scientific reports

Check for updates

OPEN

A lightweight rice pest detection algorithm based on improved YOLOv8

Yong Zheng[1,2], Weiheng Zheng[1,2]✉ & Xia Du[1]

Timely and accurate detection of rice pests is highly important for pest control, as well as for improving rice yield and quality. However, owing to the high interclass similarity, significant intraclass age differences, and complex backgrounds among different pests, accurately and rapidly identifying a variety of rice pests via deep neural network models poses a significant challenge. To address this issue, this paper presents a fast and accurate method for rice pest detection and identification named Rice-YOLO (You Only Look Once). This model is based on YOLOv8-N and incorporates an efficient detection head designed for the complex characteristics of pests. Additionally, deep supervision layers were introduced into the network, along with the incorporation and improvement of the dynamic upsampling module. The experimental data included the large-scale pest public dataset IP102 and the sixteen-class rice pest dataset R2000. The experimental results demonstrated that Rice-YOLO outperformed previous object detection algorithms, with 78.1% mAP@0.5, 62.9% mAP@0.5:0.95, and 74.3% F1 scores.

Keywords Rice pest detection, YOLOv8, Object detection, Deep learning, Computer vision

With the continuous increase in the global population, the demand for food is also constantly rising. Rice is a staple food for more than half of the world's population[1]. It plays a crucial role in agricultural production, and its high yield characteristics can effectively alleviate the problem of food shortages. Rice cultivation not only provides a livelihood for farmers but also drives the development of the agricultural economy. However, rice is susceptible to various pests during its growth process. These pests can cause damage to rice leaves, reduce the moisture content, and lead to bacterial infections, all of which severely affect the yield and quality of rice[2,3]. According to the International Rice Research Institute, pests cause an average annual loss of 37% in rice production for farmers. Therefore, the detection and prevention of rice pests are urgently needed.

Traditional pest detection methods rely primarily on manual identification, which places certain demands on the judgement abilities of personnel and identification equipment[4]. Additionally, manual identification not only requires significant human labour but also carries the risk of erroneous judgement by staff; this can affect the selection of pesticides and potentially lead to environmental pollution.

With the development of computer technology, traditional image processing techniques have been widely used to identify specific pests[5]. Manually designed image features alleviate the labour and time issues associated with pest detection and improve the efficiency of pest identification. However, traditional image processing techniques are often suitable only for the detection of specific pests in simple scenarios, making it difficult for these methods to be widely applied in various complex environments.

With the development of deep learning, convolutional neural networks, which can automatically extract image features, have gained widespread attention. In recent years, many classic deep learning architectures, such as AlexNet[6,7], VGGNet[8,9], Inception[10,11], ResNet[12,13], and DenseNet[14,15], have been applied to agricultural pest detection.

Owing to the small interclass differences, complex backgrounds, and significant variations in the appearance of the same type of pest at different developmental stages, designing an efficient algorithm for rice pest detection has become a major challenge.

Ali et al.[16] proposed a lightweight pest detection method that is based on Faster R-CNN and uses MobileNet as the backbone network, which improves the model's ability to identify pests. Wang et al.[17] proposed an agricultural pest detection algorithm based on an improved Faster R-CNN. They combined an improved feature pyramid network with the backbone network, used the bilinear interpolation method of the ROI align

[1]Xiamen University of Technology, Fujian 361024, China. [2]Hunan Provincial Key Laboratory of Remote Sensing Monitoring for Eco-environment in Dongting Lake Area, Changsha 410004, Hunan, China. ✉email: zhengweiheng@xmut.edu.cn

algorithm instead of the rounding quantization in the ROI pooling algorithm for calculations, and introduced a convolutional block attention module. The improved model has better feature extraction capabilities and significantly improved detection accuracy. Li et al.[18] proposed a greenhouse tiny pest detection method based on Faster R-CNN. By employing a transfer learning strategy, they enhanced the robustness of pest detection under greenhouse conditions, enabling rapid collection of pest count information. The aforementioned algorithms are two-stage detection algorithms. They first search for regions in the image that may contain objects and then perform object localization and classification on these candidate regions. Compared with one-stage algorithms, two-stage detection algorithms have higher accuracy but slower detection speed.

Chu et al.[19] proposed a grain storage pest detection model based on YOLOv5. This model incorporates a bidirectional feature pyramid and a channel attention mechanism, and improves the nonmaximum suppression algorithm using distance intersection over union. The improved model can achieve grain storage pest detection in complex storage environments. Xiang et al.[20] proposed an improved pest recognition model based on YOLOv5. This model incorporates the squeeze-and-excitation attention mechanism and ConvNeXt module, and constructs a feature extraction module for small sample learning based on the stacked residual structure of the standard bottleneck module. The improved model offers higher detection accuracy and robustness. Yang et al.[21] proposed a tea plantation pest detection model based on YOLOv7. This model utilizes deformable convolutions and a dynamic attention mechanism, incorporates a new implicit decoupling head, and uses a softened nonmaximum suppression algorithm in place of the original nonmaximum suppression algorithm in model predictions. The improved model has fewer parameters and can achieve fast and accurate tea pest detection. One-stage algorithms do not have a candidate box screening process; instead, they directly regress the position coordinates of the object box and the classification probability of the object; as entirely end-to-end object detection algorithms, this gives them a significant advantage in terms of detection speed.

Researchers have already applied one-stage object detection models to agricultural pest detection, but these models still lack efficiency. Therefore, it is necessary to develop an accurate and efficient algorithm for rice pest detection. The YOLO series of algorithms are mainstream one-stage object detection algorithms. This study uses YOLOv8 for rice pest detection and proposes an improved version, named Rice-YOLO, that achieves precise detection of rice pests. The main contributions of this study are as follows:

(1) Lightweight Detection Head: Given the diverse types of pests and complex back-grounds, we propose a lightweight detection head that not only improves the detection performance of the model but also reduces hardware overhead.
(2) Deep Supervision Technology: We introduce deep supervision technology, integrating multiple detection heads for joint detection; this enhances the model's perception of complex pest-related information without adding an extra inference burden.
(3) Dynamic Upsampling Module: We incorporate a dynamic upsampling module and improve the sampling offset strategy to enhance sampling flexibility and the model's feature upsampling capability.

Materials and methods

Dataset

Owing to the current scarcity of rice pest datasets, the experimental data include both the large public crop pest dataset IP102 and the R2000 rice pest dataset, which are collectively referred to as IPR16 in this study.

IP102

IP102[22] is a large-scale benchmark dataset for pest identification, covering 102 common crop pests with over 75,000 images. The data encompass images of pests throughout their entire lifecycles. This dataset is characterized by significant intraclass variation, small intraclass variation, and complex backgrounds, making the object detection task based on IP102 a substantial challenge. For this study, only the rice-related pest data from IP102 were used, which comprise 14 classes and 1248 images. Importantly, the publicly available IP102 dataset contains duplicate and misclassified images. Therefore, the rice pest data from IP102 were screened and cleaned, resulting in a preprocessed dataset containing 16 categories and 532 images.

R2000

The R2000 dataset originated from our previous work, which was constructed by referencing the rice pest data from IP102, and shares the same data characteristics. The images were sourced from common image search engines, including Google, Flickr, and Bing etc. The use of internet-collected images is a widely adopted method for constructing large public datasets such as IP102, ImageNet[23], and Microsoft COCO[24]. The R2000 rice pest dataset contains 16 categories and 2024 images.

IPR16

The limited amount of data restricts the model's ability to acquire comprehensive information, thereby reducing the scientific value and reliability of the experiments. Therefore, in this study, the above two datasets with similar characteristics were merged, to form IPR16, which includes 16 categories and 2556 images. The details regarding the categories and the number of samples are shown in Table 1.

The number of instances for each category is shown in Fig. 1. As shown in the figure, there is a significant variation in the number of instances across different categories. This variation indicates that the dataset exhibits a characteristic similar to a long-tailed distribution. This more realistic and challenging dataset imposes greater demands on the learning capabilities of the model.

The bounding boxes for the instances in each category are shown in Fig. 2. The horizontal axis represents the number of instances corresponding to different bounding box widths, and the vertical axis represents the

Class ID	Pest species	IP102 (Rice Pest)		R2000		IPR16	
		Images	Instances	Images	Instances	Images	Instances
1	Rice leaf roller	90	94	74	77	164	171
2	Pink rice borer	15	15	147	149	162	164
3	Asiatic rice borer	55	57	88	93	143	150
4	Yellow rice borer	37	37	103	108	140	145
5	Rice leaf caterpillar	8	9	113	119	121	128
6	Small brown plant hopper	26	26	98	98	124	124
7	White backed plant hopper	30	31	69	70	99	101
8	Brown plant hopper	39	42	52	65	91	107
9	Rice leaf hopper	45	47	160	182	205	229
10	Rice gall midge	36	38	131	150	167	188
11	Paddy stem maggot	21	21	109	113	130	134
12	Rice stem fly	8	8	362	409	370	417
13	Grain spreader thrips	10	10	137	225	147	235
14	Rice weevil	92	92	116	143	208	235
15	Rice shell pest	15	15	146	160	161	175
16	Rice horned caterpillar	5	5	119	120	124	125
Total		532	547	2024	2281	2556	2828

Table 1. Categories and sample counts of rice-related pests in IPR16.

number of instances corresponding to different bounding box heights. The bars on the axes reflect the quantity of bounding boxes present in the dataset. Different colours of the bounding boxes indicate different categories. There is significant variation in the bounding boxes within the same category. For example, the red bounding boxes in the figure have a range of different sizes. However, there are similarities in the bounding boxes across different categories, as observed with the bounding boxes of varying colours that are similar in size; this highlights the challenges of complex bounding box regression in detecting rice pests.

Figure 3 shows the distribution of the centre positions of all the instance bounding boxes in the images. The horizontal and vertical axes represent the percentages of instance bounding box centre positions corresponding to the image dimensions. Each instance bounding box centre is depicted as a pixel point in the figure, with darker pixel colours indicating a higher concentration of centre points within that area. Most instances are located near the centre of the images, but some instances are positioned farther from the centre. From the relationship between the positions and the number of instances, it can be observed that IPR16 also exhibits a long-tailed distribution of instance object locations. This characteristic imposes greater demands on the model's object localization capabilities.

Figure 4 shows the size distribution of the bounding boxes for all the instances. The horizontal and vertical axes represent the width and height, respectively, of the bounding box. Each instance bounding box is represented by a pixel point in the figure, where darker pixel colours indicate more bounding boxes within that pixel area. The figure, clearly shows that the pixels are widely scattered across the two-dimensional coordinate space. The relationships between the widths and heights of the bounding boxes and their quantities indicate that IPR16 features a variety of aspect ratios.

Most of the bounding boxes have relatively similar aspect ratios, whereas a few have significantly different aspect ratios. This variation is due to interclass similarities and considerable differences in appearance and duration across different lifecycle stages.

During the bounding box regression process, the shape and scale factors of the regression samples significantly affect the regression results. Therefore, the diverse bounding boxes in IPR16 present a substantial challenge for the model's bounding box regression capabilities.

Rice-YOLO

Fast detection is a key feature of the YOLO series, making these models widely applicable across various domains. Over time, through the efforts of different research teams, the YOLO family has undergone multiple iterations, such as YOLOv3[25], YOLOv4[26], YOLOv5[27], YOLOv6[28], YOLOv7[29], YOLOv8[30], YOLOv9[31] and YOLOv10[32].

YOLOv8, which was developed by Ultralytics (the creators of YOLOv5), has not only excellent inference speed but also outstanding object localization and identification capabilities. The architecture and size are illustrated in Fig. 5, which was created by GitHub user RangeKing[33]. Considering the requirements for lightweight and real-time pest detection, this study selects the smallest model, YOLOv8-N, as the baseline model.

Owing to the high similarity between different pest categories, significant appearance differences in appearance at various life stages, and high similarity between pests and their backgrounds, the YOLOv8-N model performs inadequately for this detection task. To address these issues, this paper presents an improved algorithm, Rice-YOLO, whose structure is illustrated in Fig. 6.

Compared with YOLOv8, Rice-YOLO integrates a newly proposed detection head, DBG, which enhances the model's ability to perceive complex pest-related information. This enhancement leads to improved detection

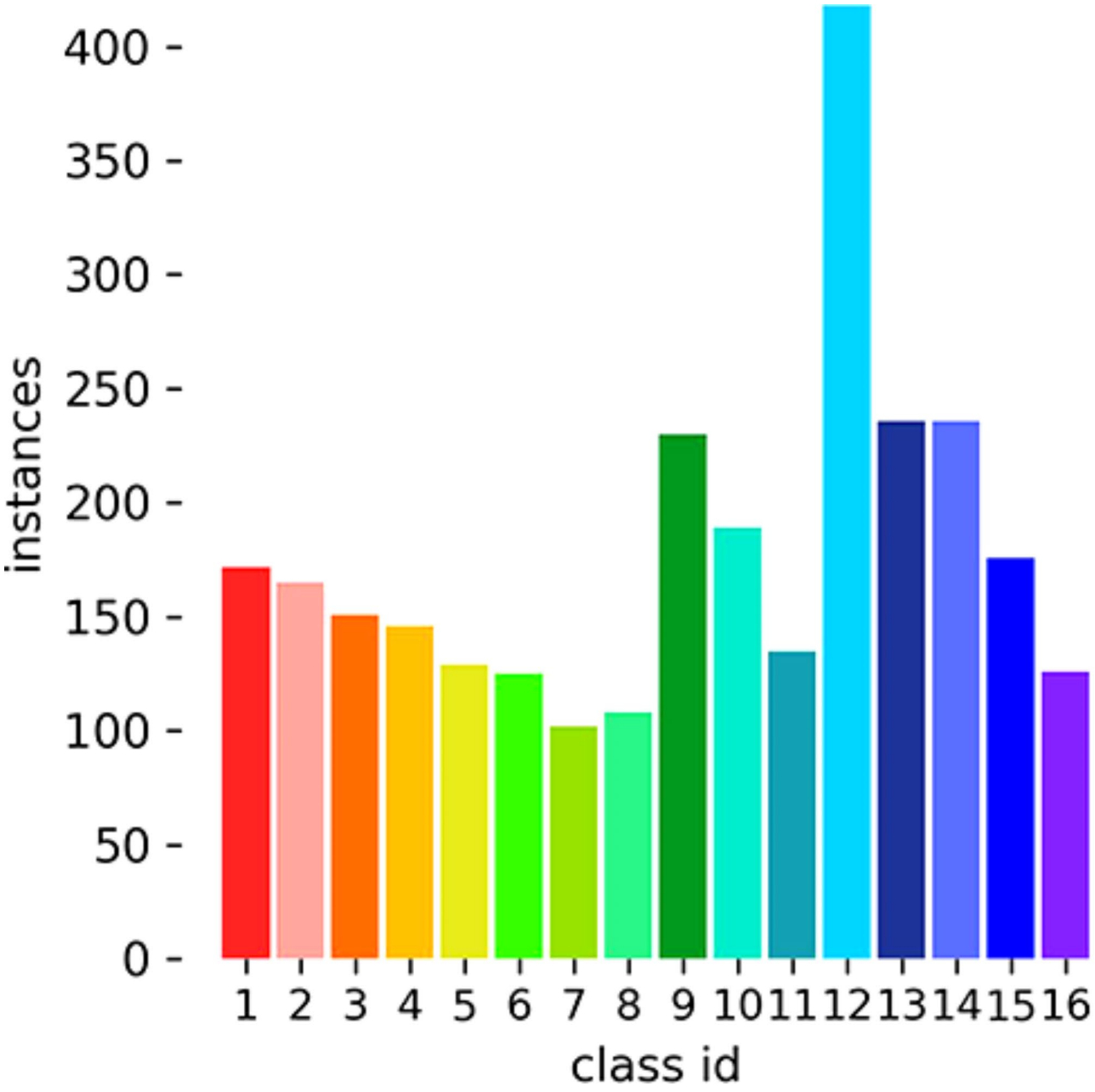

Fig. 1. Number of instances in each category.

performance while simultaneously reducing the number of parameters and computational load. Additionally, Rice-YOLO implements intermediate layer supervision with Aux-DBG, which incurs no additional hardware overhead during model inference; furthermore, it incorporates an improved point-based sampling method to replace the original upsampling method, thereby enhancing the model's upsampling capabilities. These enhancements enable Rice-YOLO to outperform YOLOv8 in terms of detection performance.

Diverse branch grouped Head

As shown in Fig. 7, based on the YOLOv8 detection head, we introduce the diverse branch block and group convolution to propose an efficient detection head, which is referred to as the diverse branch grouped head (DBG) in this paper. In the figure, k, s and g denote the kernel size, stride and number of groups, respectively.

During model training, the diverse branch block, with its branches of varying scales and complexities, enriches the diversity of the feature space. During model inference, the complex structure of the diverse branch block can be equivalently transformed into a single convolution, ensuring that the model incurs no additional hardware overhead while delivering better performance.

Rich feature maps can enhance network performance; however, an excess of feature information may also increase redundancy, potentially leading to model overfitting. To mitigate the negative effect of redundant

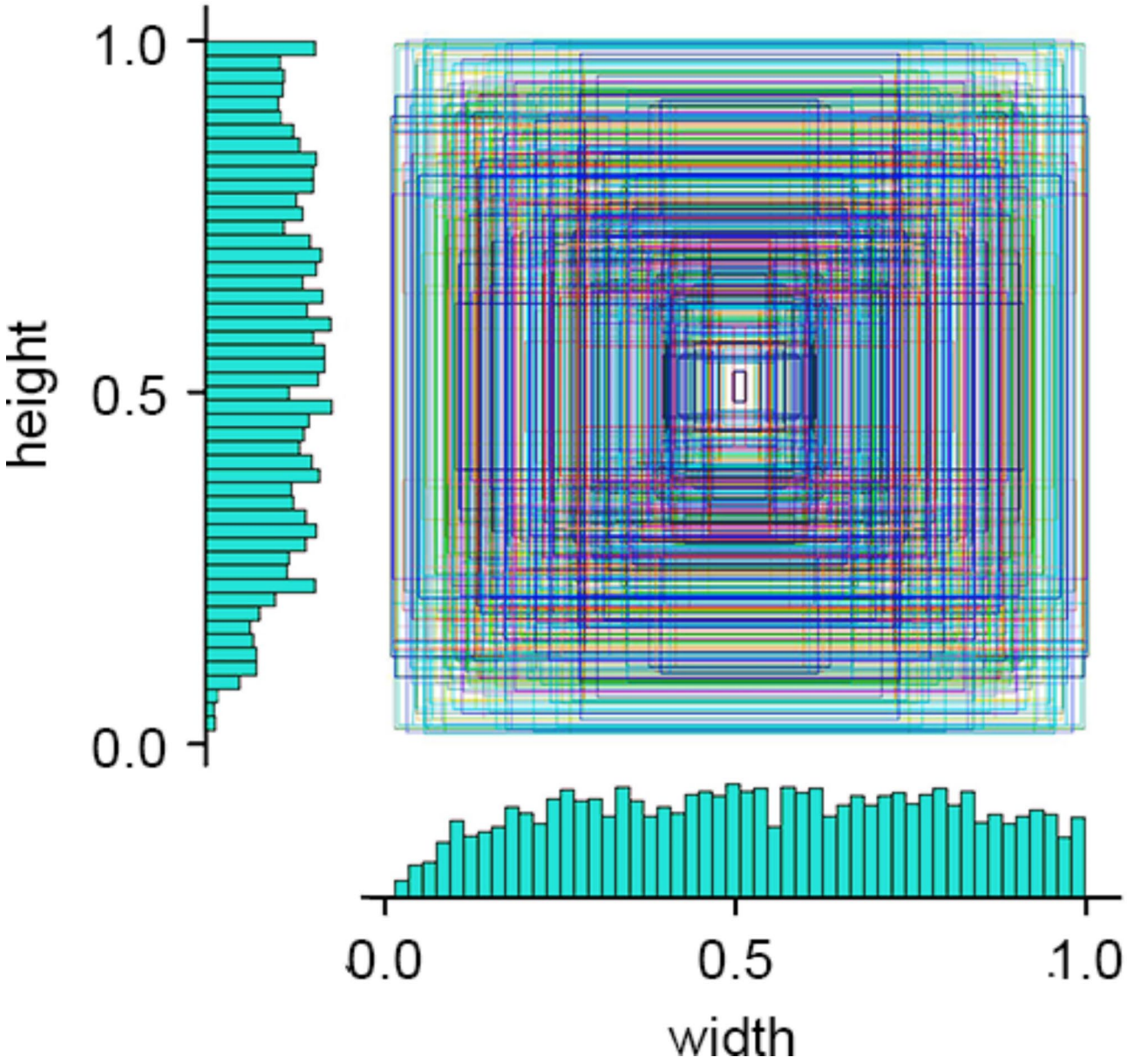

Fig. 2. Bounding boxes for instances in each category.

convolutional kernels, group convolution is employed to replace standard convolution. Each group in group convolution has different inputs, resulting in lower overlap among the features extracted by each group. This effectively reduces redundant information and improves model performance.

In Rice-YOLO, the DBG detection head is used to replace the original YOLOv8 detection head. This enhancement improves the model's ability to perceive complex information related to pests, thereby improving its detection performance while reducing the number of parameters and computational load.

Intermediate layer supervision

To achieve better training results, we add intermediate-layer supervision to the model. This supervision enables gradients to flow more effectively through the network during backpropagation, which helps prevent gradient vanishing and improves convergence.

Therefore, we further introduce the concept of the auxiliary head [29] and apply the DBG head as an intermediate supervisory layer, which we call the auxiliary DBG head (Aux-DBG). As shown in Fig. 8, the auxiliary DBG head is added during the training phase as an intermediate supervisory layer and is removed during the prediction phase, ensuring that there is no additional hardware overhead for model inference.

During the training phase, the intermediate supervisory detection head can reflect the learning performance of the hidden layers. Providing additional gradients helps the network learn better representations.

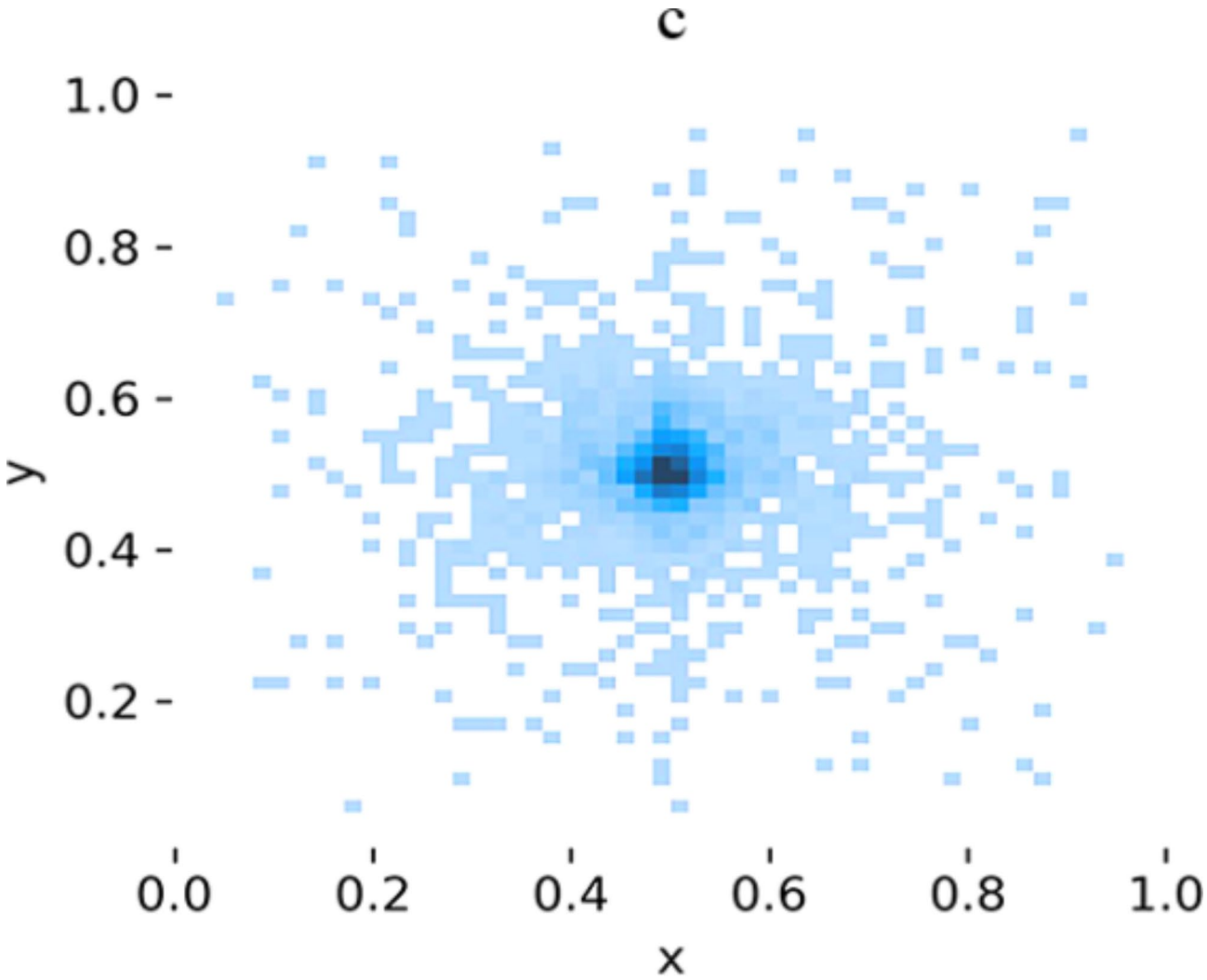

Fig. 3. Positions of bounding boxes for instances in each category.

In Rice-YOLO, the auxiliary DBG head is added as a supervisory layer to the intermediate layers of the network; thisenables the intermediate layers to receive more comprehensive training, effectively mitigating issues such as gradient vanishing and slow convergence that often occur during the training of deep neural networks.

DySample–hardtanh

DySample + is an efficient upsampler constructed from a point sampling perspective, which optimizes the scope constraint of the offset range based on DySample (DS). We base our method on DySample+ (DS+) and introduce the hardtanh function to replace the original dynamic range factor distribution function, namely, the sigmoid function, thereby improving the offset range suppression strategy. We refer to this improved method as DySample–hardtanh (DSHT). Compared with the sigmoid function, hardtanh further increases the flexibility of the offsets and is easier to compute. The structure of DySample–hardtanh is shown in Fig. 9.

Given an upsampling scale factor of $scale$ and a feature map X of size $H \times W \times C$, where H, W and C denote the height, width and number of channels, respectively, DySample–hardtanh generates sampling offsets based on X. These are combined with the initial sampling positions to create the sampling set, and the final upsampling result is obtained through bilinear interpolation. During the creation of the sampling set, DySample–hardtanh generates sampling offsets and scaling factors via different linear mappings of the input features. These are then combined with the initial sampling grid after pixel shuffling to produce the final offset.

Importantly, the initial offset generation positions are uniformly distributed. The offset amounts must be constrained by scaling factors to reduce overlap in the offset regions.

In Rice-YOLO, we employ the efficient upsampling module DySample–hardtanh to enhance the model's upsampling capability; this enables more valuable rice pest feature information to be output through feature fusion.

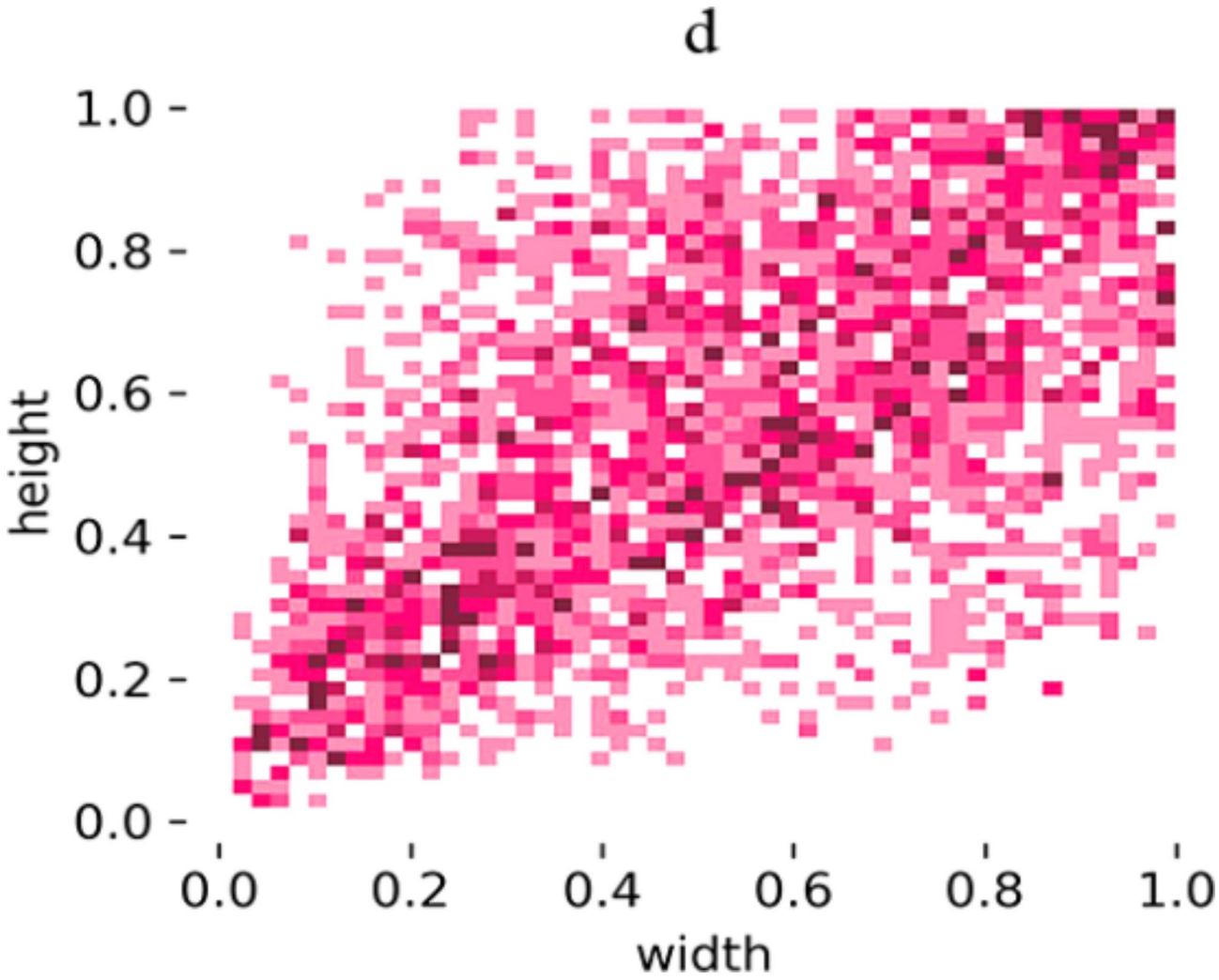

Fig. 4. Widths and heights of the bounding boxes for instances in each category.

Performance evaluation method

In this study, we selected the mean average precision (mAP), F1 score, detection speed, and floating-point operations per second (FLOPs) as evaluation metrics for model performance assessment.

The number of FLOPs represents the computational complexity and is used to measure the model complexity. Precision represents the proportion of objects predicted by the model, where TP denotes the correct positive prediction rate and FP denotes the rate at which the model incorrectly identifies negative instances as positive.

$$\text{Presicion (P)} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

Recall represents the proportion of instances correctly detected by the model. FN denotes the rate at which the model incorrectly predicts positive instances as negatives.

$$\text{Recall (R)} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

The mean average precision (mAP) represents the average precision across all classes and is used to evaluate the overall detection accuracy of the model. Here, C represents the number of classes of pests, and i denotes the ith class of pests.

$$\text{AP}_\text{i} = \int_0^1 \text{P}_\text{i}(\text{R}_\text{i})\text{dR}_\text{i} \tag{3}$$

$$\text{mAP} = \frac{1}{C}\sum\nolimits_{i=1}^{C}\text{AP}_\text{i} \tag{4}$$

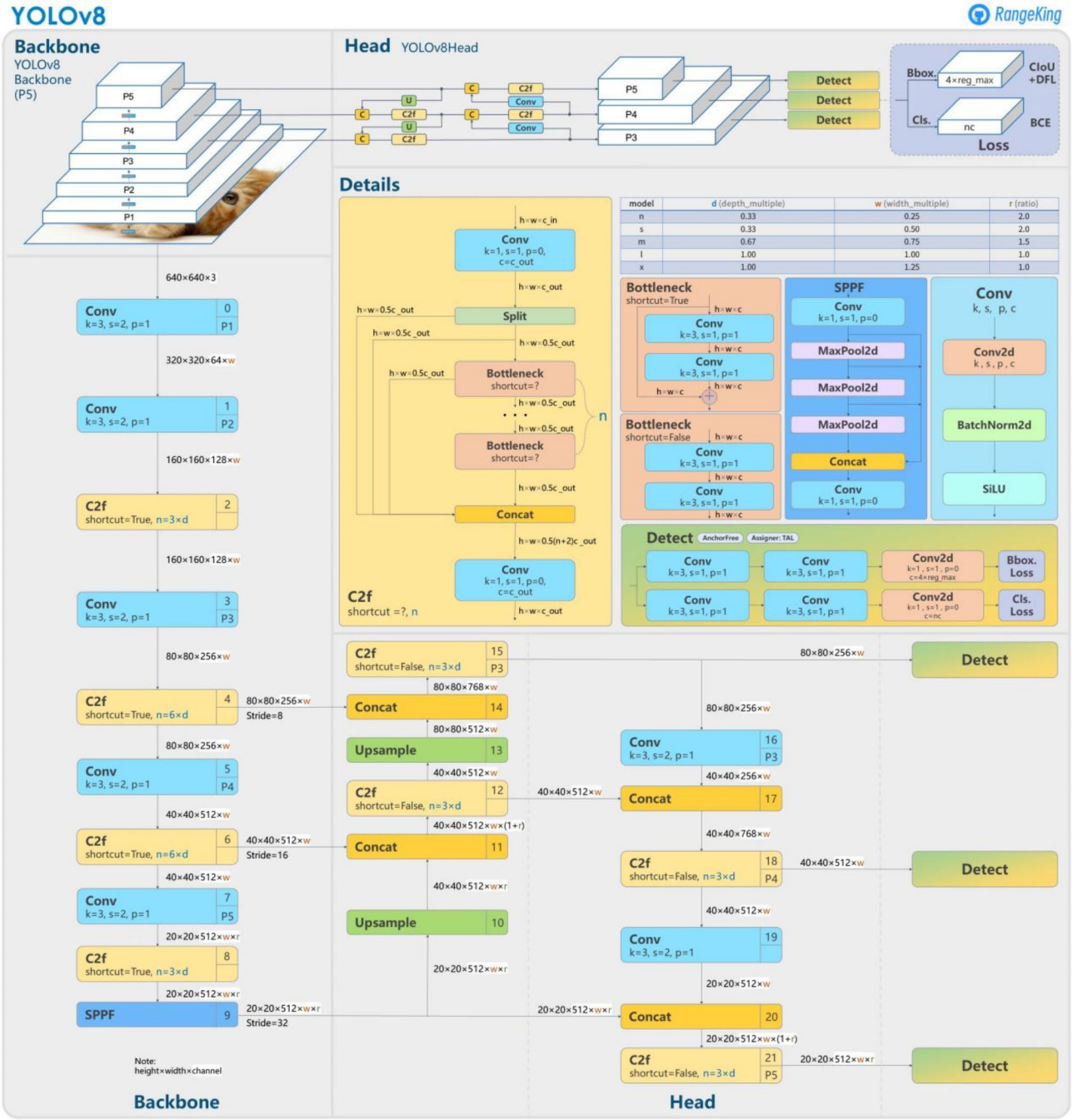

Fig. 5. Structural diagram of YOLOv8.

The F1-score represents a combined evaluation of precision and recall for the model.

$$\text{F1 score} = 2 \times \frac{\text{P} \times \text{R}}{\text{P} + \text{R}} \tag{5}$$

Speed represents the average time taken by the model to detect each image and is used to measure the model's detection speed. Preprocess, inference, and postprocess represent the times taken by the model during the preprocessing, inference, and postprocessing stages, respectively.

$$\text{Speed} = \text{preprocess} + \text{inference} + \text{postprocess} \tag{6}$$

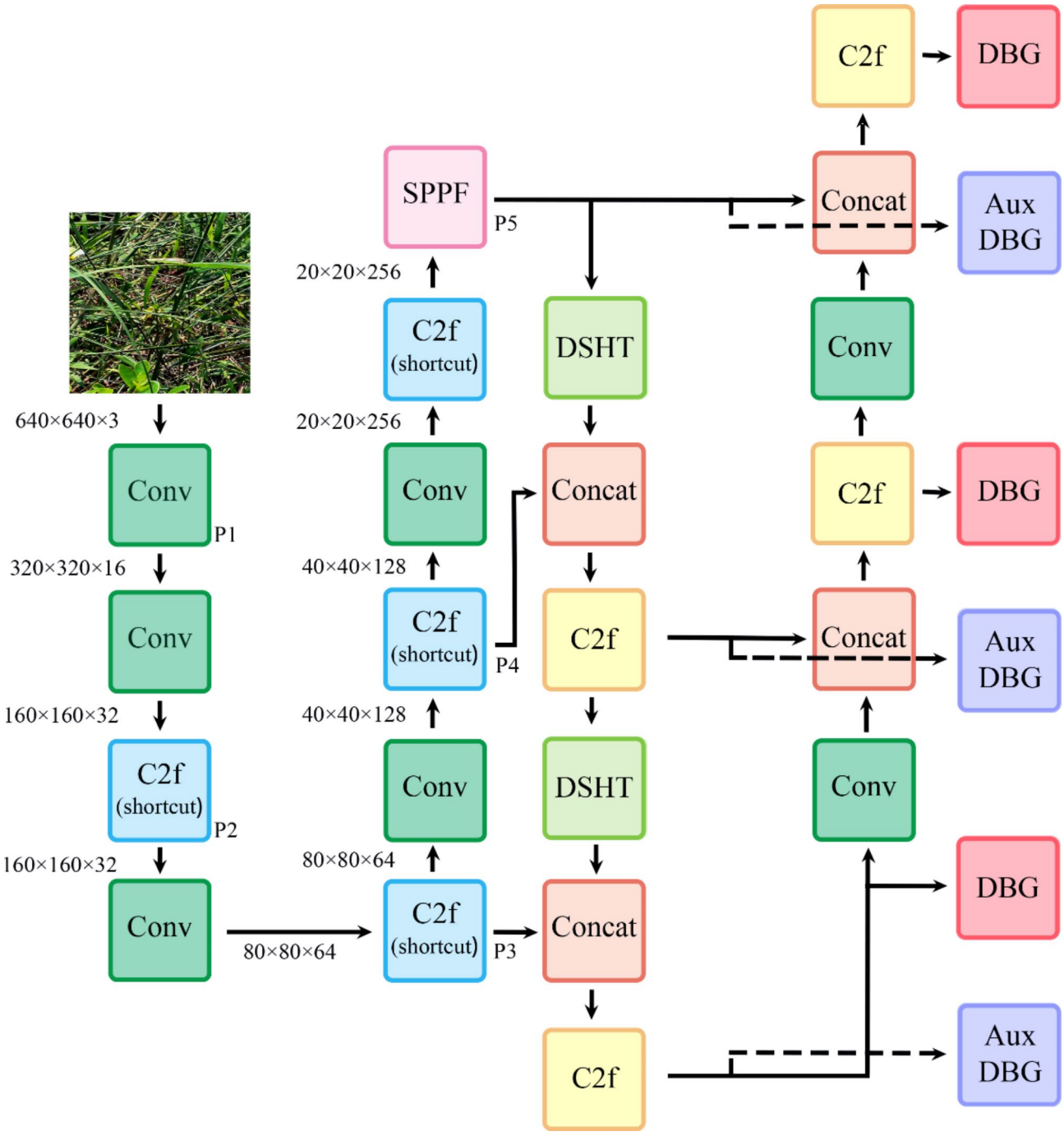

Fig. 6. Structure diagram of Rice-YOLO.

Experiments and discussion

Data processing

In this study, we employed a fivefold cross-validation method, where IPR16 was divided into five subsets, as shown in Table 2.

The dataset was divided into five folds for cross-validation, as illustrated in Fig. 10. During model training, three folds were chosen for training, one fold was chosen for validation, and one fold was chosen for testing. The final experimental results represent the average of the fivefold cross-validation results.

The hardware specifications and environmental configuration for the experiments are shown in Table 3, and the model hyperparameter settings are detailed in Table 4.

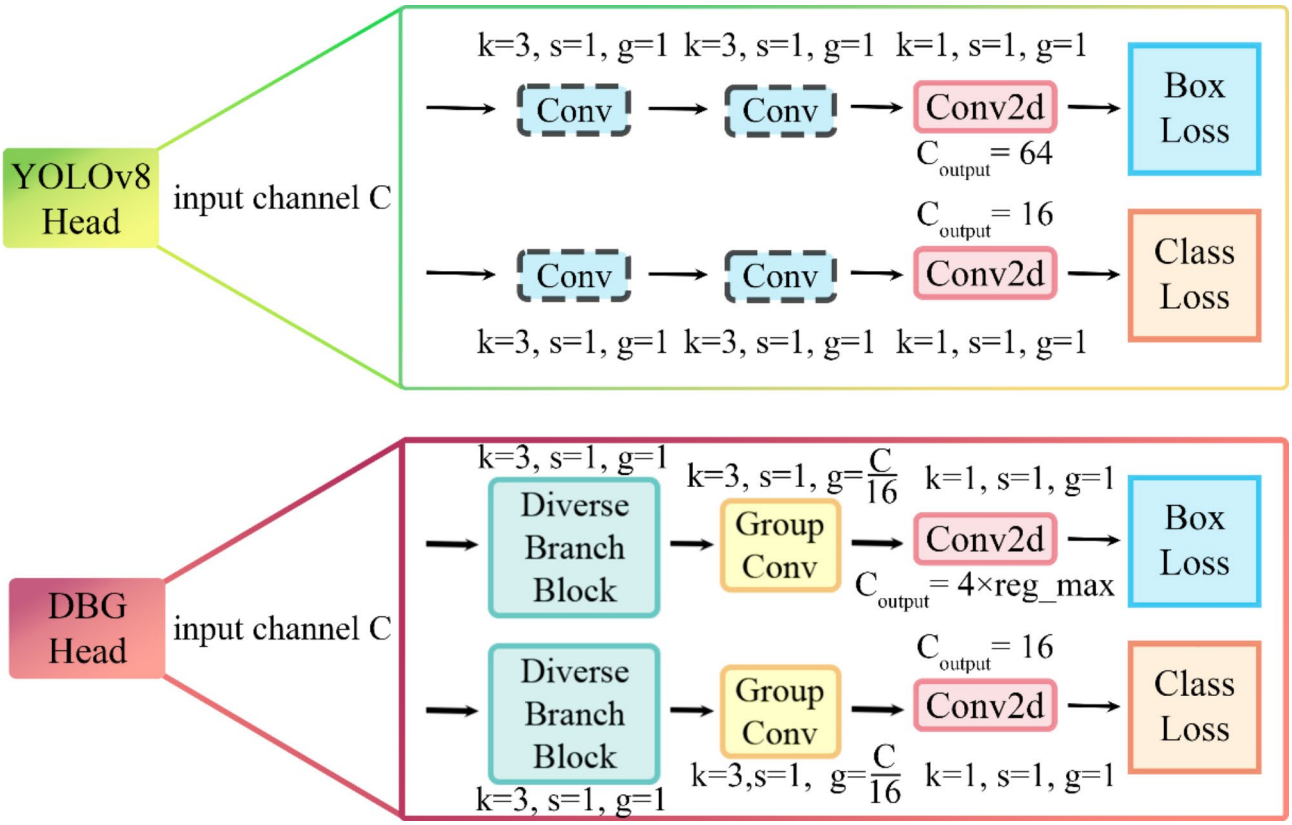

Fig. 7. Structural comparison of the YOLOv8 head and DBG head.

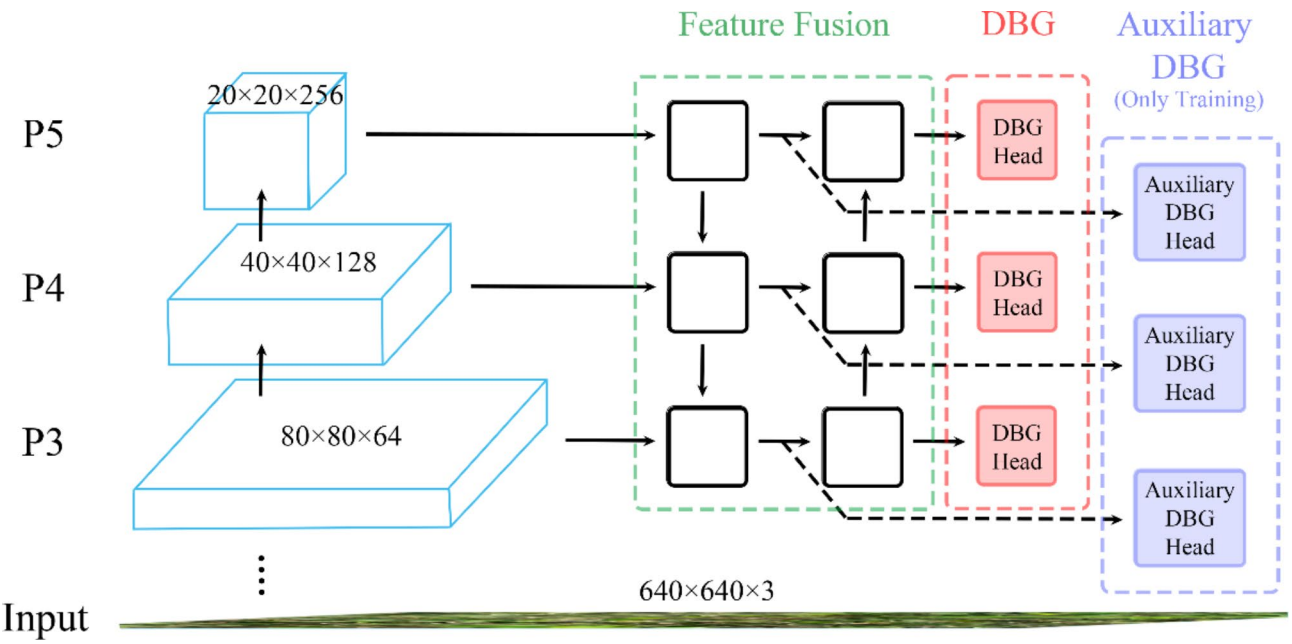

Fig. 8. Detailed information on the auxiliary DBG head in the model.

Rice-YOLO detection results

Based on the IPR16 dataset, we analysed the detection results of Rice-YOLO, as shown in Fig. 11; Table 5. Different colours represent different categories of pests, with the horizontal axis representing the recall and the vertical axis representing the precision. The area under the curve indicates the average precision, meaning

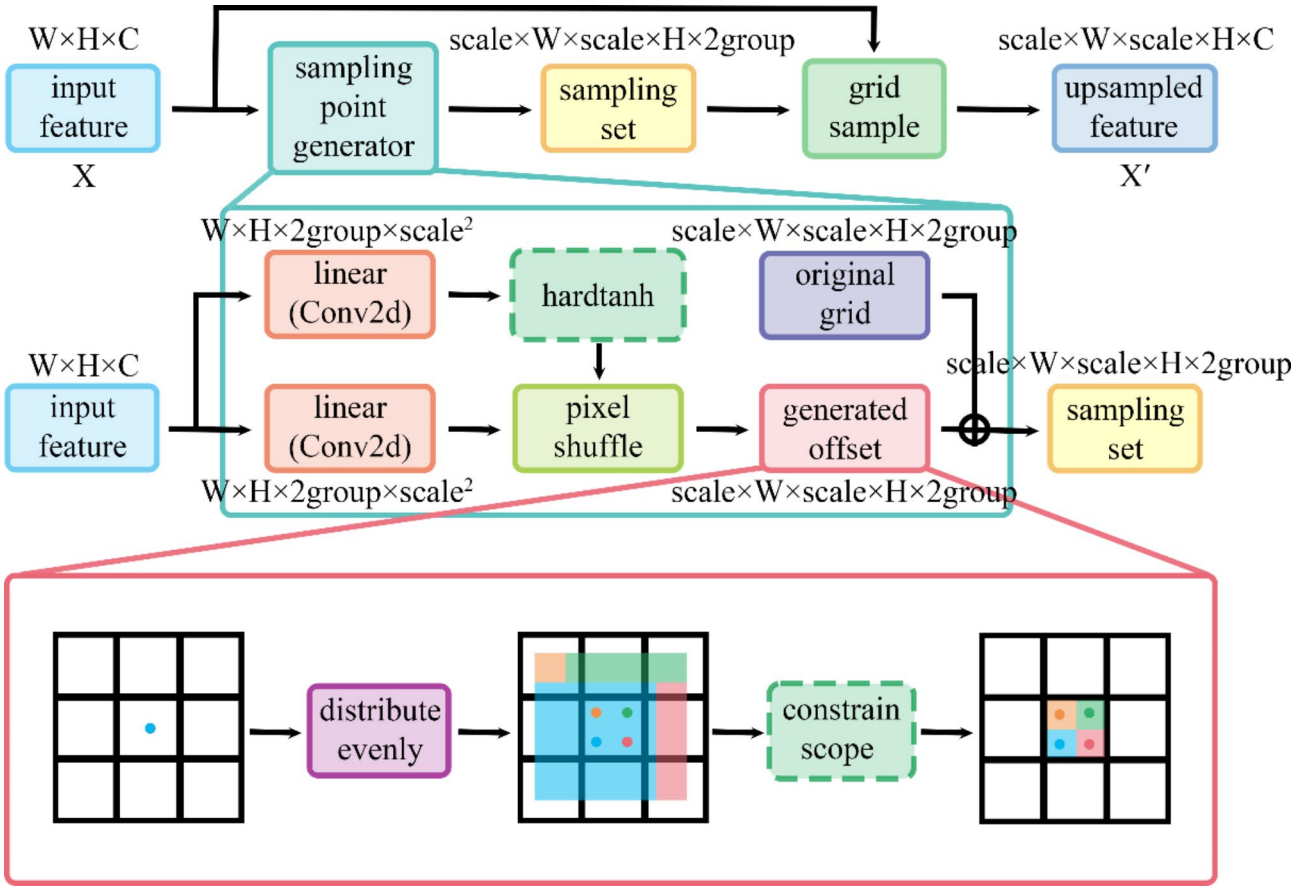

Fig. 9. Architecture of DySample–hardtanh.

Class ID	Pest species	Images					Instances				
		Fold 01	Fold 02	Fold 03	Fold 04	Fold 05	Fold 01	Fold 02	Fold 03	Fold 04	Fold 05
1	rice leaf roller	33	33	33	33	32	34	34	34	35	34
2	pink rice borer	33	32	32	32	33	33	34	32	32	33
3	asiatic rice borer	28	29	30	29	27	29	30	31	32	28
4	yellow rice borer	29	28	27	27	29	30	30	28	27	30
5	rice leaf caterpillar	23	24	24	25	25	24	25	29	25	25
6	small brown plant hopper	26	25	25	24	24	26	25	25	24	24
7	white backed plant hopper	20	20	19	20	20	21	21	19	20	20
8	brown plant hopper	17	18	19	19	18	23	19	21	20	24
9	rice leaf hopper	41	41	41	41	41	47	42	46	43	51
10	rice gall midge	33	33	33	33	35	33	42	43	34	36
11	paddy stem maggot	26	26	27	26	25	27	27	27	28	25
12	rice stem fly	75	74	73	74	74	86	82	77	81	91
13	grain spreader thrips	29	30	30	29	29	48	50	41	46	50
14	rice weevil	42	42	41	42	41	49	48	47	49	42
15	rice shell pest	32	32	32	32	33	33	33	40	34	35
16	rice horned caterpillar	25	25	25	25	24	26	25	25	25	24
Total		512	512	511	511	510	569	567	565	555	572

Table 2. Fivefold division of the IPR16 dataset.

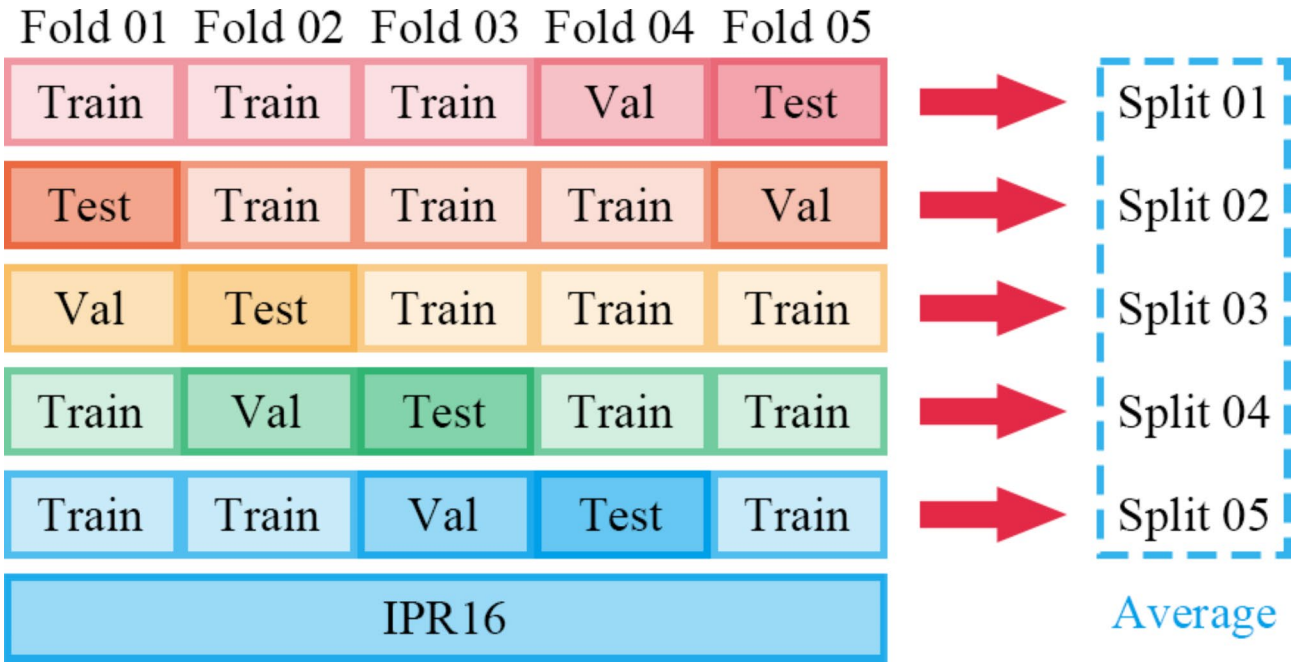

Fig. 10. Fivefold cross-validation of IPR16.

Parameters	Configuration
CPU	i3-10105 F
GPU	NVIDIA GeForce RTX 3070
GPU memory size	8G
Operating systems	Windows 10
Deep learning architecture	Pytorch2.0.1 + Cuda 11.8 + cudnn 8.7

Table 3. Hardware specifications and Environmental Configuration.

Parameters	Setup
Epoch	300
Batch size	32
Image size	640 × 640
Optimizer	AdamW
Initial learning rate	0.01
Final learning rate	0.01
Momentum	0.937
Weight decay	0.0005

Table 4. Training configuration.

that the larger the area is, the better the performance. A comparison of the PR curves reveals that Rice-YOLO achieved excellent detection performance for most types of pests, with an mAP50 of 78.1% and the highest AP of 90.6%. However, owing to the high interclass similarity of pests and the significant appearance changes across different life stages, the model still faces challenges in detecting and identifying certain types of pests. This highlights potential directions for future improvements.

Comparison of different detection models

On the IPR16 dataset, we compared Rice-YOLO with mainstream models in the field of object detection, such as YOLOv3, YOLOv4, YOLOv5, YOLOv6, and YOLOv7. The comparison results are shown in Table 6. Rice-YOLO has 2.8 million parameters and 7.2 billion floating-point operations (FLOPs). On the IPR16 dataset, it achieved an mAP50 of 78.1%, an mAP50:95 of 62.9%, and an F1 score of 74.3%. Detailed comparisons with classic models demonstrate that Rice-YOLO outperformed the vast majority of these traditional detection models.

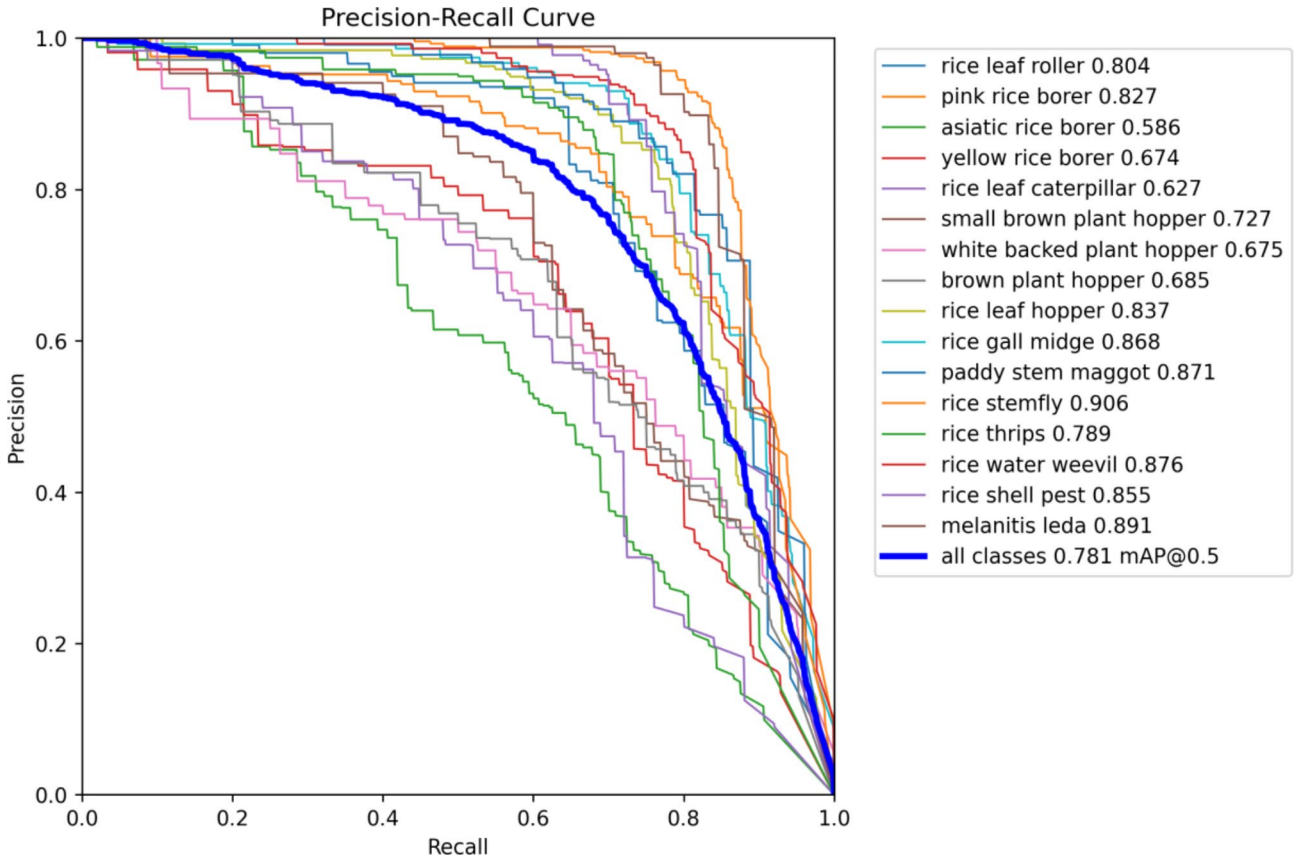

Fig. 11. P–R curves and AP values for various pests.

Class id	*P*	*R*	mAP 50	mAP 50:95	F1
1	77.92%	73.12%	80.44%	64.32%	75.4%
2	72.86%	76.90%	82.72%	63.38%	74.8%
3	63.34%	49.90%	58.64%	47.80%	55.8%
4	70.54%	65.08%	67.44%	55.28%	67.7%
5	73.30%	52.34%	62.66%	45.52%	61.1%
6	66.22%	67.80%	72.72%	64.30%	67.0%
7	60.48%	67.22%	67.52%	57.04%	63.7%
8	64.16%	62.24%	68.50%	58.82%	63.2%
9	87.14%	73.68%	83.68%	64.22%	79.8%
10	86.96%	78.74%	86.78%	64.94%	82.6%
11	80.80%	78.62%	87.10%	**76.04%**	79.7%
12	**93.54%**	**82.28%**	**90.60%**	72.20%	**87.5%**
13	87.46%	68.78%	78.92%	57.04%	77.0%
14	87.60%	79.84%	87.64%	*73.88%*	83.5%
15	*89.36%*	77.26%	85.50%	72.74%	82.9%
16	87.94%	*81.12%*	*89.06%*	68.70%	*84.4%*

Table 5. Detection performance of Rice-YOLO on different pests in the IPR16 dataset. Significant values are in bold and italics.

The model comparison reveals that YOLOv3-Tiny, benefiting from its streamlined structure, had the fastest inference speed. In contrast, although YOLOv5-N and YOLOv5-N-BiFPN have fewer parameters and lower computational complexity, their more complex structures resulted in slower inference speeds than that of YOLOv3-Tiny. YOLOv9 exhibits slower inference speeds, while YOLOv10 demonstrates poor detection performance. In contrast, YOLOv8-N strikes a balance between model size and inference speed while achieving higher detection accuracy. A comparison of Rice-YOLO with classic models reveals that it offers the best

Model	Params (M)	Layers	*P*	*R*	mAP 50	mAP 50:95	F1	Speed (ms)	FLOPs (G)
FCOS-Resnet18	19.13	–	33.4%	49.8%	38.6%	25.5%	40.0%	45.5	133
SSD-Lite-Mobilenetv2	3.3	–	46.2%	40.1%	40.1%	23.9%	42.9%	24.2	**0.704**
SSD300	25.8	–	59.9%	64.7%	66.9%	40.2%	67.4%	20.8	31.174
SSD512	26.6	–	70.3%	69.3%	73.4%	44.0%	69.8%	32.6	89.647
Faster-Rcnn-Resnet18	28.7	–	56.1%	61.0%	59.5%	34.3%	58.4%	49.4	136
YOLOv3-Tiny	8.7	38	68.6%	61.0%	64.6%	34.6%	64.5%	**5.2**	12.9
YOLOv4-Tiny	6.1	99	27.0%	59.4%	42.0%	22.7%	37.1%	*7.1*	16.5
YOLOv5-N	**1.8**	157	65.6%	62.2%	65.3%	42.5%	63.9%	10.2	*4.2*
YOLOv5-N-BiFPN	*1.8*	157	67.5%	61.9%	66.5%	43.2%	64.5%	10.9	4.2
YOLOv5-S	7.1	157	73.5%	65.4%	70.4%	47.2%	69.2%	10.7	15.9
YOLOv5-M	20.9	212	70.4%	61.4%	66.4%	45.3%	70.4%	13.6	48.1
YOLOv6-N	4.2	142	74.7%	63.6%	71.2%	53.3%	68.7%	12.7	11.8
Gold-YOLO-N	5.6	–	43.1%	49.4%	45.8%	29.3%	46.1%	18.9	12.1
YOLOv7-Tiny	6.0	200	72.8%	64.5%	69.5%	45.3%	68.4%	31.8	13.2
YOLOv7-Tiny-Silu	6.0	200	71.3%	62.4%	67.3%	43.5%	66.6%	31.4	13.2
YOLOv8-N	3.0	168	71.6%	68.7%	73.1%	57.8%	70.1%	11.8	8.2
YOLOv8-N-BiFPN	2.0	193	73.5%	63.9%	71.8%	55.7%	68.4%	13.1	7.1
YOLOv8-N-Ghost	1.7	312	67.7%	60.2%	66.3%	49.6%	63.7%	11.5	5.1
YOLOv8-S	11.1	168	74.6%	*69.5%*	*74.9%*	*60.3%*	*72.0%*	12.3	28.5
YOLOv8-M	25.8	218	*75.9%*	64.4%	72.2%	57.1%	69.6%	13.9	78.7
YOLOv8-X	68.1	268	70.6%	68.7%	72.0%	56.4%	69.6%	34.3	257.5
YOLOv9-T	2.0	486	69.6%	59.3%	65.7%	52.4%	64.0%	27.1	7.6
YOLOv10-N	2.3	229	67.4%	61.9%	67.7%	52.1%	64.5%	8.9	6.5
Rice-YOLO (ours)	2.8	172	**78.1%**	**70.9%**	**78.1%**	**62.9%**	**74.3%**	11.2	7.2

Table 6. Performance Comparison of Object Detection Algorithms on the IPR16 dataset. Significant values are in bold and italics.

detection accuracy and relatively fast inference speed. Overall, Rice-YOLO achieves a good balance among speed, accuracy, and computational complexity.

We juxtaposed the detection results of select high-performing models with those of Rice-YOLO, as depicted in Fig. 12. Overall, other models exhibited more instances of missed detections and false-negatives. In contrast, Rice-YOLO showed improvements in this area, with fewer instances of missed pest objects and classification errors. Additionally, the predicted bounding boxes of Rice-YOLO more accurately matched the actual objects.

Additionally, we utilized LayerCAM[34] to visualize the detection layers (P3, P4, and P5) of different models, as depicted in Fig. 13. The visualizations of various models clearly reveal that some models exhibit severe dispersion of heatmaps, making it challenging to focus on actual objects. A scattered distribution of heatmaps indicates that the model struggles to effectively learn and recognize target features. Consequently, during the prediction phase, the model's results often deviate significantly from reality. Conversely, concentrated heatmaps reflect the model's focus on object features and indicate the centre positions of the predicted bounding boxes. In summary, across detection layers of varying scales, some models' heatmaps deviate from actual targets, whereas Rice-YOLO's heatmaps align more closely with the actual objects.

Ablation study of each function module within Rice-YOLO

To assess the contribution of each improvement to Rice-YOLO, we conducted ablation experiments, and the results are presented in Table 7. The results of the ablation experiments, clearly show that different improvement modules have varying effects on various aspects of the Rice-YOLO model. The metrics obtained in the ablation experiments clearly indicate that each improvement made to the model is necessary, as they all have a positive effect on the final detection results.

Compared with YOLOv8, in Rice-YOLO, the utilization of the DBG head resulted in a reduction of 6.28% in the number of model parameters and a decrease of 11.98% in the computational complexity, while increasing mAP50 by 2.6%, mAP50:95 by 2.2%, and F1 score by 1.6%. This finding indicates that replacing the original detection head with the DBG head improved the model's inference capability and reduced its hardware consumption.

The addition of Aux-DBG as a supervisory layer in the middle layers of the model network increased the mAP50 by 1.8%, the mAP50:95 by 2.4%, and the F1 score by 1.8% without adding any extra inference burden.

Substituting DySample–hardtanh for the original upsampling module improved the model's upsampling capability, resulting in a 0.6% increase in mAP50, a 0.5% increase in mAP50:95, and a 0.8% increase in the F1 score.

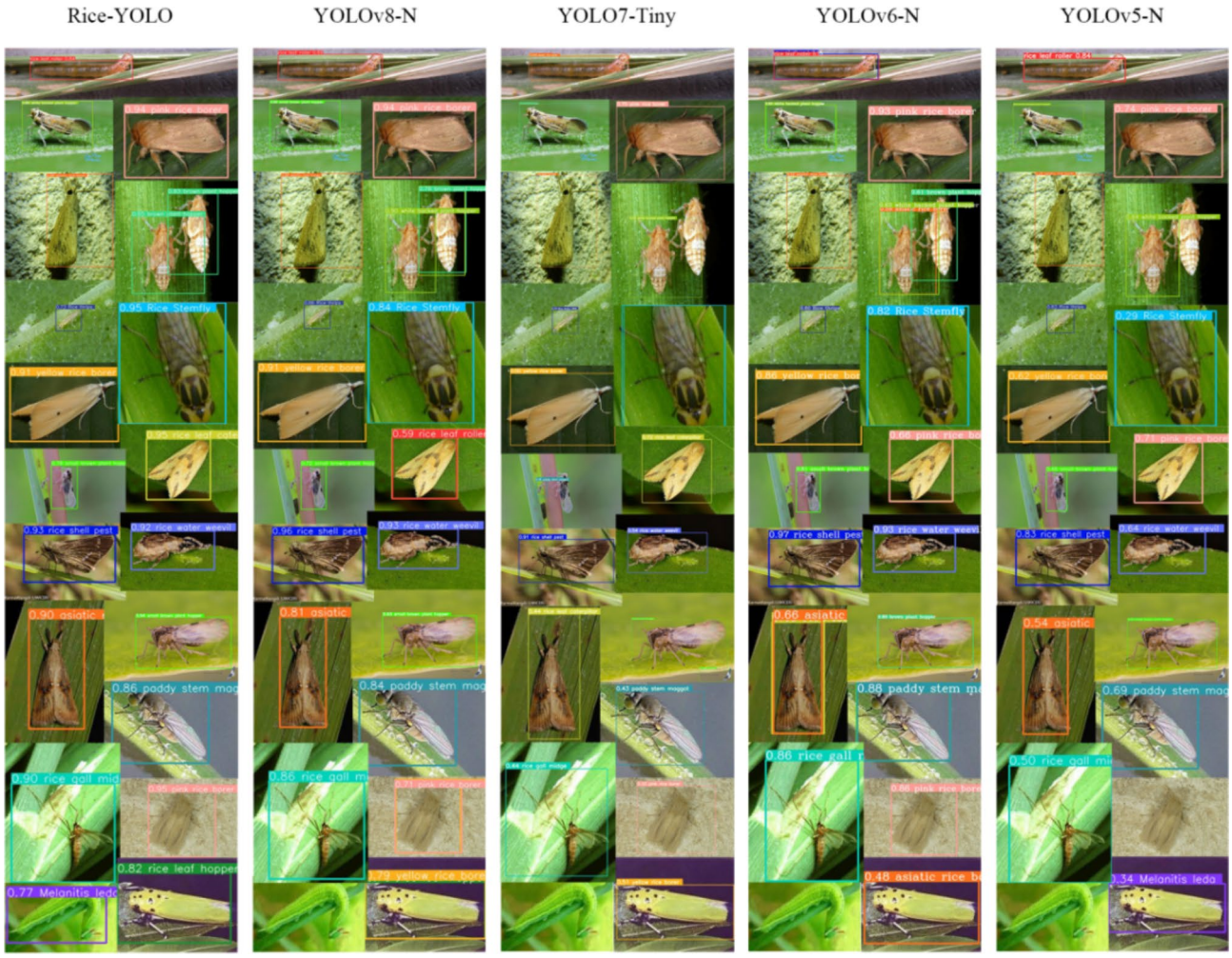

Fig. 12. Comparison of model detection results.

Furthermore, to compare the effect of the offset suppression strategy between DySample–hardtanh and DySample+, we visualized different-scale upsampling layers, as shown in Fig. 14. The visualization results indicate that, in the task of rice pest detection, the more flexible DySample–hardtanh outperforms DySample+.

Conclusions

In this study, we propose an accurate and fast rice pest detection method called Rice-YOLO to address the complexity of rice pest backgrounds, large interclass variations, and small interclass differences. Rice-YOLO achieves end-to-end efficient detection of rice pests, striking a good balance among speed, accuracy, and computational complexity. This method builds upon YOLOv8 by incorporating an efficient detection head that is tailored to the complex characteristics of pests and integrating deep supervision layers into the network, along with utilizing improved dynamic upsampling modules.

Currently, crop pest datasets are scarce. Data augmentation can be employed to expand data samples, and image generation techniques can be utilized to increase data richness. Owing to the inherent similarities among different crop pests, future research will explore training models based on multiple crop pest datasets and subsequently transferring the trained models to specific crop pest datasets for fine-tuning.

In the future, Rice-YOLO can be integrated with agricultural smart devices to transform and upgrade intelligent greenhouses and smart farms by combining intelligent sensing systems and detection systems in agricultural greenhouses and crop growing areas. This advancement will contribute to the development of the agricultural economy.

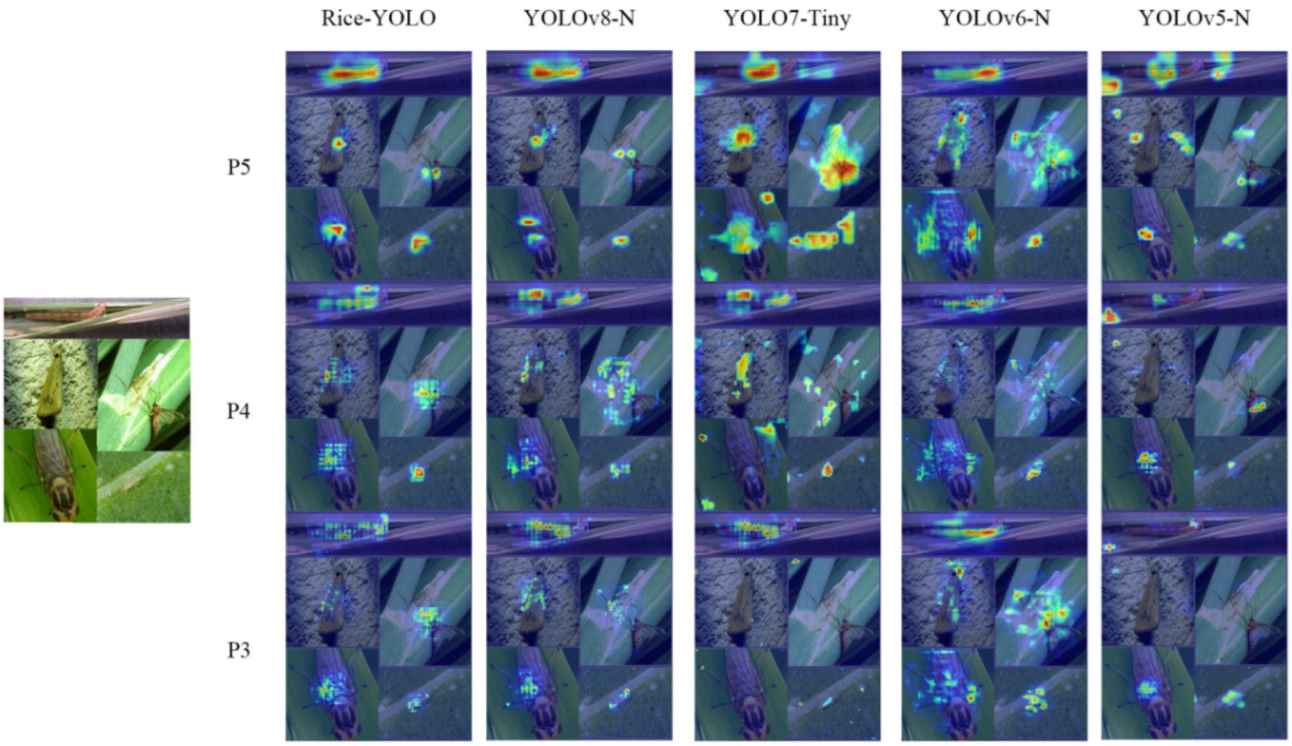

Fig. 13. Visualization of different scale detection layers.

v8	DBG	Aux DBG	DS	DS+	DSHT	Params (M)	Layers	*P*	*R*	mAP 50	mAP 50:95	F1	Speed (ms)	FLOPs (G)
√						3.0088	168	71.6%	68.7%	73.1%	57.8%	70.1%	11.6	8.10
√	√					2.8198	168	77.3%	66.8%	75.7%	60.0%	71.7%	**10.7**	**7.13**
√	√	√						77.9%	*69.5%*	77.5%	*62.4%*	73.5%		
√	√	√	√			2.8322	170	*79.5%*	69.4%	*77.7%*	62.3%	*74.1%*	11.3	*7.15*
√	√	√		√		2.8445	172	**79.7%**	67.4%	76.7%	62.1%	73.0%	*11.2*	7.17
√	√	√			√			78.1%	**70.9%**	**78.1%**	**62.9%**	**74.3%**		

Table 7. Ablation experiments of Rice-YOLO on IPR16.

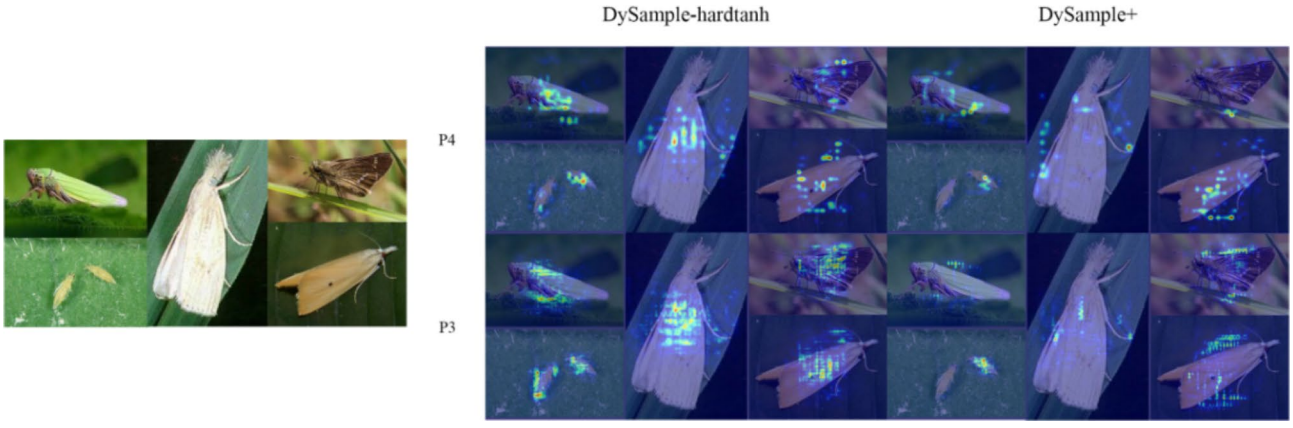

Fig. 14. Visualization of upsampling layers at different scales.

Data availability

The self-built dataset R2000 can be publicly available on https://github.com/awsomespark. Publicly available datasets IP102 were used in this study. IP102 can be found here: https://github.com/xpwu95/IP102. Additional supplements are available on request from zhengyongxmut@163.com.

Received: 7 June 2024; Accepted: 27 November 2024
Published online: 02 December 2024

References

1. Peng, S., Tang, Q. & Zou, Y. Current status and challenges of rice production in China. *Plant. Prod. Sci.* **12** (1), 3–8. https://doi.org/10.1626/pps.12.3 (2009).
2. Maddhi, S., Dodda, R., Naik, A. C. & Sinduja, K. Mitigating agricultural challenges: A comprehensive study on the impact of crop diseases on rice production in India. In *International Conference on Artificial Intelligence and Smart Energy.* 81–92. https://doi.org/10.1007/978-3-031-61475-0_7 (2024).
3. A Padmakumari, P., Kota, S. & Sundaram, R. M. Current status of host plant resistance to insects in rice and future perspectives. *Plant. Resist. Insects Major Field Crops*. 69–122. https://doi.org/10.1007/978-981-99-7520-4_4 (2024).
4. Elsayed, M. Z., Hasoon, A., Zidan, M. K. & Ayyad, S. M. Role of AI for plant disease detection and pest detection. *Int. Telecommun. Conf.* **2024**, 824–829. https://doi.org/10.1109/ITC-Egypt61547.2024.10620496. (2024).
5. Chuang, C. L. et al. Automatic X-ray quarantine scanner and pest infestation detector for agricultural products. *Comput. Electron. Agric.* **77** (1), 41–59. https://doi.org/10.1016/j.compag.2011.03.007 (2017).
6. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Commun. ACM*. **60** (6), 84–90. https://doi.org/10.1145/3065386 (2017).
7. Tang, R., Aridas, N. K., Talip, M. S. A. & Xinzheng, Y. Design of greenhouse vegetable pest and disease identification method based on improved AlexNet model. https://doi.org/10.21203/rs.3.rs-4343182/v1 (2024).
8. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. https://doi.org/10.48550/arXiv.1409.1556 (2014).
9. Devi, D., Devakadacham, S. R., Saveetha, D. & Manikandan, J. Enhancing insect species identification in agriculture using fusion BiLSTM network and VGG-16 CNN architecture. In *International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)* 1–6 (2024).
10. Szegedy, C. et al. Going deeper with convolutions. In *Proceedings of the IEEE Conf. Comput. Vis. Pattern Recogn.* 1–9 (2015).
11. Li, C. et al. An advancing GCT-Inception-ResNet-V3 model for arboreal pest identification. *Agronomy* **14** (4), 864. https://doi.org/10.3390/agronomy14040864 (2024).
12. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778 (2016).
13. Prabha, R. & Selvan, K. Modified RESNET50 with attention module for detection and classification of pests in vegetable crops. *J. Adv. Res. Appl. Sci. Eng. Technol.* 67–86. https://doi.org/10.37934/araset.63.1.6786 (2024).
14. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 4700–4708 (2017).
15. Yin, J., Zhang, H., Chen, Z. & Li, J. Detecting emerald ash borer boring vibrations using an encoder-decoder and improved DenseNet model. *Pest Manag. Sci.* https://doi.org/10.1002/ps.8442 (2024).
16. Ali, F., Qayyum, H., Iqbal, M. J. & Faster-PestNet a lightweight deep learning framework for crop pest detection and classification. *IEEE Access.* https://doi.org/10.1109/ACCESS.2023.3317506 (2023).
17. Wang, Z., Qiao, L. & Wang, M. Agricultural pest detection algorithm based on improved faster RCNN. In *International Conference on Computer Vision and Pattern Analysis (ICCPA 2021)*, vol. 12158, 104–109. https://doi.org/10.1117/12.2626859 (SPIE, 2022).
18. Li, W. et al. Field detection of tiny pests from sticky trap images using deep learning in agricultural greenhouse. *Comput. Electron. Agric.* **183**, 106048. https://doi.org/10.1016/j.compag.2021.106048 (2021).
19. Chu, J., Li, Y., Feng, H., Weng, X. & Ruan, Y. Research on multi-scale pest detection and identification method in granary based on improved YOLOv5. *Agriculture*. **13** (2), 364. https://doi.org/10.3390/agriculture13020364 (2023).
20. Xiang, Q. et al. Yolo-Pest: an insect pest object detection algorithm via CAC3 module. *Sensors* **23** (6), 3221. https://doi.org/10.3390/s23063221 (2023).
21. Yang, Z., Feng, H., Ruan, Y. & Weng, X. Tea tree pest detection algorithm based on improved Yolov7-Tiny. *Agriculture* **13** (5), 1031. https://doi.org/10.3390/agriculture13051031 (2023).
22. Wu, X., Zhan, C., Lai, Y. K., Cheng, M. M. & Yang, J. Ip102: A large-scale benchmark dataset for insect pest recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* 8787–8796 (2019).
23. Deng, J. et al. Imagenet: a large-scale hierarchical image database. *IEEE Conf. Comput. Vis. Pattern Recognit.* 248–255. https://doi.org/10.1109/CVPR.2009.5206848 (2009).
24. Lin, T. Y. et al. Microsoft Coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, Proceedings, Part V 13* 740–755. https://doi.org/10.1007/978-3-319-10602-1_48 (Springer, 2014).
25. Redmon, J. & Farhadi, A. Yolov3: An incremental improvement. https://doi.org/10.48550/arXiv.1804.02767 (2018).
26. Bochkovskiy, A., Wang, C. Y. & Liao, H. Y. M. Yolov4: optimal speed and accuracy of object detection. https://doi.org/10.48550/arXiv.2004.10934 (2020).
27. Jocher, G. et al. Ultralytics/Yolov5: V7. 0-Yolov5 Sota realtime instance segmentation. *Zenodo*. https://ui.adsabs.harvard.edu/abs/2022zndo…J/abstract (2022).
28. Li, C. et al. YOLOv6: A single-stage object detection framework for industrial applications. https://doi.org/10.48550/arXiv.2209.02976 (2022).
29. Wang, C. Y., Bochkovskiy, A. & Liao, H. Y. M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* 7464–7475 (2023).
30. Solawetz, J. F. What is YOLOv8? The ultimate guide (2023).
31. Wang, C. Y., Yeh, I. H. & Liao, H. Y. M. Yolov9: learning what you want to learn using programmable gradient information. *arXiv Preprint* (2024). arXiv:2402.13616.
32. Wang, A. et al. Yolov10: real-time end-to-end object detection. *arXiv Preprint* (2024). arXiv:2405.14458.
33. Range & King Github Blog. https://github.com/RangeKing.
34. Jiang, P. T. et al. Exploring hierarchical class activation maps for localization. *IEEE Trans. Image Process.* **30**, 5875–5888. https://doi.org/10.1109/TIP.2021.3089943 (2021).

Acknowledgements
This research was funded by the Natural Science Foundation of Fujian Province, China [Grant number 2021J05259], and the Open Project of Hunan Provincial Key Laboratory of Remote Sensing Monitoring of Ecological Environment in Dongting Lake Area, Grant number DTH Key Lab.2022-15.

Author contributions
Conceptualization and methodology, Y.Z. and W.Z.; software, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing and visualization, Y.Z.; validation and supervision, W.Z. and X.D.; project administration and funding acquisition, W.Z.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to W.Z.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.