International Journal of Engineering and Technology Innovation, vol. 15, no. 1, 2025, pp. 11-25

Pest-YOLO: A YOLOv5-Based Lightweight Crop Pest Detection Algorithm

Wanbo Luo[*]

Department of Artificial Intelligence, Leshan Vocational and Technical College, Leshan, China

Received 19 May 2024; received in revised form 17 June 2024; accepted 19 June 2024

DOI: https://doi.org/10.46604/ijeti.2024.13748

Abstract

Traditional crop pest detection methods face the challenge of numerous parameters and computations, making it difficult to deploy on embedded devices with limited resources. Consequently, a lightweight network is an effective solution to this issue. Based on you only look once (YOLO)v5, this paper aims to design and validate a lightweight and effective pest detector called pest-YOLO. First, a random background augmentation method is proposed to reduce the prediction error rate. Furthermore, a MobileNetV3-light backbone replaces the YOLOv5n backbone to reduce parameters and computations. Finally, the Convolutional Block Attention Module (CBAM) is integrated into the new network to compensate for the reduction in accuracy. Compared to the YOLOv5n model, the pest-YOLO model's Parameters and Giga Floating Point Operations (GFLOPs) decrease by about 33% and 52.5% significantly, and the Frames per Second (FPS) increase by approximately 11.1%. In contrast, the Mean Average Precision (mAP50) slightly declines by 2.4%, from 92.7% to 90.3%.

Keywords: pest detection, YOLOv5n, MobileNetV3, CBAM

1. Introduction

China is a major agricultural country with over 1,400 common crop diseases and pests. The characteristics of pest disasters include fast outbreaks and enormous losses [1]. Therefore, timely and accurate detection of pests energizes farmers to avoid losses and increase crop yields through appropriate preventive measures. In 2023, the General Office of the Ministry of Agriculture and Rural Affairs issued the notice of "2023 Action Plan to Ensure a Good Harvest as Seizing Food from Pests". Crop diseases and insect pests such as wheat, rice, and corn are estimated to reappear in 2023. The affected area reached 3.15 billion acres with a year-on-year increase of 24%. Yield losses exceeded 360 billion kilograms. Major diseases and insect pests in rapeseed and soybean are estimated to occur in 110 million acres, an increase of 15% year-on-year, threatening grain harvest and vegetable oil self-sufficiency [2].

Traditional pest monitoring relies on manual identification by insect experts or technicians, which is highly subjective and labor-intensive [3]. Recently, machine vision equipment that integrates image processing and network communication has been widespread. Computer vision technology provides a new method for surveillance of crop pests, significantly improving monitoring efficiency [4]. The intelligent pest detection system integrates artificial intelligence, the Internet of Things (IoT), and big data technology to realize crop pest surveillance [5]. In the system, cameras with artificial intelligence algorithms automatically capture pest images and detect pests. The detection results are transmitted through IoT remotely.

In recent years, deep learning algorithms have been applied widely in computer vision, including pest detection and classification [6]. Deep learning-based methods have demonstrated robust performance, rendering them the optimal solution for agricultural pest detection [7]. Therefore, this paper proposes a lightweight pest detection model that covers a wide range of crop areas, achieving a balance between accuracy and speed. The main contributions of this paper are:

---

[*] Corresponding author. E-mail address: boboluo504@gmail.com

English language proofreader: Yen-Chun Hsieh

(1) A random background data augmentation method is proposed to compensate for the lack of background images. Sufficient background images can reduce the number of false positive instances, thereby increasing the detection accuracy of the model.
(2) A more lightweight backbone, MobileNetV3-light, is obtained by appropriately reducing the channel number of the MobileNetV3-small backbone layers. The MobileNetV3-light backbone replaces the you only look once (YOLO)v5n backbone, reducing the parameters and computations of the detection model significantly.
(3) After replacing the backbone, the Convolutional Block Attention Module (CBAM) is integrated into the new network to compensate for the decline in detection accuracy.
(4) By exploring agricultural IoT technology, a pest detection framework covering numerous crop areas is proposed to meet the practical industry application.

The rest of this paper is organized as follows: Section 2 presents a review of recent relevant literature. Section 3 details the methodology employed in this study. Section 4 presents the results and discussion. Finally, concluding remarks are proposed in Section 5.

2. Related Works

Although deep learning-based pest detection methods have made significant progress, there are still several shortcomings in their practical application in industry. Firstly, some methods involve randomly capturing high-resolution pest images of crop areas through high-definition cameras. The images are then transmitted to servers with considerable computing power for pest detection. Although these methods achieve high detection accuracy, their detection speed is relatively slow. Furthermore, uploading numerous pest images to servers for processing significantly increases the transmitting bandwidth consumption of IoT, thereby limiting pest detection areas.

In addition, some methods utilized models with more parameters and computations to improve detection accuracy. The models necessitate the detection terminal to be equipped with considerable computing power and graphic memory. As a result, these terminals are expensive and difficult to deploy on a large scale in agricultural IoT. Furthermore, public pest datasets suffer from the problem of missing background images and an adequate number of pest images, leading to missed detection and false detection.

In 2023, Liang et al. [8] utilized Visual Geometry Group (VGG)16 and ResNet50 to establish a recognition model with 18 diseases of four crops: apple, corn, grape, and tomato. Single-crop and multi-crop multi-disease detection models were built through data preprocessing, data augmentation, parameter optimization, and cross-validation. The proposed method demonstrated a recognition accuracy rate of 96%, which was superior to that of ResNet50.

Ju et al. [9] proposed a method based on the masked autoencoding learning paradigm for agricultural crop disease and insect pest classification in 2024. The proposed method addressed the accuracy shortcomings of existing detection algorithms. The input crop image was subjected to local random content masking, semantic feature extraction, and global context reconstruction of high-dimensional mapping. The algorithm entirely mined implicit representations of high-level semantics of images to train a more robust model with fewer data samples. Compared to the ResNet50 baseline network, the statistical accuracy rate of the proposed method increased from 90.48% to 95.24%.

In 2023, Yang et al. [10] selected the Keras framework in TensorFlow to implement an image classification system. The focal loss function was adopted to solve the problem of low recognition accuracy caused by imbalanced datasets. ResNet50, ResNet101, MobileNetV2, and VGG16 were utilized as feature extraction backbones to identify seven rice diseases, respectively. The ResNet50 with the focal loss function achieved 98.06% top-1 classification accuracy. Luo and Hu [11] developed a pest detection system in 2024. The system combined deep learning and IoT technologies for remote detection of pests and diseases, thereby improving the efficiency of pest control work. The system utilized the YOLOv5 network and transfer learning to train and learn the characteristics of forest and farmland common pests, thereby achieving efficient detection and identification.

Given the limitations of fruit farmers' ability to identify pests, Wang and Xu [12] proposed a method that can effectively detect multiple categories of pests in 2023. The proposed method used the single-stage object detection algorithm, YOLOv5s, for navel orange pest detection, achieving a Mean Average Precision (mAP50) of 81.46% for nine types of pest detection. Liu et al. [13] proposed a rice disease and pest identification model called multi-scale dual branch (MSDB)-ResNet in 2023. The model utilized a multi-scale dual-branch structure based on ResNet. Based on the ResNet model, ConvNeXt residual blocks were introduced to optimize the calculation proportion of residual blocks. A dual branch structure was constructed to extract disease features of different sizes from the input disease image by adjusting the convolution kernel size of each branch. The MSDB-ResNet achieved a recognition accuracy of 99.10%, which was 2.42 percentage points higher than the original ResNet model.

In 2022, He et al. [14] proposed an improved method for pest detection based on YOLOv5. The proposed method utilized weighted bidirectional feature fusion technology to enrich the semantic information of feature maps at all levels and modified the adaptive anchor calculation method to improve the accuracy. The mAP50 of the proposed model reached 0.923 in 20 economic forest pest categories, and the model inference speed achieved a Frames per Second (FPS) of 64.9. In 2022, Aladhadh et al. [15] proposed an efficient pest detection method that accurately localized the pests and classified them according to their desired class label. The YOLOv5s model was enhanced through several modifications, such as extending the cross-stage partial network (CSP) module, improving the select kernel (SK) in the attention module, and modifying the multiscale feature extraction mechanism. Compared to the YOLOv5 model, the proposed model achieved the best experiment results.

Assiri et al. [16] proposed an automated insect detection and classification using the Pelican optimization algorithm with deep learning called AIDC-POADL in 2024. Firstly, the AIDC-POADL technique employed the DenseNet-121 model to learn complex features within the input images. Furthermore, the hyperparameter selection of the DenseNet-121 model was developed through the use of the Pelican optimization algorithm. Finally, the multilayer perceptron model distinguished the insects into various classes. The experimental outcomes indicated that the AIDC-POADL method yielded superior recognition results compared to other approaches.

In 2024, Hussain and Srikaanth [17] proposed a novel farmland fertility algorithm with a deep learning-based automated rice pest detection and classification (FFADL-ARPDC) technique. Firstly, the FFADL-ARPDC employed bilateral filtering to remove noise and enhance contrast. Furthermore, images of rice crops were processed using the NASNetLarge deep learning architecture to extract image features. Finally, the model accurately categorizes 14 types of pests using an Elman recurrent neural network (ERNN). The FFADL-ARPDC model exceeds existing pest detection methods with an accuracy of 97.58.

3. Methodology

Firstly, a pest detection framework was proposed based on the agriculture IoT. Secondly, a random background data augmentation method was employed to obtain diverse background images. Thirdly, the pest-YOLO algorithm was proposed by replacing the YOLOv5n backbone and integrating CBAMs.

3.1. Detection framework

The integration of agricultural IoT technology with virtual private network (VPN) technology was investigated to establish reliable and secure network connections for detecting pests in large areas [18]. The VPN establishes a private network on a public network for encrypted communication. Therefore, server resources in each crop area are integrated through VPN technology to form a distributed server cluster that energizes training of the pest detection model. A real-time pest detection framework covering numerous crop areas based on YOLOv5n was proposed in this paper, as shown in Fig. 1.

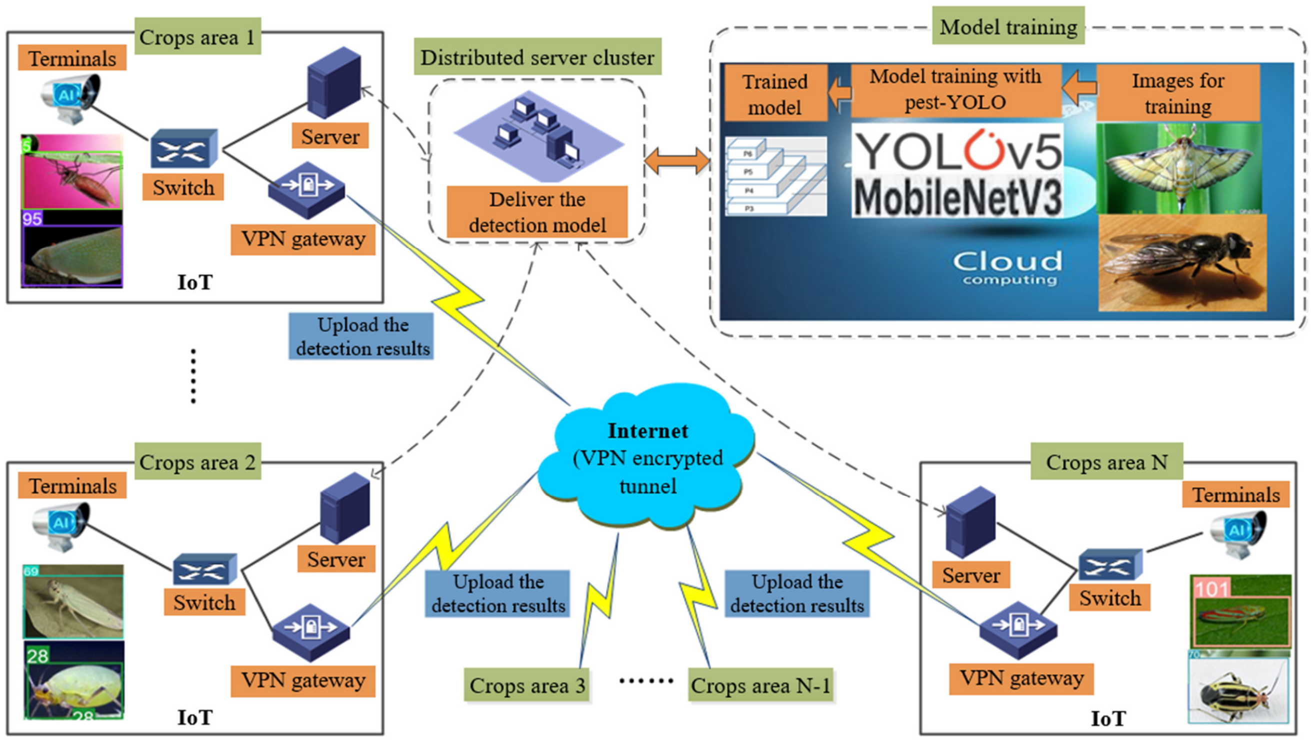


Fig. 1 Crop pest detection framework based on YOLOv5n

First, the distributed server cluster leveraged the public pest dataset to train an initial pest detection model. The model was then transmitted to terminals in crop areas via the IoT. Additionally, terminals with the trained models conducted real-time pest detection. Terminals only uploaded the detection results to the server, significantly reducing the bandwidth consumption of the IoT. Managers regularly collected images in complex environments of various crop areas. These images are annotated and used to train the new model, which increases the robustness of the detection model and improves detection accuracy.

3.2. *Data augmentation*

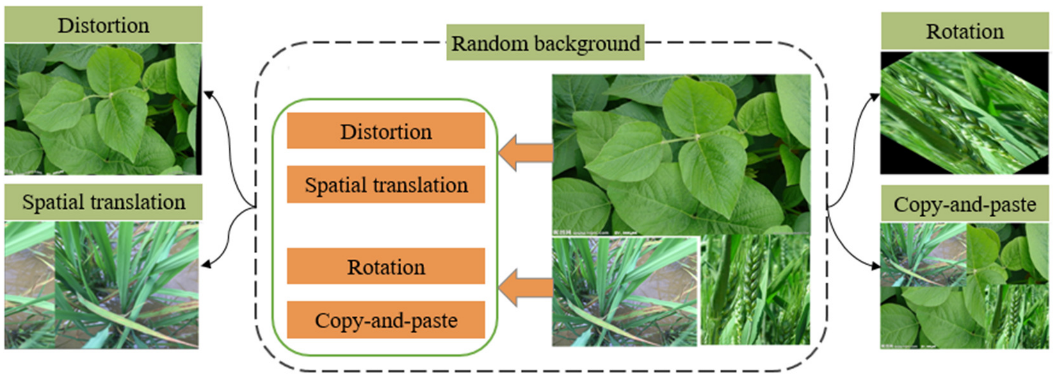


Fig. 2 Random background data augmentation method

This paper employed eight pest classes from the public IP102 pest dataset for model training: rice leaf roller, grub, wireworm, aphids, blister beetle, Miridae, Unaspis yanonensis, and Cicadellidae. IP102 was a large-scale benchmark dataset for pest identification, which annotated 102 pest categories using the PASCAL VOC format [19]. Therefore, the dataset labels were converted to YOLO format. Image rotation, spatial transformation, and distortion methods were used for data augmentation to increase the robustness of the detection model. Since certain pests exhibit a similar coloration to the background, a specific number of background images can help the trained model minimize the impact of the background. Therefore, crop background images, such as crop roots, stems, leaves, etc., were added to the dataset to reduce false positive prediction results, thereby increasing detection accuracy. Since the background image contained no detection objects, this paper proposed a random background data augmentation method to compensate for the lack of background images, as shown in Fig. 2.

The official guidance of YOLOv5 for achieving optimal training outcomes recommends incorporating approximately 1% to 10% of background images to mitigate the false positive rate. Therefore, 50 background images were collected. In addition, the translation method in the OpenCV library was used to achieve spatial transformation of images. Similarly, the rotation method was used to achieve the rotation of images. Sine and cosine functions map pixels to achieve image distortion. Moreover, the scale, crop, flip, rotation, and affine transformation methods were employed to facilitate the copy-and-paste of images. Finally, 50 new background images are randomly generated using the above methods. The final dataset contained 5,904 images, consisting of a training dataset of 5,000 images and a validation dataset of 904 images. The total number of labels in the dataset was 7,403, and the number of labels for each pest class was 190, 676, 533, 1,395, 1,013, 1,268, 546, and 1,782, respectively.

3.3. Pest-YOLO structure

The structure of the proposed lightweight pest-YOLO model is shown in Fig. 3. This structure contains the same three modules as YOLOv5n: the backbone for feature extraction, the neck for enhancing feature extraction and feature fusion, and the head for object prediction. First, data augmentation methods, such as image distortion, rotation, spatial translation, and random background, were used to increase the number of training images. In addition, a MobileNetV3-light backbone with fewer parameters and computations was obtained by appropriately reducing the number of channels of the MobileNetV3-small backbone layers, and the new backbone replaced the YOLOv5n backbone. Finally, CBAMs were effectively integrated into the new backbone to obtain the backbone of the pest-YOLO model. Similarly, the pest-YOLO neck was obtained by inserting CBAMs after concatenation layers from the YOLOv5n neck. The head of pest-YOLO used the same head as YOLOv5n to predict objects.

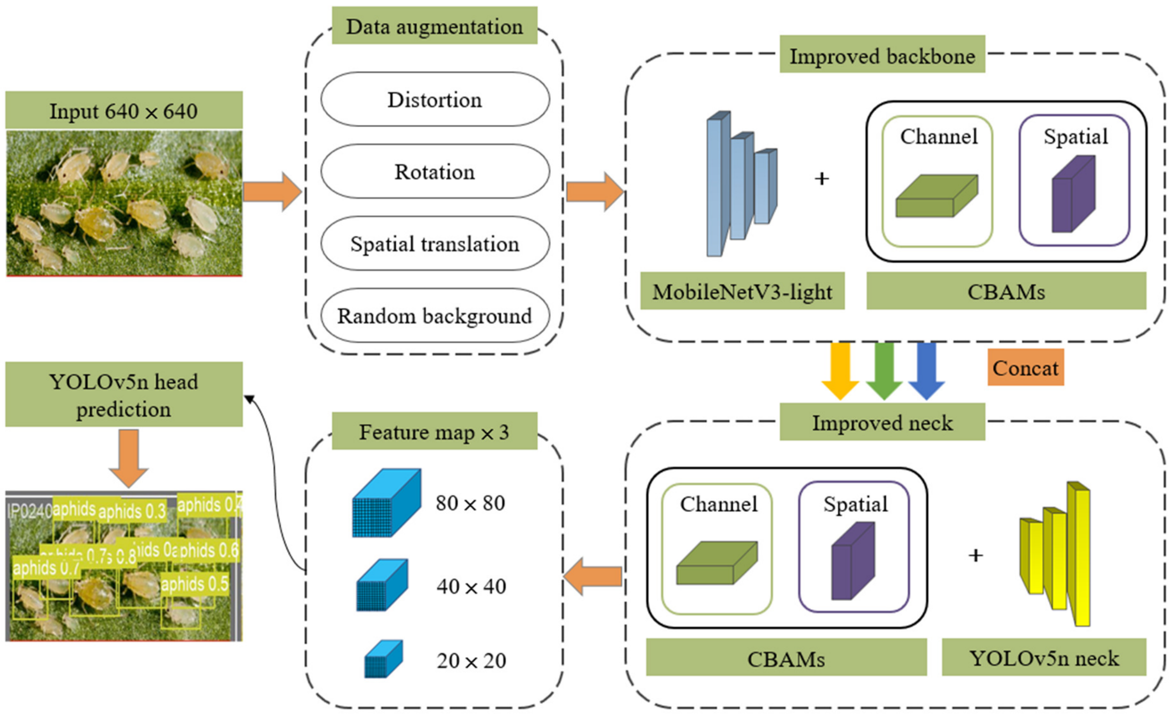


Fig. 3 Pest-YOLO structure

3.4. Replacing backbone

Based on the MobileNetV3-small backbone, this paper further reduced the number of parameters and computations by appropriately reducing the number of channels in the middle feature layer of the backbone to obtain the MobileNetV3-light backbone. A comparison of the two backbone structures (input size is 640 × 640) is shown in Table 1.

Table 1 Structure comparison of two backbones

Structure	MobileNetV3-small backbone			MobileNetV3-light backbone		
	Output shape	Parameters	GFLOPs	Output shape	Parameters	GFLOPs
conv2d	16 × 320 × 320	464	0.10	16 × 320 × 320	464	0.10
bneck 1	16 × 160 × 160	744	0.02	**8** × 160 × 160	600	0.01
bneck 2	24 × 80 × 80	9,280	0.15	**16** × 80 × 80	4,064	0.06
bneck 3	24 × 80 × 80	-	-	**16** × 80 × 80	-	-
bneck 4	40 × 40 × 40	180,032	0.31	**24** × 40 × 40	70,560	0.12
bneck 5	40 × 40 × 40	-	-	**24** × 40 × 40	-	-
bneck 6	40 × 40 × 40	-	-	**24** × 40 × 40	-	-
bneck 7	48 × 40 × 40	-	-	**24** × 40 × 40	-	-
bneck 8	48 × 40 × 40	-	-	**24** × 40 × 40	-	-
bneck 9	96 × 20 × 20	680,040	0.30	**48** × 20 × 20	182,200	0.08
bneck 10	96 × 20 × 20	-	-	**48** × 20 × 20	-	-
bneck 11	96 × 20 × 20	-	-	**48** × 20 × 20	-	-
conv2d	576 × 20 × 20	56,448	0.02	**288** × 20 × 20	14,400	0.01
		Total: 927,008	Total: 0.9		Total: 272,288	Total: 0.38

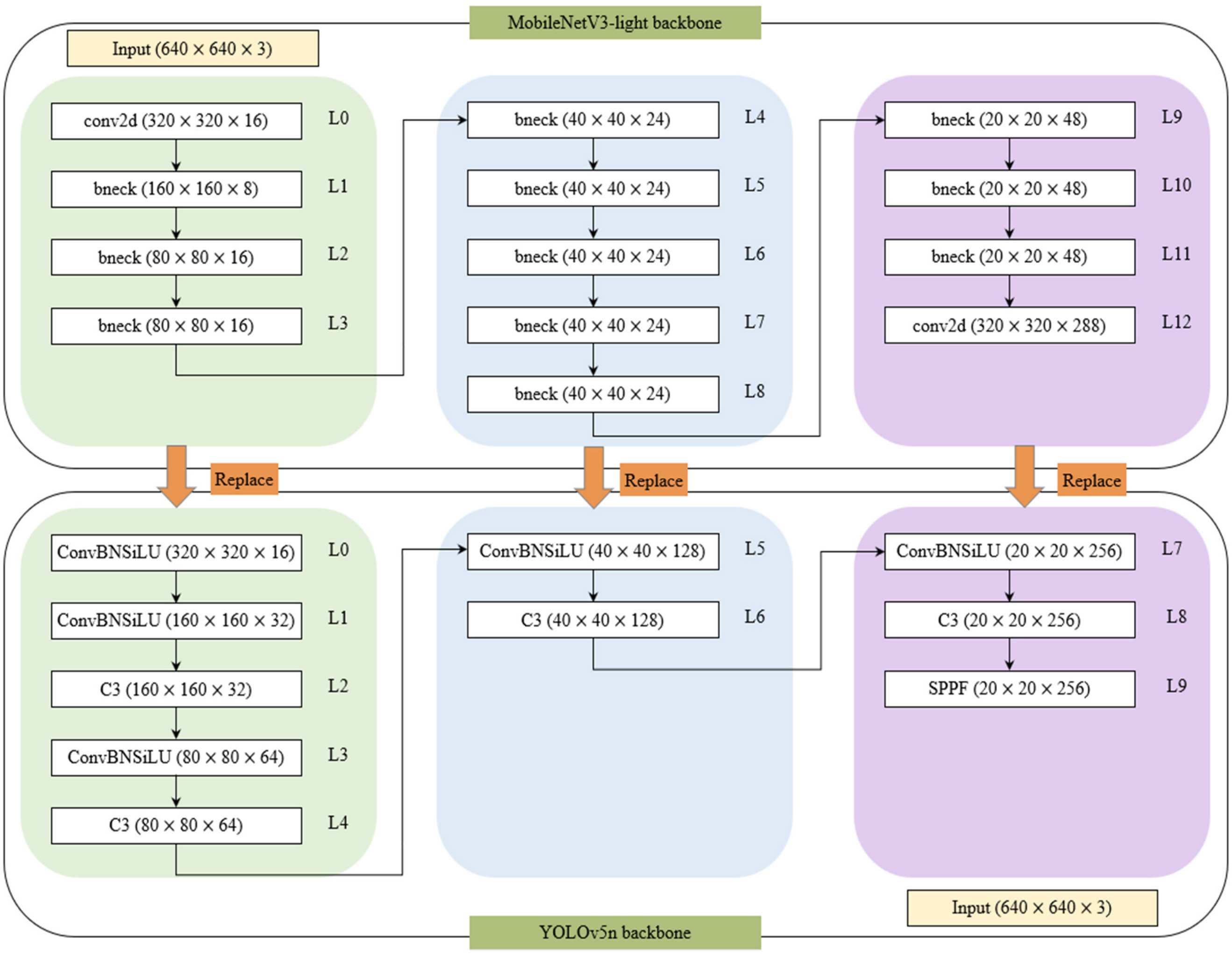


Fig. 4 The method of replacing the backbone

Where the output shape is presented as channel × height × width. In the bneck 1 structure, the number of output channels was reduced from 16 to 8 with a decrease of 144 in parameters. In the case of the bneck 2 and 3 structures, the number of output channels was reduced from 24 to 16, resulting in a reduction of 5,216 in parameters. Similarly, in the case of the bneck 4 to 8 structures, the number of output channels was reduced from 48 to 24, resulting in a reduction of 109,472 in parameters. In the case of the last three bneck structures, the number of output channels was reduced from 96 to 48, resulting in a reduction of 497,840 in parameters. In the last convolutional layer, the parameters were reduced to 42,048. Finally, the two backbones had the same depth, while the final number of output channels of the MobileNetV3-light backbone was reduced by half. The MobileNetV3-small backbone had 927,008 parameters and 0.9 Giga Floating Point Operations (GFLOPs), while the MobileNetV3-light backbone only had 272,288 parameters and 0.38 GFLOPs.

Since the YOLOv5n backbone outputs three feature maps of 80 × 80, 40 × 40, and 20 × 20 sizes for subsequent object prediction, the replaced backbone also outputs these three-size feature maps. First, layers 0 to 3 of the MobileNetV3-light backbone replaced layers 0 to 4 of the YOLOv5n backbone to output a feature map with a size of 80 × 80. In addition, layers 4 to 8 of the MobileNetV3-light backbone replaced layers 5 to 6 of the YOLOv5n backbone to output a feature map with a size of 40 × 40. Finally, layers 9 to 12 of the MobileNetV3-light backbone replaced layers 7 to 8 of the YOLOv5n backbone to output a feature map with a size of 20 × 20. The shape of feature maps represents height × width × channel. Fig. 4 shows the replacement of the YOLOv5n backbone with the MobileNetV3-light backbone.

3.5. Integrating CBAM

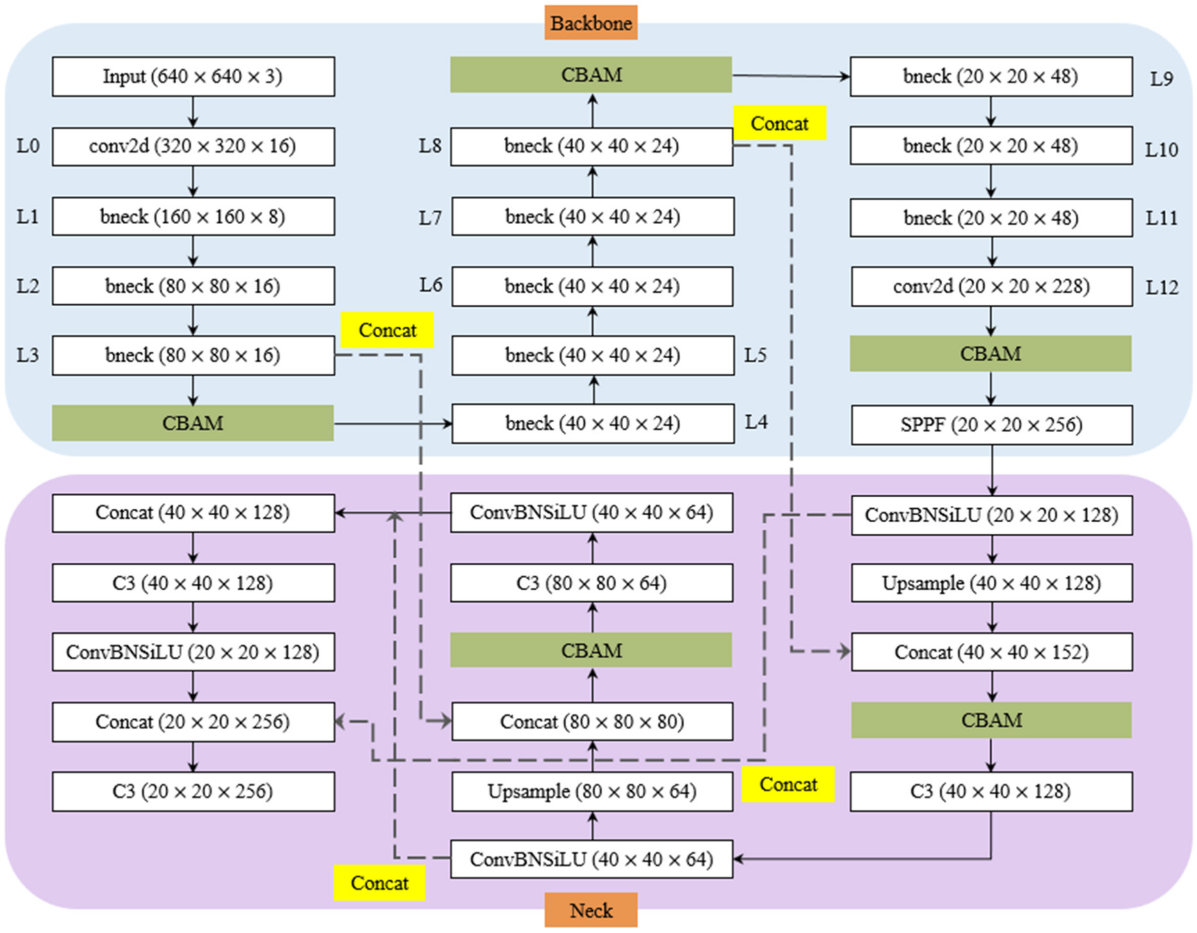


Fig. 5 The method of integrating CBAMs

In computer vision, an attention mechanism pays attention to particular areas and discards irrelevant areas of an image. The channel attention mechanism focuses on what features are, and the spatial attention mechanism focuses on where features are. CBAM is a lightweight module blending these two types of attention, which can be integrated into any neural network to improve performance [20]. The CBAM module sequentially generates attention feature maps in the two dimensions of channel and space. Two feature maps multiply with the previous original input feature maps for adaptive feature correction to generate the final feature map. This paper explored methods of integrating CBAMs through a series of ablation experiments. The effective method is shown in Fig. 5. Where the shape of feature maps represents height × width × channel. In the backbone, CBAMs were inserted after the output feature layer with sizes of 80 × 80, 40 × 40, and 20 × 20, respectively. In the neck, CBAMs were inserted after the Concatenation layer with output sizes of 40 × 40 and 80 × 80, respectively.

4. Results and Discussion

Firstly, three pest detection models, YOLOv5n, YOLOv5n-MobileNetV3, and pest-YOLO, were trained on the customized pest dataset. Furthermore, a comparative analysis was conducted to evaluate the accuracy and complexity of the three models. Finally, the performance of the pest-YOLO algorithm was compared to that of two state-of-the-art lightweight algorithms.

4.1. Model training results

The experiments were conducted on the PyTorch framework with a version of 2.0.1. The version of YOLOv5n was 7.0. The hyperparameters were set to the default values of the YOLOv5n algorithm. The Graphics Processing Unit (GPU) used was the NVIDIA GeForce RTX 3060 with 12 gigabytes of graphics memory. The Central Processing Unit (CPU) used was the Intel Core i7-13700KF with a frequency of 3.4 GHz. The Compute Unified Device Architecture (CUDA) version used was 11.8. The shape of the input image was 640 × 640 × 3. The batch size and training epochs were set to 32 and 500, respectively.

The YOLOv5n, YOLOv5n-MobileNetV3, and pest-YOLO models were respectively trained on the pest dataset. The YOLOv5n model used the original YOLOv5n network, while the YOLOv5n-MobileNetV3 model replaced the backbone of the YOLOv5n network. The pest-YOLO model used an improved network. During the training process of the three models, the losses for bounding box, objectness, and classification gradually decreased, while the metrics for precision, recall, mAP50, and mAP50-95 gradually increased.

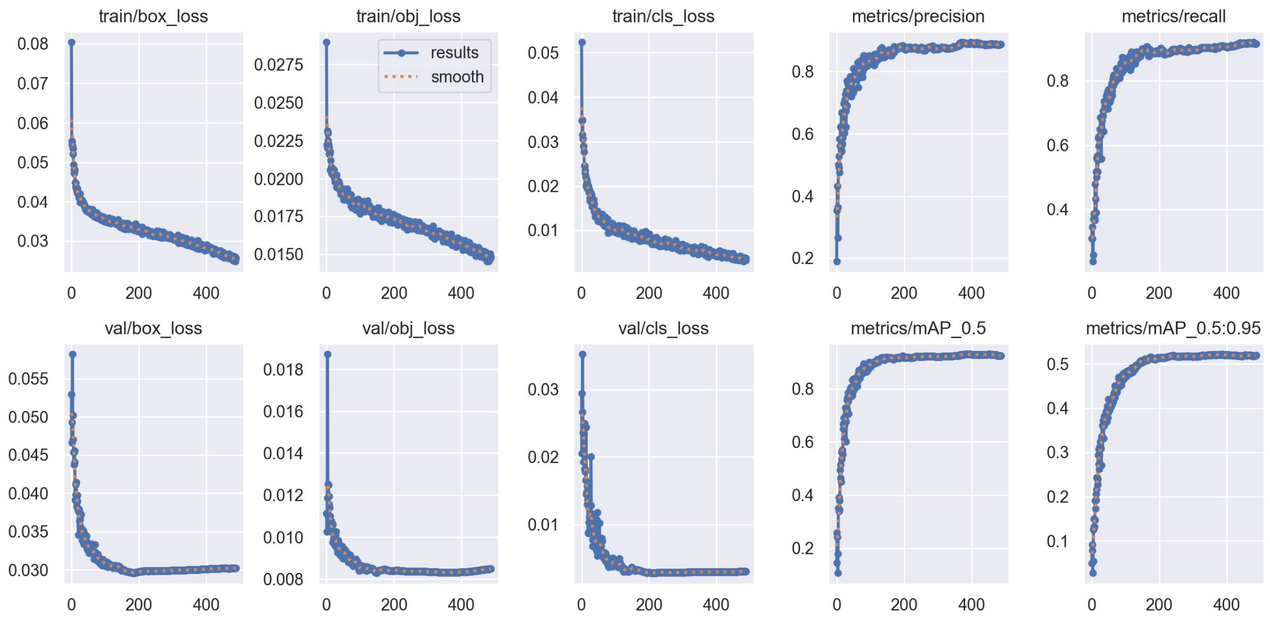


Fig. 6 Training results of the YOLOv5n model

Fig. 6 displays the training results of the YOLOv5n model. During the training process, the bounding box loss, objectness loss, and classification loss all gradually decreased. Similarly, during the validation process, these losses decreased to lower values at the 200th epoch and remained stable during the subsequent training process. The precision metric reached a higher value at the 200th epoch and peaked around the 400th epoch. The recall metric attained a high value at the 200th epoch and gradually increased during the following 300 epochs. Both mAP50 and mAP50-95 metrics achieved a high value at the 200th epoch and changed slightly in subsequent training epochs.

Fig. 7 shows the training results of the YOLOv5n-MobileNetV3 model. During the training process, the trends in three-class loss changes were similar to those of the YOLOv5n model. During the validation process, the YOLOv5n-MobileNetV3 model achieved a low object loss value at the 400th epoch, whereas the YOLOv5n model achieved a low value at the 200th epoch. The precision, recall, mAP50, and mAP50-95 metrics showed similar changes to those of the YOLOv5n model.

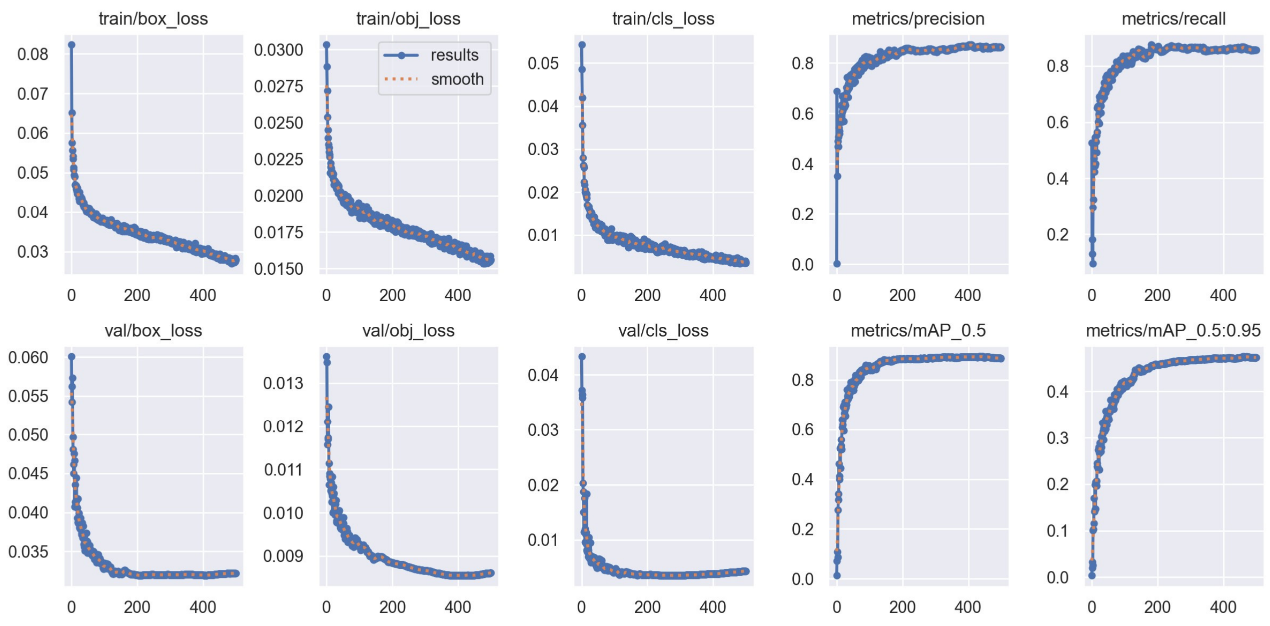


Fig. 7 Training results of the YOLOv5n-MobileNetV3 model

Fig. 8 displays the training results of the pest-YOLO model. During the training and validation processes, the three-class loss trend of the pest-YOLO model was similar to that of the YOLOv5n model. However, the precision metric showed an upward mutation at the 330th epoch, while the recall metric showed a downward mutation. These two metrics did not change significantly during the subsequent training process.

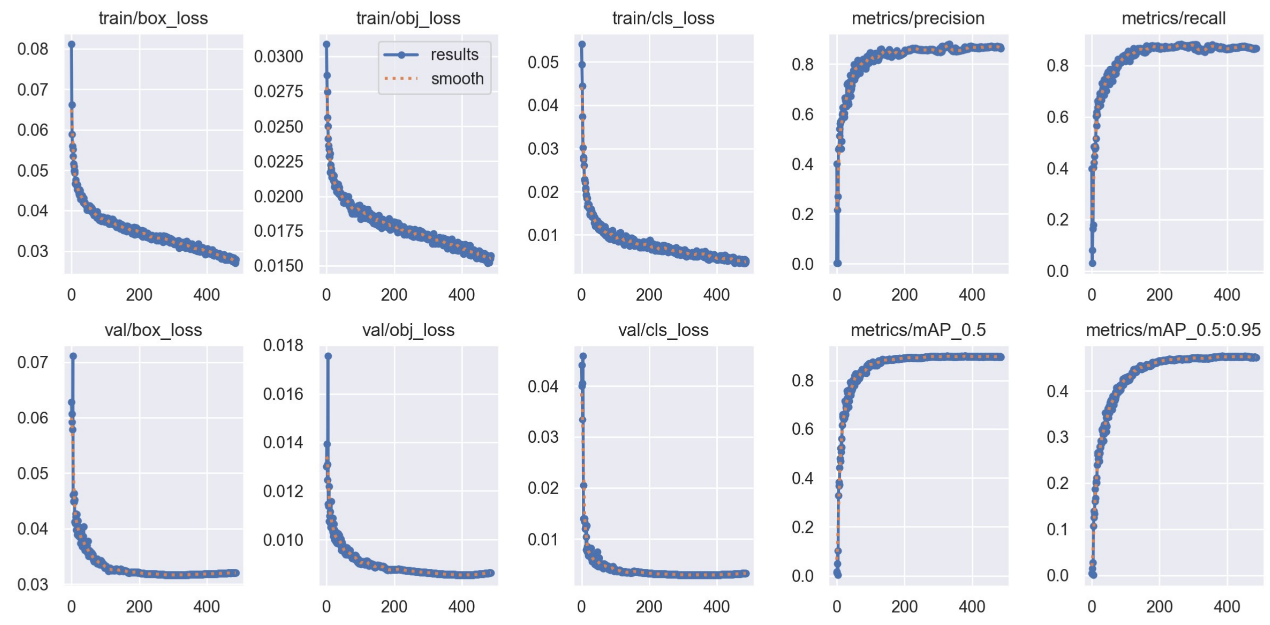


Fig. 8 Training results of the pest-YOLO model

4.2. Validation results

The validation dataset consisted of 904 images and was used to validate the performance of three models. The mAP50 metric is crucial for evaluating model accuracy, with higher values indicating greater precision. Table 2 displays the validation results of all three models across eight classes and their respective parameters for better evaluation.

Table 2 Validation results of three models

Model	Precision	Recall	mAP50	mAP50-95	Parameters
YOLOv5n	0.895	0.91	0.927	0.525	1,769,989
YOLOv5n-MobileNetV3	0.861	0.87	0.895 (**-3.2%**)	0.476	1,171,493 (**-33.8%**)
Pest-YOLO	0.864	0.874	0.903 (**-2.4%**)	0.478	1,185,972 (**-33%**)

The YOLOv5n model achieved a precision of 0.895, recall of 0.91, mAP50 of 0.927, and mAP50-95 of 0.522. The above metrics of the YOLOv5n-MobileNetV3 model were 0.861, 0.87, 0.895, and 0.476, respectively, while the metrics of the pest-YOLO model were 0.864, 0.874, 0.903, and 0.478, respectively. In terms of the parameters metric, the YOLOv5n model had 1,769,989 parameters, compared to 1,171,493 parameters for the YOLOv5n-MobileNetV3 model and 1,185,972 parameters for the pest-YOLO model. Fig. 9 compares the relevant metrics of the three models on the validation dataset. The values of parameters and training time were normalized for evaluation.

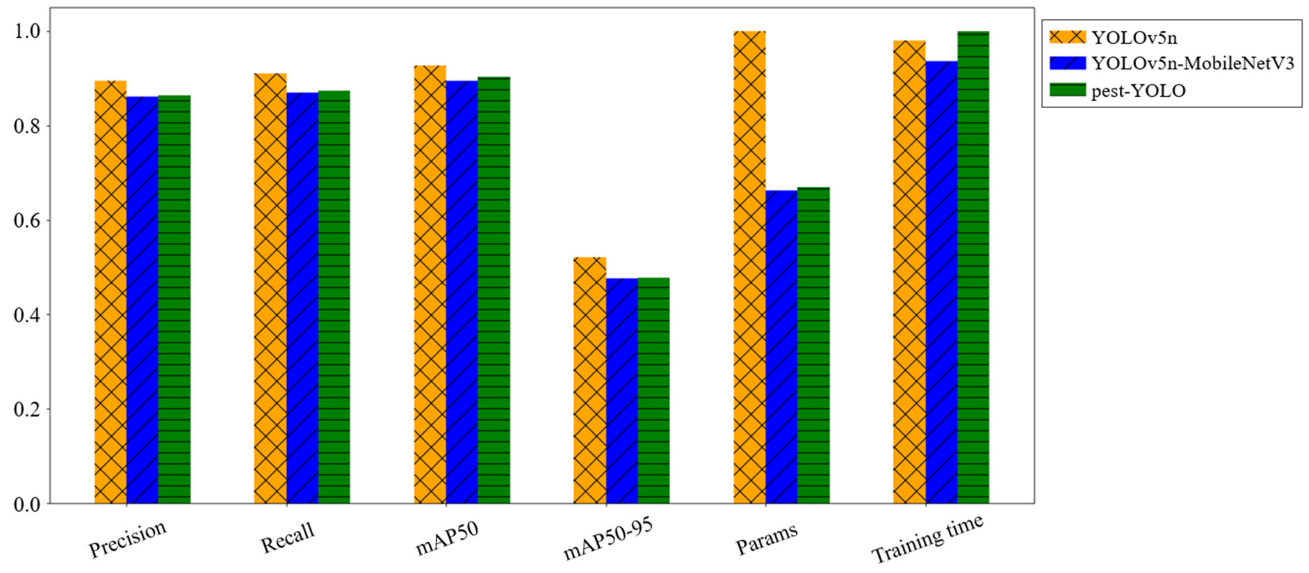


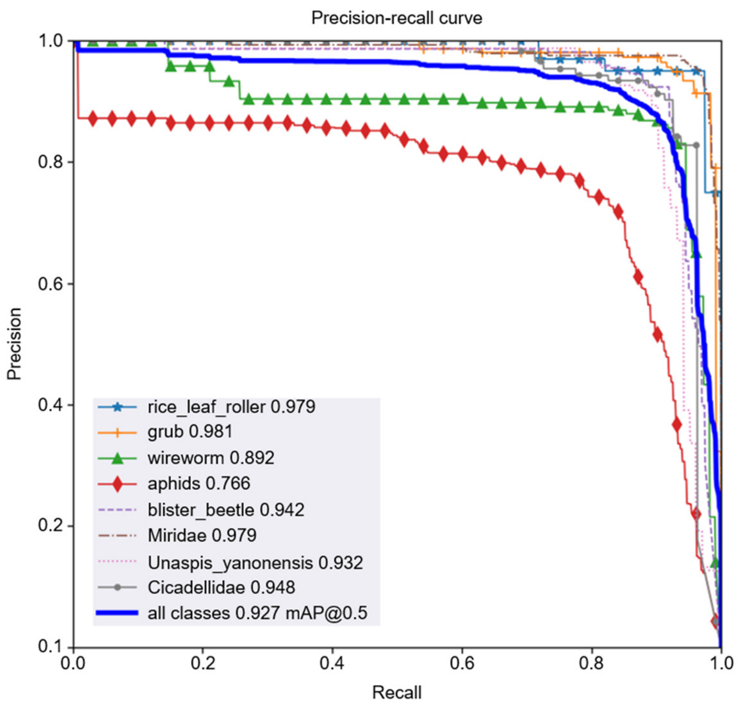
Fig. 9 Comparison of the metrics of the three models on the validation dataset

The YOLOv5n model outperformed other models in terms of detection accuracy. For the crucial mAP50 metric, the YOLOv5n model achieved the highest score of 92.7% compared to the YOLOv5n-MobileNetV3 model with a score of 89.5%, and the pest-YOLO model with a score of 90.3%. However, the parameters of the YOLOv5n-MobileNetV3 and pest-YOLO models decreased significantly by 33.8% and 33%, respectively, compared to the YOLOv5n model. This reduction in parameters results in lower graphic memory consumption. Therefore, the YOLOv5n-MobileNetV3 and pest-YOLO models are suitable for deployment on agriculture IoT terminals. Furthermore, the pest-YOLO model compensated for the decrease in detection accuracy by integrating CBAM of fewer parameters, which resulted in an increase of 0.8% in mAP50 compared to the YOLOv5n-MobileNetV3 model. Additionally, the training times for all three models were similar. Fig. 10 displays the Precision-Recall curve for three models: YOLOv5n, YOLOv5n-MobileNetV3, and pest-YOLO.

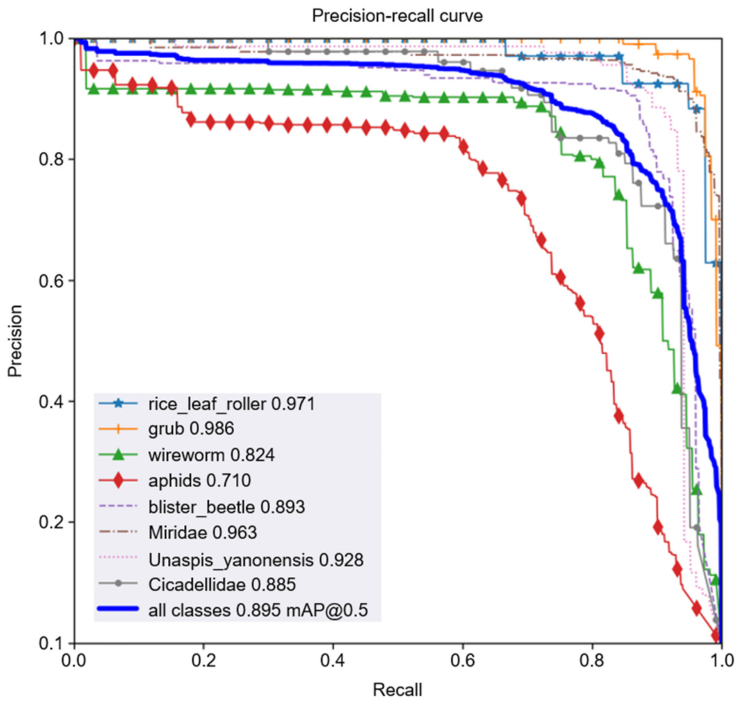
The YOLOv5n, YOLOv5n-MobileNetV3, and pest-YOLO models achieved the mAP50 of 92.7%, 89.5%, and 90.3%, respectively. Fig. 10(a) shows the mAP50 for each class in the YOLOv5n model. For the classes 'rice_leaf_roller', 'grub', 'wireworm', 'aphids', 'blister_beetle', 'Miridae', 'Unaspis_yanonensis', and 'Cicadellidae', the mAP50 values were 0.979, 0.981, 0.892, 0.766, 0.942, 0.979, 0.932, and 0.948. After replacing the backbone, the mAP50 of each class in the YOLOv5n-

MobileNetV3 model decreased, as shown in Fig. 10(b). The mAP50 for each class in the YOLOv5n-MobileNetV3 model was 0.971, 0.986, 0.824, 0.710, 0.893, 0.963, 0.928, and 0.885. In particular, the mAP50 of the 'Cicadellidae' class decreased by 6.3%.

However, after integrating CBAM, the pest-YOLO model paid more attention to the classes with lower mAP50, especially the 'wireworm' and 'Cicadellidae' classes, as demonstrated in Fig. 10(c). The mAP50 for each class in the pest-YOLO model was 0.920, 0.970, 0.858, 0.735, 0.910, 0.972, 0.937, and 0.922. The mAP50 for the 'wireworm' and 'Cicadellidae' classes increased by more than 3% compared to the YOLOv5n-MobileNetV3 model after integrating CBAM, achieving a balanced detection accuracy for each class. In addition, the 'aphid' class had a lower mAP50 than the other classes in all three models due to some aphids being incorrectly predicted as backgrounds.

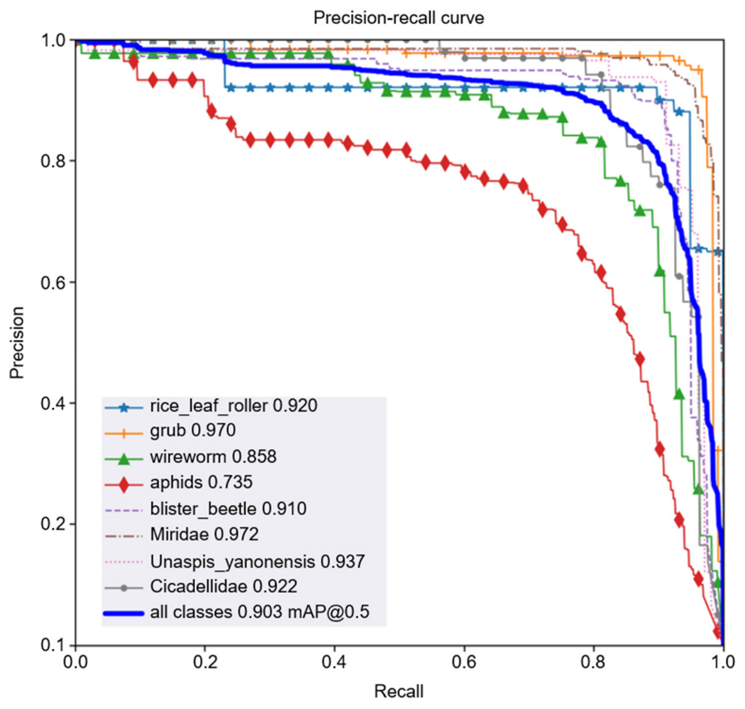


(a) YOLOv5n



(b) YOLOv5n-MobileNetV3

Fig. 10 Precision-Recall curves of three models

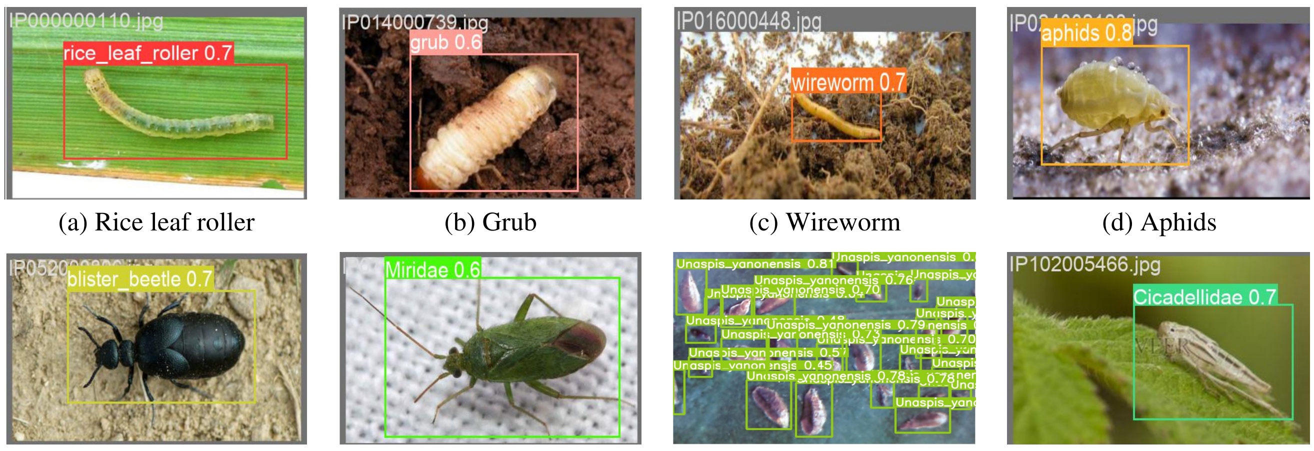


(c) pest-YOLO

Fig. 10 Precision-Recall curves of three models (continued)

4.3. Effectiveness, speed evaluation, and comparative experimental results

Pest images evaluated the effectiveness of the pest-YOLO model. Pests in sample images were enclosed in bounding boxes with confidence values, as shown in Fig. 11. Where, Fig. 11(a) to Fig. 11(h) present the detection result of rice leaf roller, grub, wireworm, aphids, blister beetle, Miridae, Unaspis yanonensis, and Cicadellidae, respectively. The pest-YOLO model can detect the eight-class pests correctly.



(a) Rice leaf roller (b) Grub (c) Wireworm (d) Aphids

(e) Blister beetle (f) Miridae (g) Unaspis yanonensis (h) Cicadellidae

Fig. 11 Detection results of the pest-YOLO model

Table 3 Comparison of Parameters and GFLOPs

Model	Parameters	GFLOPs	mAP50 (all classes)
YOLOv5n	1,769,989	4.2	0.927
YOLOv5n-MobileNetV3	1,171,493 (**-33.8%**)	1.9 (**-54.8%**)	0.895 (**-3.2%**)
pest-YOLO	1,185,972 (**-33%**)	2.0 (**-52.5%**)	0.903 (**-2.4%**)

Although the mAP50 is a crucial metric for a model, the metrics of Parameters and GFLOPs are usually evaluated as to whether a model can be deployed on the terminals with limited computation power. The graphic memory of a terminal should be larger than the value of the parameters of a model, ensuring the model can run on the terminal. Theoretically, the value of

GFLOPs should be as small as possible to reduce computations and boost the inference speed. Table 3 compares the parameters and GFLOPs metrics of three models. The YOLOv5n-MobileNetV3 and pest-YOLO models show a significant decrease in parameters and GFLOPs compared to the YOLOv5n model. However, the mAP50 of these two models only slightly decreased. The pest-YOLO model, in particular, shows a decrease of approximately 33% and 52.5% in parameters and GFLOPs, respectively, while the mAP50 only slightly decreased by 2.4%. After integrating CBAM, the mAP50 of the pest-YOLO model increased by 0.8% compared to the YOLOv5n-MobileNetV3 model, with a slight increase in parameters and GFLOPs.

The validation dataset images evaluated the three models' detection speeds, respectively. Latency assesses model detection speed, which contains the time spent in pre-processing, inference, and Non-Maximum Suppression (NMS) processes. The smaller the latency, the faster the detection speed. FPS is the reciprocal of latency and represents the number of images detected per second. Table 4 presents the mean detection speed per image for the three models. Both pest-YOLO and YOLOv5n-MobileNetV3 models achieved the highest FPS of 370, while the YOLOv5n model reached an FPS of 333. The FPS of the pest-YOLO model increased by approximately 11.1% compared to the YOLOv5n model. Due to integrating CBAMs, the parameters and GFLOPs of the pest-YOLO model slightly increased. However, the model achieved the same FPS as the YOLOv5n-MobileNetV3 model.

Table 4 Comparison of detection speed metrics

Model	Pre-process (ms)	Inference (ms)	NMS (ms)	Latency (ms)	FPS
YOLOv5n	0.1	2.0	0.9	3.0	333
YOLOv5n-MobileNetV3	0.1	1.8	0.8	2.7	370
pest-YOLO	0.1	2.0	0.6	2.7	370 **(+11.1%)**

Table 5 compares the performance of three lightweight backbones. Compared to the GhostNet and LCNet backbones, the pest-YOLO model with MobileNetV3 backbone demonstrated the highest mAP50 of 90.3%. Concurrently, the pest-YOLO model exhibited the lowest number of parameters and GFLOPs with the fastest FPS of 370. Therefore, the proposed model, pest-YOLO, achieved a good balance between speed and precision by comparing the effectiveness and speed of these models.

Table 5 Performance comparison of three lightweight backbones

Model	Precision	Recall	mAP50	mAP50-95	Parameters	GFLOPs	FPS
YOLOv5n-GhostNet	0.853	0.838	0.891	0.474	1,362,697	3.2	222
YOLOv5n-LCNet	0.869	0.841	0.894	0.496	1,188,349	2.8	250
pest-YOLO	0.864	0.874	0.903	0.478	1,185,972	2.7	370

4.4. Discussion

Firstly, this paper utilized the default hyperparameters to train the pest-YOLO model. The default values were optimized for YOLOv5 training from scratch on the Common Objects in Context (COCO) dataset. Hyperparameter evolution is a method of hyperparameter optimization using a genetic algorithm (GA) for optimization. However, evolution is generally expensive and time-consuming, as the base scenario is trained hundreds of times, possibly requiring hundreds or thousands of GPU hours. Given that the pest-YOLO model achieved satisfactory accuracy with the default hyperparameters, several hyperparameters were fine-tuned instead of hyperparameter evolution, including the class loss gain, box loss gain, and image mixup augmentation. The experimental results demonstrated that fine-tuning the class loss and box loss gains did not achieve better accuracy. Conversely, using mixup augmentation instead of mosaic augmentation was associated with lower accuracy.

Secondly, the pest-YOLO model can be extended to detect 102 pests in multiple crops. A pest detection model for detecting 102 pests is available at https://github.com/bobo504/pest-yolo. The model achieved a mAP50 of 55.9% in all 102 pests. The lack of sufficient training images for certain pests has resulted in a corresponding reduction in accuracy. Furthermore, to achieve greater accuracy, it is recommended that a limited number of pest species be utilized to train the model, thereby facilitating the practical application of the model. Thirdly, the current real-time detection model is still in the laboratory stage and will be deployed on embedded devices for field detection in future research. Furthermore, the objective is to develop a pest detection system with practical applications for agricultural pest control. The system combines detection terminals, VPN, and agricultural IoT.

5. Conclusion

This paper proposes a lightweight pest detection architecture that utilizes VPN and agricultural IoT technology to cover large-scale crop areas. The proposed pest-YOLO detector is customized for pest detection, which combines the advantages of YOLOv5n and MobileNetV3, uses the MobileNetV3-light backbone to lightweight the YOLOv5n backbone, and effectively integrates five CBAMs to compensate for the reduction in detection accuracy. Evaluation results demonstrated the pest-YOLO model achieved a good balance of precision and speed. Since the proposed model possesses fewer parameters, a faster speed, and high precision, the model is suitable for deploying on terminals in agricultural IoT to conduct pest detection tasks.

Funding

This research was funded by the Science and Technology Bureau of Leshan, Leshan, China (grant number 22NZD016).

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] J. L. Mu, B. Ma, Y. F. Wang, Z. Ren, S. X. Liu, and J. X. Wang, “Review of Crop Disease and Pest Detection Algorithms Based on Deep Learning,” Transactions of the Chinese Society for Agricultural Machinery, vol. 54, no. s2, pp. 301-313, 2023. (In Chinese)

[2] S. Dong, J. Du, L. Jiao, F. Wang, K. Liu, Y. Teng, et al., “Automatic Crop Pest Detection Oriented Multiscale Feature Fusion Approach,” Insects, vol. 13, no. 6, article no. 554, 2022.

[3] T. Domingues, T. Brandão, and J. C. Ferreira, “Machine Learning for Detection and Prediction of Crop Diseases and Pests: A Comprehensive Survey,” Agriculture, vol. 12, no. 9, article no. 1350, 2022.

[4] P. Kartikeyan and G. Shrivastava, “Review on Emerging Trends in Detection of Plant Diseases Using Image Processing With Machine Learning,” International Journal of Computer Application, vol. 174, no. 11, pp. 39-48, 2021.

[5] C. T. Wang, W. J. Liang, Q. W. Guo, H. Zhong, Y. Gan, and D. Q. Xiao, “Review on Computer-Vision-Based Detection of Agricultural Pests,” Journal of Chinese Agricultural Mechanization, vol. 44, no. 7, pp. 207-213, 2023. (In Chinese)

[6] C. Li, T. Zhen, and Z. Li, “Image Classification of Pests With Residual Neural Network Based on Transfer Learning,” Applied Sciences, vol. 12, no. 9, article no. 4356, 2022.

[7] G. Kang, L. Hou, Z. Zhao, and B. Lang, “Research on the Application of Convolutional Neural Network Based on YOLO Algorithm in Pest Small Target Detection,” 3rd Asia-Pacific Conference on Communications Technology and Computer Science, pp. 131-135, 2023.

[8] W. J. Liang, J. Cao, C. L. Sun, H. X. Cao, and W. Y. Zhang, “Research and Development of Crop Diseases Intelligent Recognition System Based on Deep Learning,” Journal of Chinese Agricultural Mechanization, vol. 44, no. 9, pp. 169-175, 2023. (In Chinese)

[9] P. Ju, Y. Song, Y. J. Zhang, Y. F. Xu, and H. Shao, “Research on Agricultural Crop Diseases and Pests Classification Based on Masked Autoencoding,” Electronic Science and Technology, vol. 37, no. 10, pp. 23-29, 2024. (In Chinese)

[10] F. F. Yang, W. C. Xu, S. D. Chen, and Y. B. Lan, “Rice Disease Image Classification Based on Fusion of Focal Loss and Typical Convolutional Neural Network Structure,” Jiangsu Agricultural Sciences, vol. 51, no. 14, pp. 198-204, 2023. (In Chinese)

[11] X. J. Luo and P. H. Hu, “Research and Implementation of Farm Insect Detection Algorithm Based on Deep Learning,” Journal of East China University of Science and Technology, vol. 50, no. 05, pp. 732-739, 2024. (In Chinese)

[12] J. Wang and C. Xu, "Research on Detection of Navel Orange Pests Based on YOLOv5s," Industrial Control Computer, vol. 36, no. 07, pp. 105-106+109, 2023. (In Chinese)

[13] Y. M. Liu, K. Hu, J. W. Nie, and T. Q. Xie, "Rice Disease and Pest Identification Based on MSDB-ResNet," Journal of South China Agricultural University, vol. 44, no. 6, pp. 978-985, 2023. (In Chinese)

[14] Y. He, D. H. Chen, and L. Peng, "Research on Object Detection Algorithm of Economic Forestry Pests Based on Improved YOLOv5," Journal of Chinese Agricultural Mechanization, vol. 43, no. 04, pp. 106-115, 2022. (In Chinese)

[15] S. Aladhadh, S. Habib, M. Islam, M. Aloraini, M. Aladhadh, and H. S. Al-Rawashdeh, "An Efficient Pest Detection Framework With a Medium-Scale Benchmark to Increase the Agricultural Productivity," Sensors, vol. 22, no. 24, article no. 9749, 2022.

[16] M. Assiri, E. S. A. Elhameed, A. Kumar, and C. Singla, "Automated Insect Detection and Classification Using Pelican Optimization Algorithm With Deep Learning on Internet of Enabled Agricultural Sector," SN Computer Science, vol. 5, article no. 576, 2024.

[17] A. Hussain and P. B. Srikaanth, "Leveraging Deep Learning and Farmland Fertility Algorithm for Automated Rice Pest Detection and Classification Model," KSII Transactions on Internet & Information Systems, vol. 18, no. 4, pp. 959-979, 2024.

[18] A. Tzounis, N. Katsoulas, T. Bartzanas, and C. Kittas, "Internet of Things in Agriculture, Recent Advances and Future Challenges," Biosystems Engineering, vol. 164, pp. 31-48, 2017.

[19] X. Wu, C. Zhan, Y. K. Lai, M. M. Cheng, and J. Yang, "IP102: A Large-Scale Benchmark Dataset for Insect Pest Recognition," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8787-8796, 2019.

[20] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," Proceedings of the European Conference on Computer Vision, pp. 3-19, 2018.