scientific reports

Check for updates

OPEN

Enhanced YOLOv8 with lightweight and efficient detection head for for detecting rice leaf diseases

Bo Gan[1,2]✉, Guolin Pu[1], Weiyin Xing[3], Lianfang Wang[1] & Shu Liang[1]✉

Detecting rice leaf diseases is essential for agricultural stability and crop health. However, the diversity of these diseases, their uneven distribution, and complex field environments create challenges for precise, multi-scale detection. While YOLO object detection algorithms show strong performance in automated detection, their feature extraction capabilities remain limited in complex agricultural settings. Moreover, their high computational demands hinder deployment on resource-constrained devices, necessitating further optimization.To overcome these issues, This paper presents G-YOLO, a novel architecture that combines a Lightweight and Efficient Detection Head (LEDH) with Multi-scale Spatial Pyramid Pooling Fast (MSPPF). The LEDH enhances detection speed by simplifying the network structure while maintaining accuracy, reducing computational demands. The MSPPF improves the model's ability to capture intricate details of rice leaf diseases at various scales by fusing multi-level feature maps. On the RiceDisease dataset, G-YOLO surpasses YOLOv8n with 4.4% higher mAP@0.5, 3.9% higher mAP@0.75, and a 13.1% increase in FPS, making it well-suited for resource-constrained devices due to its efficient design.

Keywords Rice leaf disease detection, Agricultural stability, Crop health, Lightweight

The Food and Agriculture Organization of the United Nations (FAO) highlights in its latest report that precision agriculture technologies are crucial for ensuring global food security[1]. These technologies enable farmers to monitor crop health, optimize input usage, and improve resource efficiency, leading to increased yields and reduced environmental impact. Given the increasing global demand for food, the timely and accurate detection of rice leaf diseases has become a critical aspect of modern agricultural management. Precise disease monitoring helps optimize management strategies, improve yields, and minimize the environmental impact of agricultural practices. However, current detection technologies often struggle to balance accuracy with real-time performance, particularly in the complex and dynamic conditions of rice fields, presenting a significant challenge in practical applications.

These challenges arise from two primary factors. First, the variability in the rice growth cycle, fluctuating weather conditions, and the wide range of diseases complicate accurate disease identification. Yang et al.[2] proposed that early rice leaf diseases appear as small spots with irregular shapes in natural scene images, and existing models have certain difficulties in accurately identifying these diseases. This results in missed detections and lower diagnostic accuracy. Second, many detection models suffer from high computational complexity and large model sizes, which restrict their efficiency in real-time applications.These limitations hinder the effective deployment of these models in resource-constrained field environments, underscoring the need for more robust, efficient, and scalable disease detection algorithms. Addressing these issues is crucial for improving the effectiveness of disease monitoring and ensuring the long-term sustainability of rice farming, particularly in the face of changing global conditions.

Deep learning methods for detecting rice leaf diseases can be broadly categorized into two types. The first category includes two-stage detection algorithms based on region proposals, such as RCNN[3], Fast R-CNN[4], and Faster R-CNN[5]. While these methods offer high accuracy, their slower processing speeds limit their use in real-time applications. The second category comprises one-stage detection methods, such as the YOLO series[6–12], which provide faster processing by directly predicting from the image, while maintaining accuracy comparable to two-stage methods.

Despite the excellent performance of YOLO algorithms in various applications, they face two main challenges in rice leaf disease detection. First, the large number of model parameters limits their efficiency on resource-

[1]Dazhou Vocational and Technical College, Dazhou 635000, China. [2]Sichuan ZhiShiYunTong Technology Co., Ltd, Chengdu 610000, China. [3]Mianyang Polytechnic, Mianyang 621000, China. ✉email: ganbcdut@163.com; liangss2003@163.com

constrained devices in field environments, where limited computational resources can lead to increased inference time and processing delays. These inefficiencies may result in real-time monitoring failures, reducing the system's ability to promptly detect and respond to disease outbreaks. Second, their ability to handle multi-scale features in complex field conditions is still limited, leaving room for further improvement in detection accuracy. Therefore, optimizing the model's computational complexity and enhancing detection accuracy are crucial objectives for achieving more efficient real-time monitoring in field conditions.

YOLOv8[12] is widely regarded as the most popular object detection algorithm in the YOLO series. Building on the strengths of its predecessors, YOLOv8 introduces several improvements to enhance its performance in complex environments. The algorithm consists of three key components: the backbone, neck, and head. The backbone extracts image features through convolutional layers, the C2f module, and the SPPF module. The C2f module improves gradient flow and strengthens feature representation, while the SPPF module enhances multi-scale feature perception, thereby improving detection accuracy for objects of various sizes. The neck integrates a Feature Pyramid Network (FPN) and a Path Aggregation Network (PAN), allowing for bidirectional feature fusion and ensuring precise detection of both small and large objects. Positioned atop the PAN, the head includes detection heads that are tailored to different object sizes. Additionally, YOLOv8 incorporates Distribution Focal Loss (DFL)[13], which refines bounding box predictions by reducing uncertainty and enhancing localization precision.

Recent developments in YOLOv8 have concentrated on improving multi-scale feature extraction, enhancing attention mechanisms, and refining loss functions to elevate both accuracy and speed in complex environments. For instance, BGF-YOLO[14] features Bi-level Routing Attention (BRA)[15] and Generalized Feature Pyramid Networks (GFPN)[16], along with an additional detection head, which greatly enhances the representation of multi-scale features. UAV-YOLOv8[17] incorporates BiFormer[15] and Wise-IoU (WIoU)[18], optimizing both feature extraction and localization for improved stability in complex settings. YOLO-SE[19] utilizes an Efficient Multi-scale Attention Module with Cross-Spatial Learning (EMA)[20] to tackle the challenges of multi-scale detection, significantly boosting accuracy for small objects. Meanwhile, MHSA-YOLOv8[21] integrates Multi-Head Self-Attention (MHSA)[22], refining the feature extraction process in demanding environments. These advancements have significantly enhanced YOLOv8's performance across a variety of application areas.

In this paper, we introduce G-YOLO, a novel object detection framework designed to enhance both the real-time performance and accuracy of YOLOv8n in detecting rice leaf diseases by incorporating the LEDH and MSPPF modules. Compared to the traditional YOLOv8n model, G-YOLO achieves significant improvements in detection accuracy while maintaining high inference speed. Specifically, G-YOLO improves mAP@0.5 from 0.684 to 0.728 and mAP@0.75 from 0.145 to 0.184, demonstrating enhanced detection precision. Additionally, the inference speed increases from 90.5 FPS to 102.35 FPS, ensuring real-time performance. The main contributions of this work are as follows:

1. Existing object detection models face challenges in detection head complexity and inference delay, which impact real-time performance. To address these issues, we propose the Lightweight and Efficient Detection Head (LEDH), a novel parallel architecture that significantly reduces computational overhead and inference time, thereby enhancing real-time disease detection while maintaining accuracy.
2. The current methods have certain limitations in multi-scale feature fusion, which may affect the stability of disease detection in complex environments. To address this issue, we propose a Multi-Scale Fast Spatial Pyramid Pooling (MSPPF) module to more efficiently integrate multi-scale features and enhance information interaction across different scales, thereby improving system robustness. The MSPPF module significantly enhances the detection capability for rice leaf diseases, performing well across various disease sizes and also being applicable to detection tasks involving small targets.
3. Our experiments demonstrate that G-YOLO outperforms YOLOv8n in both accuracy and inference speed, making it highly suitable for deployment on resource-constrained devices, such as drones or edge computing platforms, in precision agriculture.

Related work

Traditional methods for rice leaf diseases detection

Early detection of rice leaf diseases and pests primarily relied on traditional image processing methods, including image enhancement, feature engineering, and classification models. Techniques such as denoising, color correction, and histogram equalization were commonly used to improve the distinguishability of diseased regions. Handcrafted features, including gradients, textures, colors, and shapes, were extracted using algorithms like Histogram of Oriented Gradients (HOG), Gray-Level Co-occurrence Matrix (GLCM), and Scale-Invariant Feature Transform (SIFT), followed by classification using models such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), or Random Forests. Disease region localization was typically performed using fixed window scanning or threshold-based segmentation to differentiate between healthy and infected leaf areas. However, these methods faced limitations in complex agricultural environments. Variations in lighting conditions, overlapping leaves, and background noise often led to unstable feature extraction, reducing detection accuracy. Additionally, sliding window-based methods incurred high computational costs, and handcrafted features struggled to generalize across different crops and disease types, compromising adaptability and robustness. With the advancement of deep learning, Convolutional Neural Network (CNN)-based approaches have gradually become the mainstream. Compared to traditional methods, CNNs can automatically learn discriminative features of plant diseases, reducing manual intervention while improving detection accuracy and generalization capability, making pest and disease monitoring more efficient and intelligent.

Deep learning-based methods for rice leaf disease detection

With the development of deep learning technologies, convolutional neural network (CNN)-based methods have become the mainstream approach for detecting rice leaf diseases. These methods, with their powerful feature learning capabilities, can maintain high detection accuracy in complex environments, especially in dynamic scenarios such as paddy field monitoring. However, existing detection methods still struggle to achieve optimal performance when facing complex backgrounds, variations in target size, object overlap, and occlusion.

To address these challenges, researchers have proposed various innovative solutions. For instance, Li et al.[23] introduced the Enhanced Feature Fusion and Target-Adaptive Network (EFFTAN), which significantly improved detection accuracy in dense spot disease detection. Haque et al.[24] applied the YOLOv5 model for rice leaf disease classification and detection, achieving excellent detection performance.

As computer vision, deep learning, and mobile computing technologies advance, rice leaf disease detection has gradually expanded to mobile platforms, such as unmanned aerial vehicles (UAVs) and handheld devices, improving detection flexibility and efficiency. Sangaiah et al.[25] proposed the G-SC method, combined with an improved YOLOv4 architecture, to enhance UAV-based disease detection in precision agriculture. Their R-UAV-Net integrates spatial and channel feature extraction blocks with attention mechanisms to optimize feature representation and improve detection performance. At the same time, Anandakrishnan et al.[26] proposed a lightweight rice seedling detection model. With a compact 9 M parameter size, it integrates 3D feature adaptation, convolutional block attention, coordinate attention, and depthwise point convolution modules, optimizing computational efficiency for UAV deployment.

Despite significant progress, existing rice leaf disease detection methods still face challenges. Many advanced models require substantial computational resources and have complex architectures, making real-time detection on resource-constrained devices difficult. Therefore, balancing high detection accuracy with computational efficiency, particularly in mobile devices and real-time monitoring systems, remains a key issue to address in this field.

Methods

G-YOLO refines the YOLOv8n[12] architecture by incorporating the LEDH and MSPPF modules, achieving a more effective balance between inference speed and detection accuracy, particularly in rice leaf disease detection tasks. While retaining the core backbone of YOLOv8n, G-YOLO introduces key innovations in the detection head and feature fusion components. This section provides an in-depth analysis of the G-YOLO architecture, detailing how the LEDH and MSPPF modules contribute to enhanced model performance.

LEDH module

The detection head in YOLO object detection models plays a pivotal role after the Feature Pyramid Network (FPN), responsible for analyzing multi-scale features to predict object bounding boxes and class labels. Traditionally, the detection head consists of two main components: a location predictor for estimating bounding box positions and a class classifier for identifying object categories. An optimized detection head significantly impacts both accuracy and inference speed, ensuring efficient real-time performance by effectively processing multi-scale features.

YOLOv8n is a lightweight version of YOLOv8, reducing the number of network layers and parameters to lower computational complexity, thereby balancing real-time performance and accuracy to some extent. It adopts a Decoupled Head structure, where classification and bounding box regression are handled by separate sub-networks. This architecture includes three detection heads, each corresponding to a different scale of feature maps to detect small, medium, and large objects. The decoupled design facilitates task specialization, improving both detection accuracy and efficiency. Additionally, it enables independent optimization of classification and localization tasks through distinct loss functions and training strategies. However, while this approach reduces task interference and enhances model stability, the decoupled structure also increases the overall parameter count and computational complexity, which may become a limiting factor for real-time performance, especially in resource-constrained environments.

To address these limitations, we introduce the LEDH module, purposefully designed to reduce computational overhead and parameter count while maintaining high detection accuracy, particularly for rice leaf disease detection. The LEDH module includes several key optimizations: shared convolutions streamline the model architecture, resulting in substantial reductions in parameters and computational demands; inspired by the Efficient Layer Aggregation Network (ELAN) structure, the module incorporates an efficient multi-layer feature aggregation mechanism, which significantly enhances the network's feature learning capability and detection accuracy, thereby improving detection performance[9]; the inclusion of multiple Conv_GN modules, which utilize Group Normalization (GN), further enhances localization and classification precision in object detection tasks[27]; and the module employs two independent convolutional branches for classification and bounding box regression, with a scaling factor layer applied to the shared convolution in the regression branch to accommodate multi-scale detection requirements. Collectively, these optimizations substantially diminish computational burden and parameter count while enhancing real-time performance and detection accuracy. The structure of the LEDH module is shown in Fig. 1(a).

G-YOLO is designed with three detection heads for detecting large, medium, and small objects, respectively. Each detection head includes an independent Conv_GN module that employs a 1×1 convolution to merge and adjust feature channels, generating its respective F1. The structure of the Conv_GN module is shown in Fig. 1(b). Each F1 is then evenly split along the channel dimension into two new feature maps: F2 and F3. F3 is sequentially processed by two Conv_GN modules: F3 is first processed by a 3×3 Conv_GN module, generating F4. Then, F4 is further processed by another 3×3 Conv_GN module to produce F5. These Conv_GN modules share weights across all detection heads. Next, F2, F3, F4, and F5 are concatenated along the channel dimension

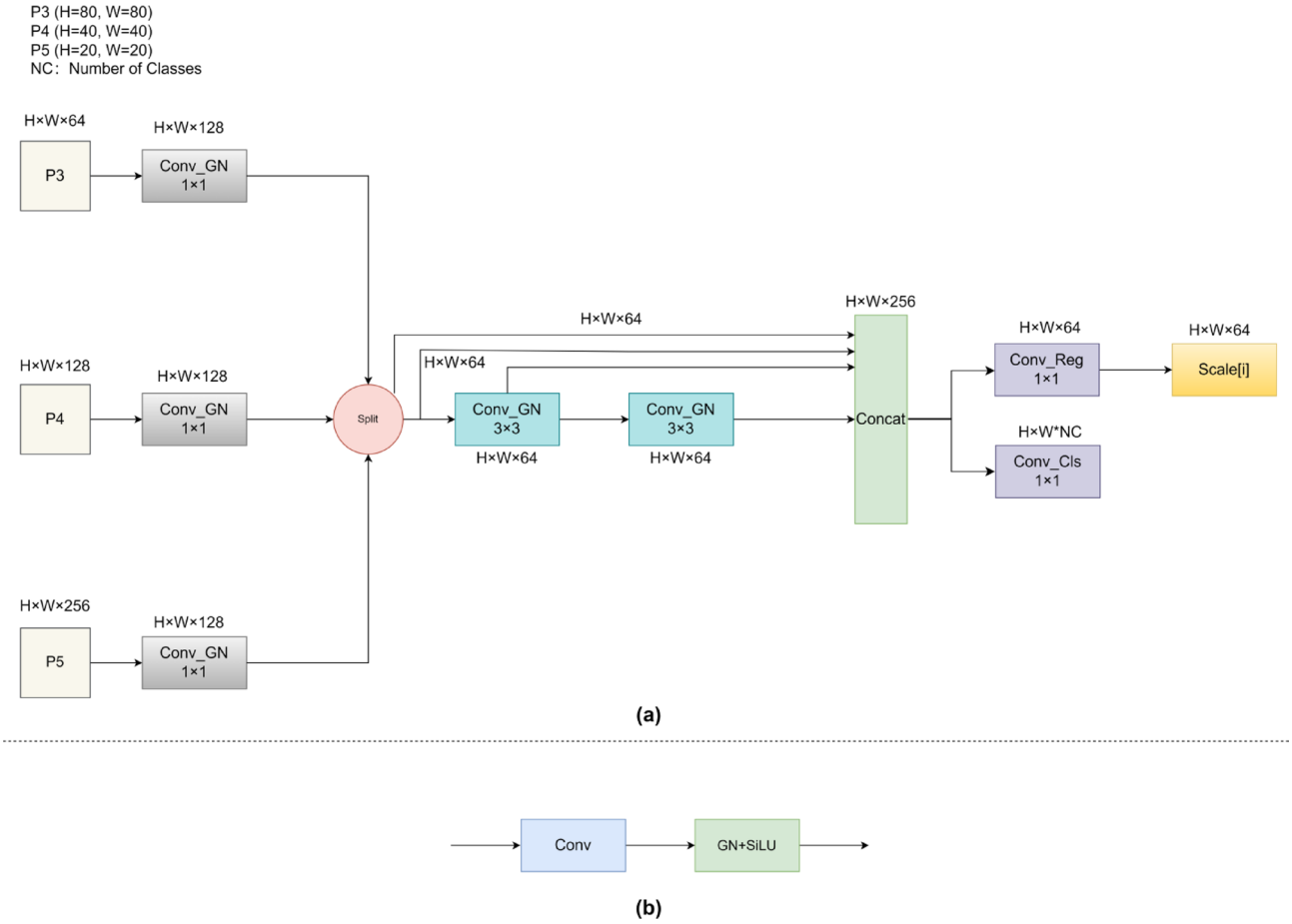

Fig. 1. (**a**) The structure of the LEDH module. (**b**) The structure of the Conv_GN module.

to form F6. F6 is then processed by two separate convolution branches, each with a 1×1 convolution, responsible for classification and bounding box regression tasks, respectively. These branches also share weights across all detection heads. To accommodate objects of varying sizes, each of the three detection heads independently learns a distinct scaling factor during training, denoted as Scale[i] (where $i = 0, 1, 2$). Each scaling factor is uniquely associated with a detection head, and these factors dynamically adjust the bounding box predictions, thereby improving the accuracy of multi-scale object detection.

MSPPF module

In the architecture of YOLOv8, the Spatial Pyramid Pooling Fast (SPPF)[12] module processes feature maps using serial max pooling operations with a fixed-size pooling kernel (e.g., 5×5). Compared to the traditional Spatial Pyramid Pooling (SPP)[28] module, which employs multiple parallel pooling paths, the SPPF module is more efficient. SPPF concatenates the feature maps produced by each pooling operation with the original feature map along the channel dimension. This approach not only significantly enhances the model's ability to capture targets across varying scales but also greatly reduces computational resource consumption by removing the overhead associated with parallel pooling paths. The SPPF design effectively balances the need for multi-scale information extraction with real-time processing requirements, allowing YOLOv8 to deliver outstanding performance in complex detection scenarios while maintaining high-speed operation. Nevertheless, there remains potential for further optimization in the area of multi-scale feature fusion.

In this paper, to address the limitations of multi-scale feature fusion in the SPPF module, we extend and optimize it by proposing the MSPPF module. Specifically, we designed the multi-scale feature concatenation (MSFC) module and integrated it with the SPPF module to form the MSPPF module. The MSFC module improves upon the existing multi-scale selective fusion (MSF) module[29], primarily by replacing the add operation in the feature fusion process with a concat operation. This enhancement significantly boosts the expressive power of multi-scale feature fusion and reduces information loss. By combining the foundational characteristics of the SPPF module with the strengths of the MSFC module, the MSPPF module demonstrates improved detection accuracy and performance in object detection tasks. The structure of the MSPPF module is shown in Fig. 2(a).

First, the feature map F is processed by the CBS module, generating F1. The structure of the CBS module is shown in Fig. 2(b). It enhances the network's feature extraction capability through a combination of convolution, batch normalization (BN), and the SiLU activation function. Then, F1 sequentially undergoes three serial max pooling operations, as follows: F1 is processed by the first 5×5 pooling kernel to generate F2; F2 is further

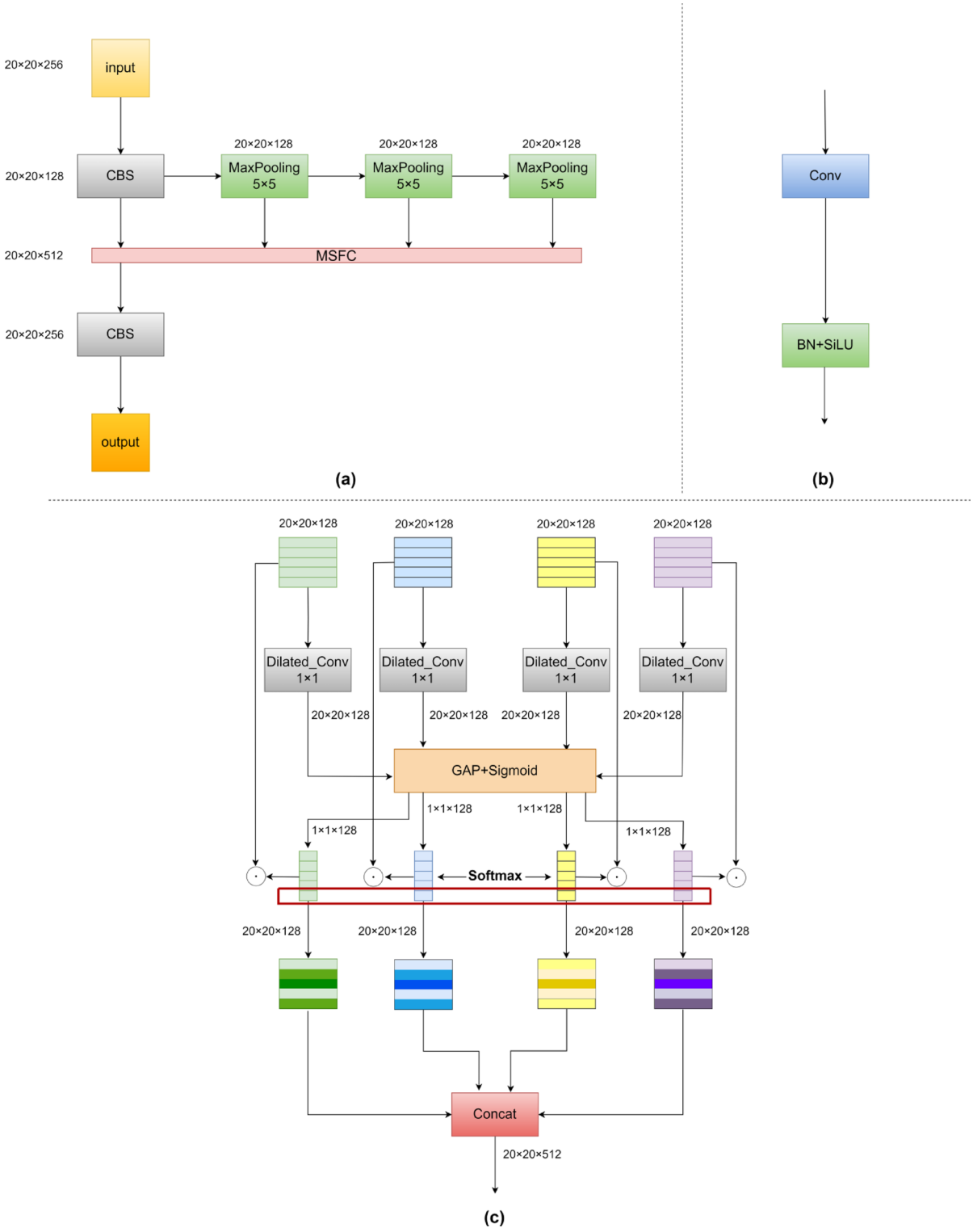

Fig. 2. (**a**) The structure of the MSPPF module. (**b**) The structure of the CBS module. (**c**) The structure of the MSFC module.

processed by the second 5×5 pooling kernel to generate F3; finally, F3 is processed by the third 5×5 pooling kernel to generate F4. The formulas are shown in Eqs. (1)-(4).

$$\mathbf{F}_1 = \mathrm{Conv}_{1\times 1}(\mathbf{F}), \mathbf{F}_1 \in \mathbb{R}^{B\times \frac{C}{2}\times H\times W}, \mathbf{F} \in \mathbb{R}^{B\times C\times H\times W} \tag{1}$$

$$\mathbf{F}_2 = \mathrm{MaxPooling}(\mathbf{F}_1), \ \mathbf{F}_1 \in \mathbb{R}^{B\times \frac{C}{2}\times H\times W} \tag{2}$$

$$\mathbf{F}_3 = \text{MaxPooling}\left(\mathbf{F}_2\right),\ \mathbf{F}_2 \in \mathbb{R}^{B\times \frac{C}{2}\times H\times W} \tag{3}$$

$$\mathbf{F}_4 = \text{MaxPooling}\left(\mathbf{F}_3\right),\ \mathbf{F}_3 \in \mathbb{R}^{B\times \frac{C}{2}\times H\times W} \tag{4}$$

Next, we input the feature maps F1, F2, F3, and F4 at different scales into the MSFC module for feature fusion, as shown in Fig. 2(c). Initially, inter-channel relationships are modeled through a 1×1 dilated convolution. Subsequently, global average pooling (GAP) is conducted on each feature map to derive the averages for each channel, and the resulting weights are transformed to fit within the 0 to 1 range via the Sigmoid activation function. Afterward, the multi-scale feature channel weights undergo normalization through the Softmax function, ensuring a balanced distribution. Ultimately, these normalized weights are multiplied with the original feature maps in an element-wise manner, and the modified feature maps are combined along the channel axis to create a new representation of multi-scale features. This feature map is then further processed by the CBS module to extract more detailed feature information, ultimately producing the output of the module. The MSFC module skillfully integrates convolutional techniques with attention mechanisms, thereby effectively blending fine image details with broader contextual information.

Experiments and results

Dataset

In this study, we utilized the publicly available RiceDisease dataset from Kaggle[30], which contains 850 images of rice leaf diseases. The dataset encompasses instances of rice leaf diseases with varying sizes and shapes, captured under diverse conditions, including low lighting and varying visibility scales. This diversity in image data effectively simulates real-world scenarios, enhancing the model's robustness and generalization capability across different conditions. The dataset covers three common rice leaf diseases: Bacterial Leaf Blight, Blast, and Brown Spot, with 434, 869, and 1878 instances, respectively. It is split into training, validation, and test sets, containing 583, 160, and 107 images of rice leaf diseases, respectively(representing approximately 69%, 19%, and 12% of the dataset).

In the analysis of the RiceDisease dataset, we conducted a detailed exploration from different perspectives through four charts. Figure 3(a) presents the histogram of the training set, showing the distribution of instances across three disease categories (Bacterial Leaf Blight, Blast, and Brown Spot), highlighting the issue of class imbalance, with Brown Spot having the most instances and bacterial Leaf Blight having the fewest. Figure 3(b) illustrates the distribution of bounding box sizes, revealing that small and medium-sized boxes dominate, providing key insights into the scale features of the targets in the dataset. Figure 3(c) visualizes the spatial distribution of bounding boxes, uncovering a trend where most labeled objects are concentrated in the center of the image, a pattern commonly observed in natural image datasets. Finally, Fig. 3(d) displays the aspect ratio distribution of the bounding boxes, indicating that most labeled objects tend to have square or vertically elongated shapes. These analyses contribute to a comprehensive understanding of the dataset.

Experimental environment

We drew on the prior research experiences of MobileNet[28,31,32] and ShuffleNet[33,34], utilizing high-performance hardware for training and testing the accuracy and complexity of our model. We also conducted real-time testing on resource-constrained devices to ensure that we could comprehensively evaluate the model's performance under different resource conditions.

All training processes and evaluations of accuracy and complexity-related metrics were conducted on a high-performance server, including Mean Average Precision (mAP), the number of parameters (Params) and model size. The use of the PyTorch framework and CUDA acceleration libraries fully leveraged the computational power of the GPU, speeding up model training and inference. Table 1 provides the configuration details of the server, while Table 2 shows the hyperparameter configuration details.

The resource-constrained device used an NVIDIA GeForce RTX 3050 Laptop GPU with 4GB of VRAM, primarily for evaluating real-time performance metrics such as inference speed, Frames Per Second (FPS), and Giga Floating Point Operations Per Second (GFLOPS). This lower-power GPU was chosen to assess the model's inference performance in environments with limited computational resources, especially when handling lightweight inference tasks. Table 3 provides the configuration details of this device.

Evaluation metrics

Model evaluation is crucial for determining a model's performance and its compatibility with the research objectives. In the rice leaf disease detection task, we employed the following evaluation metrics to comprehensively measure the performance of lightweight models: mAP, Params, model size, GFLOPS, inference time, and FPS. All experimental results in this study were obtained on the test set to ensure transparency and comparability of the results. The formulas for these evaluation metrics are shown in Eqs. (5)-(9).

1. **mAP**: mAP is a crucial metric for evaluating object detection models. It is derived by calculating the Average Precision (AP) for each class and then averaging the AP values across all classes to assess the model's overall performance. AP for each class is determined from the precision-recall curve, allowing mAP to capture both precision (P) and recall (R) at various recall levels, providing a comprehensive evaluation of detection capability. mAP@0.5 refers to the mean Average Precision at an Intersection over Union (IoU) threshold of 0.5, measuring how well the model predicts bounding boxes with at least 50% overlap with ground truth. mAP@0.75 uses a 75% IoU threshold, setting a higher bar for localization accuracy by requiring more overlap between predicted and actual boxes. In these calculations, TP denotes true positives, FP denotes false

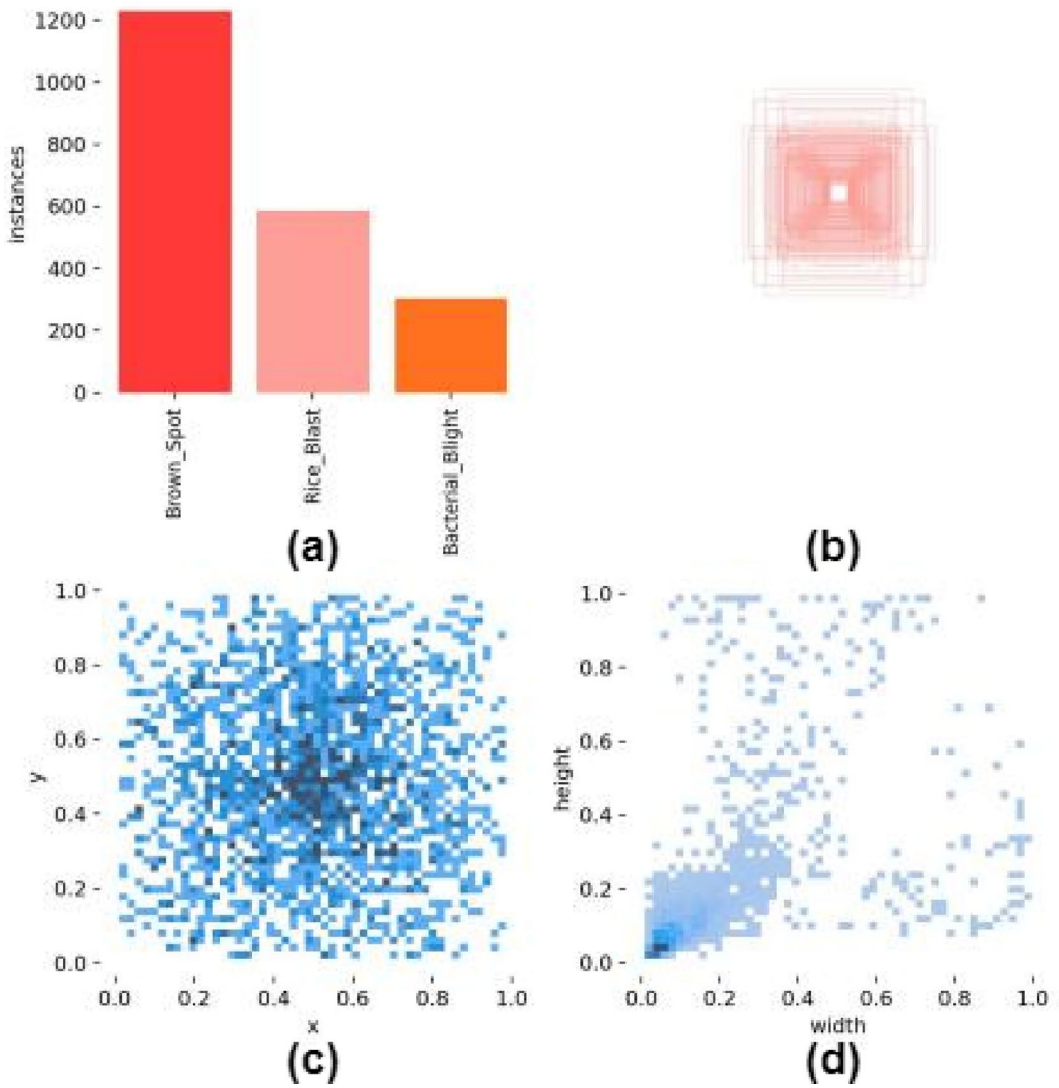

Fig. 3. (**a**) Histogram of instance distribution across disease categories.(**b**) Bounding Box Size Distribution Analysis. (**c**) Spatial Distribution of Bounding Boxes in Images. (d) Aspect Ratio Distribution of Bounding Boxes.

Device	Configuration
GPU	NVIDIA GeForce RTX 3090
VRAM	24GB
Operating System	Ubuntu18.04
Framework	PyTorch 1.11
CUDA	11.3
Python	3.8

Table 1. High-Performance server Configuration.

positives, FN denotes false negatives, P(r) represents precision at recall r, AP_i indicates the average precision for class i, and C represents the total number of classes.

$$P = \frac{TP}{TP + FP} \tag{5}$$

$$R = \frac{TP}{TP + FN} \tag{6}$$

Parameter	Value
Epoch	500
Patience	50
Batch Size	16
Image Size	640
Pretrained	False
Optimizer	AdamW
Initial Learning Rate	0.001

Table 2. Hyperparameter Settings.

Device	Configuration
GPU	NVIDIA GeForce RTX 3050 Laptop
VRAM	4GB
Operating System	Windows 11
Framework	PyTorch 1.11
CUDA	11.3
Python	3.8

Table 3. Resource-Constrained device Configuration.

$$\mathrm{AP} = \int_0^1 \mathrm{P}\left(\mathrm{r}\right) \mathrm{dr} \tag{7}$$

$$\mathrm{mAP} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{C}} \mathrm{AP_i}}{\mathrm{C}} \tag{8}$$

2. **Params**: The parameter count indicates a model's complexity and computational requirements. Fewer parameters usually suggest a more lightweight model, enabling it to function more efficiently on devices with limited resources, improving both training and inference speeds. For rice leaf disease detection, models with reduced parameters excel in resource-limited settings, cutting down computational and storage demands, which boosts system responsiveness and overall processing performance.
3. **Model Size**: The model size influences storage requirements, loading time, and computational cost. Smaller models save storage space, speed up loading, and lower computational expenses. In rice leaf disease detection, model size directly affects deployment and operational efficiency, particularly in resource-limited environments.
4. **GFLOPS**: GFLOPS quantifies a model's computational capability, indicating the number of floating-point operations performed per second. Lightweight models generally have lower GFLOPS values, reflecting reduced computational complexity and optimized efficiency. A lower GFLOPS value signifies effective optimization of resource requirements and power consumption, enhancing model efficiency in resource-constrained environments while still providing reasonable performance.
5. **Inference Time**: Inference time is the duration needed for a model to analyze a single image, measured in milliseconds (ms). Reduced inference times enhance the model's responsiveness in real-time applications, improving user experience. In rice leaf disease detection, shorter inference times enable quick processing of input data and provide immediate feedback, which is vital for applications requiring rapid detection and response.
6. **FPS**: FPS indicates the number of image frames the model processes each second and is essential for evaluating its real-time processing capability. A higher FPS results in a smoother experience, which is vital for real-time rice leaf disease detection.

$$\mathrm{FPS} = \frac{1000}{\text{Inference Time}} \tag{9}$$

where Inference Time is the inference time measured in milliseconds.

Model training results

In this study, we compared the accuracy of YOLOv8n and G-YOLO, using YOLOv8n as the baseline model. To avoid overfitting and enhance the model's generalization ability, we employed Early Stopping and Mosaic augmentation during training. Early Stopping helps prevent overfitting by monitoring the model's performance on the validation set, halting training when the performance no longer shows significant improvement. The patience period was set to 50 epochs, with a total training epoch limit of 500. Mosaic augmentation, by randomly scaling, cropping, and stitching four images together, enhanced the diversity of the training data, helping the

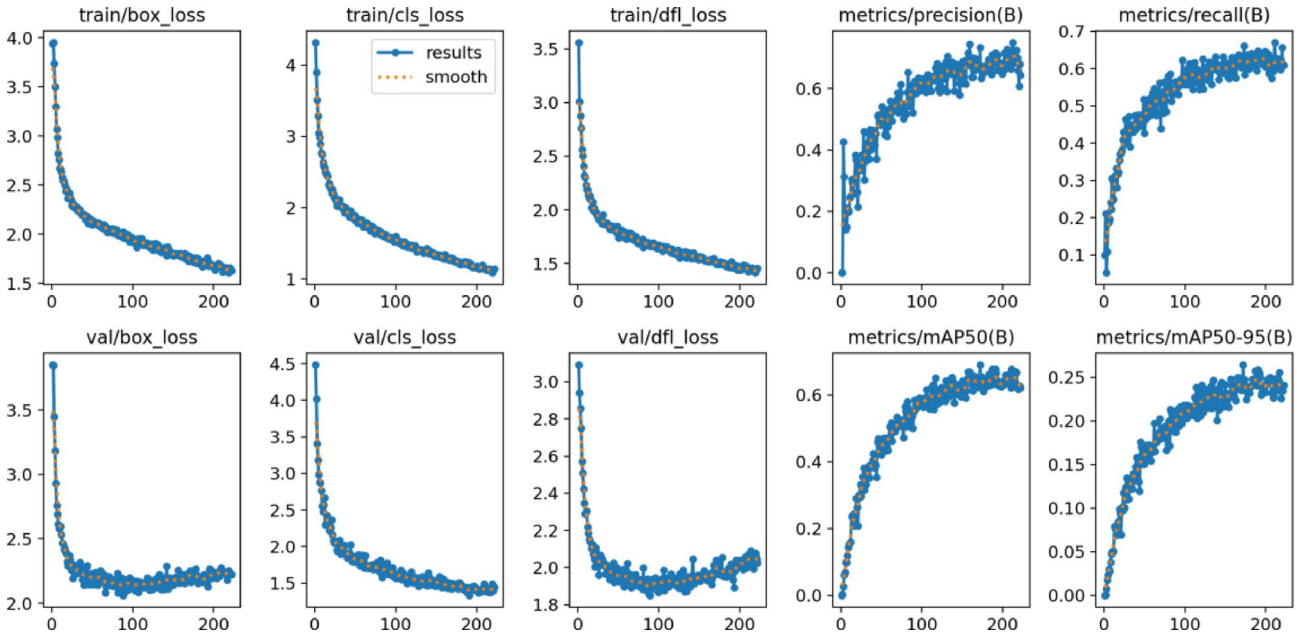

Fig. 4. Model Training Results.

Model	Mosaic	mAP@0.5	mAP@0.75
G-YOLO	×	0.696	0.157
G-YOLO	√	**0.728**	**0.184**

Table 4. Performance comparison of G-YOLO with and without mosaic augmentation.

model generalize better, especially in handling class imbalance. It increased the exposure of underrepresented classes, improving the model's ability to learn from them and reducing bias toward the majority class. As shown in Fig. 4, the G-YOLO model demonstrated rapid convergence in both training and validation losses within the first 200 epochs, and the performance stabilized thereafter. Notably, despite setting the maximum training epochs to 500, the training process was terminated early due to Early Stopping when the validation performance stopped improving, effectively preventing overfitting.

To further evaluate the impact of Mosaic augmentation, we compared the performance of G-YOLO with and without this augmentation on the RiceDisease test set. The results, summarized in Table 4, focus on mAP@0.5 and mAP@0.75, which measure overall detection accuracy and localization precision, respectively. In the experiments, '✓' indicates that Mosaic was applied, while '×' indicates it was not.

As shown in Table 4, Mosaic augmentation improves detection performance, increasing both mAP@0.5 and mAP@0.75. The enhancement in mAP@0.75 suggests better localization precision under stricter IoU thresholds. Based on these results, all subsequent experiments incorporate Mosaic augmentation as a standard preprocessing step.

Figures 5(a) and 5(b) display the PR curves for the YOLOv8n and G-YOLO algorithms on the RiceDisease test set. The PR curve for G-YOLO surpasses that of YOLOv8n, demonstrating that the G-YOLO model outperforms YOLOv8n. Moreover, G-YOLO enhances the mAP@0.5 metric by 4.4% compared to YOLOv8n.

Ablation study

We conducted a series of ablation experiments to assess the contributions of various modules in the G-YOLO model and to analyze the impact of these modules on detection performance through extensive experimental evaluation. We used the original YOLOv8n without any enhancement modules as the baseline and evaluated the effects of each improvement module on the RiceDisease dataset. In the experiments, The original YOLOv8n without enhancement modules was used as the baseline, and the effect of each improvement module was evaluated on the RiceDisease dataset. In the experiments, '✓' marks an enabled module, while '×' marks a disabled one.

From Table 5, it can be observed that integrating the MSPPF and LEDH modules into the original YOLOv8 network improves the experimental results to varying degrees for the RiceDisease dataset. Specifically, adding the MSPPF module resulted in a relative increase of 0.8% in mAP@0.5 and a 1% increase in mAP@0.75, indicating that this module better captures and integrates multi-scale contextual information. Adding the LEDH module reduced the model size by 1.2 MB, increased FPS by 17.4%, and improved mAP@0.5 and mAP@0.75 by 0.7% respectively. This shows that the module not only effectively reduces the model size and improves real-

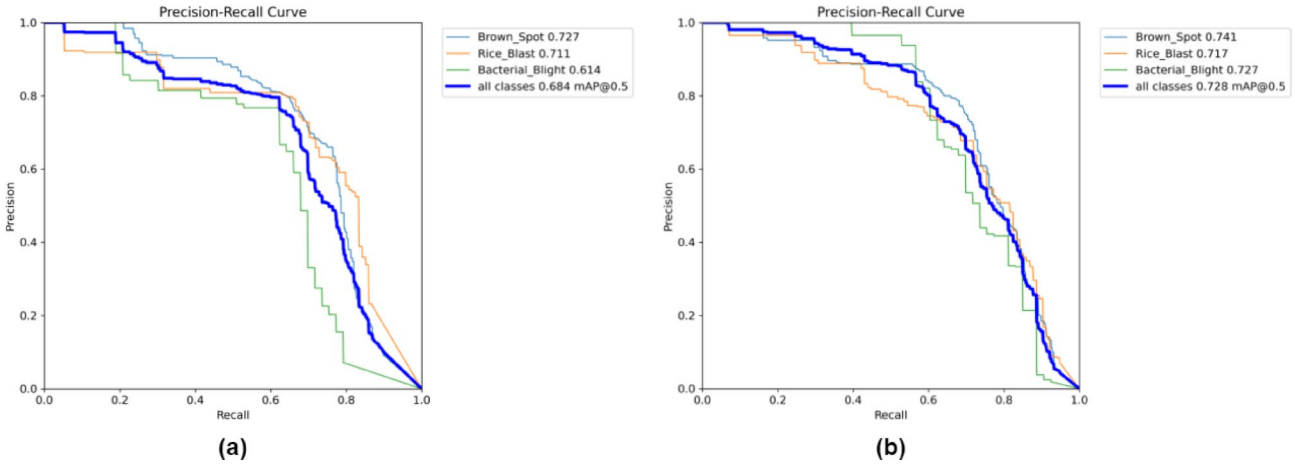

Fig. 5. (**a**) The PR curve of YOLOv8n. (**b**) The PR curve of G-YOLO.

Model	MSPPF	LEDH	mAP@0.5	mAP@0.75	Params (MB)	Model Size(MB)	GFLOPS	Inference Time(ms)	FPS
YOLOv8n	×	×	0.684	0.145	3.01	6.2	8.2	11.05	90.50
	√	×	0.692	0.155	3.34	6.9	8.3	11.24	88.97
	×	√	0.691	0.152	**2.40**	**5.0**	**6.9**	**9.41**	**106.27**
G-YOLO	√	√	**0.728**	**0.184**	2.73	5.7	7.0	9.77	102.35

Table 5. Ablation experiment results on the ricedisease test set.

Model	mAP@0.5	mAP@0.75	Params(MB)	Model Size(MB)	GFLOPS	Inference Time(ms)	FPS
YOLOv3-tiny	0.625	0.134	12.13	24.4	19.0	10.12	98.81
YOLOv5n	0.717	0.157	2.50	5.2	7.2	10.95	91.32
YOLOv6n	0.679	0.153	4.23	8.7	11.9	11.65	85.84
YOLOv8n	0.684	0.145	3.01	6.2	8.2	11.05	90.50
YOLOv9t	0.678	0.148	**2.00**	**4.6**	7.9	19.97	50.08
YOLOv10n	0.649	0.145	2.70	5.7	8.4	13.49	74.13
YOLOv11n	0.692	0.163	2.59	5.5	**6.6**	**9.24**	**108.23**
G-YOLO(ours)	**0.728**	**0.184**	2.73	5.7	7.0	9.77	102.35

Table 6. Comparative experiment results on the ricedisease test set.

time inference speed but also significantly enhances target localization accuracy and classification precision for tasks with fewer classes. The proposed G-YOLO algorithm achieves the best performance in both mAP@0.5 and mAP@0.75, with improvements of 4.4% and 3.9%, respectively. Additionally, the model size is reduced by 0.5 MB, and FPS increases by 13.1%.

Comparative experiments

In this study, since G-YOLO is a lightweight model, we compared it with several other lightweight YOLO models, including YOLOv3-tiny[35], YOLOv5n[36], YOLOv6n[37], YOLOv8n[12], YOLOv9t[38], YOLOv10n[39], and YOLOv11n[40]. The experimental results are shown in Table 6. G-YOLO achieved the best performance in terms of mAP@0.5 and mAP@0.75, while its performance in GFLOPS, inference time, and FPS is very close to that of YOLO11n. Additionally, compared to the baseline model YOLOv8n, G-YOLO reduced the model size by 0.5 MB, demonstrating its effectiveness in reducing the model size while maintaining high performance.

To evaluate the detection performance of the G-YOLO model, we selected representative sample images from the RiceDisease test set and presented the detection results of various models, including YOLOv3-tiny, YOLOv5n, YOLOv6n, YOLOv8n, YOLOv9t, YOLOv10n, YOLOv11n, and G-YOLO. The detection results of all models were compared using the same confidence threshold (conf = 0.3) and Intersection over Union (IoU) threshold (IoU = 0.5). Figure 6 illustrates the comparison of detection results.

From the figures, it can be observed that the detection performance of G-YOLO is comparable to that of other YOLO models, but it exhibits certain advantages in specific cases. Specifically, Fig. 6(a) shows that

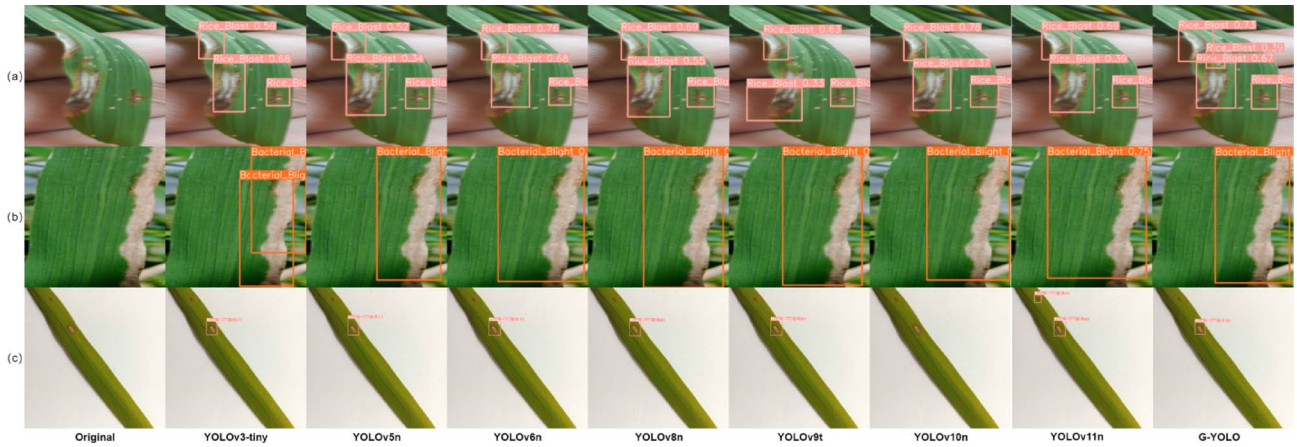

Fig. 6. The comparison of detection results.

G-YOLO detects rice leaf diseases more accurately, significantly reducing missed detections, while other models experience missed detections in the same region. Figure 6(b) indicates that G-YOLO annotates the diseased areas more precisely, improving detection accuracy compared to some models. Figure 6(c) demonstrates that G-YOLO successfully highlights a small and distinct disease spot with a confidence score of 0.81, while YOLOv10n fails to recognize this target in the visualization, and YOLOv11n produces a false positive.

Conclusion

This study proposes an improved object detection algorithm, G-YOLO, based on YOLOv8n, which significantly enhances the performance of rice leaf disease detection on the RiceDisease dataset by integrating the LEDH and MSPPF modules. Specifically, the introduction of the LEDH module effectively reduces the model size by 1.2 MB while increasing the FPS by 17.4%. In terms of detection accuracy, the LEDH module improves mAP@0.5 and mAP@0.75 by 0.7% each, indicating a notable enhancement in target localization precision and classification accuracy for detecting rice leaf diseases. Additionally, the integration of the MSPPF module improves the model's ability to capture multi-scale contextual information, increasing mAP@0.5 by 0.8% and mAP@0.75 by 1%. This demonstrates that the MSPPF module better integrates multi-scale features, thereby improving the model's detection performance.

The integration of these optimization strategies endows G-YOLO with a significant advantage in the task of rice leaf disease detection, achieving the best performance in mAP@0.5 and mAP@0.75, with improvements of 4.4% and 3.9%, respectively. Additionally, the model size is reduced by 0.5 MB, and the FPS is increased by 13.1%.

Beyond performance gains, G-YOLO demonstrates distinct advantages for agricultural applications. Its lightweight architecture and enhanced detection precision make it well-suited for deployment on resource-constrained edge devices such as drones and handheld agricultural monitoring systems, facilitating real-time disease detection across large-scale rice fields. The improved detection accuracy supports timely disease identification, which is crucial for effective disease management and yield protection. Furthermore, the model's robustness against variations in lighting conditions and complex field backgrounds enhances its practical applicability, addressing challenges that often hinder conventional detection methods.

Although G-YOLO has achieved significant results in rice leaf disease detection, subsequent research needs to explore its adaptability in multi-crop disease detection to verify its generalization ability in diverse agricultural environments. Additionally, integrating G-YOLO with edge computing frameworks holds the potential to further enhance its real-time processing capabilities. Future research can also leverage techniques such as domain adaptation and self-supervised learning to improve the model's robustness in complex agricultural scenarios.

By demonstrating how precise network design and module optimization significantly improve detection performance in complex rice leaf disease scenarios, this study provides valuable insights for the development of more efficient and scalable plant disease detection solutions.

Data availability

All the data mentioned in the paper are available through the corresponding author.

Received: 26 October 2024; Accepted: 11 June 2025
Published online: 01 July 2025

References

1. FAO. The State of Food and Agriculture 2022. FAO. (2022). https://www.fao.org/publications/sofa/2022/en
2. Yang, L. et al. GoogLeNet based on residual network and attention mechanism identification of rice leaf diseases. *Comput. Electron. Agric.* **204**, 107543 (2023).

3. Girshick, R., Donahue, J., Darrell, T. & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR* 580–587. https://doi.org/10.1109/CVPR.2014.81 (2014).
4. Girshick, R. & Fast, R-C-N-N. *ICCV*,1440–1448. (2015). https://doi.org/10.1109/ICCV.2015.169
5. Ren, S., He, K., Girshick, R., Sun, J. & Faster, R-C-N-N. Towards real-time object detection with region proposal networks. *TPAMI* **39**, 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031 (2017).
6. Redmon, J. & Farhadi, A. YOLOv3: An incremental improvement. Preprint at (2018). https://arxiv.org/abs/1804.02767
7. Li, C. et al. YOLOv6: A single-stage object detection framework for industrial applications. Preprint at (2022). https://arxiv.org/abs/2209.02976
8. Li, C. et al. YOLOv6 v3.0: A full-scale reloading. *Preprint At.* https://doi.org/10.48550/arXiv.2301.05586 (2023).
9. 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. *CVPR*, 7464–7475. (2023). https://doi.org/10.1109/CVPR52729.2023.00721
10. Wang, C. Y., Yeh, I. H. & Liao, H. Y. M. YOLOv9: Learning what you want to learn using programmable gradient information. Preprint at (2024). https://arxiv.org/abs/2402.13616
11. Wang, A. et al. YOLOv10: Real-time end-to-end object detection. Preprint at (2024). https://arxiv.org/abs/2405.14458
12. Ultralytics YOLOv8. (2024). https://github.com/ultralytics/ultralytics/tree/v8.1.47
13. Li, X. et al. Generalized focal loss: learning qualified and distributed bounding boxes for dense object detection. *NeurIPS* **33**, 21002–21012 (2020).
14. Kang, M., Ting, C. M., Ting, F. F. & Phan, R. C. W. BGF-YOLO: enhanced YOLOv8 with multiscale attentional feature fusion for brain tumor detection. *MICCAI* **15008**, 35–45. https://doi.org/10.1007/978-3-031-72111-3_4 (2024).
15. Zhu, L. et al. BiFormer: Vision transformer with bi-level routing attention. *CVPR*, 10323–10333 (2023).
16. Jiang, Y. et al. GiraffeDet: A heavy-neck paradigm for object detection. *ICLR* (2022).
17. Wang, G. et al. UAV-YOLOv8: A small-object-detection model based on improved YOLOv8 for UAV aerial photography scenarios. *Sensors* **23**, 7190. https://doi.org/10.3390/s23167190 (2023).
18. Tong, Z., Chen, Y., Xu, Z. & Yu, R. Wise-IoU: Bounding box regression loss with dynamic focusing mechanism. Preprint at (2023). https://arxiv.org/abs/2301.10051
19. Wu, T., Dong, Y. & YOLO-SE Improved YOLOv8 for remote sensing object detection and recognition. *Appl. Sci.* **13**, 12977. https://doi.org/10.3390/app132412977 (2023).
20. Ouyang, D. et al. Efficient multi-scale attention module with cross-spatial learning. *ICASSP* **1** (5). https://doi.org/10.1109/ICASSP49357.2023.10096516 (2023).
21. Li, P. et al. Tomato maturity detection and counting model based on MHSA-YOLOv8. *Sensors* **23**, 6701. https://doi.org/10.3390/s23156701 (2023).
22. Vaswani, A. et al. Attention is all you need. *Preprint At.* https://doi.org/10.48550/arXiv.1706.03762 (2017).
23. Li, Z. et al. Rice leaf disease detection based on enhanced feature fusion and target adaptation. *Plant. Pathol.* **73**, 846–858 (2024).
24. Haque, M. E., Rahman, A., Junaeid, I., Hoque, S. U. & Paul, M. Rice leaf disease classification and detection using YOLOv5. Preprint at (2022). https://arxiv.org/abs/2209.01579
25. Sangaiah, A. K. et al. R-UAV-Net: enhanced YOLOv4 with graph-semantic compression for transformative UAV sensing in paddy agronomy. *IEEE Trans. Cogn. Commun. Netw.* **11**, 1197–1209 (2025).
26. Anandakrishnan, J. et al. Precise Spatial prediction of rice seedlings from large-scale airborne remote sensing data using optimized Li-YOLOv9. *IEEE J. Sel. Top. Appl. Earth Obs Remote Sens.* **18**, 2226–2238 (2024).
27. Tian, Z., Shen, C., Chen, H. & He, T. FCOS: A simple and strong anchor-free object detector. *TPAMI* **44**, 1922–1933. https://doi.org/10.1109/TPAMI.2020.3032166 (2022).
28. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L. C. MobileNetV2: inverted residuals and linear bottlenecks. *CVPR*, 4510–4520 (2018).
29. Xie, L. et al. SHISRCNet: Super-resolution and classification network for low-resolution breast cancer histopathology image. *MICCAI* **14224**, 15–25. https://doi.org/10.1007/978-3-031-43904-9_3 (2023).
30. Shrestha, N. L. Rice disease dataset. (2021). https://www.kaggle.com/datasets/nischallal/rice-disease-dataset
31. Howard, A. G. et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. Preprint at (2017). https://arxiv.org/abs/1704.04861
32. Howard, A. et al. Searching for MobileNetV3. *ICCV*, 1314–1324 (2019).
33. Zhang, X., Zhou, X., Lin, M. & Sun, J. ShuffleNet: an extremely efficient convolutional neural network for mobile devices. *CVPR*, 6848–6856 (2018).
34. Ma, N., Zhang, X., Zheng, H. T. & Sun, J. ShuffleNet V2: practical guidelines for efficient CNN architecture design. *ECCV*, 116–131 (2018).
35. Ultralytics. YOLOv3-tiny. (2024). https://github.com/ultralytics/ultralytics/tree/v8.1.47
36. Ultralytics. YOLOv5n. (2024). https://github.com/ultralytics/ultralytics/tree/v8.1.47
37. Ultralytics. YOLOv6n. (2024). https://github.com/ultralytics/ultralytics/tree/v8.1.47
38. Ultralytics. YOLOv9t. (2024). https://github.com/ultralytics/ultralytics/tree/v8.2.69
39. Ultralytics. YOLOv10n. (2024). https://github.com/ultralytics/ultralytics/tree/v8.2.69
40. Ultralytics YOLOv11n. (2024). https://github.com/ultralytics/ultralytics/tree/v8.3.0

Acknowledgements
The authors would like to thank the anonymous reviewers for their critical comments and suggestions for improving the manuscript.

Author contributions
Methodology, B.G., G.P. and S.L.; Dataset preparation, B.G. and S.L.; Experiments, B.G., S.L. and G.P.; Original draft, B.G. and S.L.; Review and editing, S.L. ,B.G. and L.W.; Visualization, G.P. and W.X.; Supervision, S.L. All authors have read and agreed to the published version of the manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Conflict of interest
The authors declare that they have no conflict of interest regarding the publication of this paper.

Additional information

Correspondence and requests for materials should be addressed to B.G. or S.L.