horticulturae

MDPI

Article

YOLOv11n-KL: A Lightweight Tomato Pest and Disease Detection Model for Edge Devices

Shibo Peng [1,†], Xiao Chen [2,†], Yirui Jiang [3], Zhiqi Jia [1], Zilong Shang [4], Lei Shi [4], Wenkai Yan [1,*] and Luming Yang [1,*]

[1] College of Horticulture, Henan Agricultural University, Zhengzhou 450002, China; p2440316215@outlook.com (S.P.); jiazhiqi2015@163.com (Z.J.)
[2] Henan Provincial Government Affairs Big Data Center, Zhengzhou 450002, China; 17603719825@163.com
[3] School of Artificial Intelligence and Big Data, Henan University of Technology, Zhengzhou 450002, China; jiangyirui@haut.edu.cn
[4] College of Information and Management Science, Henan Agricultural University, Zhengzhou 450002, China; augustus@stu.henau.edu.cn (Z.S.); shilei@henau.edu.cn (L.S.)
* Correspondence: wenkaiyan@henau.edu.cn (W.Y.); lumingyang@henau.edu.cn (L.Y.)
† These authors contributed equally to this work.

Abstract

Frequent occurrences of pests and diseases in tomatoes severely restrict yield and quality improvements. Traditional detection methods are labor-intensive and prone to errors, while advancements in deep learning provide a promising solution for rapid and accurate identification. However, existing deep learning-based models often face high computational complexity and a large number of parameters, which hinder their deployment on resource-constrained edge devices. To overcome this limitation, we propose a novel lightweight detection model named YOLOv11n-KL based on the YOLOv11n framework. In this model, the feature extraction capability for small targets and the overall computational efficiency are enhanced through the integration of the Conv_KW and C3k2_KW modules, both of which incorporate the KernelWarehouse (KW) algorithm, and the Detect_LSCD detection head is employed to enable parameter sharing and adaptive multi-scale feature calibration. The results indicate that YOLOv11n-KL achieves superior performance in tomato pest and disease detection, balancing lightweight design and detection accuracy. The model achieves an mAP@0.5 of 92.5% with only 3.0 GFLOPs and 5.2 M parameters, reducing computational cost by 52.4% and improving mAP@0.5 by 0.9% over YOLOv11n. With its low complexity and high precision, YOLOv11n-KL is well-suited for resource-constrained applications. The proposed YOLOv11n-KL model offers an effective solution for detecting tomato pests and diseases, serving as a useful reference for agricultural automation.

Keywords: tomato pests and diseases; YOLOv11n; object detection; lightweight model

Check for updates

Academic Editor: Araceli Peña

Received: 21 November 2025
Revised: 19 December 2025
Accepted: 22 December 2025
Published: 30 December 2025

Copyright: © 2025 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) license.

1. Introduction

Tomato (*Solanum lycopersicum* L.) is an economically significant crop, rich in vitamins and also consumed as a fruit. As one of the most widely cultivated vegetables globally, tomatoes exhibit high yield, broad adaptability, and substantial nutritional value. However, like other crops, tomatoes are affected by various diseases and pests during their growth cycle. Tomato diseases include viral, nematode, physiological, bacterial, and fungal diseases. Major tomato pests include leaf miners, greenhouse whiteflies, alfalfa noctuid moths, tobacco caterpillars, and cotton bollworms, among others [1]. The occurrence of these

diseases and pests significantly impacts tomato production, causing estimated annual yield losses of 10–20% [2,3]. Therefore, early detection and prevention are crucial. Traditional methods for detecting diseases and pests rely entirely on the grower's experience or expert guidance. These methods are slow, inefficient, costly, highly subjective, and often inaccurate and untimely [4]. With the rapid development of artificial intelligence technology, novel approaches have emerged for crop disease and pest identification. Among these, efficient image recognition technologies have shown great potential by enhancing detection efficiency, reducing costs, and improving accuracy. In particular, deep learning has become a key research focus worldwide, as it enables automatic feature extraction and robust recognition of specific targets in complex environments [5–9].

Recently, deep learning-based object detection algorithms have shown great potential for identifying crop diseases and pests. For example, Chen et al. integrated channel attention mechanisms and lightweight up-sampling operators to significantly improve multi-disease recognition in complex environments, achieving an mAP@0.5 of 92.1% while reducing model parameters and computational cost by 23.9% and 31.2%, respectively [10]. Wen et al. developed the Pest-YOLO model, which combined multi-scale feature fusion with attention mechanisms to improve detection and counting accuracy for dense and tiny pests [11]. Mo et al. introduced YOLONDD with a novel NMIoU loss function, improving multi-scale detection and robustness to size variations [12]. Likewise, Cai et al. combined a lightweight multi-scale attention module (MSA) and an EfficientViT backbone with an EIOU loss, achieving higher accuracy for small-object detection [13]. Fang et al. developed the Pest-ConFormer hybrid model, which integrates traditional image analysis with advanced global modeling to improve detection accuracy [14]. Sun et al. proposed the E-TomatoDet network, which enhances tomato disease recognition under challenging conditions by integrating global disease distribution patterns with local lesion textures [15]. Moreover, other studies have explored advanced loss functions, multi-scale attention modules, novel backbone networks, and specialized detection heads to enhance accuracy for small and multi-scale targets, often achieving substantial improvements in recognition performance [16–18].

Despite these advancements, current models still face notable limitations when implemented in real-world agricultural scenarios. Most existing studies emphasize laboratory experiments and prioritize detection accuracy through network innovations, often overlooking computational efficiency and deployment feasibility [19]. Even state-of-the-art models, including the latest YOLO variants, require substantial computational resources and memory, creating obstacles for real-time inference on edge devices such as the Jetson Nano and Raspberry Pi. Consequently, these models fail to fully satisfy the practical requirements of low-power, real-time monitoring in agriculture.

To overcome these challenges, the development of lightweight models has become a critical research direction. Techniques such as depthwise separable convolution [20], channel pruning [21], and knowledge distillation [22] are widely adopted to reduce model size and computational cost while preserving generalization capability. For example, DP-Net and MobileNetV3 integrated with coordinate attention achieved comparable accuracy while substantially reducing parameters and improving inference speed on embedded devices [23]. Similarly, Jiang et al. incorporated coordinate attention into MobileNetV3, resulting in a 7.5% increase in inference speed and improved recognition accuracy on the PlantVillage dataset [24]. Lightweight YOLO variants aim to balance accuracy and efficiency [25–27]; however, their adaptability to complex field environments remains limited.

Building on these insights, this study introduces YOLOv11n-KL, a lightweight detection model specifically designed for tomato pest and disease identification in resource-constrained edge environments. YOLOv11n-KL incorporates three major innovations: (1)

the Conv_KW module, which utilizes a cross-layer shared KernelWarehouse (KW) [28] combined with an attention mechanism to dynamically generate convolutional weights for small targets, thereby enhancing feature extraction while reducing computational complexity; (2) the C3k2_KW module, which extends the dynamic kernel concept to multi-branch feature fusion, efficiently reuses kernels to reduce computational complexity while maintaining sufficient receptive field coverage; (3) the Detect_LSCD head, a lightweight shared convolution detection (LSCD) module featuring adaptive multi-scale calibration, which reduces redundancy while ensuring accurate detection of both small and large disease regions. Collectively, these designs enable YOLOv11n-KL to achieve an optimal balance among accuracy, efficiency, and generalization, offering a practical solution for real-time edge deployment in intelligent agriculture.

2. Materials and Methods

2.1. Tomato Pest and Disease Dataset

To construct a representative dataset for tomato pest and disease detection, this study integrated image resources from multiple publicly available sources, including AI Challenger 2018 and the Agricultural Pest and Disease Image Library of the Chinese Academy of Sciences. Following a standardized workflow encompassing data cleaning, quality control, and expert annotation, redundant, blurred, and low-resolution images were removed. All retained images were manually verified by experienced agricultural experts to ensure the accuracy and consistency of annotations. After preprocessing, a total of 10,429 high-quality images ($640 \times 640 \times 3$ resolution) were preserved to establish the final dataset.

The dataset comprises 13 categories of tomato pests and diseases, covering both fruit- and leaf-related symptoms. Specifically, the fruit-related categories consist of healthy fruit, *Sunscald*, *Blossom End Rot*, *Fruit Cracking*, *Spotted Wilt Virus* (*Fruit*), and *Bacterial Spot* (*Fruit*). The leaf-related categories include *Mosaic Virus*, *Late Blight*, *Early Blight*, *Leaf miner Damage*, *Leaf Mold*, *Septoria Leaf Spot*, *Spider Mite Damage*, *Yellow Leaf Curl Virus*, and healthy leaf. This diversity enables the model to maintain strong generalization ability and robustness when applied to complex agricultural scenarios. To facilitate model training and evaluation, the dataset was randomly partitioned into training, validation, and test subsets at an 8:1:1 ratio (Table 1). This partitioning strategy ensures an adequate number of samples for model optimization while minimizing potential bias caused by data imbalance.

Table 1. Number of samples of each pest and disease type.

Category	Quantity	Training Set	Validation Set	Test Set
Sunscald Fruit	1482	1186	148	148
Blossom End Rot Fruit	1255	1004	126	126
Tomato Spotted Wilt Virus Fruit	1023	818	102	102
Fruit Cracking	931	745	93	93
Bacterial Spot of Fruit	528	422	53	53
Healthy Fruit	1083	866	108	108
Tomato Mosaic Virus Leaf	699	559	70	70
Late Blight of Leaf	603	482	60	60
Early Blight of Leaf	600	480	60	60
Leaf miner Damage	589	471	59	59
Leaf Mold	475	380	48	48
Septoria Leaf Spot	418	334	42	42
Spider Mite Damage	303	242	30	30
Yellow Leaf Curl Virus	286	229	29	29
Healthy Leaf	154	123	15	15
Total	10,429	8343	1043	1043

2.2. Improvement Strategies

2.2.1. YOLOv11n-KL Model

YOLOv11 is a state-of-the-art algorithm for object detection, offering faster detection speed, higher accuracy, and greater computational efficiency than its predecessors. The YOLOv11 series includes five network models: YOLOv11n, YOLOv11s, YOLOv11m, YOLOv11l, and YOLOv11x. Among them, YOLOv11n serves as a lightweight variant specifically optimized for real-time detection on resource-constrained embedded platforms. Building upon the core design principles of previous YOLO models, YOLOv11n incorporates an enhanced C3k2 module within its backbone to facilitate deeper feature extraction and more efficient gradient propagation while maintaining a lightweight structure. This enhancement significantly improves the model's capability to recognize and capture object features. Furthermore, the integration of the Spatial Pyramid Pooling Fast (SPPF) and Cross-layer Spatial Attention (C2PSA) modules enables more efficient contextual information aggregation and strengthens the fusion between semantic and spatial representations. The neck employs a hybrid Feature Pyramid Network and Path Aggregation Network (FPN + PAN) architecture, constructing multi-scale feature pathways that enhance detection robustness across varying object sizes. The head network comprises three decoupled detection heads, each responsible for predicting classification scores and bounding box regression coordinates. Leveraging task-alignment mechanisms, the head network enhances both classification and localization accuracy by jointly optimizing classification scores and Intersection over Union (IoU), thereby suppressing low-quality prediction boxes.

To further improve detection accuracy while reducing computational complexity and parameter count, this study presents an enhanced version of the YOLOv11n model, referred to as YOLOv11n-KL. The overall architecture of the proposed model is illustrated in Figure 1. First, the C3k2_KW module replaces the original C3k2 module in the backbone of YOLOv11n. This modification effectively reduces computational cost while preserving robust feature extraction capability, allowing faster inference without compromising detection accuracy. In addition, the Conv_KW module replaces the standard convolutional block. Through optimized kernel design, this module enhances the representational capacity for small-target features, thereby improving the model's ability to accurately identify subtle pest and disease symptoms on tomato leaves and fruits. Finally, the Detect_LSCD detection head is employed to replace the original one. This module enables parameter sharing and adaptive multi-scale feature calibration, which not only reduces computational complexity but also enhances localization and classification precision. Owing to these architectural refinements, the proposed YOLOv11n-KL model achieves an optimal balance between lightweight design and high detection performance. It demonstrates improved adaptability to resource-constrained environments while ensuring reliable and accurate detection of tomato pests and diseases, thus meeting the practical requirements of large-scale agricultural monitoring and intelligent pest management.

2.2.2. Conv_KW Module

Conv_KW is a convolutional module built upon a dynamic KernelWarehouse (KW) mechanism. It adopts an attention-driven dynamic weight generation strategy that substantially reduces computational complexity while improving both feature representation and adaptability to varying inputs. The core concept reconstructs static convolutional weights into a set of dynamically combinable Kernel Cells, enabling adaptive generation of convolutional weights according to input features through a cross-layer shared Warehouse mechanism. The structural design of Conv_KW is illustrated in Figure 2.

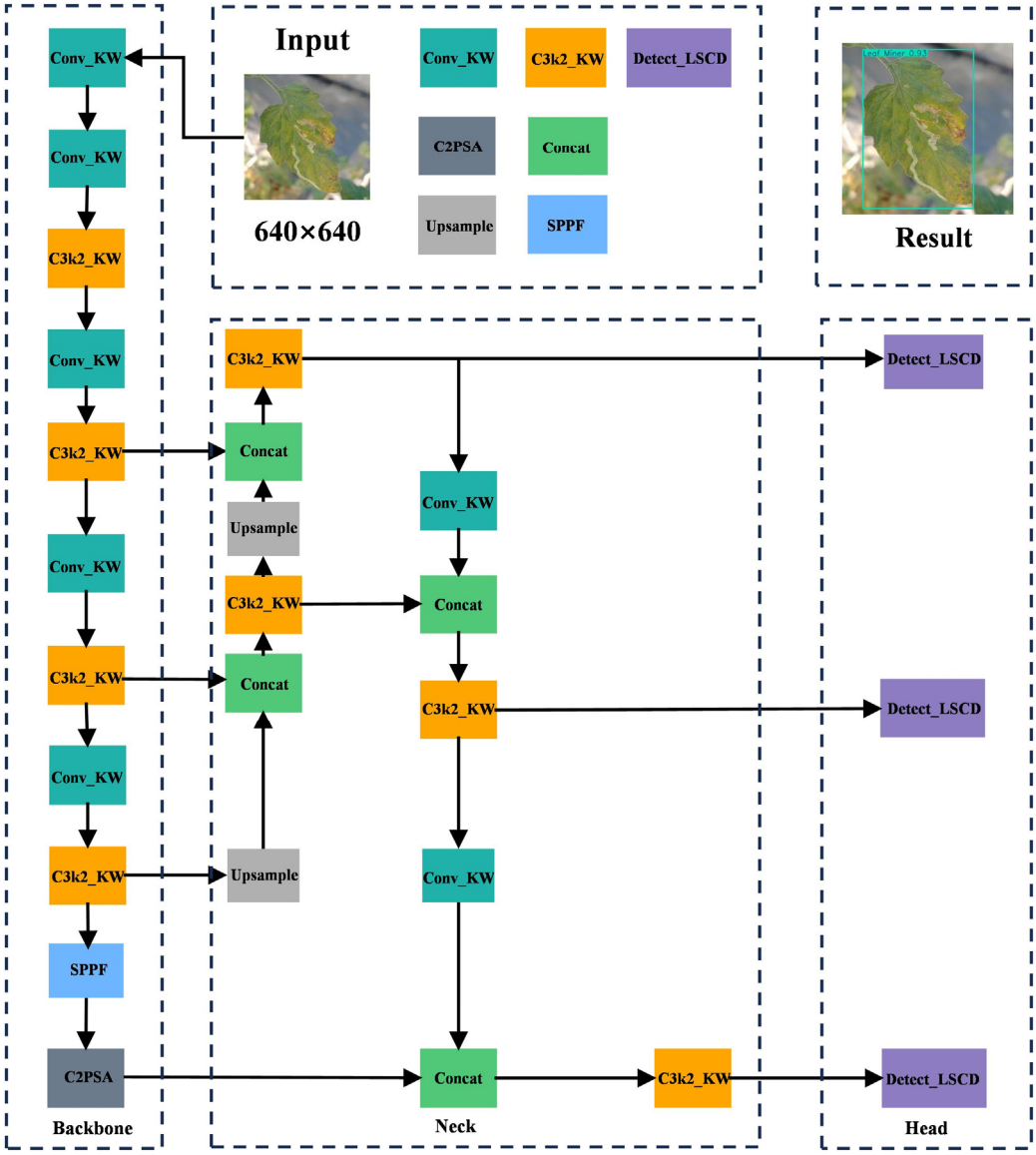


Figure 1. The architecture of the proposed YOLOv11n-KL network.

In this study, the Conv_KW module is integrated into both the Backbone and Neck networks of YOLOv11n to strengthen the model's ability to extract discriminative features from multi-scale targets in complex agricultural scenes. During training, the module dynamically aggregates feature responses from multiple convolutional kernels through an attention mechanism, whereas during inference, it operates equivalently as a single convolutional layer. This design effectively balances representational power during training with computational efficiency during inference. The dynamic convolution computation process is formulated as follows:

First, the spatial dimensions are compressed using Global Average Pooling (GAP) to extract channel-wise statistical features, as shown in Equation (1):

$$z = GAP(x) \tag{1}$$

Subsequently, the attention weights are computed through two fully connected (FC) layers followed by a batch normalization (BN) layer, as formulated in Equation (2):

$$\alpha = \sigma\left(\frac{FC2(BN(FC1(z)))}{T}\right) \tag{2}$$

where T denotes a temperature hyperparameter that controls the smoothness of the weight distribution, and σ represents the Sigmoid function. The final output is obtained by performing an attention-weighted summation over multiple convolutional kernels, as shown in Equation (3):

$$Conv_{kw(x)} = \sum_{i=1}^{n} \alpha_i \cdot (W_i * x) \tag{3}$$

where W_i represents the i-th convolutional kernel in the warehouse, n denotes the total number of kernels, and $*$ indicates the convolution operation.

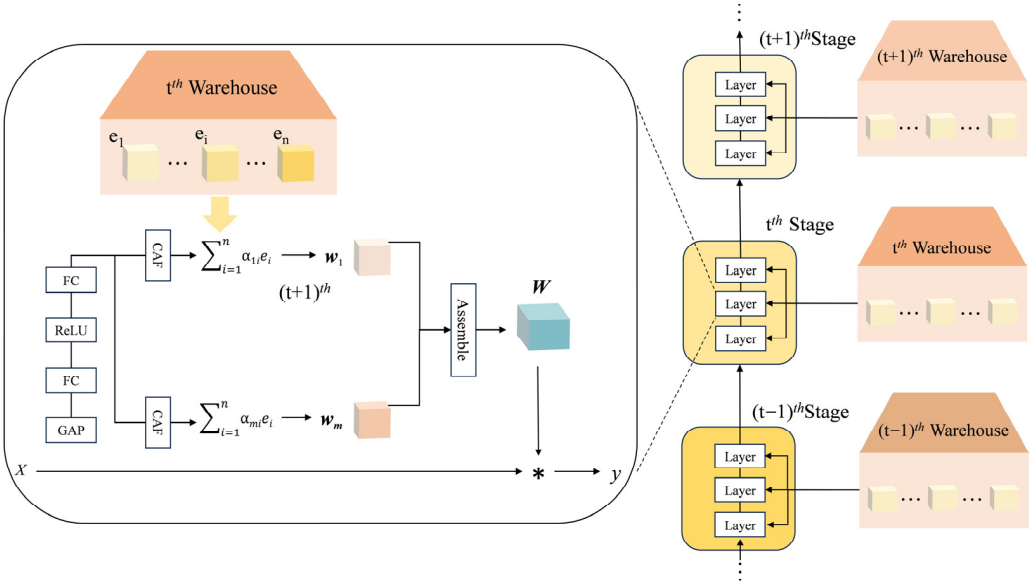


Figure 2. A schematic overview of the KernelWarehouse (KW) integrated into a convolutional network.

Through kernel sharing and the attention mechanism, this architecture substantially reduces both computational parameters and complexity. Meanwhile, it enhances the model's feature discrimination capability for targets of varying scales and morphologies, making it particularly effective for detecting and identifying pest and disease instances in agricultural imagery. The pseudocode of Conv_KW (Algorithm 1) outlines its implementation workflow.

Algorithm 1 Conv_KW Module

Input: Feature map X ∈ ℝˆB × C_{in} × H × W
Output: Enhanced feature map Y ∈ ℝˆB × C_{out} × H × W
1. **Initialize Kernel Warehouse:**
2. Create N convolution kernels $W_1, W_2, \ldots, W_N$ ∈ ℝˆC_{out} × C_{in} × K × K
3. All kernels share the same weight storage space
4. **Compute Global Feature Descriptor:**
5. Apply global average pooling to X:
6. G = AvgPool(X) ∈ ℝˆB × C_{in} × 1 × 1
7. **Compute Attention Weights:**
8. Apply first fully connected layer:
9. F1 = FC1(G) ∈ ℝˆB × R × 1 × 1 // R is reduction dimension
10. Apply batch normalization:
11. BN = BatchNorm(F1)
12. Apply second fully connected layer:

13. F2 = FC2(BN) $\in \mathbb{R}^{\wedge}B \times N \times 1 \times 1$
14. Apply softmax with temperature parameter:
15. α = Softmax(F2/T) $\in \mathbb{R}^{\wedge}B \times N \times 1 \times 1$ // T controls weight distribution
16. **Dynamic Kernel Generation:**
17. Compute weighted sum of warehouse kernels:
18. W_dynamic = $\Sigma(\alpha_i \times W_i)$ for i = 1. . .N
19. **Apply Dynamic Convolution:**
20. Perform convolution with dynamically weighted kernel:
21. Y = Conv(X, W_dynamic) + b
22. Optional: Apply activation function
23. Return Y

2.2.3. C3k2_KW Module

The C3k2_KW module is an enhanced version of the classical C3k2 (Cross-Stage Partial with kernel size 2) architecture that efficiently extracts features (Figure 3). This module substantially enhances feature representation and parameter efficiency while preserving the benefits of multi-branch feature fusion. Its key innovation lies in incorporating a KernelWarehouse mechanism, which stores a compact set of base kernels that can be dynamically assembled within a single convolutional operation. By dynamically combining these kernels with input-dependent weights, C3k2_KW enables richer multi-scale receptive fields without increasing the parameter count. This approach substantially reduces memory usage and computational load while maintaining detection accuracy.

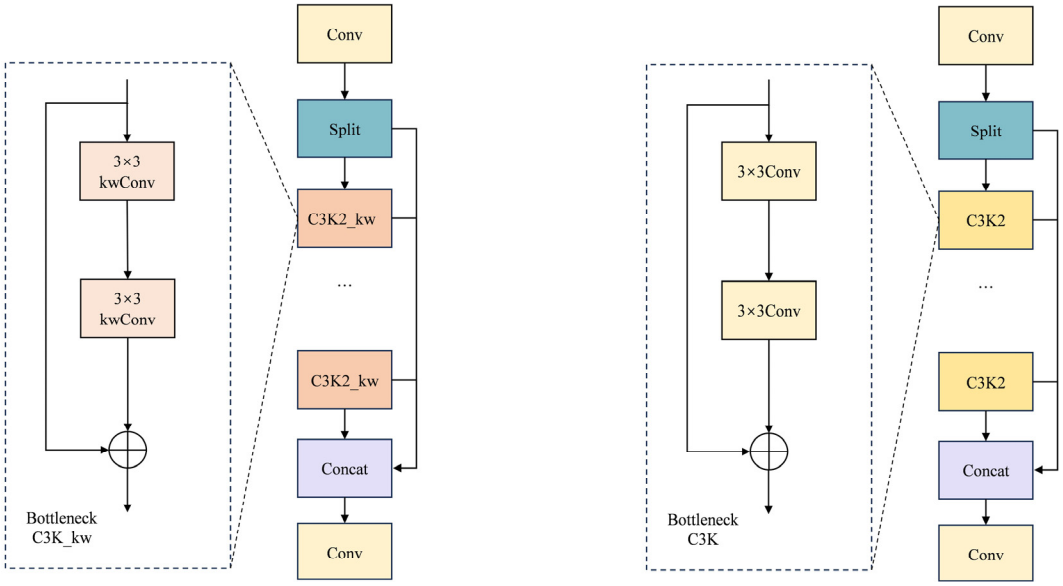


Figure 3. Structural comparison between the C3k2_KW module and the standard C3k2 module.

Traditional optimization approaches usually improve performance by increasing network depth or width, which substantially raises parameter count and computational requirements, rendering them unsuitable for resource-limited devices. The C3k2_KW module tackles these challenges with three structural innovations: (1) standard convolutional layers are replaced with Conv_KW layers, allowing dynamic allocation of kernels based on input features and improving adaptability to scale and structural variations; (2) a cross-layer kernel sharing strategy, managed by a Warehouse Manager for Kernel Cells, enhances parameter reuse efficiency; and (3) an attention-driven weight generation mechanism increases the discriminative power and generalization of kernel combina-

tions. The pseudocode of C3k2_KW (Algorithm 2) outlines its implementation workflow.

Algorithm 2 C3k2_KW Module

Input: Feature map $X \in \mathbb{R}^{\hat{}}B \times C_{in} \times H \times W$
Output: Enhanced feature map $Y \in \mathbb{R}^{\hat{}}B \times C_{out} \times H \times W$
1. **Initialize Parameters:**
2. C_{out}: Number of output channels
3. shortcut: Whether to use residual connection (default: False)
4. g: Number of groups for group convolution
5. e: Expansion ratio (default: 0.25)
6. C3k: Whether to use C3k or Bottleneck blocks
7. **Split Input Feature:**
8. Create two parallel branches from input X
9. Branch1: Identity mapping (if shortcut is True)
10. Branch2: Main processing branch
11. **Main Processing Branch:**
12. a. First KWConv Layer:
13. Apply KWConv with dynamic kernel:
14. X1 = KWConv(X, channels = $C_{in} \times e$)
15. Apply activation function (SiLU)
16. b. Create ModuleList of Bottleneck_KW blocks:
17. For each block in the list:
18. i. Apply Bottleneck_KW with dynamic kernels
19. ii. Update feature map
20. c. If C3k = True:
21. Replace Bottleneck_KW with C3k_KW blocks
22. Each C3k_KW contains multiple Bottleneck_KW sub-blocks
23. **Merge Branches:**
24. Combine outputs from branch1 and branch2:
25. if shortcut and $C_{in} == C_{out}$:
26. Y = X + X1 // Residual connection
27. else:
28. Y = X1 // No residual connection
29. **Final Processing:**
30. Apply Conv layer to ensure consistent output channels
31. Apply batch normalization
32. Apply activation function
33. Return Y

In this study, C3k2_KW is integrated into both the Backbone and Neck networks of YOLOv11n. This integration preserves the original feature fusion capability while reducing computational complexity and computational overhead. When processing pest and disease targets of varying scales and complex backgrounds, the module more accurately extracts distinctive features and effectively suppresses background interference. Consequently, it enhances detection accuracy for small targets while sustaining high inference efficiency. C3k2_KW is plug-and-play, providing a reliable framework for designing lightweight detection models, particularly suitable for real-time visual tasks on edge devices.

2.2.4. Detect_LSCD Module

In the YOLO series, the detection head generally consists of three branches, each processing information for the same object at different scales. Traditionally, these branches function independently, which may lead to inefficient parameter utilization and a higher risk of overfitting. To overcome these limitations, this study introduces the lightweight shared convolutional detection header (Detect_LSCD) module. The structure of this module is shown in Figure 4. Detect_LSCD achieves systematic optimization from three perspectives: (1) It adopts a grouped shared convolutional architecture, which applies uniform convolutional weights across multiple detection layers, thereby substantially reducing the number of parameters; (2) a lightweight Conv_GN module (Group Normalization convolution) replaces standard convolutions, further lowering computational cost and improving training stability; and (3) learnable scaling layers are incorporated to dynamically calibrate feature responses across hierarchical levels, mitigating feature conflicts caused by weight sharing and ensuring consistent multi-scale detection performance. The implementation process of the Detect_LSCD module is detailed in Algorithm 3, which outlines its pseudocode.

Algorithm 3 Detect_LSCD Module

Input: Multi-scale feature map list X ∈ R^{B × C × H × W} (from P3, P4, P5)
Number of classes nc, Hidden channel count hidc, Input channel tuple ch
Output: Object detection results (bounding box coordinates and class probabilities)
1. Initialize detection head parameters
2. Create Conv_GN module list (one for each feature layer)
3. Create shared convolution layer share_conv (contains 2 cascaded Conv_GN modules)
4. Create bounding box prediction layer cv2 and class prediction layer cv3
5. Initialize scale factor list and DFL layer
6. **Define forward propagation function**
7. out = []
8. for i in range(len(x)):
9. Step 1: Process each feature layer with dedicated Conv_GN
10. xi = self.cv1[i](x[i])
11. Step 2: Apply shared convolution layer for feature processing
12. xi = self.share_conv(xi)
13. Step 3: Compute bounding box and class features
14. bbox_feat = self.cv2[i](xi)
15. cls_feat = self.cv3[i](xi)
16. Step 4: Concatenate features and add to output list
17. fi = torch.cat((bbox_feat, cls_feat), 1)
18. out.append(fi)
19. if training mode:
20. return tuple(out)
21. **Inference mode**
22. **Generate anchors and perform bounding box decoding**
23. z = []
24. for i in range(len(out)):
25. bs, _, ny, nx = out[i].shape
26. out[i] = out[i].view(bs, self.no, ny, nx)
27. Step 1: Apply scale factors and DFL
28. if self.dynamic[i]:
29. out[i] = out[i] ∗ self.scale[i]

```
        Step 2: Generate grid coordinates and compute anchors
        grid = self._make_grid(nx, ny, i)
        y = out[i].sigmoid()
        Step 3: Decode bounding box coordinates
        y[..., 0:2] = (y[..., 0:2] * 2 − 0.5 + grid) * self.stride[i]
        y[..., 2:4] = ((y[..., 2:4] * 2) ** 2) * self.anchor_grid[i]
        z.append(y.view(bs, −1, self.no))
    return torch.cat(z, 1)
Define helper function _make_grid(nx, ny, i)
    Generate grid coordinates for bounding box decoding
    return coordinate grid
```

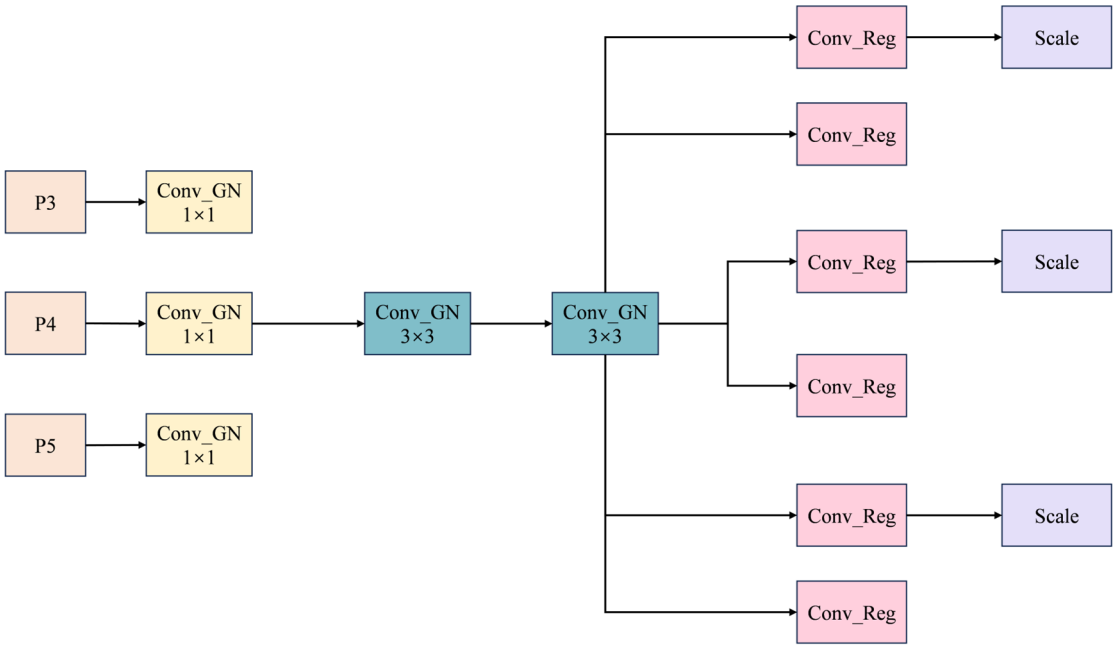


Figure 4. The architecture of Detect_LSCD head.

2.3. Model Training and Evaluation Metrics

2.3.1. Experimental Environment

The experiments in this study were conducted using the PyTorch (V2.2.2) deep learning framework on a Windows 11 operating system for both model training and testing. The primary hardware and software configurations of the experimental platform are detailed in Table 2.

Table 2. Experimental platform configuration.

Environment	Item	Specifications
Hardware Environment	CPU	Intel Xeon Platinum 8352V Processor
	GPU	NVIDIA GeForce RTX 4090
	Memory	32 GB
	Video Memory	24 GB
Software Environment	Operating System	Windows 11
	Deep Learning Framework	PyTorch 2.2.2
	CUDA	12.1
	Programming Language	Python 3.10.14

2.3.2. Training Parameters

In the model training conducted in this study, the resolution of all input images was adjusted to 640 × 640 pixels. The specific training hyperparameter settings are detailed in Table 3.

Table 3. Model training hyperparameter settings.

Parameter	Value	Parameter	Value
epochs	200	optimizer	SGD
patience	100	weight decay	0.0005
batch	32	momentum	0.937
imgsz	640	workers	8
lrf	0.01	close mosaic	0

2.3.3. Evaluation Metrics

To comprehensively evaluate the model's detection performance, this study employs several key metrics, including *Precision* (*P*), *Recall* (*R*), *F1-score*, and mean Average Precision (mAP@0.5), which serve as the main indicators of detection accuracy. In addition, the number of parameters and computational complexity, measured in Giga Floating-point Operations per second (GFLOPs), are used to evaluate the model's overall complexity. Collectively, these metrics serve as the evaluation criteria for assessing the algorithm's overall performance, as defined in Equations (4)–(8).

$$P = \frac{TP}{TP + FP} \tag{4}$$

$$R = \frac{TP}{TP + FN} \tag{5}$$

$$AP = \int_0^1 P(R)dR \tag{6}$$

$$mAP@0.5 = \frac{\sum_{i=1}^{C} APi}{C} \tag{7}$$

$$F1 - score = 2 \times \frac{P \times R}{P + R} \tag{8}$$

where TP represents true positives; FP represents false positives; FN represents false negatives; AP_i indicates the average precision of the i-th category; the area under the PR curve corresponds to the AP value; i denotes the category index, and C is the total number of categories.

3. Results

3.1. Model Comparison and Evaluation

To comprehensively evaluate the performance of the YOLOv11n-KL model in detecting tomato pests and diseases, we conducted a comparative analysis of multiple object detection models. The benchmark included the two-stage detector Faster R-CNN and several lightweight variants of the YOLO series. All models were trained and evaluated under identical conditions using our proprietary dataset to ensure fairness and objectivity in comparison. Detailed quantitative results are presented in Table 4.

As shown in Table 4, the proposed YOLOv11n-KL model achieves the highest overall performance among all evaluated object detection models. Specifically, it achieves an mAP@0.5 of 92.5%, which is 0.9% higher than that of the baseline YOLOv11n model, and significantly surpasses other lightweight YOLO variants and the two-stage Faster R-CNN

detector. In terms of precision and recall, YOLOv11n-KL attains values of 89.9% and 86.3%, respectively, yielding the highest F1-score (87.7%) among all models. Despite its superior detection accuracy, YOLOv11n-KL maintains outstanding computational efficiency. It retains the same parameter size (5.2 M) as YOLOv11n while substantially reducing computational complexity, with GFLOPs decreasing from 6.3 to 3.0. These results demonstrate that the proposed improvements effectively enhance detection performance without increasing model size, thereby improving both processing speed and energy efficiency.

Table 4. Comparison of experimental results.

Module	mAP@0.5 (%)	P (%)	R (%)	F1 (%)	Parameters (M)	GFLOPs
Faster R-CNN	84.7	78.3	71.8	75.1	28.4	470.5
YOLOv5s	89.5	88.6	81.8	84.4	5.0	7.2
YOLOv8n	90.8	88.0	84.6	85.9	5.4	6.9
YOLOv10n	90.5	89.0	83.4	85.5	5.5	8.2
YOLOv11n	91.6	87.0	86.6	86.3	5.2	6.3
YOLOv12n	91.0	87.7	85.6	86.2	5.3	6.5
YOLOv11n-KL	92.5	89.9	86.3	87.7	5.2	3.0

Overall, the results confirm that YOLOv11n-KL achieves an optimal balance between accuracy and efficiency, making it highly suitable for real-time tomato pest and disease detection in resource-limited environments, such as edge or embedded agricultural systems.

3.2. Exploring the Impact of Conv_KW Module Embedding Strategies

To systematically evaluate the embedding strategy for the Conv_KW module, this study implemented three comparative schemes: (1) embedding only in the Backbone network; (2) embedding only in the Neck network; (3) simultaneous embedding in both the Backbone and Neck networks (Backbone + Neck). The impact of each strategy on model performance and computational efficiency is shown in Table 5.

Table 5. Performance comparison of embedding the Conv_KW module at different network locations.

Module	Optimization Position	mAP@0.5 (%)	P (%)	R (%)	F1 (%)	Parameters (M)	GFLOPs
YOLOV11n	—	91.6	86.7	86.6	86.3	5.2	6.3
Conv_KW	Backbone	92.1	87.5	87.6	87.4	5.3	4.9
	Neck	91.7	90.5	84.6	86.9	5.3	6.1
	Backbone + Neck	92.7	89.5	87.1	88.0	5.3	4.6

Note: The symbol "—" indicates that module optimization was not applied.

The experimental results demonstrate that the Conv_KW module significantly improves detection accuracy while enabling model lightweighting, effectively overcoming the traditional trade-off between reduced computation and performance degradation. When embedded solely in the Backbone network, the module resulted in a 0.5% improvement in mAP@0.5 and a 22.2% reduction in computational load (GFLOPs). This improvement underscores the efficiency of the Conv_KW module in enhancing feature extraction while reducing computational cost. In contrast, embedding the module solely in the Neck network resulted in only marginal gains in mAP@0.5 (a 0.1% increase) and a decrease in Recall (from 86.6% to 84.6%), indicating that this strategy is less effective in achieving an optimal balance between detection accuracy and computational efficiency.

Notably, when the Conv_KW module was simultaneously embedded in both the Backbone and Neck networks, the model achieved the highest performance. This Backbone +

Neck embedding strategy led to a 1.1% increase in mAP@0.5 and a 27.0% reduction in computational load, with only a slight increase of 0.1 M parameters. This demonstrates a strong synergistic effect between the Backbone and Neck, where the two components collaborate to significantly enhance feature extraction, fusion, and overall model efficiency. The results strongly suggest that embedding the Conv_KW module in both stages maximizes detection accuracy and computational efficiency, making it the optimal embedding strategy.

3.3. Analysis of C3k2_KW Module Embedding Strategies

To systematically evaluate the optimal embedding strategy for the C3k2_KW module within the model architecture, this study implemented a two-stage validation procedure. The first stage was performed on the baseline YOLOv11n model to analyze the intrinsic characteristics of the module. Experimental results show that embedding C3k2_KW at a single location effectively reduces computational complexity. Specifically, the "Backbone-only" strategy achieved a 0.4% improvement in accuracy while reducing computational load by 19.1%, preliminarily validating its capacity for efficient feature representation and lightweighting (Figure 5a,b).

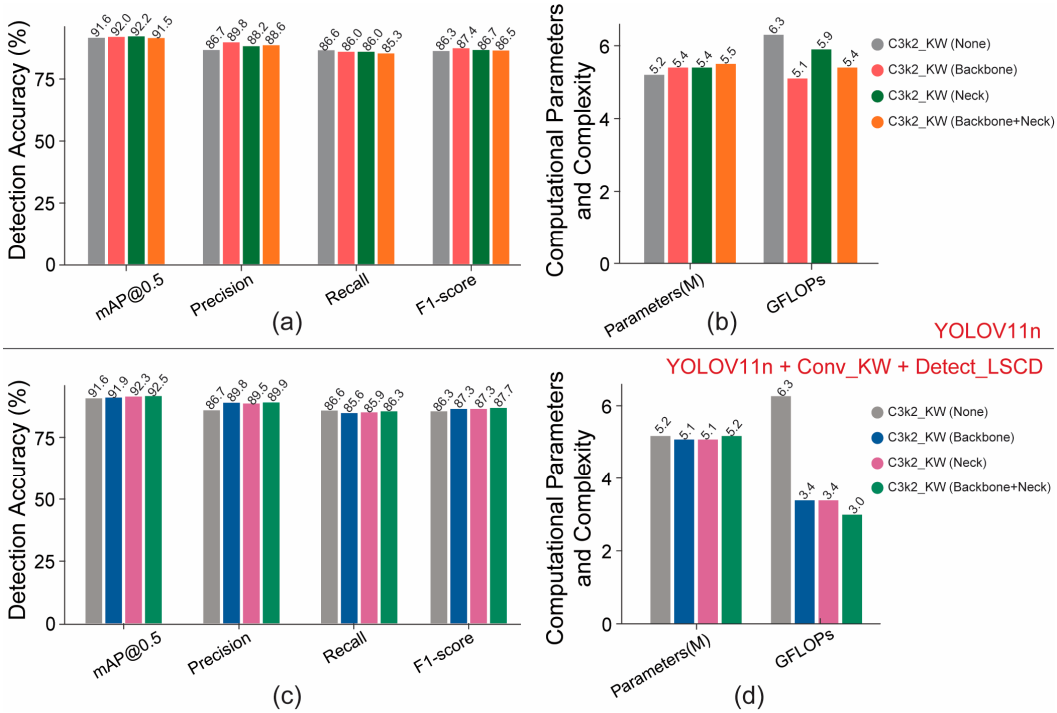


Figure 5. Performance and computational impact of different C3k2_KW module embedding strategies in YOLOv11n and the improved YOLOv11n-KL framework. (**a**,**b**) The effects of embedding the C3k2_KW module at different positions within the baseline YOLOv11n model on prediction accuracy and computational complexity. (**c**,**d**) The effects of embedding the C3k2_KW module at different positions within the improved YOLOv11n model (Conv_KW + Detect_LSCD) on prediction accuracy and computational complexity.

A more significant finding was identified in the second stage. When integrated into the fully improved framework, which already included Conv_KW and Detect_LSCD, the module's behavior changed notably (Figure 5c,d). Within this optimized architecture, the synergistic "Backbone + Neck" embedding strategy enabled the model to achieve a notable performance improvement. Despite a 52.4% reduction in computational load, the detection accuracy (mAP@0.5) improved by 0.9%. This result suggests that C3k2_KW does not function as an independent optimization unit but rather forms a highly synergistic "modular ecosystem" with Conv_KW and Detect_LSCD. This ecosystem enhances

efficiency across feature extraction, fusion, and decoding through multi-level, cross-stage optimization, effectively alleviating the trade-off between accuracy and computational cost in lightweight design. As a result, the "Backbone + Neck" embedding strategy became the core architectural decision for the final YOLOv11n-KL model.

3.4. Ablation Experiments and Analysis

To evaluate the contribution of each module to the overall performance of the YOLOv11n-KL model, a series of ablation experiments was conducted, and the results are summarized in Figure 6. The baseline model without any of the three modules (C3k2_KW, Conv_KW, or LSCD) achieved an mAP@0.5 of 91.6%, precision of 86.7%, recall of 86.6%, and F1-score of 86.3%, with a model size of 5.2 M parameters and 6.3 GFLOPs. When the C3k2_KW module was introduced individually, the model maintained a comparable mAP@0.5 (91.5%) but exhibited a notable improvement in precision (88.6%) while slightly reducing GFLOPs to 5.4, indicating enhanced computational efficiency. Incorporating the Conv_KW module alone yielded the most significant gain in overall accuracy, increasing mAP@0.5 to 92.7%, recall to 87.1%, and F1-score to 88.0%, while also reducing GFLOPs to 4.6, demonstrating its strong contribution to both accuracy and efficiency.

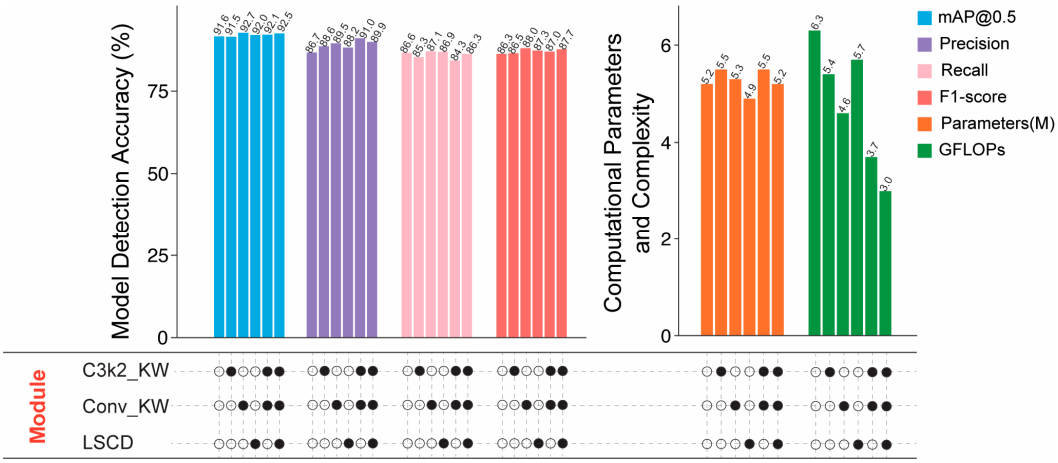


Figure 6. Ablation study of C3k2_KW, Conv_KW, and LSCD modules on detection performance and computational complexity in YOLOv11n-KL framework. The UpSet plot visualizes the relationships among different combinations of algorithms. The upper bar chart presents the prediction accuracy and computational complexity associated with each module combination. In the lower panel, the solid dots aligned with the horizontal algorithm labels indicate the module components that constitute the models represented by the corresponding vertical bars.

The addition of the LSCD module alone resulted in a moderate improvement (mAP@0.5 = 92.0%, F1-score = 87.3%) and reduced model complexity (parameters = 4.9 M, GFLOPs = 5.7). When both C3k2_KW and Conv_KW were used together, the model achieved high precision (91.0%) and a relatively low GFLOPs (3.7), indicating a favorable balance between detection accuracy and computational cost. Finally, the combination of all three modules (C3k2_KW + Conv_KW + LSCD) achieved an optimal overall performance, with an mAP@0.5 of 92.5%, precision of 89.9%, recall of 86.3%, F1-score of 87.7%, and the lowest computational cost (3.0 GFLOPs). These results demonstrate that the joint integration of the three modules enables a synergistic enhancement of both detection accuracy and model efficiency, effectively mitigating the traditional trade-off between precision and computational complexity in lightweight detection models.

3.5. Architectural Generalization and Scalability Validation

To evaluate the scalability and architectural generalization of the proposed KL (C3k2_KW + Conv_KW+ LSCD) improvement strategy, the modules were embedded into two YOLOv11 variants: the lightweight YOLOv11n and the medium-scale YOLOv11s. As shown in Figure 7, consistent performance improvements were observed across both architectures. For YOLOv11n, integrating the KL module increased mAP@0.5 from 91.6% to 92.5% while reducing computational cost by 52.4% (from 6.3 GFLOPs to 3.0 GFLOPs) without increasing the number of parameters. Similarly, YOLOv11s-KL achieved a higher mAP@0.5 of 93.6%, accompanied by a 45.5% reduction in GFLOPs (from 21.3 to 11.6) and a slight improvement in the F1-score.

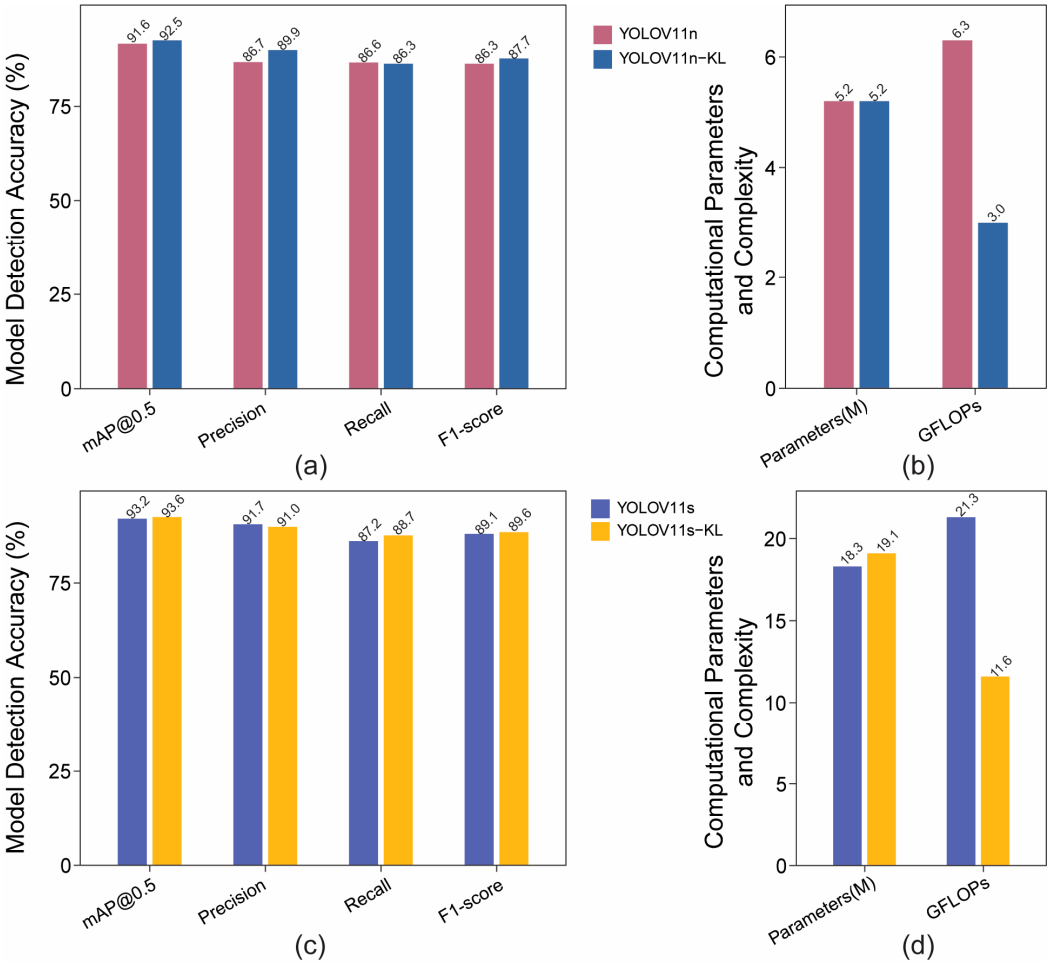


Figure 7. Scalability and architectural generalization of the KL (C3k2_KW + Conv_KW + LSCD) enhancement across YOLOv11n and YOLOv11s architectures. (**a**,**b**) The effects of the presence or absence of the KL module in the YOLOv11n model on prediction accuracy and computational complexity. (**c**,**d**) The effects of the presence or absence of the KL module in the YOLOv11s model on prediction accuracy and computational complexity.

These results confirm that the KL-enhanced design consistently improves detection accuracy while significantly reducing computational load across different model scales. This demonstrates that the dynamic kernel composition and computational sharing mechanisms of the KL structure enhance feature extraction efficiency in an architecture-independent manner. The strong adaptability across various network depths and widths underscores the method's scalability and suitability for deployment under diverse resource constraints, ranging from embedded devices to edge computing platforms.

3.6. Visual Analysis of Detection Performance and Model Attention

To evaluate the detection capability and internal reasoning of the proposed model, we performed a comparative visual analysis of detection outputs and class activation maps (CAMs) across multiple architectures (Figure 8). The YOLOv11n-KL model consistently exhibits superior localization and classification performance, accurately detecting small-scale disease spots with precise boundaries. Its attention maps display highly concentrated activations aligned with actual pathological regions, indicating that the model effectively prioritizes disease-relevant features. In contrast, conventional models (YOLOv5s, YOLOv8n) exhibit diffuse and less discriminative attention, frequently activating healthy regions and producing fragmented detections, whereas YOLOv11n and YOLOv12n show moderate improvements but remain less focused.

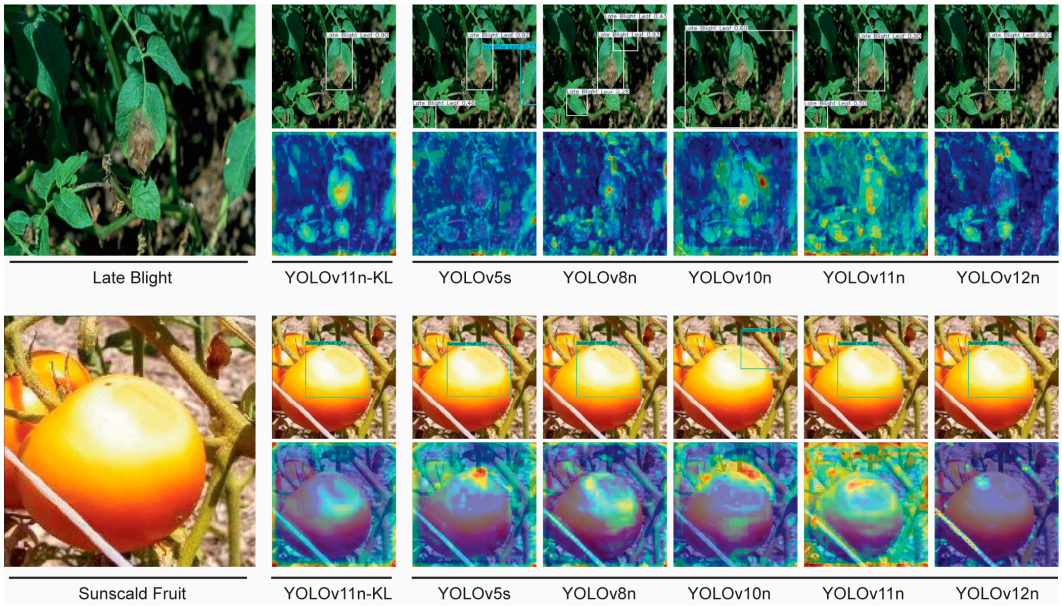


Figure 8. Comparative visualization of detection results and class activation maps across different models.

These visual observations corroborate the quantitative results and underscore the contributions of the Conv_KW and C3k2_KW modules. The Conv_KW module enhances multi-scale feature discrimination, whereas the C3k2_KW module facilitates efficient propagation of these focused activations. Consequently, YOLOv11n-KL achieves higher detection accuracy along with more interpretable reasoning. The alignment between model attention and pathological regions provides mechanistic insight and supports reliable, transparent decision-making, which is particularly valuable for practical agricultural applications.

4. Discussion

The development of efficient and accurate object detection models for agricultural applications has emerged as a central focus in intelligent agriculture research [29]. This study proposes and validates YOLOv11n-KL, a novel lightweight model designed to address the critical challenge of deploying accurate tomato pest and disease detection systems on resource-constrained edge devices. Building on the YOLOv11 framework, the approach introduces three dedicated modules that work synergistically to achieve superior performance while maintaining computational efficiency.

The core innovation of this study lies in the strategic integration of the Conv_KW, C3k2_KW, and Detect_LSCD modules. The Conv_KW module introduces a dynamic KernelWarehouse mechanism that significantly enhances feature extraction capabilities,

particularly for small, irregularly shaped disease spots commonly observed in agricultural imagery. This approach aligns with recent advancements in dynamic convolution [30] and specifically addresses the challenges of agricultural target detection. Ablation studies demonstrate that the Conv_KW module alone contributes substantially to detection accuracy improvement while reducing computational requirements (Figure 6). The C3k2_KW module extends the dynamic kernel concept to feature fusion operations, enabling more efficient multi-scale feature processing. This innovation is particularly valuable when combined with the other proposed modules, producing a synergistic effect that surpasses the performance of individual components. Experimental results show that the complete YOLOv11n-KL model achieves a 52.4% reduction in computational load while improving detection accuracy, effectively addressing the efficiency-accuracy trade-off that has constrained previous approaches [31].

Comparative analysis shows that YOLOv11n-KL outperforms not only the baseline YOLOv11n but also other state-of-the-art models, including YOLOv8s and YOLOv10n (Table 4). The model achieves a superior balance between precision (89.9%) and recall (86.3%), yielding the highest F1-score (87.7%) among the compared methods. These results are particularly notable given the model's high efficiency, requiring only 3.0 GFLOPs for inference. This performance advantage is even more pronounced compared to two-stage detectors such as Faster R-CNN [32], which typically demand substantially greater computational resources. The generalizability of this approach is demonstrated through experiments with the YOLOv11s architecture (Figure 7), where consistent performance improvements confirm the scalability of the proposed modules. This architectural independence suggests that the KL enhancement strategy can be effectively applied across various model scales, offering flexibility for different deployment scenarios. Visual analysis of attention maps (Figure 8) further corroborates the model's enhanced capability, revealing more focused and precise attention on pathological regions compared to other methods.

Although the results are promising, certain limitations warrant consideration. The model's performance under extreme conditions, such as heavy occlusion or atypical lighting variations, requires further investigation. Additionally, as with many deep learning approaches [33], generalization to entirely new environmental conditions remains a challenge, necessitating further research. Moreover, in the present study, key hyperparameters were selected based on empirical experience and commonly adopted settings rather than exhaustive hyperparameter tuning, which may limit the model's ability to reach its optimal performance. Future work should emphasize large-scale real-world validation and integration with emerging technologies, such as neural architecture search [34], as well as systematic hyperparameter optimization strategies [35], for further optimization.

5. Conclusions

YOLOv11n-KL represents a notable advancement in lightweight vision models for agricultural applications. By achieving an optimal balance between detection accuracy and computational efficiency, the model offers a practical solution for real-time pest and disease monitoring in edge computing environments. The proposed architectural modifications offer practical insights for efficient deep learning and may support improvements in precision agriculture applications. Future research will focus on large-scale real-world deployment and validation under diverse field conditions, along with further optimization through automated model design and hyperparameter tuning. In addition, integrating the proposed framework with multimodal sensing data and adaptive learning mechanisms may further enhance its robustness and applicability in real-world agricultural scenarios.

Author Contributions: Conceptualization, S.P. and X.C.; Methodology, S.P.; Software, S.P. and Z.S.; Formal analysis, S.P.; Investigation, S.P., Z.J. and L.S.; Resources, S.P.; Data curation, S.P. and X.C.;

Writing—original draft, S.P. and X.C.; Writing—review and editing, Y.J., W.Y. and L.Y.; Visualization, S.P. and W.Y.; Supervision, W.Y. and L.Y.; Project administration, W.Y. and L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: Major Science and Technology Special Project of Henan Province (241100110200); China Postdoctoral Science Foundation (2024M760814); and Postdoctoral Fellowship Program of CPSF (China Postdoctoral Science Foundation) (GZC20230720).

Data Availability Statement: All resources used in this study are publicly accessible. The implementation code for the YOLOv11n-KL model can be found at https://github.com/pengshibo-png/Yolov11KL (accessed on 20 November 2025). Due to the fact that the tomato pest and disease dataset constitutes a valuable research resource and involves data usage restrictions, it cannot be publicly released at this stage. However, the processed dataset will be made available by the authors upon reasonable request.

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

References

1. Liu, J.; Wang, X. Tomato diseases and pests detection based on improved Yolo V3 convolutional neural network. *Front. Plant Sci.* **2020**, *11*, 898. [CrossRef] [PubMed]
2. Iftikhar, Y.; Mubeen, M.; Sajid, A.; Zeshan, M.A.; Shakeel, Q.; Abbas, A.; Bashir, S.; Kamran, M.; Anwaar, H. Effects of tomato leaf curl virus on growth and yield parameters of tomato crop. *Arab J. Plant Prot.* **2021**, *39*, 79–83. [CrossRef]
3. Ferrero, V.; Baeten, L.; Blanco-Sánchez, L.; Planelló, R.; Díaz-Pendón, J.A.; Rodríguez-Echeverría, S.; Haegeman, A.; de la Peña, E. Complex patterns in tolerance and resistance to pests and diseases underpin the domestication of tomato. *New Phytol.* **2020**, *226*, 254–266. [CrossRef]
4. Panchal, A.V.; Patel, S.C.; Bagyalakshmi, K.; Kumar, P.; Khan, I.R.; Soni, M. Image-based plant diseases detection using deep learning. *Mater. Today Proc.* **2023**, *80*, 3500–3506. [CrossRef]
5. Feng, Y.; Liu, C.; Han, J.; Lu, Q.; Xing, X. IRB-5-CA net: A lightweight, deep learning-based approach to wheat seed identification. *IEEE Access* **2023**, *11*, 119553–119559. [CrossRef]
6. Han, K.; Zhang, N.; Xie, H.; Wang, Q.; Ding, W. An improved strategy of wheat kernel recognition based on deep learning. *DYNA Ing. Ind.* **2023**, *98*, 91–97. [CrossRef] [PubMed]
7. Wang, X.; Liu, J.; Zhu, X. Early real-time detection algorithm of tomato diseases and pests in the natural environment. *Plant Methods* **2021**, *17*, 43. [CrossRef]
8. Iftikhar, M.; Kandhro, I.A.; Kausar, N.; Kehar, A.; Uddin, M.; Dandoush, A. Plant disease management: A fine-tuned enhanced CNN approach with mobile app integration for early detection and classification. *Artif. Intell. Rev.* **2024**, *57*, 167. [CrossRef]
9. Wrat, G.; Ranjan, P.; Mishra, S.K.; Jose, J.T.; Das, J. Neural network-enhanced internal leakage analysis for efficient fault detection in heavy machinery hydraulic actuator cylinders. *J. Mech. Eng. Sci.* **2025**, *239*, 1021–1031. [CrossRef]
10. Chen, W.; Zheng, L.; Xiong, J. Algorithm for crop disease detection based on channel attention mechanism and lightweight up-sampling operator. *IEEE Access* **2024**, *12*, 109886–109899. [CrossRef]
11. Wen, C.; Chen, H.; Ma, Z.; Zhang, T.; Yang, C.; Su, H.; Chen, H. Pest-YOLO: A model for large-scale multi-class dense and tiny pest detection and counting. *Front. Plant Sci.* **2022**, *13*, 973985. [CrossRef]
12. Mo, L.; Xie, R.; Ye, F.; Wang, G.; Wu, P.; Yi, X. Enhanced tomato pest detection via leaf imagery with a new loss function. *Agronomy* **2024**, *14*, 1197. [CrossRef]
13. Cai, H.; Li, J.; Hu, M.; Gan, C.; Han, S. EfficientViT: Lightweight Multi-Scale Attention for High-Resolution Dense Prediction. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–6 October 2023; pp. 17256–17267.
14. Fang, M.; Tan, Z.; Tang, Y.; Chen, W.; Huang, H.; Dananjayan, S.; He, Y.; Luo, S. Pest-ConFormer: A hybrid CNN-Transformer architecture for large-scale multi-class crop pest recognition. *Forests Expert Syst. Appl.* **2024**, *255*, 124833. [CrossRef]
15. Sun, H.; Fu, R.; Wang, X.; Wu, Y.; Al-Absi, M.A.; Cheng, Z.; Chen, Q.; Sun, Y. Efficient deep learning-based tomato leaf disease detection through global and local feature fusion. *BMC Plant Biol.* **2025**, *25*, 311. [CrossRef] [PubMed]
16. Ye, Y.; Chen, Y.; Xiong, S. Field detection of pests based on adaptive feature fusion and evolutionary neural architecture search. *Comput. Electron. Agric.* **2024**, *221*, 108936. [CrossRef]
17. Gao, W.; Zong, C.; Wang, M.; Zhang, H.; Fang, Y. Intelligent identification of rice leaf disease based on YOLO V5-EFFICIENT. *Crop Prot.* **2024**, *183*, 106758. [CrossRef]
18. Liu, W.; Zhai, Y.; Xia, Y. Tomato leaf disease identification method based on improved YOLOX. *Agronomy* **2023**, *13*, 1455. [CrossRef]
19. Albahar, M. A survey on deep learning and its impact on agriculture: Challenges and opportunities. *Agriculture* **2023**, *13*, 540. [CrossRef]

20. Zhao, Z.; Song, A.; Zheng, S.; Xiong, Q.; Guo, J. DSC-HRNet: A lightweight teaching pose estimation model with depthwise separable convolution and deep high-resolution representation learning in computer-aided education. *Int. J. Inf. Technol.* **2023**, *15*, 2373–2385. [CrossRef]
21. Chen, R.; Qi, H.; Liang, Y.; Yang, M. Identification of plant leaf diseases by deep learning based on channel attention and channel pruning. *Front. Plant Sci.* **2022**, *13*, 1023515. [CrossRef]
22. Zhang, X.; Liang, K.; Zhang, Y. Plant pest and disease lightweight identification model by fusing tensor features and knowledge distillation. *Front. Plant Sci.* **2024**, *15*, 1443815. [CrossRef]
23. Zhao, L.; Wang, L. A new lightweight network based on MobileNetV3. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 1–15. [CrossRef]
24. Jiang, Y.; Tong, W. Improved lightweight identification of agricultural diseases based on MobileNetV3. *arXiv* **2025**, arXiv:2207.11238.
25. Song, Q.; Yao, B.; Xue, Y.; Ji, S. MS-YOLO: A lightweight and high-precision YOLO model for drowning detection. *Sensors* **2024**, *24*, 6955. [CrossRef]
26. Yang, M.; Fan, X. YOLOv8-Lite: A lightweight object detection model for real-time autonomous driving systems. *ICCK Trans. Emerg. Top. Artif. Intell.* **2024**, *1*, 1–16. [CrossRef]
27. Jia, L.; Wang, T.; Chen, Y.; Zang, Y.; Li, X.; Shi, H.; Gao, L. MobileNet-CA-YOLO: An improved YOLOv7 based on the MobileNetV3 and attention mechanism for Rice pests and diseases detection. *Agriculture* **2023**, *13*, 1285. [CrossRef]
28. Li, C.; Yao, A. Kernelwarehouse: Rethinking the design of dynamic convolution. *arXiv* **2024**, arXiv:2406.07879. [CrossRef]
29. Dhanasekar, S. A comprehensive review on current issues and advancements of Internet of Things in precision agriculture. *Comput. Sci. Rev.* **2025**, *55*, 100694. [CrossRef]
30. Zhang, Y.; Zhang, H.; Huang, Q.; Han, Y.; Zhao, M. DsP-YOLO: An anchor-free network with DsPAN for small object detection of multiscale defects. *Expert Syst. Appl.* **2024**, *241*, 122669. [CrossRef]
31. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2004**, arXiv:2004.10934.
32. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Access* **2016**, *39*, 1137–1149. [CrossRef]
33. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
34. Chen, G.; Hou, Y.; Cui, T.; Li, H.; Shangguan, F.; Cao, L. YOLOv8-CML: A lightweight target detection method for Color-changing melon ripening in intelligent agriculture. *Sci. Rep.* **2024**, *14*, 14400. [CrossRef] [PubMed]
35. Stefenon, S.F.; Seman, L.O.; Klaar, A.C.R.; Ovejero, R.G.; Leithardt, V.R.Q. Hypertuned-YOLO for interpretable distribution power grid fault location based on EigenCAM. *Ain Shams Eng. J.* **2024**, *15*, 102722. [CrossRef]

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.