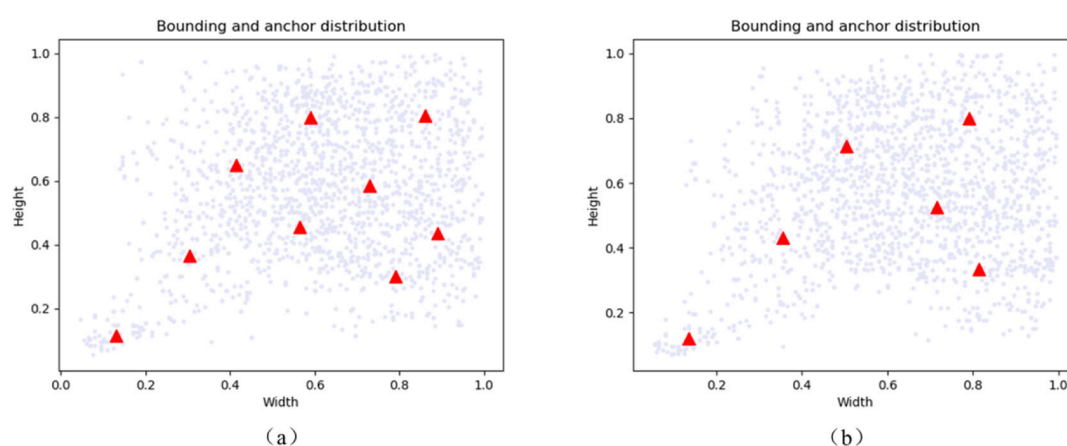
(172, 270), (237, 191), (243, 324), (314, 241), (324, 110), (349, 341), (366, 164)]. In both cases, each tuple denotes the width and height (in pixels) of the corresponding anchor box.

The Tiny version of the COCO dataset has 6 default anchor boxes by default, which are [(10, 14), (23, 27), (37, 58), (81, 82), (135, 169), (344, 319)]. The anchor boxes obtained through K-means clustering in this dataset are [(56, 51), (149, 180), (214, 299), (299, 218), (332, 332), (339, 139)].

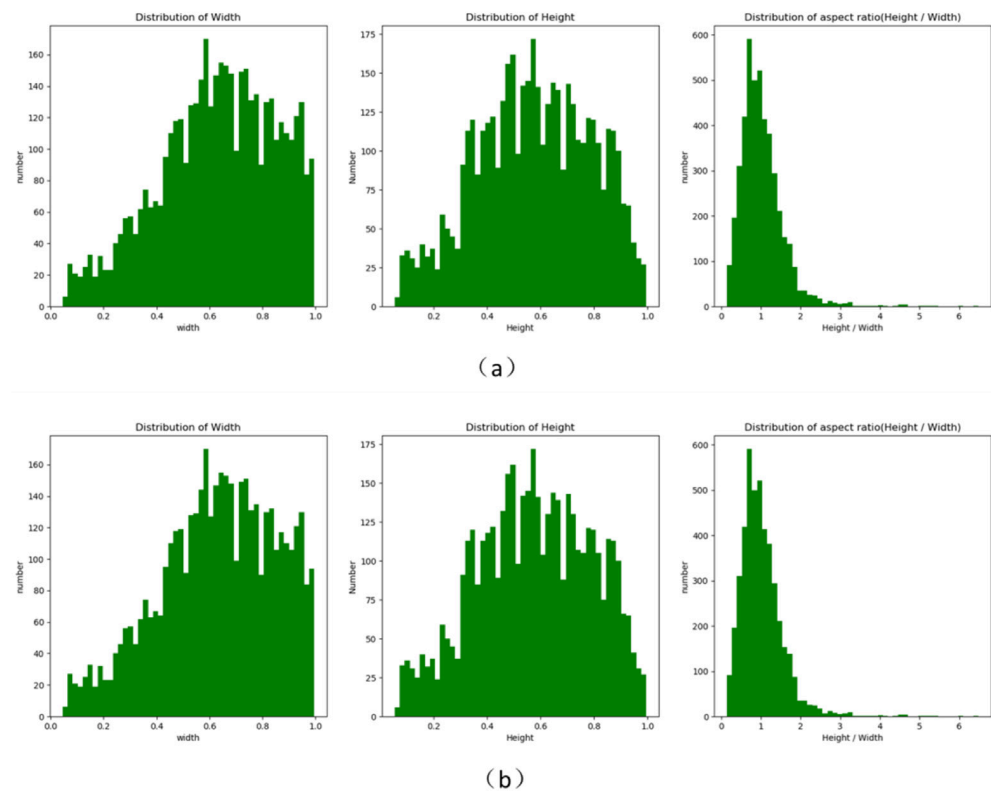
Set the file path (ANNOTATIONS\_PATH) and the number of clusters (CLUSTERS) to be clustered, as well as whether to normalize the object coordinates relative to the image width and height scale (BBOX\_NORMALIZE = False). Read the object coordinates from all the .xml files, then run K-means clustering and save the results. As shown in Table 7, the data distribution of the recommended aspect ratios obtained by the pre-selection box clustering of the dataset in this paper when the clusters are equal to 9 and 6 is presented. Figure 14a,b, respectively, show the clustering results when the number of clusters is 9 and 6, with the red triangles representing the clustering results. Figure 15a,b, respectively, show the distribution statistics of the length and width values and aspect ratios of the data boxes when the number of clusters is 9 and 6.

**Table 7.** Recommended aspect ratios.

| YOLO-Kmeans |        |              | YOLO-Tiny-Kmeans |        |              |
|-------------|--------|--------------|------------------|--------|--------------|
| Width       | Height | Height/Width | Width            | Height | Height/Width |
| 0.130       | 0.115  | 0.9          | 0.135            | 0.120  | 0.9          |
| 0.315       | 0.365  | 1.2          | 0.360            | 0.430  | 1.2          |
| 0.570       | 0.455  | 0.8          | 0.815            | 0.335  | 0.4          |
| 0.830       | 0.320  | 0.4          | 0.505            | 0.720  | 1.4          |
| 0.415       | 0.650  | 1.6          | 0.710            | 0.525  | 0.7          |
| 0.675       | 0.635  | 0.9          | 0.800            | 0.800  | 1.0          |
| 0.845       | 0.510  | 0.6          |                  |        |              |
| 0.580       | 0.830  | 1.4          |                  |        |              |
| 0.860       | 0.805  | 0.9          |                  |        |              |



**Figure 14.** The K-means clustering results of the pest dataset preselection box. (a) the clustering result when the number of clusters is equal to 9; (b) the clustering result when the number of clusters is equal to 6. In the figure, the light purple scatter points represent all ground-truth bounding boxes within the dataset, and the red triangles correspond to anchor boxes generated by K-means clustering.



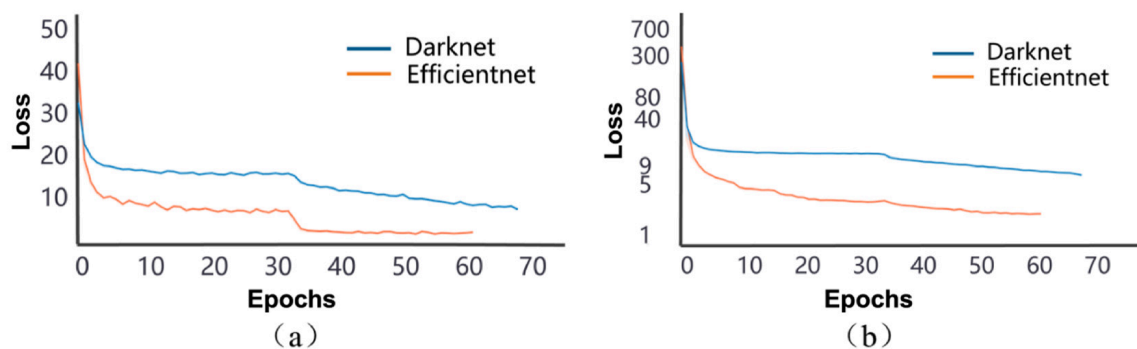
**Figure 15.** Distribution chart of length and width values, as well as aspect ratios, for the preselected bounding boxes across different clusters. (a) the distribution statistics of the width and height values and the aspect ratio of the preselected boxes when the number of clusters is 9; (b) the distribution statistics of the width and height values and the aspect ratio of the preselected boxes when the number of clusters is 6.

### 3.2. Analysis of Object Detection Results of the YOLOv3 Backbone Feature Extraction Network

DarkNet-53 and EfficientNet were employed as the backbone networks for feature extraction. The pest dataset was loaded into the model for training, and the test set was used to evaluate the performance of target classification and recognition under each backbone architecture. As shown in Figure 16, the loss curves for both backbone networks exhibit a consistent downward trend during training and validation, ultimately converging to stable values. This convergence indicates that both architectures possess strong fitting capacity and generalization ability on the proposed model and the pest dataset used in this study. Moreover, a comparative analysis of the loss curves reveals that EfficientNet achieves faster loss reduction and reaches convergence earlier than DarkNet-53. Among them, the loss value of the model with Darknet as the backbone network converges to the interval [9.1, 9.2] as the number of iterations increases, while the loss of the model with Efficientnet as the backbone converges to the interval [2.4, 2.5]. When Efficientnet is used as the backbone, the final loss value of the network is much smaller than that of the model with Darknet as the backbone.

After ensuring the generalization ability of the fitting, the overall performance of the model under different backbone networks is analyzed to obtain the optimal backbone network. This model is to identify 37 types of pests, so the mean average precision  $mAP = \sum_{i=1}^{37} AP(i)$  (where  $i \in [1, 37]$ ,  $AP(i)$  represents the average precision of each type). The table in Section 3.3 presents a comparison of the performance of different backbone feature extraction networks when IoU is set to 0.5, including the mean average localization accuracy, average recognition error rate, and the time consumed for testing each pest image. It can be seen from Table 8 that the mean average accuracies of the two backbone networks

are comparable. Under the condition of a single NVIDIA GeForce RTX 5080Ti GPU, the detection time for a single pest image only differs by 0.006 s. The average recognition error rate obtained under the Efficientnet network is 0.432% lower than that of Darknet. Meanwhile, under the same number of training iterations, the original YOLOv3-Darknet baseline has 60.68 M parameters and 134.09 GFLOPs computational consumption, while our improved YOLOv3-EfficientNet-B0 only retains 14.35 M parameters with 31.44 GFLOPs. The sharp drop in model complexity shortens the training time per epoch from 3.77 min to 2.08 min on NVIDIA GeForce RTX 5080 Ti, corresponding to a 48.98% reduction in training duration. Therefore, it can be concluded that the YOLOv3 model with Efficientnet as the backbone has better overall performance. Therefore, the EfficientNet backbone significantly reduces computational costs while maintaining high accuracy, a performance characteristic consistent with prior lightweight agricultural detection research [46,47]. This makes it feasible to deploy real-time monitoring systems on resource-limited edge devices in farmland, such as portable diagnostic instruments, thereby advancing food safety control nodes to the forefront of production [48].



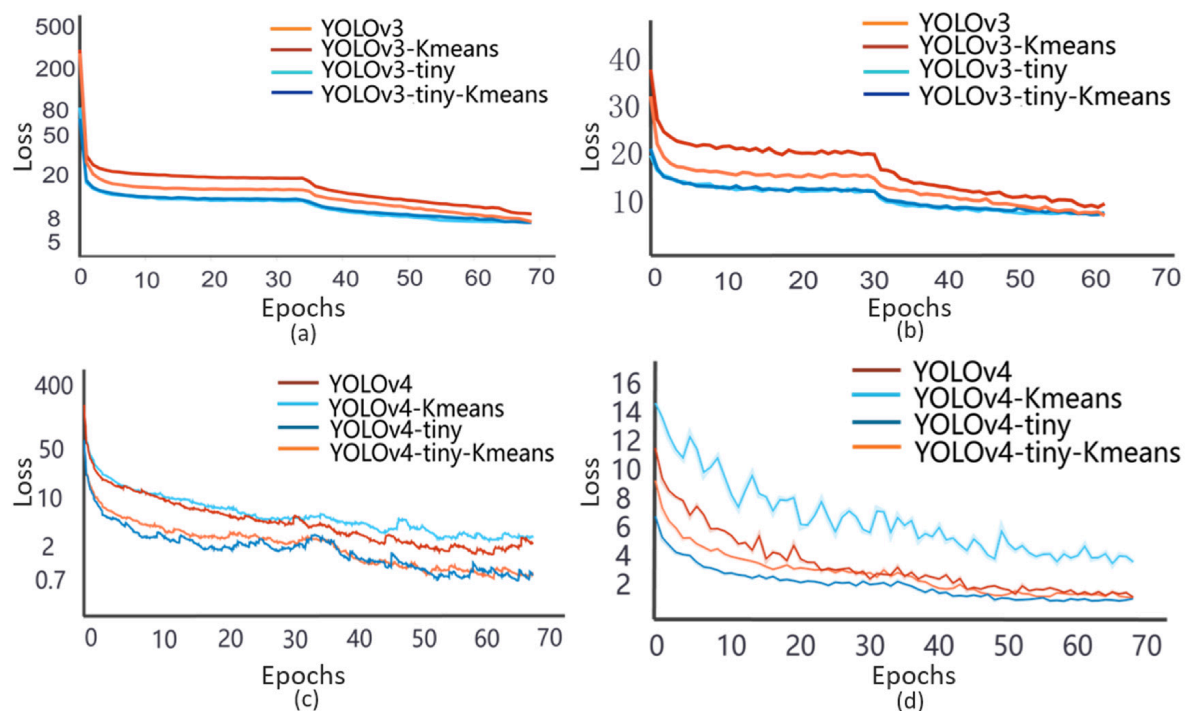
**Figure 16.** Training and validation loss curves of YOLOv3 with different backbone networks. (a) Training loss curves of YOLOv3 with different backbone networks; (b) Validation loss curves of YOLOv3 with different backbone networks.

**Table 8.** Comparison of performance test results of YOLOv3 with different backbone networks.

| Main Trunk Feature Extraction Network | Total Params (M) | GFLOPs (416 × 416) | Mean Accuracy (%) | Mean Recognition Error Rate (%) | Training Time (min) | Testing Time (s) |
|---------------------------------------|------------------|--------------------|-------------------|---------------------------------|---------------------|------------------|
| Darknet                               | 60.68            | 134.09             | 98.93             | 2.00                            | 245                 | 0.054            |
| Efficientnet                          | 14.35            | 31.44              | 98.89             | 1.57                            | 125                 | 0.048            |

### 3.3. Analysis of Detection Results with Different Anchor Settings

The anchor box parameters derived from the COCO dataset and those obtained via K-means clustering on the damaged insect image dataset were separately employed to configure the corresponding hyperparameters for these four models, followed by model training and validation. The results are presented in Figure 17. Specifically, Figure 17a–d depict the training and validation loss curves of YOLOv3, YOLOv4, YOLOv3-tiny, and YOLOv4-tiny, respectively, under both anchor box configurations. In all cases, the loss curves exhibit a consistent downward trend and ultimately converge, indicating that all four network architectures achieve robust fitting performance and satisfactory generalization capability on the insect damage dataset introduced in this work.



**Figure 17.** Training and validation loss curves of different YOLO models under different anchor box parameter settings. (a) Training loss curves of YOLOv3 and YOLOv3-Tiny under different anchor box settings; (b) Validation loss curves of YOLOv3 and YOLOv3-Tiny under different anchor box settings; (c) Training loss curves of YOLOv4 and YOLOv4-tiny; (d) Validation loss curves of YOLOv4 and YOLOv4-tiny.

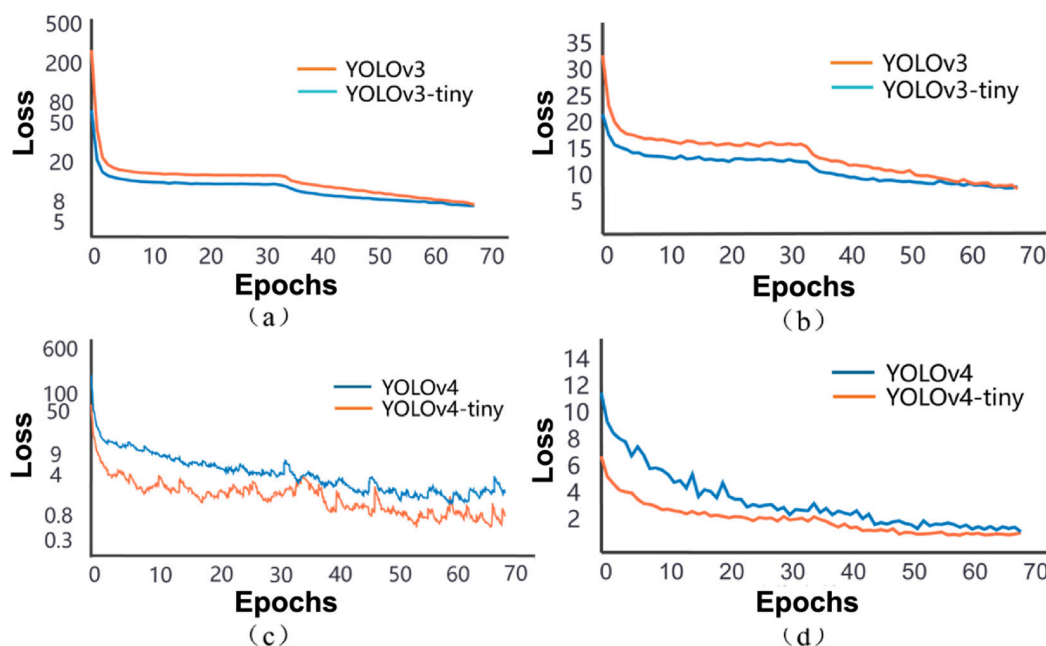
After verifying the model's generalization capability, a comparative analysis was performed to evaluate the overall performance of different network architectures under varying anchor box configurations; the results are summarized in Table 9. With regard to training time and inference time per pest image, the four network architectures evaluated under two distinct anchor box settings exhibited only minor numerical variations, with no prominent runtime gaps between the two anchor schemes. For YOLOv3, training time increases marginally from 245 to 248 min, while inference latency remains constant at 0.054 s. YOLOv4 exhibits identical training durations (549 min) under both anchor schemes, with inference latency varying only slightly from 0.079 to 0.070 s. Similarly, the lightweight variants YOLOv3-tiny and YOLOv4-tiny demonstrate negligible absolute differences in both training and inference runtimes across anchor settings, with no substantial or consistent speed degradation attributable to anchor choice. In contrast, consistent and statistically meaningful disparities are evident in recognition quality metrics: across all four network architectures, models trained with standard COCO anchors consistently achieve higher average accuracy and lower average recognition error rates than their counterparts trained with K-means anchors derived from our pest-specific dataset. As a representative case, YOLOv3 trained with COCO anchors attains an average accuracy of 98.93% and a recognition error rate of 2.00%, whereas the same architecture using pest-specific K-means anchors achieves only 97.93% accuracy and a 2.43% error rate. This performance advantage of COCO anchors is robustly replicated across YOLOv3, YOLOv4, YOLOv3-tiny, and YOLOv4-tiny. Consequently, the anchor box configuration from the COCO dataset is deemed more suitable for the proposed model and the target pest image dataset.

**Table 9.** Comparison of performance test results of different network structures under different anchor settings.

| Network Structure | Anchor Box Type | Average Accuracy Rate (%) | Average Recognition Error Rate (%) | Training Time (min) | Test Time (s) |
|-------------------|-----------------|---------------------------|------------------------------------|---------------------|---------------|
| yolov3            | coco anchors    | 98.93                     | 2.00                               | 245                 | 0.054         |
|                   | K-means anchors | 97.93                     | 2.43                               | 248                 | 0.054         |
| yolov3-tiny       | coco anchors    | 98.52                     | 1.81                               | 228                 | 0.030         |
|                   | K-means anchors | 97.93                     | 2.43                               | 248                 | 0.054         |
| yolov4            | coco anchors    | 98.14                     | 1.97                               | 549                 | 0.079         |
|                   | K-means anchors | 94.52                     | 5.76                               | 549                 | 0.070         |
| yolov4-tiny       | coco anchors    | 95.58                     | 2.78                               | 222                 | 0.029         |
|                   | K-means anchors | 94.17                     | 3.78                               | 196                 | 0.038         |

### 3.4. Analysis of Detection Results for Different Network Structures

The pest dataset was separately trained on five object detection models: YOLOv3-Darknet53, YOLOv3-Efficientnet, YOLOv3-tiny, and YOLOv4-tiny. The corresponding test set was employed to evaluate and compare the performance of each model in terms of target classification and localization accuracy. The quantitative results are presented in Figure 18a,b, both YOLOv3 and YOLOv3-tiny exhibit a consistent downward trend in their training and validation loss curves, with losses converging to nearly identical values. This suggests that both architectures possess strong generalization capability on the present pest dataset. Similarly, as illustrated in Figure 18c,d, YOLOv4 and YOLOv4-tiny also demonstrate monotonically decreasing training and validation losses that ultimately converge, indicating robust generalization performance across both models. Notably, however, YOLOv4-tiny achieves faster convergence and attains a lower final loss compared to its full-sized counterpart, YOLOv4.



**Figure 18.** Loss curves for training and validation of different YOLO network structures. (a) Training loss curves of YOLOv3 and YOLOv3-Tiny; (b) Validation loss curves of YOLOv3 and YOLOv3-Tiny; (c) Training loss curves of YOLOv4 and YOLOv4-Tiny; (d) Validation loss curves of YOLOv4 and YOLOv4-Tiny.



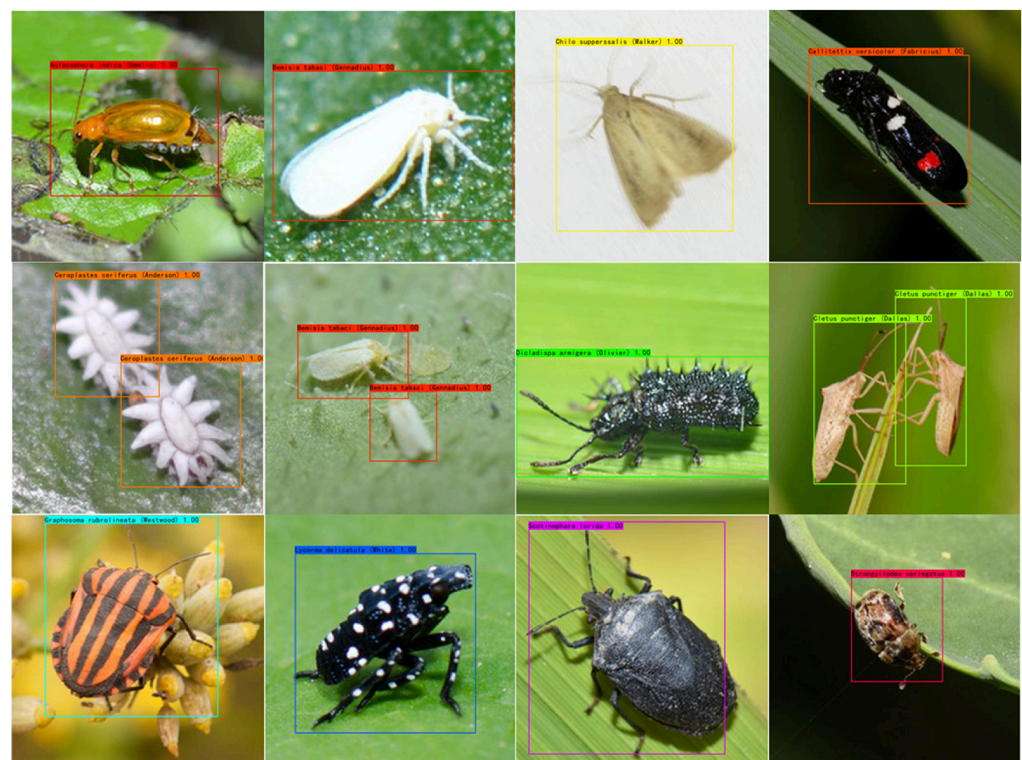
To further quantify the comprehensive detection capacity of different network structures, the mAP, MLR, training time, and the time consumed for testing each pest image were analyzed and compared, respectively. The experimental results are presented in Table 10. In terms of average classification accuracy, YOLOv3-Darknet53 achieves the highest value of 98.93%. The proposed YOLOv3-Efficientnet only suffers a negligible drop of 0.04 percentage points, reaching 98.89%, which ranks second among all models. The accuracy of YOLOv3-tiny and YOLOv4 falls to 98.52% and 98.14%, respectively, while YOLOv4-tiny delivers an obvious performance degradation at merely 95.58%. When evaluating the average recognition error rate, YOLOv3-EfficientNet achieves an optimal value of 1.57%, the lowest among all comparative models. This result surpasses YOLOv3-Tiny (1.81%), YOLOv4 (1.97%), the original YOLOv3-Darknet53 (2.00%), and YOLOv4-Tiny (2.78%), demonstrating that the EfficientNet backbone enhanced with attention mechanisms exhibits superior robustness in distinguishing morphologically similar pest individuals against complex field backgrounds. Beyond superior recognition stability, the proposed architecture also yields prominent gains in training computational efficiency, as quantified by total training runtime. All training experiments were executed on an NVIDIA GeForce RTX 5080 Ti GPU. YOLOv3-EfficientNet completes training in merely 125 min, far outperforming other full-scale detection backbones: YOLOv3-Darknet53 consumes 245 min of training time, whereas YOLOv4 requires as long as 497 min. Though the two lightweight tiny variants reduce total training overhead to 228 min (YOLOv3-tiny) and 196 min (YOLOv4-tiny), their streamlined network design incurs dramatic declines in classification accuracy and anti-interference capability. Furthermore, we validate the inference efficiency gap between all models by comparing single-image testing latency to fully characterize the speed-versus-accuracy trade-off. In terms of per-image inference latency, YOLOv3-EfficientNet processes each pest sample within 0.048 s, representing a clear speed improvement over the original YOLOv3-Darknet53 (0.054 s). While YOLOv3-tiny and YOLOv4-tiny deliver the fastest inference speeds at 0.030 s and 0.029 s respectively, their marginal latency advantage is accompanied by substantial deterioration in detection precision—a typical trade-off for lightweight architectures. On the contrary, the full-size YOLOv4 model suffers the slowest inference performance, with a single-image processing time of 0.079 s.

**Table 10.** Comparison of experimental results on performance of different network structures.

| Network Structure   | Average Accuracy Mean (%) | Average Recognition Error Rate (%) | Training Time (min) | Test Time (s) |
|---------------------|---------------------------|------------------------------------|---------------------|---------------|
| YOLOv3-Darknet53    | 98.93                     | 2.00                               | 245                 | 0.054         |
| YOLOv3-Efficientnet | 98.89                     | 1.57                               | 125                 | 0.048         |
| YOLOv3-tiny         | 98.52                     | 1.81                               | 228                 | 0.030         |
| YOLOv4              | 98.14                     | 1.97                               | 497                 | 0.079         |
| YOLOv4-tiny         | 95.58                     | 2.78                               | 196                 | 0.029         |

Based on the above results, a comprehensive comparison of accuracy, recognition robustness, training cost and inference efficiency confirms that YOLOv3-Efficientnet achieves the best overall pest detection performance, which aligns with the key findings of recent lightweight agricultural detection research that have established EfficientNet-integrated YOLO architectures as optimal solutions for balancing precision and computational efficiency [46,47]. It only sacrifices a tiny amount of classification accuracy to obtain the minimum recognition error rate, while drastically cutting training time and accelerating image inference, striking a superior balance between detection precision and computational burden. Representative visualized detection outputs are displayed in Figure 19.

Consistent with the quantitative metrics, YOLOv3-Efficientnet maintains high classification precision and precise bounding-box localization on both single-pest samples and most multi-pest overlapping images. This well-balanced integration of robust recognition stability and moderate computational overhead confers significant practical deployment value upon the model within the precision agriculture sector, aligning with the core operational requirements of modern large-scale crop plantations and grassroots agricultural monitoring stations [26,49,50]. Conventional full-scale networks like YOLOv3-Darknet53 and YOLOv4 achieve satisfactory detection accuracy but impose excessive hardware requirements, which cannot run stably on low-power portable field terminals and aerial drone edge devices widely used in actual agricultural production. Conversely, while lightweight variants, such as YOLOv3-tiny and YOLOv4-tiny, offer ultra-fast inference speeds, their diminished capacity to discriminate between small-sized and morphologically similar pests results in frequent missed detections and misclassifications under complex field conditions, thereby undermining the reliability required for long-term pest early-warning systems integral to IPM workflows [51]. In contrast, the YOLOv3-EfficientNet architecture proposed in this study achieves a favorable trade-off between robust anti-interference recognition performance and low inference latency. This enables its practical deployment on portable edge devices in agricultural field settings, facilitating real-time, on-site detection and early warning of crop diseases and pests in the future.

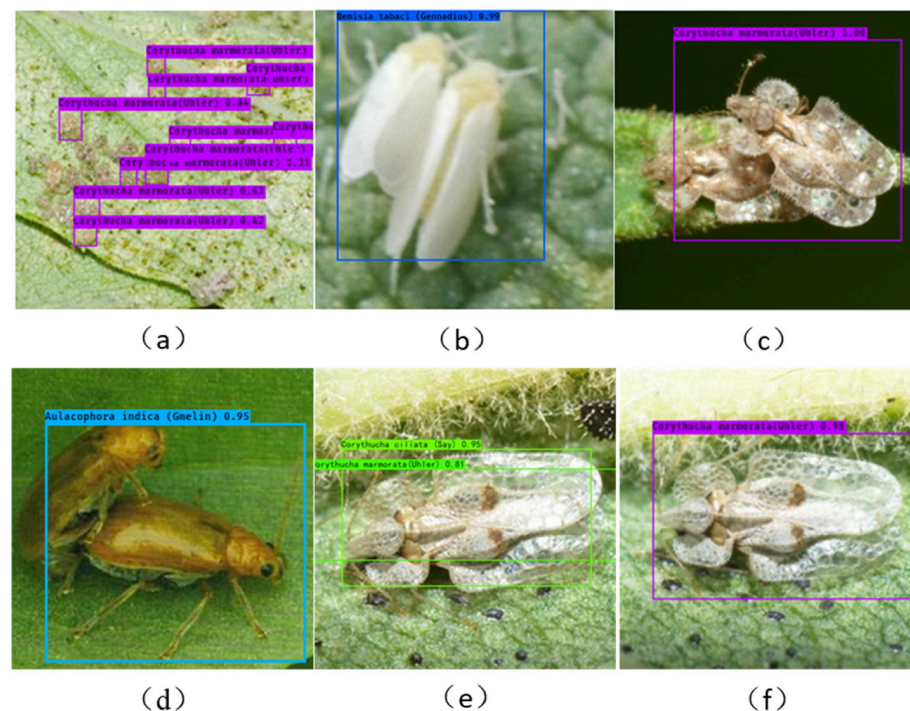


**Figure 19.** Partial image detection results illustrating pest infestations.

### 3.5. Analysis of Abnormal Test Results

In the detection results, a phenomenon of missed detection for small targets was observed, as illustrated in Figure 20a. Small targets exhibit limited pixel-level features, rendering their detection heavily reliant on shallow-layer feature representations [52,53]. In contrast, deeper layers capture more sophisticated semantic representations; however, this enhanced abstraction inherently entails a trade-off between spatial resolution and semantic richness, well-documented limitation in recent YOLO-based detection studies [21]. As YOLO performs predictions based on global contextual features, the progressively

downscaled feature maps in its convolutional backbone pose a critical limitation: by the final layer, the spatial resolution becomes insufficient to accurately localize and regress small objects, thereby contributing to their frequent omission during inference [54]. This finding aligns with EP-YOLO's research on micro-pest detection, which confirms that continuous strided convolutions in YOLO backbones inevitably destroy scarce pixel information of tiny pests as the network deepens [17]. In addition to the inherent small-target missing detection issue caused by feature resolution degradation, the YOLO model also presents a typical prediction defect in dense and overlapping pest scenarios. For multi-object images, the detection results revealed that multiple objects were erroneously merged into a single detection instance, as shown in Figure 20b–d. The YOLO model takes the entire image as the input of the network and divides the entire image into  $S \times S$  grid cells. Each grid cell predicts  $B$  bounding boxes. If the center of a certain object falls within this grid cell, then this grid cell is responsible for predicting this object. In cases where there is object overlap or tight adhesion in the image, the center points of multiple objects may fall within the same grid cell. The model treats multiple objects as a whole for classification and prediction, resulting in the aforementioned abnormal situation, which has been repeatedly validated in dense pest detection applications [12,55].



**Figure 20.** Schematic diagram of abnormal results in insect pest image detection. (a) Small targets missed; (b–d) Inaccurate segmentation of adhered and overlapping images; (e,f) Classification errors.

Abnormal species misclassification cases observed in the detection results are presented in Figure 20e,f. Image classification fundamentally relies on discriminative features, which are automatically learned by the convolutional layers of the model. However, due to the presence of visually similar pest specimens in the experimental dataset, the feature representations extracted during convolutional feature learning become highly comparable, thereby increasing the likelihood of misclassification. Specifically, in our experiments, *Corythucha ciliata* (Say) was erroneously classified as *Corythucha marmorata* (Uhler).

In summary, future improvements targeting the detection accuracy of small and densely clustered objects will enable earlier and more precise identification of pest and



disease infestations. This is crucial for establishing a green production model that prioritizes prevention and reduces pesticide use.

#### 4. Conclusions

Timely and precise pest monitoring is essential for safeguarding crop production and ensuring the stability of intensive agricultural systems. To overcome the limitations of traditional pest monitoring methods in terms of efficiency, accuracy, and real-time performance, this study explores and validates an intelligent pest identification solution based on an improved YOLO framework. The key findings of the study are as follows:

First, in terms of model architecture optimization, replacing the YOLOv3 backbone network with the lightweight and efficient EfficientNet significantly reduces model complexity and training time (by approximately 50%) while maintaining high detection accuracy (mAP@0.5: 98.89%) and achieving a lower average recognition error rate (1.57%). This demonstrates the potential of lightweight networks for real-time field pest monitoring applications in large-scale crop production.

Second, regarding target size adaptability, this study found that, for the pest dataset used, the general anchor box parameters from the COCO dataset outperformed the customized anchor boxes generated by K-means clustering. This suggests that in some crop pest monitoring scenarios, widely validated general prior knowledge may be more practical than complex fine-tuning for specific pest datasets.

Finally, comprehensive evaluations show that the YOLOv3-EfficientNet model strikes the best balance between speed and accuracy, with a mAP@0.5 of 98.89%, an average recognition error rate of 1.57%, and an average detection time per image of only 0.048 s, a total parameter count of merely 14.35 M, and GFLOPs of 31.44 under the  $416 \times 416$  input resolution. Such lightweight computational overhead and outstanding detection performance fully meeting the requirements for real-time pest monitoring in field crop environments.

In conclusion, this study verifies the feasibility of using the improved YOLO model for efficient and accurate pest identification in field crops. This solution provides an automated monitoring tool for front-end pest management in intensive agriculture, enabling early pest detection and precise prevention and control. It helps reduce over-reliance on broad-spectrum chemical pesticides, thereby mitigating the adverse impacts of excessive pesticide use on farmland ecosystems and crop quality. Ultimately, this research provides technical support for promoting sustainable crop production and building precision agricultural systems.

#### 5. Future Prospects

Although this study achieved promising detection performance, several avenues for future investigation remain to further enhance the system's practicality, robustness, and applicability in precision agriculture. It is worth clarifying that all performance evaluations of the proposed model in the current research are limited to offline tests on static annotated image datasets, and we have not yet carried out systematic continuous verification and long-term operational testing under authentic open-field agricultural environments. Real farmland scenarios contain complex unconstrained variables that static datasets cannot fully replicate, including cross-regional differences in crop varieties, drastic diurnal and seasonal fluctuations of natural illumination, random leaf occlusion, variable weather conditions, and dynamic pest population distributions across different growth cycles of field crops. We will deploy the optimized YOLOv3-EfficientNet model on portable field-monitoring devices and agricultural drones, and conduct multi-regional, multi-season, on-site, real-time detection trials. During these trials, we will systematically record practical performance metrics such as field-level recognition accuracy, the miss rate

for small pests, and the misclassification rate among visually similar pest species under authentic agricultural production conditions. Subsequently, we will iteratively refine the model using streaming image data collected in situ to bridge the performance gap between offline dataset evaluation and real-world agricultural applications.

Future work could focus on improving model efficiency and deployability to enable real-time inference on resource-constrained edge devices such as smartphones and drones. Such advancements would lower the technical barrier to adoption and broaden the system's operational scope in field settings. At the same time, the integration of multimodal data including hyperspectral and multispectral imaging, as well as environmental sensor measurements is expected to enable early, and even latent-stage, pest detection. This advancement would represent a paradigm shift from conventional visible identification to proactive early warning, thereby holding fundamental significance for safeguarding the quality of food raw materials at the source. To address the current model's limitations, future research should prioritize the development of more robust feature extraction architectures, anchor-free detection frameworks, and fine-grained recognition techniques. Such advancements are essential to improve the model's generalization performance in highly complex, real-world field environments. Ultimately, this detection technology can be seamlessly integrated with pest prediction models, agrological knowledge graphs, and precision agriculture systems, thereby enabling a closed-loop, intelligent plant protection framework that encompasses intelligent sensing, evidence-based decision-making, and precise actuation. This evolution signifies a paradigm shift in research focus: from developing standalone identification tools toward delivering holistic prevention-and-control solutions that unify real-time monitoring, early warning, and adaptive management. Such integration provides a robust technical foundation and systemic support for precision agriculture.

**Author Contributions:** C.S. and H.Z. conceived the project. H.Z. wrote the manuscript, C.W. and X.L. have drafted the work. H.Z. edited the English version. J.T. and J.L. substantively revised it. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by Innovative Research Group Project of National Natural Science Foundation of China (51821003), the Excellent Youth foundation of Shanxi Province (202103021222011), the Key research and development project of Shanxi Province (202202020101002), the fundamental research program of Shanxi Province (202303021211150).

**Data Availability Statement:** The raw data supporting the conclusions of this article will be made available by the authors on request.

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

## References

1. Albahri, G.; Alyamani, A.A.; Badran, A.; Hijazi, A.; Nasser, M.; Maresca, M.; Baydoun, E. Enhancing Essential Grains Yield for Sustainable Food Security and Bio-Safe Agriculture through Latest Innovative Approaches. *Agronomy* **2023**, *13*, 1709. [\[CrossRef\]](#)
2. Lenné, J.; Wood, D. Crop Diversity in Agroecosystems for Pest Management and Food Production. *Plants* **2024**, *13*, 1164. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Savary, S.; Willocquet, L.; Pethybridge, S.J.; Esker, P.; McRoberts, N.; Nelson, A. The global burden of pathogens and pests on major food crops. *Nat. Ecol. Evol.* **2019**, *3*, 430–439. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Rhouma, A.; Hajji-Hedfi, L.; Khaire, P.B. Comprehensive analysis of Botryosphaeriaceae-induced panicle and shoot blight and its management strategies. *DYSONA-Appl. Sci.* **2025**, *6*, 40. [\[CrossRef\]](#)
5. Góngora, C.E.; Silva, M.D. Sustainable Strategies for the Control of Crop Diseases and Pests to Reduce Pesticides. *Agronomy* **2024**, *14*, 2158. [\[CrossRef\]](#)
6. El Fakhouri, K.; Boulamtat, R.; Sabraoui, A.; El Bouhssini, M. The Chickpea Pod Borer, *Helicoverpa armigera* (Hübner): Yield Loss Estimation and Biorational Insecticide Assessment in Morocco. *Agronomy* **2022**, *12*, 3017. [\[CrossRef\]](#)



7. Godavari, H.; Madhuri, E.V.; Tulasi, B.; Manoj, M.S.; Manideep, S.; Keerthika, N.; Paschapur, A.U. Precision Farming Solutions: Integrating Technology for Sustainable Pest Management. *J. Adv. Biol. Biotechnol.* **2024**, *27*, 33–54. [[CrossRef](#)]
8. Singh, A.; Shraogi, N.; Verma, R.; Saji, J.; Kar, A.K.; Tehlan, S.; Ghosh, D.; Patnaik, S. Challenges in current pest management practices: Navigating problems and a way forward by integrating controlled release system approach. *Chem. Eng. J.* **2024**, *498*, 154989. [[CrossRef](#)]
9. Shao, X.; Zhang, Q.; Zhang, B.; Xie, Z.; Xu, K. Biological Control of Insect Pests in Agroecosystems: Current Challenges, Innovative Strategies, and Future Directions. *Agriculture* **2026**, *16*, 597. [[CrossRef](#)]
10. Wu, Y.; Chen, L.; Yang, N.; Sun, Z. Research Progress of Deep Learning-Based Artificial Intelligence Technology in Pest and Disease Detection and Control. *Agriculture* **2025**, *15*, 2077. [[CrossRef](#)]
11. Wang, P.; Abisado, M.; Chen, Y.; Zeng, X. Deep-learning-based Detection of Citrus Pests and Diseases via Image Recognition. *Sens. Mater.* **2025**, *37*, 4691–4704. [[CrossRef](#)]
12. 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](#)]
13. Xiong, P.; Zhang, C.; He, L.; Zhan, X.; Han, Y. Deep learning-based rice pest detection research. *PLoS ONE* **2024**, *19*, e0313387. [[CrossRef](#)] [[PubMed](#)]
14. 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](#)]
15. Li, K.; He, S.; Wang, J. A Pest Detection Algorithm Based on Improved YOLO. In *International Symposium on Intelligence Computation and Applications*; Springer Nature: Singapore, 2024. [[CrossRef](#)]
16. Yang, X.; Shi, J.; Zhang, J. Gated Channel Attention Mechanism YOLOv3 Network for Small Target Detection. *Adv. Multimed.* **2022**, *2022*, 8703380. [[CrossRef](#)]
17. Tang, Y.; Wang, J.; Sheng, W.; Bian, J. EP-YOLO: An Enhanced Lightweight Model for Micro-Pest Detection in Agricultural Light-Trap Environments. *Sensors* **2026**, *26*, 2607. [[CrossRef](#)] [[PubMed](#)]
18. Lv, J.; Li, W.; Fan, M.; Zheng, T.; Yang, Z.; Chen, Y.; He, G.; Yang, X.; Liu, S.; Sun, C. Detecting Pests From Light-Trapping Images Based on Improved YOLOv3 Model and Instance Augmentation. *Front. Plant Sci.* **2022**, *13*, 939498. [[CrossRef](#)] [[PubMed](#)]
19. Liu, B.; He, F.; Du, S.; Li, J.; Liu, W. An advanced YOLOv3 method for small object detection. *J. Intell. Fuzzy Syst. Appl. Eng. Technol.* **2023**, *45*, 5807–5819. [[CrossRef](#)]
20. Huang, X.; Zhou, R. Rice pest detection via multi-scale edge network and wavelet attention enhancement. *Front. Plant Sci.* **2026**, *17*, 1748419. [[CrossRef](#)] [[PubMed](#)]
21. Zhu, Y.; Han, Y.; Yin, Y.; Zhao, S.; Lan, Y.; Huang, D. An improved YOLOv8n model for in-field detection of pests and diseases in pakchoi. *Front. Plant Sci.* **2025**, *16*, 1730683. [[CrossRef](#)] [[PubMed](#)]
22. Wang, X.; Liu, J.; Chen, Q. An advanced deep learning method for pepper diseases and pests detection. *Plant Methods* **2025**, *21*, 70. [[CrossRef](#)] [[PubMed](#)]
23. Han, S.; Liu, W.; Wang, S.; Zhang, X.; Zheng, S. Improving Small Object Detection in Tobacco Strands Using Optimized Anchor Boxes. *IEEE Access* **2025**, *13*, 208763–208772. [[CrossRef](#)]
24. Zhong, Y.Y.; Wang, J.F.; Peng, J.; Zhang, L. Anchor Box Optimization for Object Detection. In *Proceedings of the 2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Snowmass, CO, USA, 1–5 March 2020; pp. 1275–1283. [[CrossRef](#)]
25. Rana, A.; Vaidya, P. YOLO-based deep learning framework for real-time multi-class plant health monitoring in precision agriculture. *Sci. Rep.* **2026**, *16*, 197. [[CrossRef](#)] [[PubMed](#)]
26. Karacaoglu, B.; Sahin, M.E. Optimized YOLO architectures for efficient Kiwi detection in precision agriculture on embedded systems. *Sci. Rep.* **2025**, *15*, 45113. [[CrossRef](#)] [[PubMed](#)]
27. Li, F.; Lu, Y.; Ma, Q.; Yin, S.; Zhao, R. GhostConv CA-YOLOv8n: A lightweight network for rice pest detection based on the aggregation of low-level features in real-world complex backgrounds. *Front. Plant Sci.* **2025**, *16*, 1620339. [[CrossRef](#)] [[PubMed](#)]
28. Dalal, M.; Mittal, P. A Systematic Review of Deep Learning-Based Object Detection in Agriculture: Methods, Challenges, and Future Directions. *Comput. Mater. Contin.* **2025**, *84*, 57–91. [[CrossRef](#)]
29. Yang, C.; Wang, Y.; Yun, L.; Wang, H.; Han, Y.; Chen, Z. IP-YOLOv8: A Multi-Scale Pest Detection Algorithm for Field-Scale Applications. *Horticulturae* **2025**, *11*, 1109. [[CrossRef](#)]
30. Xie, C.; Zhang, J.; Li, R.; Li, J.; Hong, P.; Xia, J.; Chen, P. Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning. *Comput. Electron. Agric.* **2015**, *119*, 123–132. [[CrossRef](#)]
31. Xie, C.; Wang, R.; Zhang, J.; Chen, P.; Dong, W.; Li, R.; Chen, T.; Chen, H. Multi-level learning features for automatic classification of field crop pests. *Comput. Electron. Agric.* **2018**, *152*, 233–241. [[CrossRef](#)]
32. Wen, C.; Guyer, D. Image-based orchard insect automated identification and classification method. *Comput. Electron. Agric.* **2012**, *89*, 110–115. [[CrossRef](#)]

33. Yun, S.; Han, D.; Oh, S.J.; Chun, S.; Choe, J.; Yoo, Y. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019. [\[CrossRef\]](#)
34. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767. [\[CrossRef\]](#)
35. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013.
36. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. [\[CrossRef\]](#)
37. Tan, M.; Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019. [\[CrossRef\]](#)
38. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016. [\[CrossRef\]](#)
39. Pang, Y.; Wang, T.; Anwer, R.M.; Khan, F.S.; Shao, L. Efficient Featurized Image Pyramid Network for Single Shot Detector. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019. [\[CrossRef\]](#)
40. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2012**, *60*, 84–90. [\[CrossRef\]](#)
41. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [\[CrossRef\]](#) [\[PubMed\]](#)
42. Liu, Z.; Gao, G.; Sun, L.; Fang, L. IPG-Net: Image Pyramid Guidance Network for Small Object Detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 13–19 June 2020; pp. 4422–4430. [\[CrossRef\]](#)
43. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017. [\[CrossRef\]](#)
44. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid Scene Parsing Network. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017. [\[CrossRef\]](#)
45. Wang, Z.; Ji, S. Smoothed dilated convolutions for improved dense prediction. *Data Min. Knowl. Discov.* **2021**, *35*, 1470–1496. [\[CrossRef\]](#)
46. Tang, R.; Jun, T.; Chu, Q.; Sun, W.; Sun, Y. Small Object Detection in Agriculture: A Case Study on Durian Orchards Using EN-YOLO and Thermal Fusion. *Plants* **2025**, *14*, 2619. [\[CrossRef\]](#) [\[PubMed\]](#)
47. Wang, Y.; Wu, M.; Shen, Y. Identifying the Growth Status of Hydroponic Lettuce Based on YOLO-EfficientNet. *Plants* **2024**, *13*, 372. [\[CrossRef\]](#) [\[PubMed\]](#)
48. Wang, Y.; Wu, M.; Shen, Y. YOLO-RP: A Lightweight and Efficient Detection Method for Small Rice Pests in Complex Field Environments. *Symmetry* **2025**, *17*, 1598. [\[CrossRef\]](#)
49. Raza, H.; Abu Bakr, M.; Khan, S.D.; Batool, H.; Ullah, H.; Ullah, M. Benchmarking YOLO Models for Crop Growth and Weed Detection in Cotton Fields. *Agriengineering* **2025**, *7*, 375. [\[CrossRef\]](#)
50. Nyakuri, J.P.; Nkundineza, C.; Gatera, O.; Nkurikiyeyezu, K.; Mwitende, G. AI and IoT-powered edge device optimized for crop pest and disease detection. *Sci. Rep.* **2025**, *15*, 22905. [\[CrossRef\]](#) [\[PubMed\]](#)
51. Mu, Y.; Luo, L.; Ni, L.; Sun, Y.; Gong, H. An efficient and accurate pest detection model based on improved YOLOv5s architecture for complex agricultural environments. *Appl. Soft Comput.* **2026**, *187*, 114329. [\[CrossRef\]](#)
52. Wang, P.; Zhao, J. SOD-YOLO: Enhancing YOLO-Based Detection of Small Objects in UAV Imagery. In Proceedings of the 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 11–15 June 2025. [\[CrossRef\]](#)
53. Qiu, Y.; Lin, Z. Lightweight model LMW-YOLO for small object detection in remote sensing images. *Sci. Rep.* **2026**, *16*, 11644. [\[CrossRef\]](#) [\[PubMed\]](#)
54. Yang, L.; Asli, B.H.S. MSConv-YOLO: An Improved Small Target Detection Algorithm Based on YOLOv8. *J. Imaging* **2025**, *11*, 285. [\[CrossRef\]](#) [\[PubMed\]](#)
55. Gao, Q.; Shi, C.; Ji, Y.; Wang, M. Crop Pest Identification and Real-Time Monitoring System Design Based on Improved YOLOv8s. *Sensors* **2026**, *26*, 404. [\[CrossRef\]](#) [\[PubMed\]](#)

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