

Article

AdaptiveLeaf: Lightweight Multi-Scale Framework for Small-Target Detection of Maize Leaf Diseases

Yu Yang ¹, Bo Mao ^{2,*} and Lei Zhang ²

¹ School of Mechanical and Electronic Engineering, Nanjing Forestry University, Nanjing 210037, China; lilaili@njfu.edu.cn

² School of Computer and Artificial Intelligence, Nanjing University of Finance and Economics, Nanjing 210023, China; 9120031040@nufe.edu.cn

* Correspondence: bo.mao@nufe.edu.cn

Abstract

Early-stage maize leaf diseases and pests are difficult to detect due to their small size, low contrast, and complex backgrounds. AdaptiveLeaf is a lightweight multi-scale framework designed to improve the detection of such small targets. It integrates an Adaptive Kernel Lightweight Block (AKL-Block) for dynamic multi-scale feature extraction, a Feature Decomposition and Reconstruction (FDR) module to recover fine details such as lesion edges and spore clusters, and a Scale-Aware Gradient Boosting Loss (SAGB-Loss) to increase the training contribution of small targets. Experiments on 10,324 field-collected maize leaf images across eight disease and pest categories show that AdaptiveLeaf achieves a mean mAP@0.5 of 75.0% over three repeated runs and increases small-target AP from 28.1% to 32.8%, using only 2.52 M parameters and 5.3 GFLOPs. The framework balances accuracy and efficiency, making it suitable for real-time field inspection and precision agriculture.

Keywords: maize leaf diseases; small-target detection; lightweight framework; AKL-Block; FDR; SAGB-Loss

1. Introduction

Maize is one of the world's three major staple crops and is frequently threatened throughout its growth cycle by diseases and pests, including rust, gray leaf spot, maize borer, and fall armyworm. In the early stages, infections and infestations typically appear as small targets with low contrast, weak texture, uneven distribution, and complex backgrounds. This combination makes timely identification and precise control particularly challenging. Accurate early detection is crucial, as it helps safeguard yield, reduce pesticide use, minimize environmental impact, and improve overall productivity [1–5]. At the same time, accurate early detection places significant demands on real-time monitoring capabilities. This is particularly relevant for smart agricultural equipment, including unmanned sprayers, intelligent harvesters, and autonomous inspection robots. With the rapid advancement of smart agriculture, visual perception systems must not only process high-resolution field images efficiently but also deliver low-latency detection on edge devices. Meeting these requirements necessitates a careful balance between model accuracy and deployment efficiency, as reflected by recent real-time agricultural detectors and YOLO series evaluations [6,7].

Current lightweight object detection methods are largely represented by the YOLO series, which is known for fast inference and ease of deployment. Recent maize leaf



Academic Editor: Xianzhou Nie

Received: 24 May 2026

Revised: 14 June 2026

Accepted: 20 June 2026

Published: 29 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

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

disease detectors, including DSTANet, BCS_YOLO, and CKM-YOLO11, have reported promising performance in field environments [8–10]. In YOLO-style networks, compact feature aggregation modules such as C2f and C3k2 are commonly used to improve feature reuse and gradient flow. To further reduce computation, researchers have developed lightweight backbones such as MobileNetV4 [11], EdgeNeXt [12], and GhostNet [13], which use channel shuffling, depthwise separable convolution, or low-cost feature generation to reduce computational load. However, these networks generally rely on fixed convolution kernels. In field images, small lesions, spore clusters, and large leaf blight areas often appear together. Fixed kernels struggle to capture features at multiple scales, leading to weak representation of small targets and limiting the network's ability to adapt dynamically to multi-scale disease patterns.

Small-target feature sparsity remains a challenge, and existing studies have proposed various multi-scale feature fusion strategies, such as Dynamic Head [14], DetectoRS with recursive feature pyramids [15], and BiFPN [16]. These methods enhance small-target feature representation through top-down or bottom-up pathways. They still introduce computational overhead, making efficient deployment on edge devices difficult. In addition, small targets often produce sparse responses in deep feature maps, while shallow features carry limited semantic information. During training, large targets dominate the regression loss, limiting the gradient contribution from small targets.

In current object detection frameworks, the loss is typically composed of classification and regression branches. Classification-reweighting losses, including Varifocal Loss and Generalized Focal Loss, address sample-quality and class imbalance by reweighting training examples [17,18], but it mainly affects the classification branch and cannot handle scale imbalance in the regression branch. Improvements in regression loss, such as GIoU, DIoU, CIoU, and EIoU, introduce finer geometric constraints by incorporating terms such as center distance and aspect ratio, which improve localization accuracy [19,20]. However, the inherent difference in loss magnitude between small and large targets remains. Some studies amplify anchors smaller than a preset threshold with a fixed factor, while others assign different loss weights across feature levels, giving shallow layers that detect small targets higher emphasis. Most of these modifications adjust only a single dimension, either feature level or object area, and area-based weighting often uses hard thresholds, leading to discontinuous gradients near the boundary.

In agricultural scenarios, small lesions, insect eggs, or spore clusters often appear mixed with leaf veins, shadows, weeds, or soil background. Existing models are prone to missing these targets. Multi-scale feature fusion, adaptive kernel selection, and loss function optimization have not yet been integrated into a systematic solution, leaving room for improvement in detection accuracy and stability under real field conditions.

The main research gaps and the corresponding design choices of AdaptiveLeaf are summarized in Table 1.

AdaptiveLeaf is introduced as a lightweight enhancement framework for detecting small maize leaf disease and pest targets. The key contributions include the following:

1. The Adaptive Kernel Lightweight Block (AKL-Block) employs global average pooling and a lightweight gating mechanism to dynamically select the most suitable convolution kernel for each input feature map. Gumbel-Softmax ensures differentiability during training, while inference activates only the optimal kernel path. Such a design allows input-level receptive field adaptation and efficient multi-scale feature representation at a substantially reduced computational cost.
2. The Feature Decomposition and Reconstruction (FDR) module separates deep features into a structural branch and a detail branch. The detail branch undergoes lightweight reconstruction via sub-pixel convolution, enhancing fine cues such as lesion edges,

- insect contours, and spore textures. Meanwhile, bilinear interpolation preserves the global structure in the structural branch, effectively balancing detail recovery and computational efficiency.
3. The Scale-Aware Gradient Boosting Loss (SAGB-Loss) combines feature-level weighting with a continuous exponential area factor to smoothly amplify regression gradients for small targets, increasing their training contribution without negatively affecting medium or large targets.

Table 1. Research gaps in lightweight maize leaf disease and pest detection and the corresponding AdaptiveLeaf design responses.

Challenge	Typical Methods	Limitation in Maize-Field Images	AdaptiveLeaf Response
Fixed lightweight backbones	MobileNetV4 [11], EdgeNeXt [12], GhostNet [13]	A fixed receptive field is not flexible enough for tiny spore clusters, insect eggs, medium lesions, and large blighted areas.	AKL-Block adaptively selects 3×3 , 5×5 , or 7×7 kernels for each input, with only one path activated during inference.
Costly feature fusion	Dynamic Head [14], DetectoRS [15], BiFPN [16]	Extra fusion paths increase FLOPs, and simple interpolation cannot explicitly recover lesion edges, insect contours, or spore textures.	FDR separates structural and detail branches and reconstructs fine local cues only in the lightweight detail branch.
Scale-imbalanced training	Varifocal Loss [17], Generalized Focal Loss [18]	Classification reweighting reduces easy-sample dominance but does not strengthen small-box regression gradients.	SAGB-Loss combines feature-level weighting with a continuous area factor to emphasize small-target regression.
Regression-loss limitation	GIoU, DIoU, CIoU, EIoU [19,20]	Geometric losses improve localization but do not fully balance small and large targets; hard thresholds may introduce discontinuities.	SAGB-Loss applies smooth scale-aware weighting and avoids abrupt threshold changes.

Experiments were conducted on a dataset of 10,324 field-collected maize leaf images covering eight disease and pest categories. AdaptiveLeaf achieves a mean mAP@0.5 of 75.0% over three repeated runs, with 2.52 M parameters and 5.3 GFLOPs. Small-target AP_s rises from 28.1% to 32.8%, while total computation is reduced by 15.9% compared with YOLOv11n. On several mainstream edge devices, AdaptiveLeaf consistently achieves stable real-time inference, demonstrating its practical utility for field inspection and precision crop management.

2. Materials and Methods

2.1. Dataset and Experimental Setup

The dataset used in this study was primarily collected by our research group through field campaigns in major maize-producing regions from June 2023 to September 2025. The collection regions covered Jilin Province (approximately 43.58° N, 126.48° E), Heilongjiang Province (approximately 48.49° N, 128.13° E), Shandong Province (approximately 36.33° N, 118.75° E), and Henan Province (approximately 33.88° N, 113.50° E). The collection conditions covered sunny, cloudy, and partially rainy weather, with variations in lighting, leaf orientation, crop density, growth stage, and weed coverage to ensure data diversity. Image acquisition did not rely on a single fixed camera model because the images were collected across multiple field campaigns; instead, multiple smartphone RGB cameras and portable industrial RGB cameras were used, with typical sensor resolutions of 2 MP and

5 MP. The shooting distance, view angle, and illumination varied with field accessibility and leaf distribution. After collection, images that were severely blurred, abnormally exposed, or contained unidentifiable targets were discarded. The remaining samples were annotated with bounding boxes for lesions, insects, and other disease and pest targets using annotation tools. This process resulted in a dataset of 10,324 original field-collected maize leaf images covering eight disease and pest categories. Table 2 summarizes the number of images, target instances, and small-target proportions for each category, where small targets are defined as instances with bounding-box areas smaller than 32^2 pixel². As shown in Table 2, the dataset covers common maize leaf diseases and pests across a range of scales, with small-target instances accounting for a notably high proportion. For the experiments, the dataset was split using stratified random sampling according to disease/pest category and object-size distribution at a ratio of 6:2:2, containing 6194 training images, 2065 validation images, and 2065 test images, respectively.

Table 2. Sample distribution of eight disease and pest categories, including the object-size statistics used for evaluation.

Category	Images	Instances	Small Instances ($<32^2$ Pixel ²)	Medium Instances (32^2 – 96^2 Pixel ²)	Large Instances ($\geq 96^2$ Pixel ²)	Small Proportion (%)
Maize Rust	1503	3964	2761	880	323	69.6
Gray Leaf Spot	1247	2638	1316	905	417	49.9
Northern Maize Leaf Blight	1189	2147	758	734	655	35.3
Leaf Blight	1083	1841	551	612	678	29.9
Maize Borer	1164	2509	1623	650	236	64.7
Fall Armyworm Larvae	1287	2895	1738	801	356	60.0
Insect Eggs	1538	4621	4074	501	46	88.2
Spore Clusters	1313	3521	3125	356	40	88.8
Total	10,324	24,136	15,946	5439	2751	66.1

The experimental platform was configured with an AMD Ryzen 7 7800X3D CPU (Advanced Micro Devices, Inc., Austin, TX, USA), 32 GB of RAM, and an NVIDIA RTX 3080 GPU (10 GB VRAM) (NVIDIA Corporation, Santa Clara, CA, USA). The software environment included PyTorch 2.0.1 and CUDA 11.8. During training, the input image resolution was set to 640×640 pixels. An SGD optimizer was used with momentum 0.937 and weight decay 5×10^{-4} . The initial learning rate was 0.01, which decayed to 0.001 following a cosine annealing schedule. Training was conducted for 200 epochs with a batch size of 16. Mosaic data augmentation was applied during the first 150 epochs with a probability of 1.0. All 10,324 images reported in the dataset are original field images. Because the augmented samples were dynamically generated online for each mini-batch and could vary across epochs, no fixed offline augmented-image set was stored, and the number of augmented samples was not treated as an additional fixed dataset size. In the AKL-Block, the Gumbel-Softmax temperature coefficient τ was linearly annealed from 1.0 to 0.1 during the first 100 epochs and then fixed at 0.1.

Evaluation metrics include mAP@0.5 (mean average precision at an IoU threshold of 0.5), small-object AP_s (area $< 32^2$ pixels²), medium-object AP_m ($32^2 \leq \text{area} < 96^2$ pixels²), large-object AP_l (area $\geq 96^2$ pixels²), number of parameters (Params), computational complexity (GFLOPs), and inference speed (FPS).

2.2. Overall Architecture of AdaptiveLeaf

Figure 1 illustrates the overall architecture of the proposed AdaptiveLeaf, which introduces improvements to three key stages of YOLOv11n:

1. **Backbone:** The original C3k2 modules are replaced by AKL-Blocks. A kernel selector performs input-level adaptive receptive field assignment, providing multi-scale representation at the cost of single-kernel inference.
2. **Feature enhancement layer:** An FDR module is inserted after the P5 feature layer. It splits deep semantic features into a structural branch and a detail branch, then feeds the reconstructed detail responses back to P4. These are fused with the original P4 features in the neck and sent to subsequent networks. The reason for applying FDR only to the P5 feature layer is explained in detail in Section 2.4.
3. **Detection head and loss function:** The detection head keeps the decoupled head with DFL design from YOLOv11n. During training, the original regression loss is replaced with SAGB-Loss, which jointly introduces level-wise weights and an area enhancement factor in a two-dimensional manner, precisely amplifying the gradient signals for small targets.

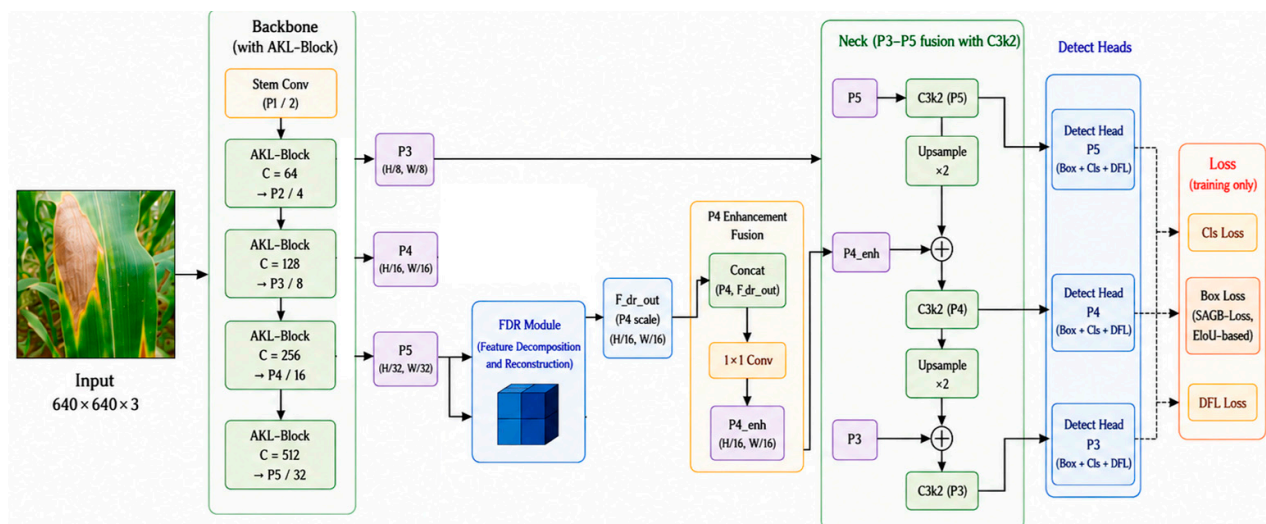


Figure 1. Overall architecture of AdaptiveLeaf. AKL-Block is embedded in the backbone for adaptive multi-scale feature extraction; FDR module reconstructs lost details from P5 to P4; SAGB-Loss supervises the detection heads during training.

AKL-Blocks in the backbone enable adaptive multi-scale feature extraction; the FDR module reconstructs details lost from P5 to P4; and SAGB-Loss supplies scale-aware supervision to the detection head during training.

2.3. AKL-Block: Adaptive Kernel Lightweight Block

The AKL-Block uses lightweight gating to pick the most suitable kernel, 3×3 , 5×5 , or 7×7 , for each input, and combines this with Ghost operations to roughly halve the FLOPs of the base convolution. Its structure is shown in Figure 2 and breaks down into four steps: Ghost channel expansion, kernel selection, single-kernel convolution, and channel recalibration with a residual connection.

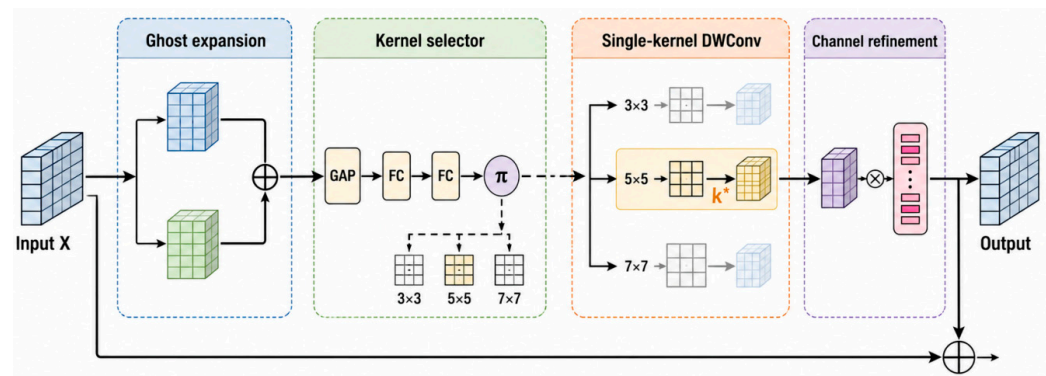


Figure 2. Structure of the proposed AKL-Block. Ghost expansion and adaptive kernel selection are combined to dynamically activate one depthwise convolution kernel (3×3 , 5×5 , or 7×7) for efficient multi-scale feature extraction.

The process works as follows.

1. Ghost channel expansion

Given an input $X \in \mathbb{R}^{C \times H \times W}$, a standard 1×1 convolution first squeezes out intrinsic features at half the original channels. The depthwise separable convolution then generates one ghost feature map from each intrinsic one, and the two groups are concatenated, keeping the total channel count unchanged [13]. This can be written as follows:

$$X_{\text{intrinsic}} = \text{Conv}_{1 \times 1}(X) \in \mathbb{R}^{C/2 \times H \times W} \quad (1)$$

$$X_{\text{ghost}} = \text{DWConv}_{3 \times 3}(X_{\text{intrinsic}}) \in \mathbb{R}^{C/2 \times H \times W} \quad (2)$$

$$X_{\text{expanded}} = \text{Concat}(X_{\text{intrinsic}}, X_{\text{ghost}}) \in \mathbb{R}^{C \times H \times W} \quad (3)$$

The depthwise separable convolution used here costs far less than a standard convolution, while the intrinsic–ghost concatenation preserves the full channel dimension.

2. Kernel selector

The selector predicts, for the whole feature map, how likely each of the three candidate kernels is to be chosen. It first applies global average pooling (GAP) to the expanded features, then pushes them through two fully connected layers that map to 3 output units, and finally applies Softmax to produce a normalized probability vector $\pi \in \mathbb{R}^3$. The selector's total parameter count is only about 0.005 M.

$$z = \text{GAP}(X_{\text{expanded}}) \quad (4)$$

$$\pi = \text{Softmax}(\text{FC}_2(\text{ReLU}(\text{FC}_1(z)))) \quad (5)$$

As illustrated in Figure 2, FC_1 compresses the C -dimensional features to $C/4$ to strip out redundancy, and FC_2 maps them to three values, one for each candidate kernel size. The three components correspond to the selection probabilities of the 3×3 , 5×5 , and 7×7 depthwise separable convolution kernels, respectively.

3. Differentiable kernel selection and inference

Kernel selection is inherently a discrete decision. Taking argmax over π would pick a single kernel for each input at inference, but argmax is non-differentiable and blocks gradient flow during training. To work around this, we adopt the Gumbel–Softmax reparameterization [21]:

$$g = \text{GumbelSoftmax}(\pi, \tau) \in [0, 1]^3 \quad (6)$$

where τ is the temperature. Early in training, τ starts at 1.0, making g act like a soft weight, e.g., [0.3, 0.4, 0.3]; the outputs of all three kernels are blended proportionally, allowing smooth gradient flow. As training proceeds, τ is gradually lowered to 0.1, pushing g toward a one-hot vector, e.g., [0.01, 0.98, 0.01], so the selector transitions smoothly from soft weighting to a hard, discrete choice. This annealing strategy keeps training differentiable while guaranteeing deterministic single-kernel selection at inference.

The feature map computed with the selected kernels is then:

$$F_{\text{selected}} = g_1 \cdot \text{DWConv}_{3 \times 3}(X_{\text{expanded}}) + g_2 \cdot \text{DWConv}_{5 \times 5}(X_{\text{expanded}}) + g_3 \cdot \text{DWConv}_{7 \times 7}(X_{\text{expanded}}) \quad (7)$$

Equation (7) describes the soft selection used during training, where all three candidate kernels participate in the forward pass to maintain gradient flow. At inference, a hard selection $k^* = \text{argmax}_i(\pi_i)$ is used, and only the winning path is activated. Accordingly, the GFLOPs and inference latency reported in this paper all refer to the single-kernel inference path; the extra branch overhead of training is not counted toward edge deployment cost. The FLOPs of the three depthwise separable convolutions are roughly $9C \times \text{HW}$ for 3×3 , $25C \times \text{HW}$ for 5×5 , and $49C \times \text{HW}$ for 7×7 . Under equal-probability selection, the average computation is about $27.7C \times \text{HW}$, 33.4% of the $83C \times \text{HW}$ required by three-way parallel fusion.

4. Channel recalibration and residual connection

A squeeze-and-excitation (SE) module performs adaptive channel-wise feature recalibration to strengthen cross-channel interactions after kernel selection [22]. The residual connection is added to maintain numerical stability during training.

$$F_{\text{out}} = X + \text{SE}(\text{Conv}_{1 \times 1}(F_{\text{selected}})) \quad (8)$$

2.4. FDR: Feature Decomposition and Reconstruction Module

In YOLOv11n, the P5 feature map undergoes five $2 \times$ downsampling steps, shrinking to $1/32$ of the input resolution. For targets smaller than 32^2 pixels², the response on P5 can collapse to a single point or disappear entirely. The usual solution is to bilinearly upsample P5 and fuse it with shallower features. However, bilinear interpolation is essentially a weighted average of neighboring pixels; it handles smooth structure well but struggles to recover the fine edges and textures that distinguish small objects from complex backgrounds.

An alternative is to upsample all P5 features with a learnable PixelShuffle operation [23], letting the network reconstruct lost details from the downsampled features. However, P5 typically has 256 or 512 channels, a full PixelShuffle would first expand the channels fourfold via 1×1 convolutions, substantially raising the parameter count and computational cost.

In maize leaf disease and pest detection, the information most easily lost during downsampling is local detail, lesion edges, insect contours, and spore textures. The global structural cues, like leaf shape, large lesion positions, and background layout, can be roughly recovered through bilinear upsampling. There is no need to apply learnable upsampling to every channel. It makes more sense to concentrate the limited computation on reconstructing the detail branch.

FDR splits P5 features into two branches with a 1×1 convolution: a structural branch that preserves global information and is upsampled by bilinear interpolation, and a detail branch that uses lightweight sub-pixel reconstruction to restore fine local responses.

In the detail branch, a 1×1 convolution expands to $4C_{\text{detail_out}}$ channels, after which PixelShuffle rearranges them into a detail feature map with $C_{\text{detail_out}} = C/8$ channels. The

learnable upsampling cost only applies to this branch. The structure of FDR is shown in Figure 3.

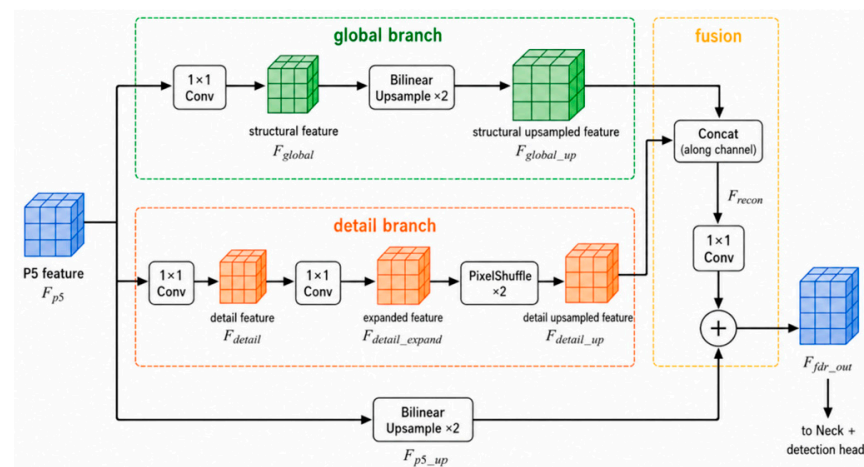


Figure 3. Structure of the proposed FDR module. Deep features are decomposed into a structural branch and a detail branch. The structural branch is upsampled by bilinear interpolation, while the detail branch uses lightweight PixelShuffle reconstruction to recover fine local responses.

As shown in Figure 3, the input is the P5 feature $F_{p5} \in \mathbb{R}^{C \times H \times W}$, where C is the number of channels (256 in YOLOv11n). The module proceeds as follows:

1. Splitting

A 1×1 convolution compresses the channels and separates the two branches:

$$F_{global} = \text{Conv}_{1 \times 1}(F_{p5}) \in \mathbb{R}^{C/4 \times H/32 \times W/32} \quad (9)$$

$$F_{detail} = \text{Conv}_{1 \times 1}(F_{p5}) \in \mathbb{R}^{3C/4 \times H/32 \times W/32} \quad (10)$$

2. Bilinear upsampling (structural branch)

The structural branch is simply upsampled by a factor of 2 using bilinear interpolation:

$$F_{global_up} = \text{BilinearUpsample}(F_{global}, \text{scale} = 2) \in \mathbb{R}^{C/4 \times H/16 \times W/16} \quad (11)$$

This step involves no learnable parameters and negligible computational cost, relying only on pixel indexing and weighted summation.

3. Sub-pixel detail reconstruction (detail branch)

The detail branch is designed to reconstruct fine local responses that may be weakened or lost after repeated downsampling. Let the detail feature map be denoted as F_{detail} . For an upscaling factor r , the PixelShuffle operation requires an input tensor with $r^2 C_d$ channels, where C_d is the number of output channels after upsampling. Therefore, a 1×1 convolution is first applied to expand the channel dimension:

$$F_{detail_expand} = \text{Conv}_{1 \times 1}(F_{detail}) \in \mathbb{R}^{r^2 C_o \times H \times W} \quad (12)$$

The PixelShuffle operator then rearranges the expanded channel responses into a higher-resolution spatial feature map:

$$F_{detail_up} = \text{PixelShuffle}_r(F_{detail_expand}) \in \mathbb{R}^{C_o \times rH \times rW} \quad (13)$$

The upscaling factor is set to $r = 2$. Accordingly, the 1×1 convolution generates four channel groups for each output channel. PixelShuffle redistributes these four groups

into a 2×2 neighborhood in the upsampled feature map. Unlike bilinear interpolation, which calculates new pixels through weighted averaging, PixelShuffle uses learnable channel responses to reconstruct spatial details. This allows the network to recover fine-grained cues such as lesion boundaries, insect contours, and spore cluster textures.

The proposed reconstruction mechanism remains lightweight because PixelShuffle is applied only to the detail branch rather than to the complete P5 feature map. The structural branch is still upsampled using parameter-free bilinear interpolation. Therefore, learnable reconstruction is concentrated on the feature components that are most relevant to small-target discrimination, while unnecessary computation on global structural information is avoided.

4. Fusion and output

The upsampled outputs from both branches are concatenated along the channel dimension:

$$F_{\text{recon}} = \text{Concat}(F_{\text{detail_up}}, F_{\text{global_up}}) \in \mathbb{R}^{(C/4+C/8) \times H/16 \times W/16} \quad (14)$$

The total channel count after concatenation is controlled by the structural-detail split ratio. A subsequent 1×1 convolution adjusts it to match the channel dimension of the P4 feature in the neck, preparing it for fusion. Meanwhile, a residual connection adds the bilinearly upsampled P5 feature to maintain an uninterrupted flow of base information:

$$F_{\text{fdr_out}} = \text{Conv}_{1 \times 1}(F_{\text{recon}}) + \text{BilinearUpsample}(F_{\text{p5}}) \quad (15)$$

This residual path effectively provides an identity mapping from P5 to P4 so that FDR only needs to learn the detail enhancement on top of bilinear upsampling. The output $F_{\text{fdr_out}}$ is then element-wise added to the corresponding P4 feature in the neck and passed to the subsequent C3k2 module and detection head.

Take a 256-channel P5 feature with $2 \times$ upsampling as an example to compare the computational cost. A full PixelShuffle approach would first expand from 256 to 1024 channels and then apply PixelShuffle to obtain a $256 \times H/16 \times W/16$ feature map. Its approximate complexity is:

$$\text{FLOPs}_{\text{full}} \approx C \times 4C \times HW/32^2 + (\text{subsequent convolution FLOPs}) \quad (16)$$

In contrast, FDR's cost comes from the structural branch, the detail branch, and the final 1×1 fusion layer. These components together are much lighter than full-channel sub-pixel upsampling.

2.5. SAGB-Loss: Scale-Aware Gradient Boosting Loss

Small targets suffer from double signal attenuation during training: their gradients are weakened both by the feature level on which they are detected and by their small area. Most existing methods tackle only one of these two dimensions. Level-wise weighting assigns different weights to different feature-map resolutions for coarse inter-layer adjustment, but within the same layer, large and small targets still share the same parameters. Area-based strategies multiply the loss of anchor boxes smaller than a preset threshold by a fixed factor, but such hard thresholds cause gradient discontinuity near the boundary and can make medium-target training unstable in later stages.

SAGB-Loss combines level-wise weights with a continuous area enhancement factor. The level weights boost shallow feature maps where small targets are detected. The exponential area factor smoothly amplifies the gradient as the target size decreases so that small targets receive steadily increasing emphasis without any sudden jumps.

For the i -th anchor on the feature map at level l , let A_k be the area of its matched ground-truth box. The area factor is defined as:

$$w_k^{\text{area}} = 1 + \alpha \cdot \exp(-A_k/A_0) \quad (17)$$

here $A_0 = 0.0025$ corresponds roughly to a small-target area threshold of 32^2 pixel^2 , and $\alpha = 2.0$ controls the maximum amplification. Under this setting, the loss for a very small target can be scaled up by about $3\times$. When the target area is much smaller than 32^2 pixel^2 , the exponential term approaches 1 and the regression loss is amplified nearly threefold; when the area far exceeds the threshold, the gain falls to nearly zero and no extra amplification is applied.

We choose EIoU as the base regression loss. EIoU models width and height errors separately and is more sensitive to deviations in small objects than CIoU:

$$\mathcal{L}_{\text{EIoU}} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}_{\text{gt}})}{c^2} + \frac{(w - w_{\text{gt}})^2}{c_w^2} + \frac{(h - h_{\text{gt}})^2}{c_h^2} \quad (18)$$

where $\rho(\cdot)$ is the Euclidean distance, $\mathbf{b}$ and $\mathbf{b}_{\text{gt}}$ are the centers of the predicted and ground-truth boxes, c is the diagonal length of the smallest enclosing box, and c_w and c_h are its width and height.

Combining the level-wise weight w_l^{layer} and the area factor, the total regression loss is:

$$L_{\text{reg}} = \frac{\sum_l \sum_k w_l^{\text{layer}} w_k^{\text{area}} L_{\text{EIoU}}}{\sum_l \sum_k w_l^{\text{layer}} w_k^{\text{area}}} \quad (19)$$

The level-wise weight decreases for higher feature levels and is defined according to the feature stride S_l :

$$w_l^{\text{layer}} = 1 + \beta \cdot \frac{S_{\text{max}}}{S_l} \quad (20)$$

where S_l is the stride of the feature map at level l , and S_{max} is the maximum stride in the network (32 in YOLOv11n). With $\beta = 4$, the weights become $w_{p3}^{\text{layer}} = 17$, $w_{p4}^{\text{layer}} = 9$, and $w_{p5}^{\text{layer}} = 5$.

The distribution focal loss (DFL) is kept unchanged. The overall training loss is a weighted sum of three terms:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{box}} \mathcal{L}_{\text{reg}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} + \lambda_{\text{dfl}} \mathcal{L}_{\text{DFL}} \quad (21)$$

with λ_{box} , λ_{cls} , and λ_{dfl} following the default settings of YOLOv11n to ensure a fair comparison with the baseline.

3. Results

3.1. Ablation Experiments

Table 3 summarizes the ablation results, with YOLOv11n serving as the baseline and each module added step by step. Replacing the baseline C3k2 module with the AKL-Block cuts GFLOPs from 6.3 to 5.0, while mAP rises from 72.2% to 73.5% and small-object AP_s gains 2.0 percentage points. The adaptive kernel selection effectively trims redundant multi-path computation without hurting accuracy.

Table 3. Ablation Study of Proposed Modules.

Scheme	AKL	FDR	SAGB	mAP@0.5 (%)	AP_s/%	AP_m/%	AP_l/%	Params/M	GFLOPs
YOLOv11n Baseline	—	—	—	72.20 ± 0.20	28.10 ± 0.50	55.80 ± 0.30	64.30 ± 0.20	2.58	6.3
Only AKL-Block	✓	—	—	73.50 ± 0.20	30.10 ± 0.40	56.20 ± 0.20	64.10 ± 0.30	2.50	5.0
Only FDR	—	✓	—	72.80 ± 0.20	28.70 ± 0.40	55.90 ± 0.30	64.00 ± 0.20	2.60	6.6
Only SAGB-Loss	—	—	✓	72.70 ± 0.20	28.50 ± 0.40	55.90 ± 0.30	64.10 ± 0.20	2.58	6.3
AKL-Block + FDR	✓	✓	—	74.30 ± 0.30	31.40 ± 0.50	56.10 ± 0.30	63.90 ± 0.30	2.52	5.3
AKL-Block + SAGB-Loss	✓	—	✓	73.70 ± 0.20	30.30 ± 0.40	56.30 ± 0.20	64.10 ± 0.30	2.50	5.0
FDR + SAGB-Loss	—	✓	✓	73.50 ± 0.20	29.20 ± 0.50	56.00 ± 0.30	63.80 ± 0.20	2.60	6.6
AdaptiveLeaf	✓	✓	✓	75.0 ± 0.32	32.8 ± 0.71	56.4 ± 0.40	64.0 ± 0.15	2.52	5.3

Adding the FDR module pushes small-object AP_s further from 30.1% to 31.4%, GFLOPs only inch up from 5.0 to 5.3, and large-object AP_l edges down from 64.1% to 63.9%.

When SAGB-Loss is also included, the full AdaptiveLeaf achieves a mean mAP@0.5 of 75.0% over three repeated runs and 32.8% AP_s, a further gain of 1.4 percentage points over the previous step. AP_l stays at 64.0%, down only 0.3 points from the baseline. AP_s rises from 28.1% to 32.8%, while large-object accuracy shows almost no real loss.

Reproducibility was checked with three random seeds (42, 123, and 2024), and the results are listed in Table 4. Reporting variation across repeated runs is important because measured performance can change with data order, initialization, and other stochastic factors. The average mAP@0.5 is 75.0% with a standard deviation of 0.32, and the standard deviation of AP_s is 0.71. The slightly larger AP_s variation is consistent with the stochastic behavior of Gumbel-Softmax sampling and dynamic loss weighting.

Table 4. Repeatability Results of the Model with Three Different Random Seeds.

Random Seeds	mAP@0.5	AP_s	AP_m	AP_l
42	75.1	33.4	56.6	64.0
123	74.6	32.0	55.9	63.9
2024	75.2	32.9	56.6	64.2
Mean ± Std	75.0 ± 0.32	32.8 ± 0.71	56.4 ± 0.40	64.0 ± 0.15

Table 5 breaks down the individual and combined gains of each module on AP_s and calculates the degree of synergy, where Overlap = joint gain – sum of individual gains.

Table 5. Gain comparison of individual and combined modules based on AP_s (%).

Module Combination	AP_s	Gain over Baseline	Sum of Individual Gains	Overlap Degree
Baseline	28.1	—	—	—
Only AKL	30.1	+2.0	—	—
Only FDR	28.7	+0.6	—	—
Only SAGB	28.5	+0.4	—	—
AKL + FDR	31.4	+3.3	+2.6	+0.7 (Positive)
AKL + SAGB	30.3	+2.2	+2.4	−0.2 (Mild Negative)
FDR + SAGB	29.2	+1.1	+1.0	+0.1 (Neutral)
AKL + FDR + SAGB	32.8	+4.7	+3.0	+1.7

As shown in Table 5, the rich feature details preserved by AKL provide a stronger foundation for FDR's reconstruction, yielding a clear positive synergy of +0.7 between the

two. In contrast, AKL and SAGB show a slight negative synergy of -0.2 , while the gains from FDR and SAGB combine in a largely additive manner. When all three modules are integrated, the overall synergy reaches $+1.7$.

3.2. Model Comparison

We keep the training configuration unchanged (without introducing FDR or SAGB-Loss, backbone replacement only) and compare AKL-Block with a C3k2 variant using a fixed 7×7 convolution, C3k2-Inception, and C3k2-SK modules. The results are summarized in Table 6.

Table 6. Comparison Among Different Multi-Scale Backbone Modules.

Scheme	mAP@0.5	AP _s	AP _m	AP _l	Params/M	GFLOPs
YOLOv11n (Baseline)	72.2	28.1	55.8	64.3	2.58	6.3
C3k2-Fixed 7×7	72.1	27.6	55.6	64.5	2.55	6.8
C3k2-Inception	73.7	30.4	56.3	64.8	2.68	7.9
C3k2-SK	73.6	30.2	56.2	64.6	2.66	7.3
AKL-Block (Ours)	73.5	30.1	56.2	64.1	2.50	5.0

The fixed 7×7 convolution model yields an AP_s of 27.6%, lower than the baseline, which indicates that a large kernel overly smooths out the fine details of small lesions. Inception and SKNet reach small-object AP_s values of 30.4% and 30.2%, slightly outperforming AKL-Block at 30.1%, reflecting the more thorough multi-scale representation offered by parallel multi-branch structures. However, their GFLOPs stand at 7.9 and 7.3, while AKL-Block achieves 73.5% mAP with only 5.0 GFLOPs. This result should be interpreted as an efficiency–accuracy trade-off: AKL-Block sacrifices only 0.3 and 0.1 percentage points of AP_s compared with Inception and SK, respectively, while substantially reducing computational complexity.

To further analyze the limitation of image-level kernel selection, a scale-mixed subset was constructed from the test set. An image was included in this subset if it contained at least one small target with a bounding-box area smaller than 32^2 pixel² and at least one large target with a bounding-box area greater than or equal to 96^2 pixel² after resizing the input image to 640×640 pixels. According to this criterion, the scale-mixed subset contained 318 images, accounting for 15.4% of the test set. The subset included images containing both tiny targets, such as spore clusters and insect eggs, and relatively large targets, such as extensive blighted areas. The results are presented in Table 7.

Table 7. Comparison between AKL-Block and Inception on Scale-Mixed Samples.

Scheme	AP _s	AP _l	mAP@0.5
C3k2-Inception	30.2	63.8	73.0
AKL-Block (Ours)	30.1	64.1	73.1

Table 7 reveals a limitation of the AKL-Block. Its small-target AP_s of 30.1% falls slightly behind Inception’s 30.2%. This is because image-level kernel selection cannot simultaneously serve tiny and large objects appearing in the same image. On the other hand, AKL-Block achieves a large-target AP_l of 64.1%, marginally higher than Inception’s 63.8%, and its overall mAP is 73.1%, which is essentially on par with Inception.

Using the AKL-Block as a common backbone, we compare FDR with several mainstream $P5 \rightarrow P4$ feature reconstruction schemes. The results are summarized in Table 8.

Table 8. Comparison of FDR with other feature enhancement methods.

Scheme	mAP@0.5	AP_s	AP_m	AP_l	Params/M	GFLOPs	Extra Params
No P5 enhancement	73.5	30.1	56.2	64.1	2.50	5.0	0
Bilinear-only	73.6	30.3	56.1	64.3	2.50	5.0	0
Deconv	74.1	31.0	56.3	64.2	2.68	5.6	0.18 M
Full-PixelShuffle	74.8	32.2	56.5	64.5	2.62	5.8	0.12 M
FDR (Ours)	74.3	31.4	56.1	63.9	2.52	5.3	0.02 M

Full PixelShuffle delivers the highest AP_s at 32.2%, surpassing FDR by 0.8 percentage points. This advantage stems from its indiscriminate learnable upsampling applied to all channels, which preserves more feature information. However, that extra capability comes at a clear cost: additional parameters of 0.12 M and a 16% increase in GFLOPs. By contrast, FDR adds only 0.02 M parameters and 6% more GFLOPs while raising AP_s from 30.1% to 31.4%, with AP_l edging down slightly from 64.1% to 63.9%. These results show that FDR strikes a good accuracy-efficiency balance under lightweight constraints.

Images containing ultra-small targets (smaller than 64 pixels) were also selected for evaluation, and the recall comparison is reported in Table 9.

Table 9. Recall comparison between FDR and Full-PixelShuffle on extreme small objects.

Scheme	Recall@0.5	Recall@0.3
No P5 enhancement	31.2%	48.5%
Full-PixelShuffle	34.8%	52.1%
FDR (Ours)	33.0%	50.4%

FDR achieves a Recall@0.5 of 33.0%, slightly below the 34.8% of Full PixelShuffle. This gap stems largely from FDR's dependence on P5 residual features; when the original P5 signal is already very faint, the reconstruction can only add limited gains. Notably, relaxing the IoU threshold to 0.3 narrows the gap, with recalls of 50.4% and 52.1% respectively.

The remaining gap is expected to be concentrated in categories dominated by extremely small responses, especially spore clusters and insect eggs. A per-category breakdown is provided in Table 10 to verify this interpretation.

Table 10. Per-category Recall@0.5 comparison for extremely small objects.

Category	Full-PixelShuffle Recall@0.5 (%)	FDR Recall@0.5 (%)	Gap (Percentage Points)
Maize Rust	35.2	33.9	1.3
Gray Leaf Spot	31.8	30.9	0.9
Northern Maize Leaf Blight	29.6	28.7	0.9
Leaf Blight	28.4	27.6	0.8
Maize Borer	36.1	34.7	1.4
Fall Armyworm Larvae	34.7	33.2	1.5
Insect Eggs	39.5	36.2	3.3
Spore Clusters	43.1	38.8	4.3
Mean	34.8	33.0	1.8

As shown in Table 10, Full-PixelShuffle achieves a slightly higher mean Recall@0.5 than FDR for extremely small objects, with an average gap of 1.8 percentage points. The gap is relatively small for lesion-related categories such as Gray Leaf Spot, Northern Maize Leaf Blight, and Leaf Blight, where the difference remains below 1.0 percentage point. In contrast, the performance gap is more evident for Insect Eggs and Spore Clusters, reaching

3.3 and 4.3 percentage points, respectively. This indicates that the limitation of FDR mainly occurs in categories dominated by extremely small and weak-textured responses. Since FDR reconstructs details from the P5 feature map with a lightweight detail branch, its performance may be constrained when the original P5 response has already become very weak after repeated downsampling. Nevertheless, FDR maintains a competitive mean Recall@0.5 while requiring fewer additional parameters and lower computational cost than Full-PixelShuffle, which is more consistent with the lightweight deployment goal of AdaptiveLeaf.

Using the network framework combining AKL-Block and FDR, we compare SAGB-Loss against several mainstream weighting approaches. Table 11 presents the results.

Table 11. Comparison of SAGB-Loss with other loss weighting strategies.

Loss Strategy	mAP@0.5	AP_s	AP_m	AP_l
CIoU (Baseline)	74.3 ± 0.20	31.4 ± 0.17	56.3 ± 0.17	64.0 ± 0.17
EIoU	74.5 ± 0.17	31.7 ± 0.17	56.4 ± 0.17	64.1 ± 0.17
EIoU + Layer-wise weighting	74.7 ± 0.17	32.2 ± 0.26	56.4 ± 0.26	64.0 ± 0.17
EIoU + Hard area threshold weighting	74.6 ± 0.30	32.8 ± 0.35	55.8 ± 0.20	64.0 ± 0.17
SAGB-Loss (Ours)	74.9 ± 0.26	32.6 ± 0.36	56.3 ± 0.21	64.2 ± 0.17

All loss strategies were rerun with three random seeds (42, 123, and 2024). Switching from CIoU to EIoU raises AP_s by 0.3 percentage points. Adding level-wise weighting further increases AP_s to 32.2%. Hard area-threshold weighting reaches the highest AP_s value, 32.8%, but AP_m decreases from 56.3% to 55.8%. SAGB-Loss reaches 32.6% AP_s while maintaining 56.3% AP_m. The numerical advantage over layer-wise weighting is modest: 0.2 points in mAP@0.5 and 0.4 points in AP_s. It should therefore be interpreted cautiously until repeated-run significance tests are completed.

Recall that α controls the maximum gradient amplification for small targets. Table 12 reports a sensitivity analysis by sweeping α across the range 0–4.0.

Table 12. Sensitivity analysis of enhancement strength α .

α	mAP@0.5	AP_s	AP_m	AP_l
0.0	74.3	31.4	56.3	64.0
1.0	74.6	32.0	56.3	64.1
2.0	74.9	32.6	56.3	64.2
3.0	74.7	32.8	56.0	64.0
4.0	74.4	32.9	55.7	63.8

As shown in Table 12, AP_s rises monotonically with α , going from 31.4% to 32.9%. Once α exceeds 2.0, AP_m slips from 56.3% to 55.7% at $\alpha = 4.0$, while AP_l also edges down from 64.2% to 63.8%. The best mAP, 74.9%, is reached at $\alpha = 2.0$. Based on this, we set $\alpha = 2.0$, which substantially strengthens small-target detection while largely preserving the accuracy on medium and large targets.

Table 13 summarizes the performance of AdaptiveLeaf and representative mainstream detectors, including RTMDet and DAMO-YOLO variants [24,25]. Additional maize- and plant-disease detection studies are included to provide a broader context for comparison [26–43]. To ensure a controlled and reproducible comparison, all reimplemented models were trained and tested on the same dataset under a unified experimental protocol. However, no exhaustive architecture-specific hyperparameter search was conducted for the comparison models. In particular, Transformer-based or hybrid detectors,

such as RT-DETR and EfficientViT, may require different warmup strategies, learning-rate schedules, training epochs, and augmentation settings to reach their optimal performance. Therefore, the results in Table 13 should be interpreted as performance under the same unified protocol rather than the best achievable performance of each architecture.

Table 13. Performance Comparison with State-of-the-Art Models.

Method	mAP@0.5	AP_s	AP_m	AP_l	Params/M	GFLOPs	FPS
Faster R-CNN (ResNet50)	66.9	19.3	48.2	60.4	137.1	370.2	21
SSD (VGG16)	72.3	24.5	53.1	63.8	26.29	62.74	41
RT-DETR (R18)	75.8	31.2	57.0	65.8	20.0	71.1	24
YOLOv5n	62.1	22.3	48.5	56.2	1.9	4.5	141
YOLOv8n	71.1	26.5	54.2	63.1	3.2	8.7	124
YOLOv11n (Baseline)	72.2	28.1	55.8	64.3	2.58	6.3	136
YOLOv11s	74.5	30.1	56.6	65.2	9.42	21.9	86
YOLOv11m	75.3	31.0	57.0	65.8	20.0	67.7	51
YOLOv11L	76.7	32.3	58.2	66.5	25.29	86.6	43
EfficientViT	73.3	29.5	55.9	64.8	3.56	6.9	128
Gold-YOLO	74.8	31.5	56.9	65.6	21.5	46.0	38
YOLO-MS (XS)	73.8	30.2	56.1	64.9	4.5	8.7	118
YOLO-MS (S)	75.6	32.0	57.2	66.0	8.1	15.4	82
DAMO-YOLO (Tiny)	73.1	29.0	55.5	64.5	6.8	14.1	95
DAMO-YOLO (Small)	74.3	30.6	56.3	65.2	11.2	23.4	68
RTMDet-Tiny	73.5	29.8	55.7	64.7	4.8	8.1	142
RTMDet-Small	74.9	31.2	56.8	65.3	8.89	14.8	118
MobileViT-YOLO	72.8	28.6	54.8	64.0	8.6	10.2	105
EMO-YOLO	73.4	29.7	55.6	64.5	5.1	8.5	122
FasterNet-YOLO	72.5	28.9	55.0	64.2	3.96	7.8	136
AdaptiveLeaf (Ours)	75.0	32.8	56.4	64.0	2.52	5.3	135

AdaptiveLeaf achieves a mean mAP@0.5 of 75.0%, slightly behind YOLOv11m (75.3%) and YOLOv11L (76.7%). Yet with only 2.52 M parameters and 5.3 GFLOPs, it uses about one-eighth the parameters and under one-tenth the computation of YOLOv11m. Small-object AP_s reaches 32.8%, topping all compared models and even outperforming the larger YOLOv11L (32.3%). Large-object AP_l sits at 64.0%, lower than the others, which is consistent with our design goal of channeling more capacity toward small targets. Relative to the baseline YOLOv11n, AP_s climbs 4.7 percentage points, mAP rises 2.8 points, and GFLOPs drop by 15.9%.

3.3. Visual Analysis

Images containing lesions at different scales are taken from the test set, and the kernel selection probabilities during inference are visualized. The 3×3 kernel is marked in blue, 5×5 in green, and 7×7 in orange. Results are shown in Figure 4.

For images dominated by small rust lesions, the selection probability of the 3×3 kernel stays close to 1.0. On images with medium-sized gray leaf spot lesions, the choice is fairly evenly split between 3×3 and 5×5 kernels. When large-area leaf blight is present, the 7×7 kernel's selection probability clearly rises. This pattern matches the expected behavior of kernel selection.

Figure 5 uses heatmaps to show how FDR sharpens small-target feature responses, taking a cluster of tiny maize rust spores as the example. The three columns display: Figure 5a the activation heatmap of the neck P4 features without FDR; Figure 5b the activation heatmap of the FDR output features; Figure 5c the difference between the two, i.e., the additional detail residuals introduced by FDR.

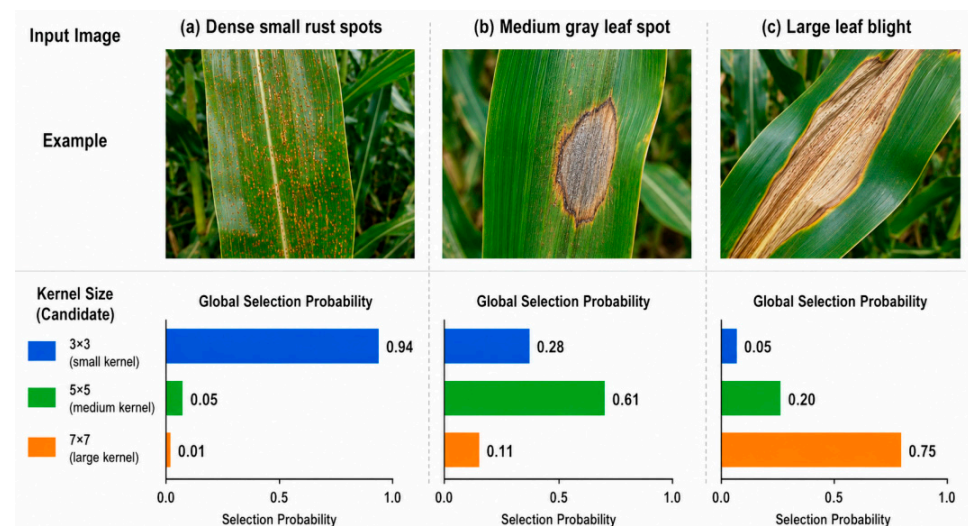


Figure 4. Visualization of kernel selection behavior of AKL-Block. The three input cases are: (a) small rust-like lesions and spore clusters, (b) medium gray leaf spot lesions, and (c) large blighted areas. The horizontal axis denotes the three input images, and the vertical axis represents the selection probabilities of the three candidate kernels.

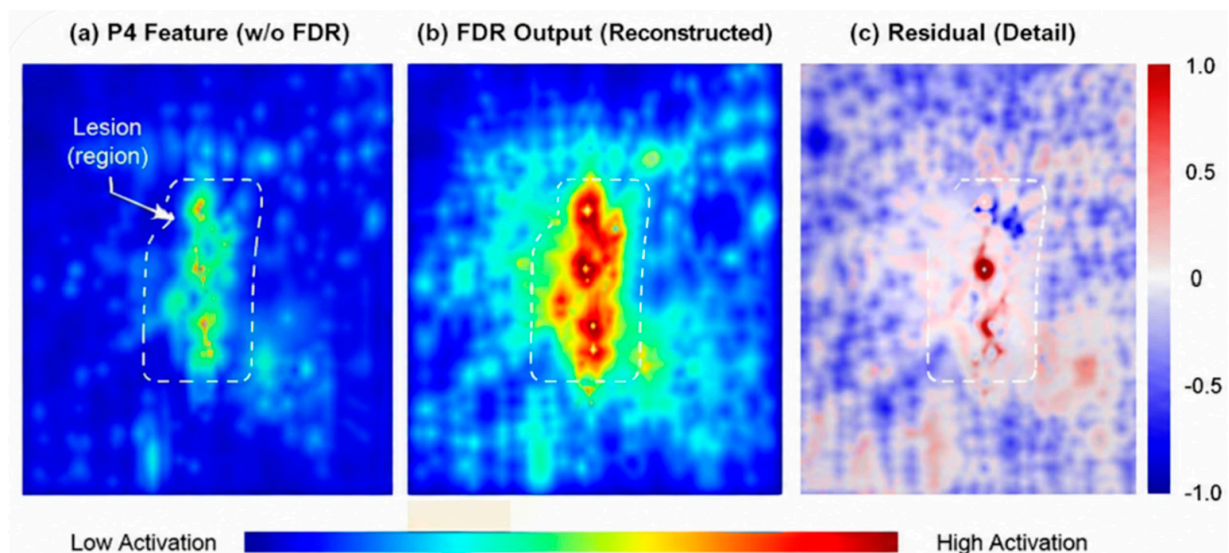


Figure 5. Feature enhancement results of the FDR module. (a) Original P4 feature map without FDR, showing weak and diffuse activation around the small lesion region. (b) Reconstructed feature output of the FDR module, where lesion boundaries and texture responses are significantly enhanced. (c) Residual detail response map, highlighting boundary- and texture-related details. Compared with the original P4 features, FDR reconstructs stronger high-frequency responses around small lesion boundaries and texture regions.

As shown in Figure 5b is that FDR noticeably strengthens the detail responses along the boundaries and in the textured areas of tiny lesions, compared with the original P4 features:

1. In the unenhanced P4 heatmap, the activation signals for the tiny lesions are both weak and scattered. There is no clear peak, and they blend into the surrounding leaf texture with very little contrast, which means they could easily be drowned out in later processing.
2. The FDR-reconstructed heatmap tells a different story: the activation intensity in those same lesion areas increases considerably. The response peaks become concentrated and stand out, making the lesion characteristics clearly visible.

3. The residual heatmap captures the extra detail that FDR brings in, with crisp, high-activation signals precisely at the lesion locations.

3.4. Edge Deployment Performance

To evaluate the deployment feasibility of AdaptiveLeaf, inference experiments were conducted on two representative edge-computing platforms: an RK3588-based edge device (Rockchip Electronics Co., Ltd., Fuzhou, Fujian, China) and an NVIDIA Jetson Orin NX 16 GB module (NVIDIA Corporation, Santa Clara, CA, USA). The main specifications are summarized in Table 14 according to the vendor datasheets [44,45].

Table 14. Main specifications of the edge-computing platforms used for deployment evaluation.

Platform	Hardware Model	CPU	GPU/AI Accelerator	Memory	Tested Precision
RK3588-based edge device	Orange Pi 5	4 × Cortex-A76 + 4 × Cortex-A55 8-core Arm	ARM Mali-G610 MC4 GPU, triple-core NPU 1024-core NVIDIA	16 GB LPDDR4×	INT8
NVIDIA Jetson Orin NX	Jetson Orin NX	Cortex-A78AE v8.2 64-bit CPU	Ampere GPU with 32 Tensor Cores	16 GB LPDDR5	FP16/INT8

For RK3588 deployment, the model was converted and executed using RKNN-Toolkit2 v2.0.0 and the RKNPU2 runtime under Ubuntu 20.04. For Jetson Orin NX deployment, inference was performed under JetPack 5.1.2, CUDA 11.4, cuDNN 8.6, and TensorRT 8.5.2. FP16 inference was enabled directly by TensorRT, while INT8 inference was obtained using post-training quantization (PTQ). The results are reported in Table 15.

Table 15. Inference performance on edge devices.

Platform	Model	Precision	Latency (ms)	Power (W)	Memory (MB)	mAP@0.5 (%)
RK3588 CPU	YOLOv11n	INT8	12.7	0.9	92	69.8
RK3588 CPU	AdaptiveLeaf	INT8	12.1	0.9	88	71.5
RK3588 NPU	AdaptiveLeaf	INT8	7.5	1.2	152	71.5
Jetson Orin	YOLOv11n	INT8	5.3	5.1	124	69.7
Jetson Orin	AdaptiveLeaf	FP16	6.8	6.2	198	74.7
Jetson Orin	AdaptiveLeaf	INT8	4.8	4.9	122	71.5

On the RK3588 NPU, AdaptiveLeaf needs only 7.5 ms per inference, which corresponds to about 133 FPS. With INT8 quantization on the Jetson Orin NX, latency drops to 4.8 ms, reaching roughly 208 FPS. The quantized mAP is 71.5%, clearly outperforming the quantized YOLOv11n baselines under comparable INT8 settings, which reach 69.8%.

3.5. Qualitative Detection Performance

Figure 6 presents detection results on typical field scenes, including dense fine rust spots, fall armyworm larvae, and leaf blight against complex backgrounds. The model holds up well in recall for dense rust spots and small lesions near leaf edges. However, when spores are smaller than roughly 10×10 pixels and overlap with bright vein textures, a small number of missed detections still happen.

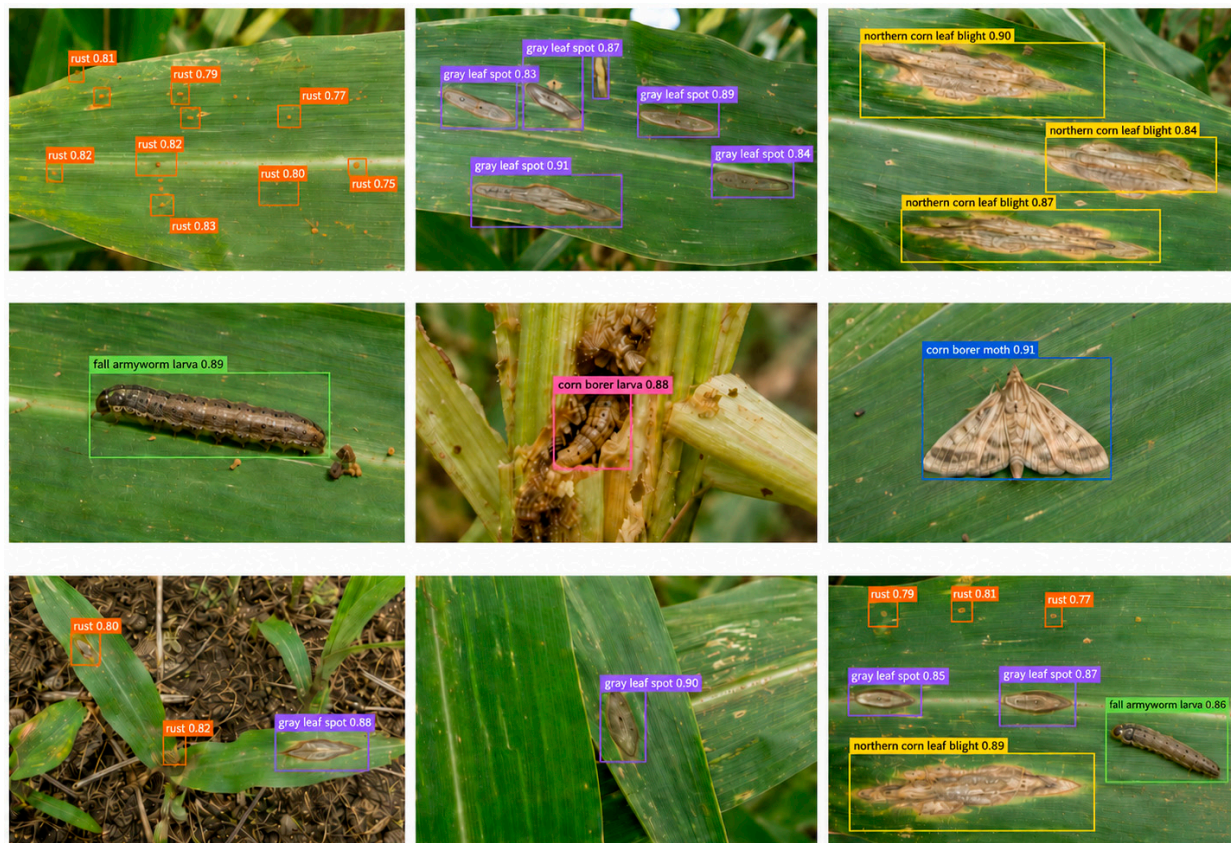


Figure 6. Qualitative detection results of AdaptiveLeaf in complex field scenarios. The proposed method achieves robust detection under dense small targets, occlusion, illumination variation, and cluttered agricultural backgrounds.

3.6. Uncertainty and Error Analysis

To comprehensively evaluate the reliability of AdaptiveLeaf under real field conditions, uncertainty and error analysis was conducted from two perspectives: stochastic training variation and field-image acquisition variation.

As reported in Table 4, three independent runs were conducted using different random seeds. The standard deviation was 0.32 for mAP@0.5 and 0.71 for AP_s. The larger variation in AP_s is expected because tiny-object localization is more sensitive to minor changes in parameter initialization, sample order, and data augmentation. In addition, the Gumbel-Softmax sampling process in AKL-Block and the dynamic weighting strategy in SAGB-Loss may introduce slight fluctuations during training. Nevertheless, the overall standard deviations remain within an acceptable range, indicating that the performance of AdaptiveLeaf is relatively stable across repeated experiments.

The field-image acquisition introduces additional measurement uncertainty. Strong sunlight, backlighting, weak illumination, leaf moisture, motion blur, shooting distance, focus variation, leaf overlap, and vein-like textures can all affect the apparent contrast, texture, and bounding-box quality of disease and pest targets. For example, strong sunlight and backlighting may cause local overexposure or pronounced shadows on maize leaves, reducing the contrast between lesion textures and the leaf background. Leaf moisture and water droplets may introduce reflective regions that resemble small lesions or spore clusters. Partial occlusion can destroy the complete contours of lesions, insects, and spore clusters, increasing localization difficulty.

To further examine robustness under complex field conditions, five representative and non-exhaustive image subsets were selected and annotated from the test set, including

strong illumination, low illumination, backlighting, partial occlusion, and different shooting angles. Because a single field image may simultaneously contain multiple characteristics, such as occlusion, backlighting, and tilted shooting, a small degree of overlap was allowed among the subsets. Therefore, the sum of the numbers of images in the five condition-specific subsets does not equal the total number of images in the valid robustness-analysis set. After excluding images that could not be reliably assigned to the predefined field-condition categories, 2049 valid test images containing 4827 target instances were retained for the robustness analysis. All models were evaluated on the original test images using the same training parameters and confidence threshold. The representative model weights trained with seed = 42 were used for testing. The results are shown in Table 16.

Table 16. Robustness evaluation under different complex field conditions.

Testing Condition	Number of Images	Number of Instances	YOLOv11n mAP@0.5 (%)	AdaptiveLeaf mAP@0.5 (%)	YOLOv11n AP_s (%)	AdaptiveLeaf AP_s (%)
Valid robustness-analysis set	2049	4827	72.2	75.0	28.1	32.8
Strong illumination	312	708	70.8	74.1	26.9	31.7
Low illumination	286	655	69.6	73.0	25.8	30.8
Backlighting	241	536	67.9	71.7	24.6	29.8
Partial occlusion	354	812	68.7	72.6	25.1	30.4
Different shooting angles	328	741	71.0	74.4	27.2	32.0

As shown in Table 16, complex field conditions reduce detection accuracy to varying degrees. The performance degradation is particularly evident under backlighting and partial occlusion. Under backlighting, the mAP@0.5 of YOLOv11n decreases to 67.9%, while AdaptiveLeaf maintains 71.7%. Under partial occlusion, AdaptiveLeaf improves mAP@0.5 from 68.7% to 72.6% and AP_s from 25.1% to 30.4%. Under backlighting, AP_s increases from 24.6% to 29.8%. These results demonstrate that AdaptiveLeaf consistently outperforms YOLOv11n under different complex field conditions.

The improved robustness can be attributed to the complementary effects of the three proposed components. AKL-Block adaptively adjusts the effective receptive field according to the input feature distribution, which helps the model handle scale and appearance variations caused by different imaging conditions. FDR compensates for partially weakened edge and texture information under complex illumination and occlusion by reconstructing fine local responses. SAGB-Loss increases the contribution of small-target samples during bounding-box regression training, improving the detection of tiny lesions, insect eggs, and spore clusters.

Despite these improvements, several challenging cases remain. Missed detections are more likely to occur when backlighting causes severe overexposure in the target region, when the occlusion area is large, or when extremely small spore clusters smaller than approximately 10×10 pixels strongly overlap with bright leaf-vein textures. These observations are consistent with the qualitative detection analysis. Future work will further improve model stability under extreme field conditions by incorporating richer illumination augmentation, occlusion-simulation augmentation, and cross-regional dataset expansion.

4. Discussion

The experimental results show that AdaptiveLeaf manages to improve small-target detection for maize leaf diseases and pests while staying lightweight. The main reason is that the improvements don't simply add more network depth or detection heads. Instead, they tackle three specific bottlenecks in small-target detection: input scale variation, deep feature detail loss, and how training gradients are distributed. Compared with scaling up

the model, this kind of targeted optimization suits field-edge scenarios much better, where computing resources are tight.

4.1. Module Effectiveness Analysis

The AKL-Block contributes most obviously to cutting computational cost. Traditional multi-scale structures often rely on parallel branches to capture different receptive fields, but they have to run all branches during inference, which can burden deployment. The AKL-Block uses input-level kernel selection so that only one candidate kernel is active per inference. This keeps multi-scale adaptability while reducing redundant computation. In the ablation study, adding the AKL-Block dropped GFLOPs from 6.3 to 5.0 and raised AP_s by 2.0 points. This suggests the input-adaptive kernel selection works well for targets that vary considerably in scale, like tiny rust spots, insect eggs, and larger blight areas.

The FDR module mainly helps by compensating for lost details in deep features. maize leaf disease and pest targets often mix with veins, shadows, and cluttered backgrounds. After repeated downsampling, the responses of tiny lesions and spore clusters tend to fade. FDR doesn't reconstruct all P5 channels at high cost; instead, it splits structural and detail information and only applies lightweight reconstruction to the detail branch. In this way, it sharpens edges, textures, and local contours with only a small computational increase. In experiments, adding FDR lifted AP_s from 30.1% to 31.4% while GFLOPs went from 5.0 to just 5.3, which suggests the design strikes a reasonable balance between accuracy gain and computational expense.

The effectiveness of SAGB-Loss comes from redistributing regression gradients toward small targets. Small targets cover only a few pixels, so a slight box shift causes a noticeable IoU drop, yet standard regression losses are easily dominated by medium and large objects. SAGB-Loss adjusts the regression loss jointly through level-wise weights and an area enhancement factor, giving small-target samples more thorough optimization during training. Compared with hard-threshold weighting, the continuous exponential form avoids sudden weight changes near the scale boundary. As a result, AP_s improves without noticeably sacrificing AP_m or AP_l. The full model reaches 32.8% AP_s, a gain of 4.7 points over the YOLOv11n baseline, indicating that this loss design provides real benefits for small-target detection.

Looking at how the modules interact, AKL-Block and FDR complement each other well. The AKL-Block improves scale adaptation in the backbone and provides a better feature foundation for later detail recovery; FDR then compensates for the local texture loss caused by downsampling. Together they enhance small-object feature quality. SAGB-Loss strengthens small-target emphasis at the supervision level, complementing the two structural modules. Compared with parallel multi-branch structures like Inception and SKNet, AKL-Block yields slightly lower small-object accuracy but costs far less computation. Compared with Full-PixelShuffle reconstruction, FDR recovers extremely tiny targets a bit less effectively but adds far fewer parameters and operations. This suggests that AdaptiveLeaf favors a practical trade-off under lightweight deployment conditions rather than chasing the highest possible accuracy.

The edge deployment results further illustrate the model's practical value. On devices like RK3588 and Jetson Orin, AdaptiveLeaf after INT8 quantization still delivers high inference speed, meeting the real-time requirements of field scouting, mobile identification, and precision control. The qualitative detection results confirm that the model reliably detects dense rust spots, fall armyworm larvae, and leaf blight areas in complex backgrounds, indicating strong potential for edge applications.

4.2. Limitations

Despite its accuracy-efficiency balance, AdaptiveLeaf still has several limitations. First, AKL-Block selects one candidate kernel for the complete input feature map. When tiny spore clusters and large blighted regions appear together, one receptive field cannot represent both scale extremes equally well. The complete-test comparison in Table 6 indicates a missed AP_s gain of 0.3 percentage points relative to C3k2-Inception. On the scale-mixed subset in Table 7, the missed AP_s gain is 0.1 points. A region-level selector could make local routing decisions for windows or candidate regions, but the additional routing overhead must be evaluated carefully.

A two-stage extension is also feasible. A first stage could locate suspicious leaf regions or coarse disease areas, and a second stage could refine tiny spore clusters and other difficult targets inside selected crops. Such a pipeline may improve extreme-scale detection, but it would also increase latency, memory use, and system complexity. The current study retains a single-stage design because its main goal is real-time edge deployment. A conditional refinement stage for uncertain regions is a suitable direction for future work.

FDR relies on the residual responses preserved in deep P5 features for detail recovery. When targets are extremely small, textures closely resemble bright vein areas, or the deep feature signal has already been severely weakened, lightweight reconstruction can only recover limited information. The qualitative results also show that when spore clusters are smaller than roughly 10×10 pixels and overlap with vein textures, the model still misses a few detections.

The area enhancement factor and level weights in SAGB-Loss depend on the target scale distribution of the dataset. If applied to different crops, different shooting distances, or different input resolutions, the relevant hyperparameters may need readjustment. Excessively high small-target weights can further improve AP_s but may also reduce the localization stability of medium and large targets.

Although the current dataset covers eight categories of maize leaf diseases and pests, the samples mainly come from specific collection regions and equipment conditions. Real field environments involve additional factors such as different varieties, growth stages, lighting conditions, mixed pest and disease occurrences, and seasonal changes. The model's long-term generalization ability still needs more validation. Moreover, this work focuses on static image detection and has not yet exploited the continuous frame information available in field inspection videos.

5. Conclusions

This paper proposes AdaptiveLeaf, a lightweight enhancement framework for detecting small targets in maize leaf diseases and pests. Building on YOLOv11n as the baseline, it improves scale adaptation through the AKL-Block, deep detail recovery through the FDR module, and small-target regression supervision through SAGB-Loss, thereby boosting detection performance on tiny lesions, insects, eggs, and spore clusters.

On a dataset of 10,324 field-collected maize leaf images with eight disease and pest categories, AdaptiveLeaf achieves a mean mAP@0.5 of 75.0% over three repeated runs, with 2.52 M parameters and 5.3 GFLOPs. Compared with YOLOv11n, AP_s rises from 28.1% to 32.8%, a gain of 4.7 percentage points, while computation drops by 15.9%. These results demonstrate that the improved framework substantially strengthens small-target detection without enlarging the model and is feasible for deployment on agricultural edge devices. Future work will focus on region-level dynamic kernel selection, cross-region sample expansion, and continuous field validation to further enhance generalization and long-term stability.

Author Contributions: Conceptualization, Y.Y. and B.M.; methodology, Y.Y.; software, Y.Y.; validation, Y.Y., B.M. and L.Z.; formal analysis, Y.Y.; investigation, Y.Y.; resources, L.Z.; data curation, Y.Y.; writing—original draft preparation, Y.Y.; writing—review and editing, B.M. and L.Z.; visualization, Y.Y.; supervision, L.Z.; project administration, B.M. and L.Z.; funding acquisition, B.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key Research and Development Program of China, grant number 2024YFF1105504 and the Integrated Pilot Special Project for Agricultural Machinery R&D, Manufacturing, Promotion and Application, grant number JSYTH01.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The dataset used in this study has been deposited in Kaggle and is available at: <https://www.kaggle.com/datasets/cdcd001/maize-leaf-diseases> (accessed on 20 May 2026).

Acknowledgments: The authors sincerely appreciate Xuesong Jiang for providing crucial project funding, experimental platforms, and valuable academic guidance throughout this research. His strong support facilitated the completion of supplementary experiments, data analysis optimization, and manuscript improvement during the major revision stage, which played an essential role in the successful completion of this study. During the preparation of this manuscript, the authors used ChatGPT (GPT-5, OpenAI) to improve the language. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the content of the publication.

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

References

1. Timilsina, S.; Sharma, S.; Kondo, S. Advancements in maize leaf disease detection, segmentation and classification: A review. *Biosyst. Eng.* **2025**, *255*, 104162. [[CrossRef](#)]
2. Gülmez, B. Advancements in maize disease detection: A comprehensive review of convolutional neural networks. *Comput. Biol. Med.* **2024**, *183*, 109222. [[CrossRef](#)] [[PubMed](#)]
3. Liu, M.; Gao, J. SLD-YOLO11: A topology-reconstructed lightweight detector for fine-grained maize–weed discrimination in complex field environments. *Agronomy* **2026**, *16*, 328. [[CrossRef](#)]
4. Murugan, T.; Noor Mohamed Badusha, N.A. Research advances in maize crop disease detection using machine learning and deep learning approaches. *Computers* **2026**, *15*, 99. [[CrossRef](#)]
5. Li, C.; Li, Y.; Yang, W.; Chen, D. ACF-YOLO: Feature enhancement and multi-scale alignment for sustainable crop small object detection. *Sustainability* **2026**, *18*, 4168. [[CrossRef](#)]
6. Chen, Y.; Yuan, X.; Wang, J.; Wu, R.; Li, X.; Hou, Q.; Cheng, M.-M. YOLO-MS: Rethinking Multi-Scale Representation Learning for Real-Time Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2025**, *47*, 4240–4252. [[CrossRef](#)] [[PubMed](#)]
7. Sapkota, R.; Meng, Z.; Churuvija, M.; Du, X.; Ma, Z.; Karkee, M. Comprehensive Performance Evaluation of YOLOv12, YOLO11, YOLOv10, YOLOv9 and YOLOv8 on Detecting and Counting Fruitlet in Complex Orchard Environments. *Agric. Commun.* **2026**, *4*, 100125. [[CrossRef](#)]
8. Gao, X.; He, L.; Liu, Y.; Wu, J.; Cao, Y.; Dong, S.; Jia, Y. DSTANet: A lightweight and high-precision network for fine-grained and early identification of maize leaf diseases in field environments. *Sensors* **2025**, *25*, 4954. [[CrossRef](#)] [[PubMed](#)]
9. Hao, S.; Gao, E.; Ji, Z.; Ganchev, I. BCS_YOLO: Research on corn leaf disease and pest detection based on YOLOv11n. *Appl. Sci.* **2025**, *15*, 8231. [[CrossRef](#)]
10. Zhu, H.; Xiao, F.; Xiang, J.; Guo, J.; Mu, H. CKM-YOLO11: A Lightweight Maize Foliar Disease Detection Model for Complex Natural Field Environments. *Sensors* **2026**, *26*, 2969. [[CrossRef](#)] [[PubMed](#)]
11. Qin, D.; Lechner, C.; Delakis, M.; Fornoni, M.; Luo, S.; Yang, F.; Wang, W.; Banbury, C.; Ye, C.; Akin, B.; et al. MobileNetV4: Universal Models for the Mobile Ecosystem. In *Computer Vision—ECCV 2024*; Springer: Cham, Switzerland, 2024; pp. 78–96. [[CrossRef](#)]
12. Maaz, M.; Shaker, A.; Cholakkal, H.; Khan, S.; Zamir, S.W.; Anwer, R.M.; Khan, F.S. EdgeNeXt: Efficiently Amalgamated CNN-Transformer Architecture for Mobile Vision Applications. In *Computer Vision—ECCV 2022 Workshops*; Springer: Cham, Switzerland, 2023; pp. 3–20. [[CrossRef](#)]
13. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. GhostNet: More Features from Cheap Operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 13–19 June 2020; pp. 1580–1589. [[CrossRef](#)]

14. Dai, X.; Chen, Y.; Xiao, B.; Chen, D.; Liu, M.; Yuan, L.; Zhang, L. Dynamic Head: Unifying Object Detection Heads with Attentions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 7369–7378. [\[CrossRef\]](#)
15. Qiao, S.; Chen, L.-C.; Yuille, A. DetectorS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 10208–10219. [\[CrossRef\]](#)
16. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and Efficient Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790. [\[CrossRef\]](#)
17. Zhang, H.; Wang, Y.; Dayoub, F.; Sunderhauf, N. VarifocalNet: An IoU-Aware Dense Object Detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 8510–8519. [\[CrossRef\]](#)
18. Li, X.; Wang, W.; Wu, L.; Chen, S.; Hu, X.; Li, J.; Tang, J.; Yang, J. Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21002–21012.
19. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 12993–13000. [\[CrossRef\]](#)
20. Zhang, Y.-F.; Ren, W.; Zhang, Z.; Jia, Z.; Wang, L.; Tan, T. Focal and Efficient IoU Loss for Accurate Bounding Box Regression. *Neurocomputing* **2022**, *506*, 146–157. [\[CrossRef\]](#)
21. Herrmann, C.; Bowen, R.S.; Zabih, R. Channel Selection Using Gumbel Softmax. In *Computer Vision—ECCV 2020*; Springer: Cham, Switzerland, 2020; pp. 241–257. [\[CrossRef\]](#)
22. Hu, J.; Shen, L.; Albanie, S.; Sun, G.; Wu, E. Squeeze-and-Excitation Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 2011–2023. [\[CrossRef\]](#) [\[PubMed\]](#)
23. Shao, G.; Sun, Q.; Gao, Y.; Zhu, Q.; Gao, F.; Zhang, J. Sub-Pixel Convolutional Neural Network for Image Super-Resolution Reconstruction. *Electronics* **2023**, *12*, 3572. [\[CrossRef\]](#)
24. Lyu, C.; Zhang, W.; Huang, H.; Zhou, Y.; Wang, Y.; Liu, Y.; Zhang, S.; Chen, K. RTMDet: An Empirical Study of Designing Real-Time Object Detectors. *arXiv* **2022**, arXiv:2212.07784. [\[CrossRef\]](#)
25. Xu, X.; Jiang, Y.; Chen, W.; Huang, Y.; Zhang, Y.; Sun, X. DAMO-YOLO: A Report on Real-Time Object Detection Design. *arXiv* **2022**, arXiv:2211.15444. [\[CrossRef\]](#)
26. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. YOLOv10: Real-Time End-to-End Object Detection. In Proceedings of the Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Vancouver, BC, Canada, 10–15 December 2024. [\[CrossRef\]](#)
27. Li, R.; Li, Y.; Qin, W.; Abbas, A.; Li, S.; Ji, R.; Wu, Y.; He, Y.; Yang, J. Lightweight Network for Corn Leaf Disease Identification Based on Improved YOLO v8s. *Agriculture* **2024**, *14*, 220. [\[CrossRef\]](#)
28. Zhu, T.; Yan, F.; Lv, X.; Zhao, H.; Wang, Z.; Dong, K.; Fu, Z.; Jia, R.; Lv, C. A Deep Learning Model for Accurate Maize Disease Detection Based on State-Space Attention and Feature Fusion. *Plants* **2024**, *13*, 3151. [\[CrossRef\]](#) [\[PubMed\]](#)
29. Kalunga, P.K.; Kunda, D.K.; Zimba, A. ZamYOLO-maize: A YOLOv8n-based deep learning framework for automated detection and classification of maize leaf diseases. *Front. Artif. Intell.* **2026**, *9*, 1764283. [\[CrossRef\]](#)
30. Alpsalaz, F.; Özüpak, Y.; Aslan, E.; Uzel, H. Classification of maize leaf diseases with deep learning: Performance evaluation of the proposed model and use of explicable artificial intelligence. *Chemom. Intell. Lab. Syst.* **2025**, *262*, 105412. [\[CrossRef\]](#)
31. Chitrainingrum, N.; Banowati, L.; Herdiana, D.; Mulyati, B.; Sakti, I.; Fudholi, A.; Saputra, H.; Farishi, S.; Muchtar, K.; Andria, A. Comparison study of corn leaf disease detection using YOLO-v5 and YOLO-v8 variants. *J. Eng. Technol. Sci.* **2024**, *56*, 62–71. [\[CrossRef\]](#)
32. Ren, Y.; Xu, Y.; Tian, H.; Zhang, Q.; Yan, M.; Zhu, R.; Xu, D.; Cheng, Q.; Wei, Q.; Song, S. Detection of maize pathogenic fungal spores based on deep learning. *Agriculture* **2025**, *15*, 1689. [\[CrossRef\]](#)
33. Askale, G.T.; Yibel, A.B.; Taye, B.M.; Wubneh, G.D. Mobile based deep CNN model for maize leaf disease detection and classification. *Plant Methods* **2025**, *21*, 72. [\[CrossRef\]](#) [\[PubMed\]](#)
34. Song, Y.; Zhang, H.; Li, J.; Ye, R.; Zhou, X.; Dong, B.; Fan, D.; Li, L. High-Accuracy Maize Disease Detection Based on Attention Generative Adversarial Network and Few-Shot Learning. *Plants* **2023**, *12*, 3105. [\[CrossRef\]](#) [\[PubMed\]](#)
35. Mall, A.; Kabra, S.; Lhila, A.; Ajmera, P. AMAizeD: An End to End Pipeline for Automatic Maize Disease Detection. *arXiv* **2023**, arXiv:2308.03766. [\[CrossRef\]](#)
36. Mustofa, S.; Hasan Munna, M.M.; Emon, Y.R.; Rabbany, G.; Ahad, M.T. A Comprehensive Review on Plant Leaf Disease Detection Using Deep Learning. *arXiv* **2023**, arXiv:2308.14087. [\[CrossRef\]](#)
37. Abade, A.S.; Ferreira, P.A.; Vidal, F.B. Plant Diseases Recognition on Images Using Convolutional Neural Networks: A Systematic Review. *Comput. Electron. Agric.* **2021**, *185*, 106125. [\[CrossRef\]](#)

38. Kanakala, S.; Ningappa, S. Detection and classification of diseases in multi-crop leaves using LSTM and CNN models. *J. Innov. Image Process.* **2025**, *7*, 161–181. [[CrossRef](#)]
39. Richey, B.; Majumder, S.; Shirvaikar, M.; Kehtarnavaz, N. Real-time detection of maize crop disease via a deep learning-based smartphone app. In *Real-Time Image Processing and Deep Learning 2020*; Kehtarnavaz, N., Carlsohn, M.F., Eds.; SPIE: Bellingham, WA, USA, 2020; Volume 11401, p. 114010A. [[CrossRef](#)]
40. Fraiwan, M.; Faouri, E.; Khasawneh, N. Classification of Corn Diseases from Leaf Images Using Deep Transfer Learning. *Plants* **2022**, *11*, 2668. [[CrossRef](#)] [[PubMed](#)]
41. Meng, Y.; Zhan, J.; Li, K.; Yan, F.; Zhang, L. A rapid and precise algorithm for maize leaf disease detection based on YOLO MSM. *Sci. Rep.* **2025**, *15*, 6016. [[CrossRef](#)] [[PubMed](#)]
42. Albahli, S.; Nawaz, M.; Javed, A.; Irtaza, A. Efficient attention-based CNN network for multi-class maize crop disease classification. *Front. Plant Sci.* **2022**, *13*, 1003152. [[CrossRef](#)] [[PubMed](#)]
43. Nan, F.; Song, Y.; Yu, X.; Nie, C.; Liu, Y.; Bai, Y.; Zou, D.; Wang, C.; Yin, D.; Yang, W.; et al. A novel method for maize leaf disease classification using the RGB-D post-segmentation image data. *Front. Plant Sci.* **2023**, *14*, 1268015. [[CrossRef](#)] [[PubMed](#)]
44. Rockchip Electronics Co., Ltd. RK3588 Brief Datasheet. 2025. Available online: <https://www.rock-chips.com/uploads/pdf/2022.8.26/192/RK3588/Brief/Datasheet.pdf> (accessed on 13 June 2026).
45. NVIDIA Corporation. NVIDIA Jetson Orin NX Series Modules Data Sheet, DS-10712-001_v1.7. 2026. Available online: <https://developer.nvidia.com/downloads/jetson-orin-nx-module-series-data-sheet> (accessed on 13 June 2026).

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