agriculture

MDPI

Article

LP-YOLO: A Lightweight Object Detection Network Regarding Insect Pests for Mobile Terminal Devices Based on Improved YOLOv8

Yue Yu [1], Qi Zhou [2], Hao Wang [2], Ke Lv [3], Lijuan Zhang [4], Jian Li [1,*] and Dongming Li [4,*]

[1] School of Information Technology, Jilin Agricultural University, Changchun 130118, China; yueyu@mails.jlau.edu.cn
[2] School of Computer Science, Xi'an Jiaotong University, Xi'an 710049, China; 2224111333@stu.xjtu.edu.cn (Q.Z.); 2222210687@stu.xjtu.edu.cn (H.W.)
[3] School of Electrical Engineering and Automation, Wuhan University, Wuhan 430072, China; 2022302191115@whu.edu.cn
[4] College of Internet of Things Engineering, Wuxi University, Wuxi 214105, China; zhanglijuan@cwxu.edu.cn
* Correspondence: lijian@jlau.edu.cn (J.L.); lidongming@cwxu.edu.cn (D.L.)

Abstract: To enhance agricultural productivity through the accurate detection of pests under the constrained resources of mobile devices, we introduce LP-YOLO, a bespoke lightweight object detection framework optimized for mobile-based insect pest identification. Initially, we devise lightweight components, namely LP_Unit and LP_DownSample, to serve as direct substitutes for the majority of modules within YOLOv8. Subsequently, we develop an innovative attention mechanism, denoted as ECSA (Efficient Channel and Spatial Attention), which is integrated into the network to forge LP-YOLO(l). Moreover, assessing the trade-offs between parameter reduction and computational efficiency, considering both the backbone and head components of the network, we use structured pruning methods for the pruning process, culminating in the creation of LP-YOLO(s). Through a comprehensive series of evaluations on the IP102 dataset, the efficacy of LP-YOLO as a lightweight object detection model is validated. By incorporating fine-tuning techniques during training, LP-YOLO(s)n demonstrates a marginal mAP decrease of only 0.8% compared to YOLOv8n. However, it achieves a significant reduction in parameter count by 70.2% and a remarkable 40.7% increase in FPS, underscoring its efficiency and performance.

Keywords: object detection; pests; lightweight network; LP-YOLO; attention mechanism

Citation: Yu, Y.; Zhou, Q.; Wang, H.; Lv, K.; Zhang, L.; Li, J.; Li, D. LP-YOLO: A Lightweight Object Detection Network Regarding Insect Pests for Mobile Terminal Devices Based on Improved YOLOv8. *Agriculture* **2024**, *14*, 1420. https://doi.org/10.3390/agriculture14081420

Academic Editors: Gniewko Niedbała, Magdalena Piekutowska, Sebastian Kujawa and Tomasz Wojciechowski

Received: 1 August 2024
Revised: 19 August 2024
Accepted: 20 August 2024
Published: 21 August 2024

1. Introduction

The proliferation of pests not only precipitates substantial economic losses in agricultural production but also catalyzes the dissemination of infectious diseases and disrupts ecological equilibria. Effective pest detection mitigates the risk of widespread infestations and diminishes both the difficulty and the expenditure associated with pest management endeavors. Moreover, precise pest identification underpins the study of pest ecological habits and reproductive patterns, thus providing a scientific basis for the pursuit of sustainable agricultural development.

Historically, pest identification has primarily relied on the specialized expertise of entomologists and taxonomists. However, with the rise of the Internet and advancements in information technology, new methodologies for identifying crop pests and diseases have surfaced. The introduction of the R-CNN algorithm [1] marked a critical juncture in applying deep learning to object detection, utilizing Convolutional Neural Networks (CNNs) for feature extraction and classification within candidate regions, significantly boosting detection accuracy. This evolution spurred the development of algorithms such as Fast R-CNN [2] and Faster R-CNN [3], which incorporate Region Proposal Networks (RPNs) to enhance detection efficiency and speed.

Recent advancements in single-stage detection algorithms, like the SSD (Single-Shot Multibox Detector) series [4] and RetinaNet [5], have notably pushed forward the field of object detection. The SSD framework introduced an innovative approach for multi-scale object detection by directly predicting objects at various scales from feature maps, enabling real-time performance. Enhancements such as SSDLite [6], which employs depthwise separable convolutions, have optimized the model for deployment on resource-constrained devices. Furthermore, models like MDSSD [7] have integrated multi-scale deconvolutional modules to bolster the detection of small objects.

RetinaNet, introduced in 2017, tackled the challenge of class imbalance in object detection with the introduction of Focal Loss, selectively down-weighting easily classifiable examples to concentrate on challenging instances, thereby enhancing the model's accuracy in scenarios with significant class imbalance. Subsequent enhancements to RetinaNet involved the adoption of more efficient backbone architectures, such as EfficientNet [8], and improvements to the Feature Pyramid Network (FPN) for better multi-scale feature representation.

Simultaneously, the YOLO (You Only Look Once) series [9] of algorithms, operating on a single-stage detection pipeline, has demonstrated exceptional speed and accuracy. The latest iteration, YOLOv10 [10], builds upon previous versions by incorporating advanced network architectures and optimization techniques, further extending the boundaries of real-time object detection performance.

In parallel, the development of attention mechanisms in deep learning has continued to evolve, enhancing model performance across a spectrum of tasks including object detection. Initially utilized in natural language processing, attention mechanisms have been increasingly applied to computer vision, significantly improving the ability of neural networks to focus on relevant features within vast quantities of visual data. Recent advancements, such as the introduction of transformer models in vision tasks, exemplified by the Vision Transformer (ViT) [11], have revolutionized how models process images, highlighting the utility of attention mechanisms in interpreting complex scenes more effectively.

Further developments include the integration of attention into existing architectures to refine their efficacy. For instance, the Bottleneck Transformers [12] incorporate transformer blocks seamlessly within convolutional networks, enhancing the representational power without compromising the efficiency inherent to CNNs. Moreover, the CrossViT [13] structure introduces dual-branch transformers to handle different scale features simultaneously, boosting performance in multi-scale object detection.

Recent advancements in pest target detection, driven by deep learning, convolutional neural network (CNN) technologies, and the integration of sophisticated attention mechanisms, have enabled more accurate pest identification and classification across diverse environments. In 2023, Zhang et al. [14] introduced an enhancement to YOLOX [15] via a Cross-Layer Transformer, achieving an AP50 of 57.7 on the IP102 dataset. Likewise, Zhu et al. [16] introduced the CBF-YOLO network, incorporating CSE-ELAN, Bi-PAN, and FFE modules, to enhance detection of common soybean pests in complex environments, achieving a new benchmark with a mean average precision (mAP) of 86.9%. However, the extensive parameter counts and complexity inherent in these algorithms render them less suitable for deployment on field mobile devices, a limitation attributed to their substantial computational demands and resource requirements. Consequently, their practical application, especially in agricultural scenarios where mobility and resource constraints are paramount, remains a formidable challenge.

In the domain of insect detection, recent investigations have also underscored the development of lightweight networks, yielding remarkable accomplishments. For instance, Xu et al. [17] developed a lightweight SSV2-YOLO-based model aimed at detecting sugarcane aphids in unstructured natural environments, contributing significantly to the field. In their methodology, they employed ShuffleNet V2 modules as replacements for the original components of the YOLOv5 network, maintaining the core structure. This adaptation, while striving for enhanced model compactness, introduced a trade-off, affecting accuracy

to some extent and presenting challenges in achieving the desired level of lightweight efficiency. Nevertheless, the aforementioned model, despite its advancements, remains insufficiently lightweight and is limited in its pest detection range. There exists a pivotal opportunity for the design of an even more streamlined network that preserves, if not enhances, detection accuracy.

In response to this challenge, we introduce LP-YOLO, a network that achieves a substantial reduction in parameters and complexity without compromising detection accuracy. To harmonize model precision with scale and simplicity, we embrace depthwise separable convolution and channel shuffle, aiming to refine the network's operational efficiency. Furthermore, the incorporation of both spatial and channel attention mechanisms serves to amplify the network's receptive capacity and adaptability while concurrently attenuating irrelevant inputs. We unveil the innovative LP_Unit and LP_DownSample modules, which are devised as strategic alternatives to the architectural components of YOLOv8, culminating in the establishment of our LP-YOLO(l) framework. Subsequent network pruning efforts yield an even more streamlined variant, LP-YOLO(s). The LP-YOLO framework encompasses two distinct models, LP-YOLO(l) and LP-YOLO(s), each available at five different scales: n, s, m, l, and x.

The contributions of this research are threefold:

- Innovative lightweight modules: We introduce two pioneering lightweight modules, LP_Unit and LP_DownSample, leveraging depthwise separable convolution to diminish computational load and parameter quantity. The implementation of channel shuffle addresses the restricted information flow between channel groups, a limitation encountered with depthwise separable convolution. These enhancements substantially elevate the network's lightweight characteristics by replacing segments of the original architectural framework.
- Efficient Channel and Spatial Attention (ECSA): To harness the computational efficacy of Efficient Channel Attention (ECA) while compensating for its omission of spatial attention, we propose the ECSA mechanism. This novel approach, in conjunction with the CBAM attention mechanism, directs the network's focus towards critical information during data processing, significantly boosting feature extraction capabilities. This integration facilitates the derivation of the LP-YOLO(l) model.
- Optimized network pruning: The LP-YOLO(l) undergoes a strategic network pruning process, followed by fine-tuning training, culminating in the LP-YOLO(s) model. We implement weight pruning to halve the output channels of recurrent backbone network modules, thereby reducing parameters and computational demands while preserving essential weights. Moreover, by augmenting the repetition of select modules, we enhance feature extraction depth. For the network's neck and head sections, we streamline by pruning neurons and connections, thus advancing the network's efficiency, compactness, and operational performance. We compare the parameter counts of the backbone and head before and after network pruning using the modified network. After pruning, the parameters of the backbone decrease from 702,013 to 251,457, a reduction of 64.2%. The head's parameters decrease from 2,074,092 to 746,009, a reduction of 64.0%. Additionally, we find that the detect layer alone accounts for 39.2% of the total parameters before pruning, and the reduction in this layer contributes to 29.2% of the total parameter reduction. Therefore, reducing the number of detection heads is necessary. On inference speed, network pruning also produces significant effects. Consequently, LP-YOLO(s) nearly matches YOLOv8n, with a minimal 0.8% mAP reduction, yet achieves a remarkable 70.2% reduction in parameter volume.

2. Material and Methods

Given the extensive content in this section, Figure 1 provides a flowchart that outlines all the steps of our approach.

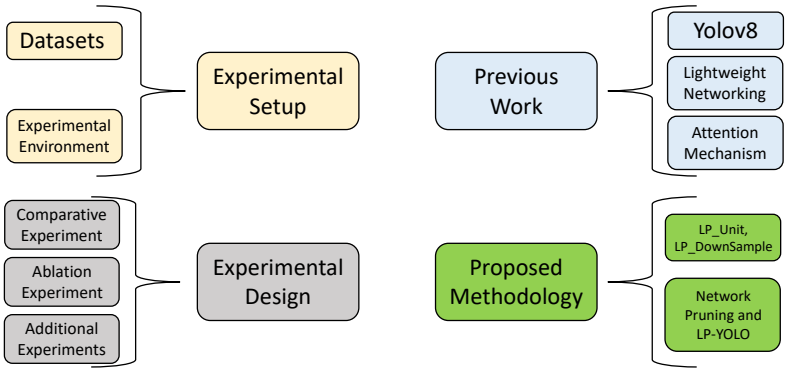


Figure 1. Material and Methods part's flow chart.

2.1. Experimental Setup

2.1.1. Datasets

For this study, we utilized the IP102 dataset, a comprehensive benchmark specifically designed for pest identification tasks. The dataset includes 102 categories of pests, comprising a total of 18,975 images that capture various developmental stages of pests, ranging from larvae to adults within these categories. To ensure effective model training and evaluation, the dataset was divided into distinct sets: 13,282 images for training, 1329 images for validation, and 4364 images for testing. The allocation of these images is graphically illustrated in Figure 2.

The IP102 dataset was obtained from https://github.com/xpwu95/IP102 (accessed on 19 August 2024), as initially described by Wu et al. in their paper on large-scale agricultural pest detection [18]. The dataset is openly accessible under the terms of its license.

Figure 2 illustrates the distribution of images across different categories in the IP102 dataset. The *X*-axis represents various categories within the dataset, where each number corresponds to a specific class. The *Y*-axis shows the number of images contained within each category. Notably, classes with a high number of images, such as Class 24 (aphids), Class 70 (blind bugs), and Class 101 (leafhoppers), have significantly larger datasets compared to others. Conversely, classes with the fewest images, such as Class 59 (grape mealybug), Class 60 (Colomerus vitis), and Class 75 (Phyllocoptes oleiverus ashmead), highlight the challenges of class imbalance within the dataset.

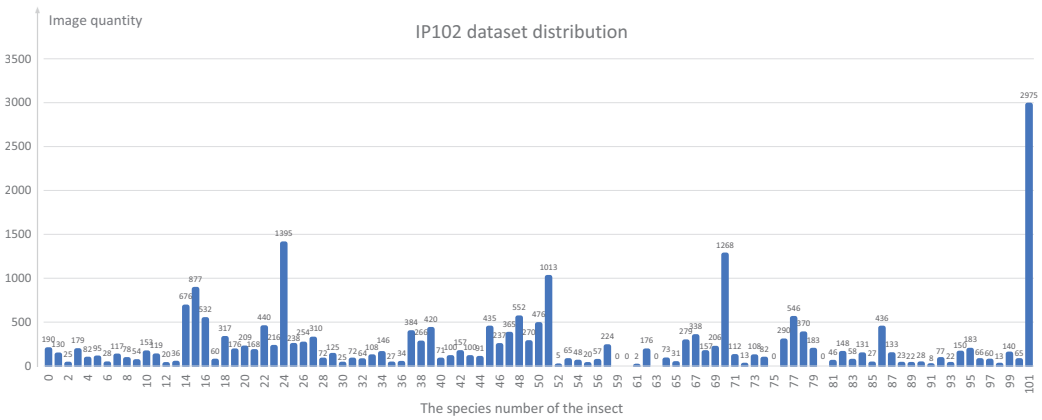


Figure 2. IP102 dataset distribution.

2.1.2. Experimental Environment

The experimental environment was configured with Python 3.8.0 and PyTorch 1.12.1. Computations were facilitated by a 3090 graphics card, operating under CUDA version 11.3.

During the training process of our proposed model, we employed early stopping as the termination condition, monitoring the validation loss to avoid overfitting. Training was

set to run for a maximum of 500 epochs; however, it was terminated earlier if the validation loss did not improve for 10 consecutive epochs. On average, the training process lasted around 250 epochs. This approach ensured that the model was sufficiently trained while preventing overfitting, allowing for a balanced evaluation of its performance.

2.2. Previous Work

2.2.1. YOLOv8

YOLO, an acronym for "You Only Look Once," represents a paradigm shift in real-time object detection within the domain of computer vision. By conceptualizing object detection as a regression problem, YOLO obviates the necessity for traditional sliding window and region proposal methodologies, thereby facilitating end-to-end model training. At the heart of YOLO's methodology is the division of the input image into a grid, within which bounding boxes and class probabilities are predicted for each grid cell. This innovative approach has not only accelerated detection speeds but also improved accuracy, rendering YOLO applicable across a multitude of real-time contexts, including autonomous vehicular navigation and surveillance systems. Through its iterative developments, YOLO has continually enhanced its accuracy, detection velocity, and architectural sophistication, thereby reaffirming its stature as a seminal algorithm in computer vision.

As the most recent iteration preceding YOLOv9, YOLOv8 augments its predecessors by integrating several significant advancements and optimizations. It utilizes a more complex network structure, Darknet-53 [19], to augment its feature extraction efficacy. Furthermore, YOLOv8 integrates novel methodologies such as the Spatial Attention Module (SAM) [20] and Path Aggregation Network (PAN) [21], aimed at refining detection precision and minimizing false positives. Its adoption of multi-scale detection techniques allows for the effective identification of objects across varying scales, thereby enhancing the model's robustness and adaptability to diverse object dimensions. YOLOv8 also introduces innovative strategies to advance detection accuracy while preserving the algorithm's hallmark rapid processing capability. Key enhancements in YOLOv8 include an evolved network architecture, optimized anchor box configurations, and sophisticated loss functions, all contributing to its exceptional performance metrics. This iteration continues to extend the frontiers of object detection technology, positioning it as an optimal solution for tasks requiring real-time, efficient computing. YOLOv8 is disseminated in five distinct models, n, s, m, l, and x, with each model representing a progressive increase in scale.

2.2.2. Lightweight Networking

Lightweight network architectures seek to balance the dual objectives of minimizing scale and computational complexity with the imperative of preserving model efficacy. Network pruning, an initial approach in this domain, strategically eliminates superfluous parameters and connections to reduce model dimensions and enhance inferential efficiency [22]. Despite its effectiveness in streamlining models, this method may inadvertently compromise accuracy. Network pruning is a technique used to optimize deep learning models by systematically removing redundant parameters and connections within the network. The primary goal of pruning is to reduce the model's size and computational complexity while maintaining, or even improving, its performance. This process involves identifying and eliminating weights that contribute minimally to the overall functionality of the network, thereby streamlining the model.

The benefits of network pruning are significant: it can lead to faster inference times, lower memory usage, and reduced energy consumption, making it especially valuable for deploying models on resource-constrained devices such as mobile phones and embedded systems. However, the pruning process must be carefully managed to avoid removing critical parameters, which could degrade the model's accuracy. When applied effectively,

network pruning can yield a more efficient model that performs comparably to its larger, unpruned counterpart. The objective function for pruning can be expressed as:

$$L_{\text{prune}} = L_{\text{task}} + \lambda \sum_{i=1}^{N} \|w_i\| \quad (1)$$

where L_{prune} represents the overall pruning loss, L_{task} denotes the task-specific loss function (e.g., classification or regression loss), λ is a regularization coefficient controlling the pruning strength, N is the total number of weights in the network, and w_i indicates the ith weight in the network, with $\|w_i\|$ typically representing the $L1$ norm (absolute value) of the weight w_i, encouraging sparsity.

In contrast, knowledge distillation harnesses the insights of larger, more complex teacher models to guide the training of more compact student models, aiming to curtail resource consumption while safeguarding performance levels [23]. This technique is beneficial for accelerating the education of student models and elevating their operational performance; however, it bears the potential drawbacks of promoting overfitting and escalating both training duration and computational demands.

The design of network architectures represents another pivotal avenue for achieving lightweight models. Prolonged exploration in neural network structures has unveiled opportunities for crafting architectures that substantially reduce parameter counts and computational requisites. This exploration has given rise to several seminal lightweight networks, including GhostNet [24], MobileNetV3 [25], EfficientDet [26], EfficientNetV2 [27], and ShuffleNet [28]. In addition to these manually crafted architectures, Neural Architecture Search (NAS) [29] offers an automated method for generating lightweight network designs. This approach utilizes Neural Architecture Search without training, enhancing the efficiency of the search process and reducing resource consumption. While NAS is recognized for its ability to efficiently design high-performance models, this method is still typically resource-intensive, and the resulting model structures may pose interpretative challenges.

In the present study, we initially apply the principles of network structure design to devise the LP_Unit and LP_DownSample modules. By integrating these modules into the existing framework, we derive the LP-YOLO(l) model, characterized by its reduced footprint. Subsequently, employing network pruning techniques, we significantly diminish network complexity, culminating in the development of the LP-YOLO(s) model.

To achieve a reduction in network complexity, we engineered the LP_Unit and LP_DownSample modules by employing depthwise separable convolution and channel shuffling techniques. Subsequently, the Efficient Channel and Spatial Attention (ECSA) mechanism was devised, integrating both spatial and channel attention to enhance processing efficiency. The integration of ECSA alongside the Convolutional Block Attention Module (CBAM) within the LP_Unit and LP_DownSample modules ensured heightened sensitivity to pivotal aspects of the input data, thereby augmenting detection precision. In the final phase, network optimization was pursued through the application of weight pruning and structured pruning techniques, aimed at refining the network's architecture, followed by meticulous fine-tuning during training. The ensuing sections delineate these measures in detail.

2.2.3. Attention Mechanism

To augment the network's ability to differentially prioritize pivotal information within the input data, thereby attenuating the focus on non-critical details, we integrate two distinct attention mechanism modules: our proposed Efficient Channel and Spatial Attention (ECSA) and the established Convolutional Block Attention Module (CBAM) [20]. Prior to detailing these innovations, an overview of foundational attention mechanism modules, characterized by their relatively simpler functionalities, is warranted.

The Convolutional Block Attention Module (CBAM) incorporates both channel and spatial attention mechanisms. The channel attention mechanism is expressed as:

$$M_c(X) = \sigma(\text{MLP}(\text{AvgPool}(X)) + \text{MLP}(\text{MaxPool}(X))) \tag{2}$$

The spatial attention mechanism is expressed as:

$$M_s(X) = \sigma(f^{7\times7}([\text{AvgPool}(X); \text{MaxPool}(X)])) \tag{3}$$

where X is the input feature map, σ denotes the sigmoid function, MLP represents a Multi-Layer Perceptron, and $f^{7\times7}$ indicates a convolution operation with a 7×7 kernel. The Efficient Channel Attention (ECA) module, depicted in Figure 3, initiates with a global average pooling operation applied to the input feature maps, transitioning their dimensions from (N, C, H, W) to (N, C, 1, 1). This process facilitates the aggregation of global contextual information, paralleling the operations inherent in the Squeeze-and-Excitation (SE) attention mechanism. Subsequently, the module computes the dimensions of the adaptive convolution kernel. This computation notably involves a 1D convolution aimed at capturing the interdependencies among channels, with the convolution kernel's dimensions determined by the following formula:

$$k = \left| \frac{log_2(c)}{\gamma} + \frac{b}{\gamma} \right| \tag{4}$$

In the described methodology, where c represents the number of channels, and γ and b serve as hyperparameters, the convolution kernel, derived from preceding calculations, is strategically applied to the feature maps. This application aims to assign distinct weights to each channel. Specifically, this allocation process is executed through element-wise multiplication between each channel's feature map and its corresponding weight, culminating in the generation of weighted feature maps. Subsequent to this step, the weighted feature maps undergo a normalization procedure, which meticulously adjusts the weight distribution across channels, thereby enhancing the model's attention mechanism.

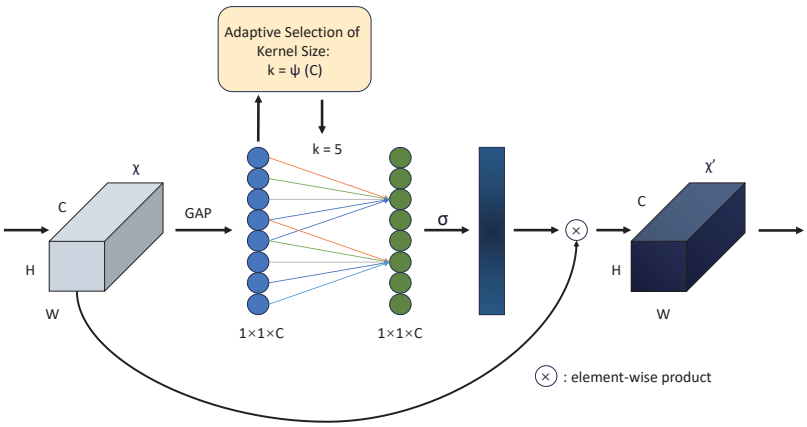


Figure 3. Diagram of ECA.

To augment the model's ability to isolate and prioritize critical features within input data, we integrate two principal attention mechanisms: the Channel Attention Module (CAM) and the Spatial Attention Module (SAM), as part of the Convolutional Block Attention Module (CBAM). The CAM assigns weights to different channels of a feature map based on their relevance, thereby emphasizing channels with informative content while suppressing those deemed less pertinent. SAM, in contrast, focuses on identifying and accentuating specific spatial locations within the feature map that are crucial for the task, by computing attention weights that spotlight significant regions. This dual-attention strat-

egy ensures a comprehensive analysis of both channel and spatial information, significantly enhancing the model's feature extraction and classification capabilities.

Furthermore, we introduce the Efficient Channel and Spatial Attention (ECSA) mechanism, which adeptly combines channel and spatial attentions. ECSA integrates Efficient Channel Attention (ECA) with spatial attention (SA), allowing the network to simultaneously refine the focus on both channel and spatial dimensions. Unlike the CBAM, which applies channel and spatial attention in sequence, ECSA performs both attention mechanisms in a unified process. This not only streamlines the attention mechanism but also enhances computational efficiency by avoiding redundant operations. While SENet focuses exclusively on channel attention, potentially missing spatial context, and the CBAM's sequential approach may introduce additional computational overhead, ECSA efficiently balances channelwise recalibration with spatial refinement. This results in improved accuracy and processing speed by directly addressing both types of information, achieving more effective feature representation with reduced computational cost.

The inclusion of these attention mechanisms aligns with the design principles of lightweight networks, prioritizing computational efficiency and minimal parameterization. As depicted in Figure 4, the architecture of the ECSA and CBAM exemplifies this strategic blend of attention-based analysis, ensuring that the network remains responsive to critical feature cues within constrained resource parameters, thus achieving heightened accuracy in detection tasks. To visually demonstrate the advantages of the CBAM attention mechanism we used and our proposed ECSA, we conducted the following heatmap experiments.

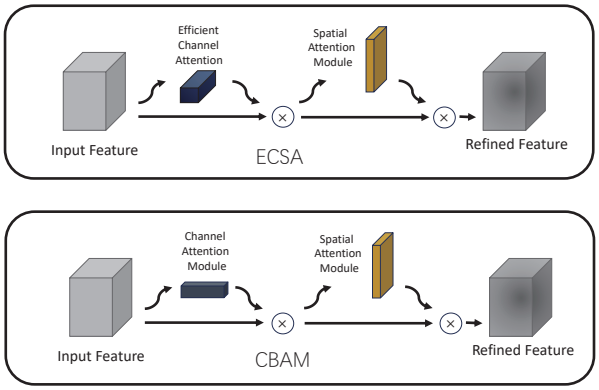


Figure 4. Diagram of ECSA and CBAM.

2.3. Proposed Methodology

2.3.1. LP_Unit, LP_DownSample

To achieve a balance between lightweight network architecture and precision retention, we developed two specialized modules: LP_Unit and LP_DownSample. These modules were strategically devised to substitute the conventional C2f and Conv modules within the original network's backbone, incorporating depthwise separable convolution as a cornerstone technology. Different from standard convolutional processes, depthwise separable convolution disaggregates the convolutional operation into two distinct stages: depthwise convolution and pointwise convolution. Initially, the input tensor undergoes feature extraction through depthwise convolution, followed by dimensionality reduction via pointwise convolution. This bifurcation significantly diminishes both computational complexity and parameter count, whilst preserving the model's accuracy. A comparative illustration of traditional convolution versus depthwise separable convolution is provided in Figure 5.

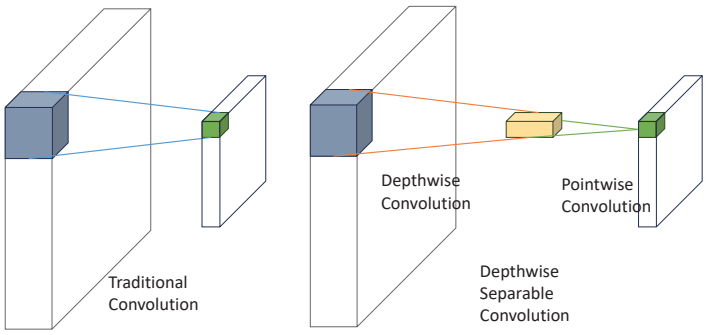


Figure 5. Contrast between traditional convolution and depthwise separable convolution.

Further augmenting our network's efficiency and model's representational capacity, the channel shuffle operation interlaces feature maps across various layers, thereby enriching feature diversity and bolstering the model's interpretability. This mechanism is crucial for promoting feature integration, mitigating overfitting risks, and streamlining model architecture. Executable through a straightforward linear transformation, channel shuffle ensures computational efficiency. The conceptual foundation of channel shuffle is depicted in Figure 6. In the channel shuffle operation, the feature map channels are shuffled to improve computational efficiency. The shuffle operation can be expressed as:

$$\text{Shuffle}(X) = \text{reshape}(X, (N, g, C/g, H, W)) \tag{5}$$

where X is the input feature map, N is the batch size, g is the number of groups, C is the number of channels, H is the height, and W is the width of the feature map.

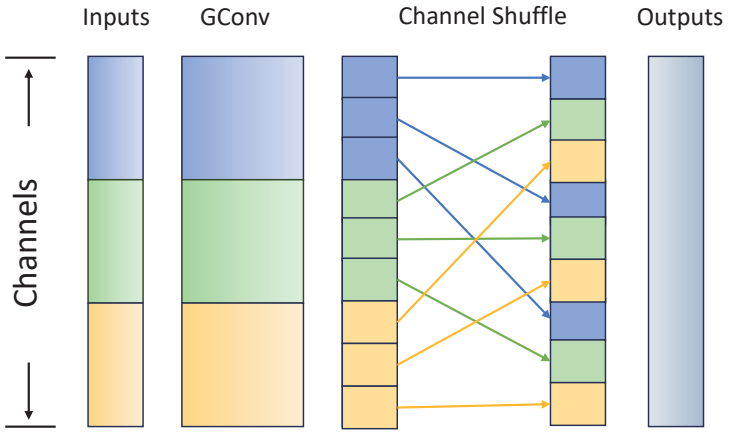


Figure 6. Diagram of channel shuffle. GConv donates group convolution, which is a type of convolution that divides the input channels into groups and applies convolution operations within each group independently. This approach can reduce model complexity and improve computational efficiency, making it suitable for efficient deep learning models.

In our design, the LP_Unit module, as illustrated in Figure 7, initiates with a channel split operation on the input feature map. Subsequently, one split undergoes processing via a 1×1 convolution layer, followed by a depthwise separable convolution (with a stride of 1), before merging with the counterpart split. The initial 1×1 convolution serves to diminish the channel count, thereby simplifying the ensuing depthwise separable convolution's complexity. After merging, the consolidated feature map is subjected to an Efficient Channel and Spatial Attention (ECSA) layer, aimed at accentuating pivotal information, consequently enhancing the model's accuracy. The process concludes with a channel shuffle operation to foster enhanced inter-channel interaction, ensuring a comprehensive feature

representation. Implementing the LP_Unit in lieu of YOLOv8's C2f module markedly diminishes parameter volume while preserving, if not augmenting, performance efficacy.

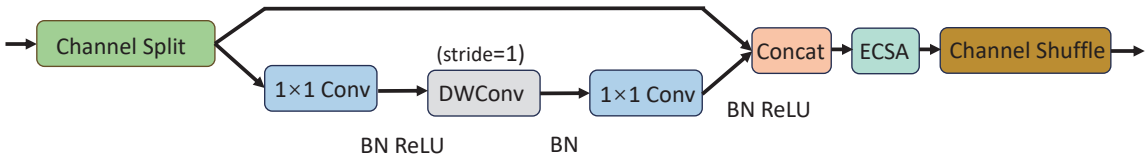


Figure 7. Structure of LP_Unit. DWConv denotes depthwise convolution. It and the following 1 × 1 convolution together form the depthwise separable convolution.

Moreover, the LP_DownSample module, depicted in Figure 8, introduces an innovative, lightweight downsampling approach intended to supplant the conventional Conv layer within the YOLOv8 backbone. This module channels the input feature map into dual pathways, bypassing the channel split. One pathway mirrors the LP_Unit's convolution branch, albeit with the depthwise separable convolution's stride adjusted to 2. Concurrently, the alternate pathway executes a depthwise separable convolution, also with a stride of 2. The amalgamation of outputs from both pathways undergoes further refinement through the Convolutional Block Attention Module (CBAM) and a subsequent channel shuffle operation, culminating in a feature map optimally poised for subsequent layers.

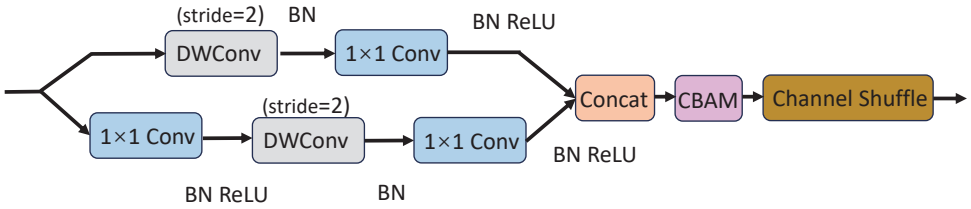


Figure 8. Structure of LP_DownSample. DWConv denotes depthwise convolution. It and the following 1 × 1 convolution together form the depthwise separable convolution.

In our analysis, we compared the LP_Unit and LP_DownSample modules with classic lightweight networks such as MobileNet and ShuffleNet. Our modules incorporate depthwise separable convolutions, a technique similar to that used in MobileNet, which is effective in reducing computational complexity. Both MobileNet and our modules employ batch normalization and ReLU activations to enhance training stability. Additionally, our LP_Unit integrates channel shuffle operations, akin to ShuffleNet, which improves feature propagation across grouped convolutions. Beyond these similarities, our approach introduces advanced attention mechanisms, specifically the CBAM and ECSA, which are not part of the standard MobileNet or ShuffleNet architectures. These attention modules significantly improve feature representation by focusing on crucial areas of the feature maps. Moreover, our use of residual connections in LP_Unit and LP_DownSample mirrors strategies from ResNet architectures, aiding in training deeper networks by addressing gradient vanishing issues. The downsampling strategy in LP_DownSample, utilizing a stride of 2, aligns with established techniques but is specifically optimized in our design to balance computational efficiency and feature extraction. Overall, while our modules share foundational elements with MobileNet and ShuffleNet, the incorporation of sophisticated attention mechanisms and flexible group configurations distinguishes our approach, offering notable advancements in lightweight neural network performance.

2.3.2. Network Pruning and LP-YOLO

In the quest to refine the YOLOv8 architecture for enhanced efficiency without compromising accuracy, we introduced the LP_Unit and LP_DownSample modules as strategic replacements for the C2f and Conv modules within the YOLOv8 backbone, extending

this modification to certain elements of the network's head as well. The resultant architecture, dubbed LP-YOLO(l), embodied our initial step towards a lightweight yet precise detection model.

Subsequent efforts to diminish the network's parameter footprint and streamline its complexity led to the adoption of network pruning strategies. Specifically, weight pruning was employed to reduce the output channels of LP_DownSample and LP_Unit—modules recurrently featured in the backbone—by fifty percent. This reduction not only trimmed the parameter count and computational demand but also eliminates less impactful weights. Concurrently, we augmented the repetition of the LP_Unit module within the backbone to enrich the network's capacity for feature extraction, enabling nuanced detail capture across varying spatial dimensions.

Furthermore, in addressing the network's neck and head components, we prioritized neuronal and junction pruning to alleviate the inherent complexity unsuitable for a streamlined network. This pruning simplified the interconnections within the neck and head, reducing the number of detection heads from three to two, thus optimizing inference speed and promoting network sparsity.

The pruned variant, designated as LP-YOLO(s), comprised five model sizes, n, s, m, l, and x, categorized according to their dimensions and parameter counts. Comparative visualizations of the YOLOv8n and LP-YOLO(s)n models, along with their respective module output parameters, as shown in Figures 9 and 10, illustrate the streamlined structure and reduced complexity of LP-YOLO(s)n, highlighting its lightweight design. Meanwhile, detailed specifications for the intermediate variant, LP-YOLO(l), are outlined in Table 1.

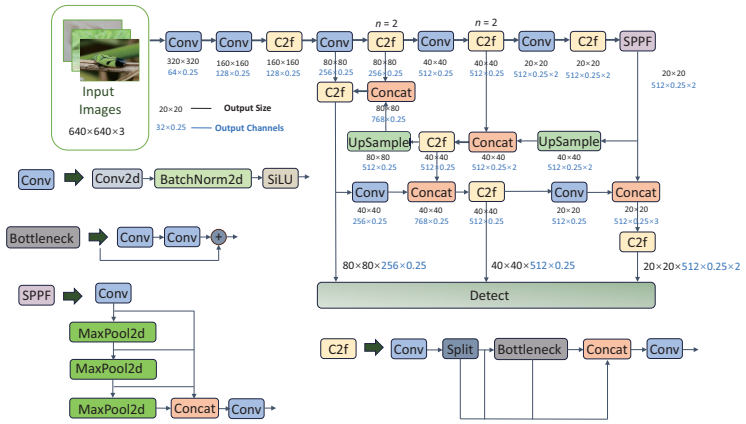


Figure 9. The overall architecture of YOLOv8n, where n = 2 means that the module is repeated twice.

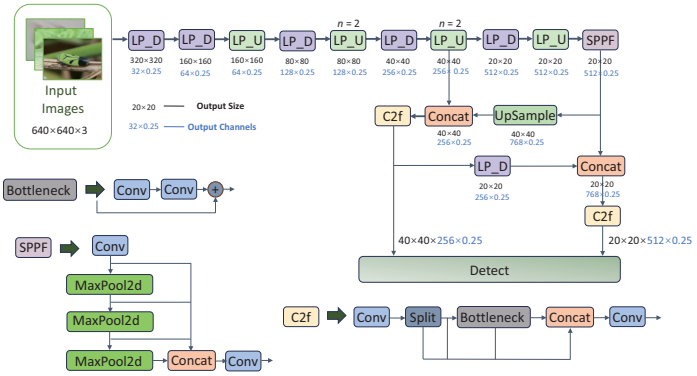


Figure 10. The overall architecture of LP-YOLOn(s)n, where n = 2 means that the module is repeated twice. Also, LP_U and LP_D refer to the LP_Unit shown in Figure 7 and the LP_DownSample shown in Figure 8, respectively.

Contrary to our initial expectations, LP-YOLO(s)n exhibited suboptimal performance during the training phase. This prompted an evaluation of potential remedial strategies, including knowledge distillation and fine-tuning. Given the inherent challenges associated with knowledge distillation—namely, the efficiency of knowledge transfer, operational complexity, and computational expenditure—we opted for fine-tuning as the more viable solution. Consequently, our approach entailed first training LP-YOLO(l), utilizing the resultant weights as a foundation for fine-tuning LP-YOLO(s). This strategy culminated in the pruned network achieving commendable convergence and enhanced performance metrics.

Table 1. Specification for the LP-YOLO(l).

Operator	Output Size	Projection	From	Output Channels	n
Images	640×640	-	-	3	-
LP_D	320×320	0	-1	$64 \times w$	1
LP_D	160×160	1	-1	$128 \times w$	1
LP_U	160×160	2	-1	$128 \times w$	$3 \times d$
LP_D	80×80	3	-1	$256 \times w$	1
LP_U	80×80	4	-1	$256 \times w$	$6 \times d$
LP_D	40×40	5	-1	$512 \times w$	1
LP_U	40×40	6	-1	$512 \times w$	$6 \times d$
LP_D	20×20	7	-1	$512 \times w \times r$	1
LP_U	20×20	8	-1	$512 \times w \times r$	$3 \times d$
SPPF	20×20	9	-1	$512 \times w \times r$	1
Upsample	40×40	10	-1	$512 \times w \times r$	1
Concat	40×40	11	$[-1, 6]$	$512 \times w \times (1 + r)$	1
C2f	40×40	12	-1	$512 \times w$	$3 \times d$
Upsample	80×80	13	-1	$512 \times w$	1
Concat	80×80	14	$[-1, 4]$	$768 \times w$	1
C2f	80×80	15	-1	$256 \times w$	$3 \times d$
LP_D	40×40	16	-1	$256 \times w$	1
Concat	40×40	17	$[-1, 12]$	$768 \times w$	1
C2f	40×40	18	-1	$512 \times w$	$3 \times d$
LP_D	20×20	19	-1	$512 \times w$	1
Concat	20×20	20	$[-1, 9]$	$512 \times w \times (1 + r)$	1
C2f	20×20	21	-1	$512 \times w \times r$	$3 \times d$
Detect	-	22	[15, 18, 21]	-	-

Illustrative of this developmental trajectory, Figure 11 delineates the entire process from the initial adaptation of YOLOv8 towards the lightweight LP-YOLO configuration, inclusive of network training phases.

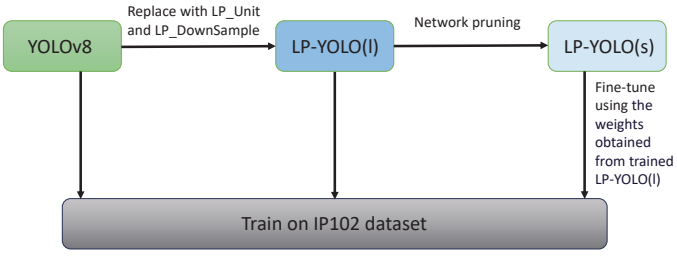


Figure 11. The whole process of lightweight YOLOv8 to obtain LP-YOLO and network training.

Table 2 delineates model performance after each major modification, vividly illustrating the changes in the model following each significant improvement.

Table 2. Model performance after each major modification.

	mAP50(%)	mAP50-95(%)	Param/M	FLOPs/G	FPS	Latency/ms
YOLOv8n	63.0	40.7	3.35	9.7	263.2	3.8
LP-YOLO(l)n	62.3	39.4	2.77	7.7	312.5	3.2
LP-YOLO(s)n	62.2	39.8	1.00	3.4	370.4	2.7

2.4. Experimental Design

2.4.1. Experimental Setup for Comparative Analysis

For alignment with mobile terminal device constraints and to streamline experimentation, we predominantly utilized the 'n' model size for both YOLOv8 and LP-YOLO frameworks. To ascertain the efficacy and lightweight properties of the developed LP-YOLO architecture, comparative analyses were conducted on the IP102 dataset, featuring LP-YOLO(s), LP-YOLO(l), and other leading lightweight networks. The YOLOv8n's backbone was substituted with five notable lightweight architectures: GhostNet, MobileNetV3-Small, MobileNetV3-Large, ShuffleNetV2-x1, and ShuffleNetV2-x2. This analysis also included LP-YOLO(s)n and LP-YOLO(l)n, alongside the scaled-down iterations of the YOLO series, namely, YOLOv5n and YOLOv8n, ensuring uniform experimental conditions and hyperparameter settings across all models.

In order to validate the efficacy of the newly developed LP-YOLO model for pest detection, it was benchmarked against an array of leading object detection frameworks within our experimental setup. This comparison included YOLOv5n, YOLOv8n, YOLOX-L, Faster R-CNN, SSD300, and RetinaNet [30], selected for their relevance and performance in state-of-the-art object detection tasks.

Additionally, we also trained, validated, and tested larger model sizes from the YOLOv8 series and our LP-YOLO(s) series, specifically using YOLOv8m, YOLOv8l, LP-YOLO(s)m, and LP-YOLO(s)l. This demonstrated that our lightweight strategy was equally applicable across different model sizes.

2.4.2. Ablation Study Procedure

To delineate the impact of various enhancement techniques on detection efficacy, ablation studies were conducted utilizing the IP102 dataset. The experimental configurations, outlined in Table 3, encompassed five distinct improvement strategies. The incorporation of LP_Unit and LP_DownSample was evaluated for their roles in substituting certain network structures. Additionally, the utilization of ECSA and CBAM attention mechanisms was examined for their integration at strategic model junctures. In scenarios devoid of LP_Unit and LP_DownSample, the ECSA and CBAM mechanisms were appended to the C2f and Conv modules, correspondingly. The foundational YOLOv8n model served as a baseline, devoid of the aforementioned enhancement techniques. The LP-YOLO(s)n model, embodying all five strategies, was juxtaposed against variants in the ablation framework. For clarity and ease of analysis, a distinct nomenclature was assigned to the models engaged in these ablation studies.

Table 3. Different improvement schemes.

	LP_Unit	LP_DownSample	ECSA	CBAM	Pruning
YOLOv8n					
YOLO-Unit	✓				
YOLO-DownSample		✓			
LP-no(l)n	✓	✓			
LP-ECSA(l)n	✓	✓	✓		
LP-CBAM(l)n	✓	✓		✓	
LP-YOLO(l)n	✓	✓	✓	✓	
LP-no(s)n	✓	✓			✓
LP-ECSA(s)n	✓	✓	✓		✓
LP-CBAM(s)n	✓	✓	✓	✓	✓
LP-YOLO(s)n	✓	✓	✓	✓	✓

Note: The symbol "✓" indicates that the corresponding module is included in the model on the left.

2.4.3. Additional Experiments

Given our objective to enhance agricultural productivity through accurate pest detection on mobile devices, we conducted additional experiments to further validate the performance and robustness of LP-YOLO. We performed a visual comparison of heatmaps generated by YOLOv8n, LP-YOLO(l)n, and LP-YOLO(s)n models to assess the impact of our Efficient Channel and Spatial Attention (ECSA) mechanism and CBAM on feature focus. Using Grad-CAM, we analyzed heatmaps from the IP102 dataset to compare how these models prioritized relevant pest features. Additionally, we tested the robustness of LP-YOLO(s)n by introducing various noise types, including Gaussian and salt-and-pepper noise, into the input images, and evaluated the model's performance by measuring changes in *mAP*50, *mAP*50-95, and *FPS*. Finally, we applied LP-YOLO(s)n to real-world video data captured in agricultural fields to validate its adaptability and effectiveness in uncontrolled environments, ensuring its practical application in mobile-based pest detection.

3. Results and Discussion

3.1. Evaluation Indicators

In evaluating the efficacy of object detection models within the domain of deep learning, several performance metrics are pivotal: recall, precision, average precision (AP), mean average precision (mAP), the total number of model parameters, floating-point operations per second ($FLOPs$), frame rate (FPS), and response time ($latency$).

The mathematical formulations for $precision$, $recall$, AP, and mAP are delineated in Equations (6)–(9). In these equations, TP (True Positives) represents the number of correctly detected objects, FP (False Positives) refers to the number of incorrectly detected objects, and FN (False Negatives) denotes the objects that were missed by the model. Average precision (AP) is calculated as the area under the precision–recall curve (Equation (8)), and mean average precision (mAP) is the mean of the AP values across all classes, providing an overall measure of the model's accuracy, as shown in Equation (9).

$$P = \frac{TP}{TP + FP} \tag{6}$$

$$R = \frac{TP}{TP + FN} \tag{7}$$

$$AP = \int_0^1 P(R)d(R) \tag{8}$$

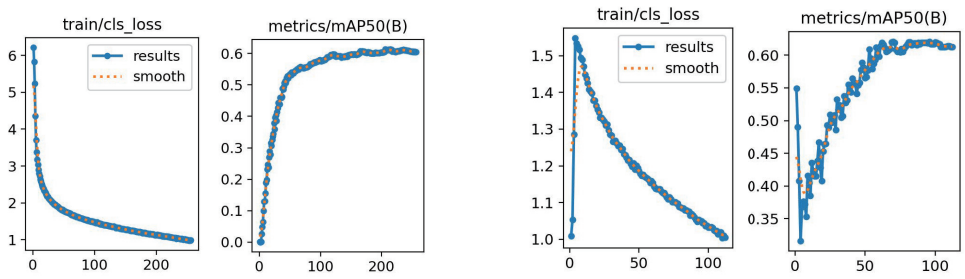
$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{9}$$

The assessment of lightweight object detection algorithms conventionally employs a suite of critical metrics. Model complexity is gauged by the number of parameters, while computational intensity is quantified through floating-point operations per second ($FLOPs$). FPS measures the algorithm's real-time processing capability, whereas $latency$ delineates the temporal delay from input reception to output generation, a factor of paramount importance for systems requiring swift responsiveness.

3.2. Model Training and Fine-Tuning Process

As seen on the Figure 12a, LP-YOLO(l)n exhibited a stable decline in loss values during training, approaching the minimum value, while mAP and other object detection metrics steadily increased towards their maximum values. The training required approximately 250 epochs in total. The training processes of other models during the experiment were similar to this. As shown in the Figure 12b, during the fine-tuning process of LP-YOLO(s)n, the early stage of training exhibited a slight performance drop in mAP for LP-YOLO(s)n. This initial decline could be attributed to the model's adjustment period, where the pruned model structure was adapting to the pre-trained weights. Even though the parameters were

consistent with those of the original, non-pruned model, the reduced complexity required the model to recalibrate its internal representations. This phenomenon was particularly evident in the early epochs. As the training progressed, typically up to around 250 epochs, the model began to stabilize, and the performance improved steadily. The use of pre-trained weights from LP-YOLO(l)n ensured that LP-YOLO(s)n could recover from the initial drop and converge effectively, as evidenced by the smooth rise in mAP. The final result was a model that, despite being significantly pruned, retained high accuracy with a reduced parameter count and enhanced FPS.



(**a**) LP-YOLO(l)n training process (**b**) LP-YOLO(s)n fine-tuning training process

Figure 12. Decline curve of the model training loss and the change curve of the mAP of (**a**) LP-YOLO(l)n and (**b**) LP-YOLO(s)n during training on the IP102 dataset (introduced in Section 2.1.1). The "results" line shows the raw values per epoch, while the "smooth" line displays a smoothed version to highlight overall trends.

3.3. Comparison and Ablation Results

Table 4 delineates the outcomes of the comparative experiments with other lightweight networks, substantiating the LP-YOLO's performance and lightweight attributes. Notably, YOLOv8n and LP-YOLO(s)n exhibited parameter counts of 3,342,336 and 996,147, respectively, indicating a 70.2% reduction in parameters for LP-YOLO(s)n relative to YOLOv8n. When benchmarked against models employing MobileNetV3 and ShuffleNetV2 backbones, our LP-YOLO(l)n and LP-YOLO(s)n variants not only exemplified enhanced lightweight features but also maintained competitive mAP values, rendering them exceptionally suited for mobile device applications. The analysis revealed LP-YOLO(l)n's competitive edge over other lightweight networks and LP-YOLO(s)n's superior lightness after pruning, without compromising mAP performance.

Table 4. Comparison of different lightweight networks.

Method	Backbone	mAP50(%)	Param/M	FLOPs/G	FPS	Latency/ms
YOLOv5n	-	62.4	2.84	8.7	277.8	3.6
YOLOv8n	-	63.0	3.34	9.6	263.2	3.8
YOLOv8n	GhostNet	58.6	2.21	5.8	333.3	3.0
YOLOv8n	MobileNetV3-L	61.5	2.37	6.2	333.3	3.0
YOLOv8n	MobileNetV3-S	59.4	1.66	3.9	370.4	2.7
YOLOv8n	ShuffleNetV2-x2	62.8	2.81	7.8	303.0	3.3
YOLOv8n	ShuffleNetV2-x1	60.3	1.95	4.7	344.8	2.9
LP-YOLO(l)n	-	62.3	2.77	7.7	312.5	3.2
LP-YOLO(s)n	-	62.2	1.00	3.4	370.4	2.7

The outcomes of the comparative experiments with state-of-the-art object detect networks are encapsulated in Table 5. Upon evaluation, it was observed that, relative to the previously lightest model, YOLOv5n, our LP-YOLO established a new benchmark for model efficiency. Specifically, LP-YOLO(l)n exhibited a reduction of 0.25 million in parameter count, 1.0 billion in floating-point operations (FLOPs), and a decrease of 0.4 ms in latency. Building upon this foundation, LP-YOLO(s)n further advanced the lightweight design initiated by LP-YOLO(l)n, achieving a 64.1% reduction in parameters alongside notable decreases in both FLOPs and latency metrics. Performance-wise, LP-YOLO(s)n demonstrated

superior mean average precision (mAP50) compared to Faster R-CNN, SSD300, and RetinaNet, while closely trailing YOLOv8n—the model with the highest mAP50 value—by a marginal 0.8%. These comparative experiments unequivocally affirmed LP-YOLO's advantage as a highly efficient lightweight object detection model. In addition, the data demonstrated that both LP-YOLO(s)m and LP-YOLO(s)l excelled in maintaining a strong balance between accuracy and efficiency when compared to larger models like YOLOv8m and YOLOv8l. Despite using significantly fewer parameters—7.54 M for LP-YOLO(s)m and 17.15 M for LP-YOLO(s)l—these models achieved competitive mAP50 scores of 75.4% and 77.1%, respectively. Moreover, both models delivered superior FPS and lower latency, with LP-YOLO(s)m reaching 174.3 FPS and LP-YOLO(s)l achieving 123.2 FPS. This performance underscored the effectiveness of our lightweight strategy, proving its suitability across various model sizes while ensuring a high processing speed and reduced computational load.

While these larger model sizes are indeed well suited for complex scenarios and intensive tasks, it is also important to consider potential limitations. Although LP-YOLO(s)n maintained competitive mAP values even with a significant reduction in parameters, there may be edge cases where this reduction could negatively impact performance. For example, in highly complex or densely populated scenes, the simplified model might struggle to capture fine details, potentially leading to decreased detection accuracy or robustness under certain conditions. Addressing these potential limitations and exploring strategies to mitigate them could provide a more comprehensive understanding of the model's performance across diverse real-world applications.

Table 5. Comparison of different algorithms for object detection.

	mAP50(%)	Param/M	FLOPs/G	FPS	Latency/ms
YOLOv8m	78.4	25.92	79.4	83.2	19.6
YOLOv8l	80.6	43.71	165.8	50.7	20.7
LP-YOLO(s)m	75.4	7.54	27.8	174.3	9.7
LP-YOLO(s)l	77.1	17.15	64.3	123.2	13.4
YOLOv5n	62.4	2.84	8.7	277.8	3.6
YOLOv8n	63.0	3.34	9.6	263.2	3.8
YOLOX-L	62.4	4.06	10.3	263.2	3.8
Faster-RCNN	54.9	12.01	23.5	88.5	11.3
SSD300	56.3	6.36	15.6	238.1	4.2
RetinaNet	59.6	9.79	17.2	138.9	7.2
LP-YOLO(l)n	62.3	2.77	7.7	312.5	3.2
LP-YOLO(s)n	62.2	1.00	3.4	370.4	2.7

The outcomes of the ablation experiments are encapsulated in Table 6. The analysis revealed that the integration of ECSA and CBAM attention mechanisms significantly augmented the mAP, with a minimal increase in parameter count, underscoring their efficacy in enhancing detection accuracy. Conversely, LP_Unit and LP_DownSample were identified as potent measures for ameliorating network complexity. Specifically, relative to the base YOLOv8n model, implementations designated as YOLO-Unit and YOLO-Down exhibited parameter reductions of 9.0% and 10.2%, respectively. Furthermore, the merits of network pruning and fine-tuning training were corroborated. Notably, LP-YOLO(s)n, in comparison to LP-YOLO(l)n, demonstrated a negligible decrement in mAP50 of 0.1%, alongside a substantial reduction in parameters of 63.9% and an enhancement in FPS of 15.6%.

Table 6. Ablation results of different methods.

	mAP50(%)	mAP50-95(%)	Param/M	FLOPs/G	FPS	Latency/ms
YOLOv8n	63.0	40.7	3.35	9.7	263.2	3.8
YOLO-Unit	62.5	40.2	3.04	7.9	277.8	3.6
YOLO-Down	62.2	40.8	3.00	8.4	285.7	3.5
LP-no(l)n	61.3	39.5	2.77	7.6	312.5	3.2
LP-ECSA(l)n	61.4	38.8	2.77	7.6	285.7	3.5
LP-CBAM(l)n	61.4	39.0	2.78	7.7	294.1	3.4
LP-YOLO(l)n	62.3	39.4	2.77	7.7	312.5	3.2
LP-no(s)n	61.2	38.5	0.99	3.3	344.8	2.9
LP-ECSA(s)n	61.7	39.0	1.00	3.3	357.1	2.8
LP-CBAM(s)n	62.0	39.2	0.99	3.4	370.4	2.7
LP-YOLO(s)n	62.2	39.8	1.00	3.4	370.4	2.7

3.4. Detection Results Visualization

This segment of the study illustrates the LP-YOLO model's efficacy through visual comparisons of prediction outcomes on the IP102 dataset. Notably, the LP-YOLO model demonstrated commendable predictive performance. Figure 13 juxtaposes partial prediction outcomes from YOLOv8n, LP-YOLO(l)n, and LP-YOLO(s)n against the ground truth for several images, showcasing the model's accuracy.

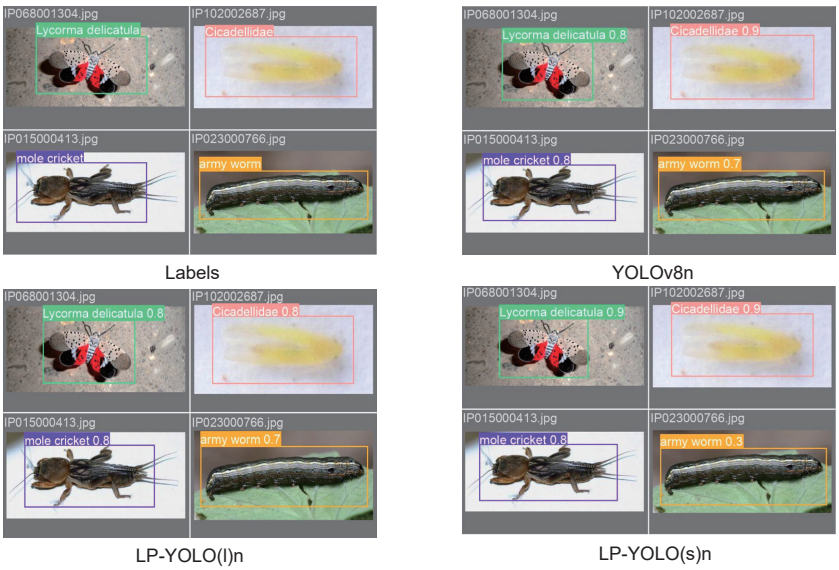


Figure 13. Pictures with labels and pictures of detection results.

However, instances of mismatches between predictions and actual labels were observed, exemplified in Figure 14. An analysis into the causes of these discrepancies revealed several contributing factors:

1. The color morphology of insects is similar to the background, which leads to the recognition failure, among which (a) and (b) in Figure 14 are typical representatives.
2. Insects, as the object of target recognition, have certain particularity. Some insects have great morphological changes during the growth process from larva to adult; Phyllocnistis Citrella Stainton (c) in Figure 14 is such an insect.
3. There are large differences among some species of insects, such as the blister beetles in Figure 14d.

We conducted detailed comparative experiments on YOLOv8n and LP-YOLO(s)n using the IP102 dataset. By comparing the prediction results of these two models, we found no significant differences in their overall performance, as both demonstrated high levels of detection accuracy and stability. Figures 15 and 16 depict some sample detection results, indicating that both YOLOv8n and LP-YOLO(s)n performed well in pest detection tasks.

However, LP-YOLO(s)n exhibited significantly fewer parameters and lower complexity compared to YOLOv8n, highlighting the advantage of our proposed LP-YOLO model.

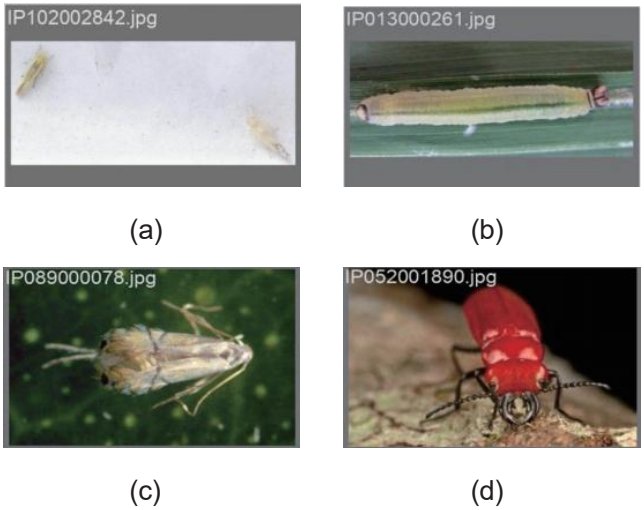


Figure 14. Examples of Prediction Failures Due to Insect and Background Similarity (**a**,**b**), Growth-Stage Morphological Changes in Phyllocnistis citrella (**c**), and Species Differences in Blister Beetles (**d**).

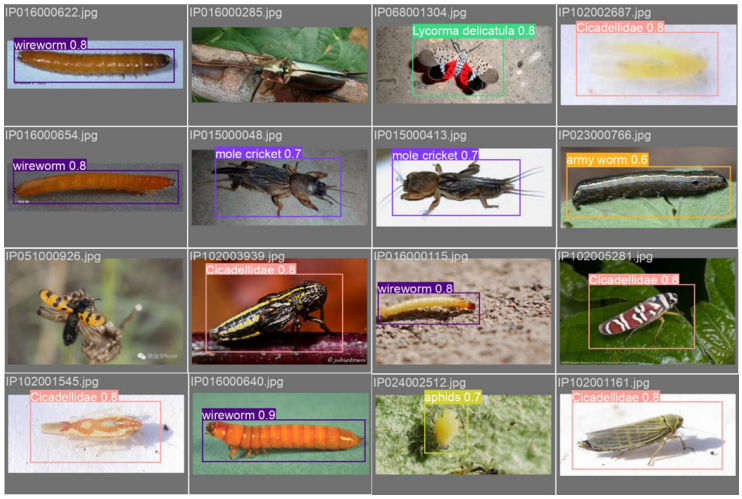


Figure 15. Detection results on YOLOv8n.

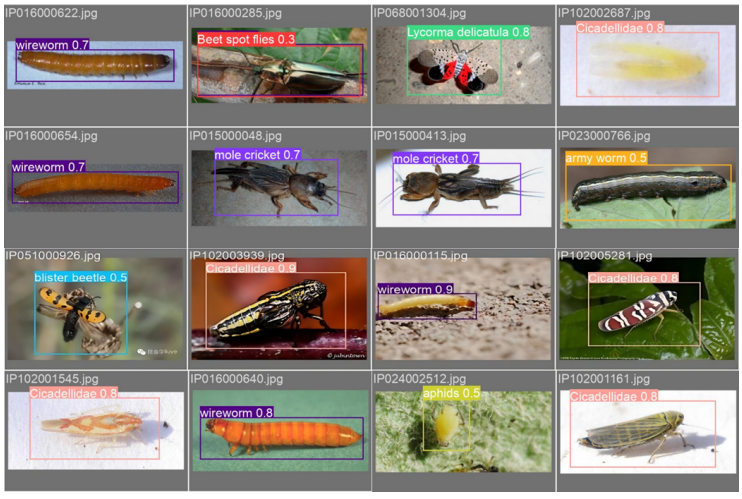
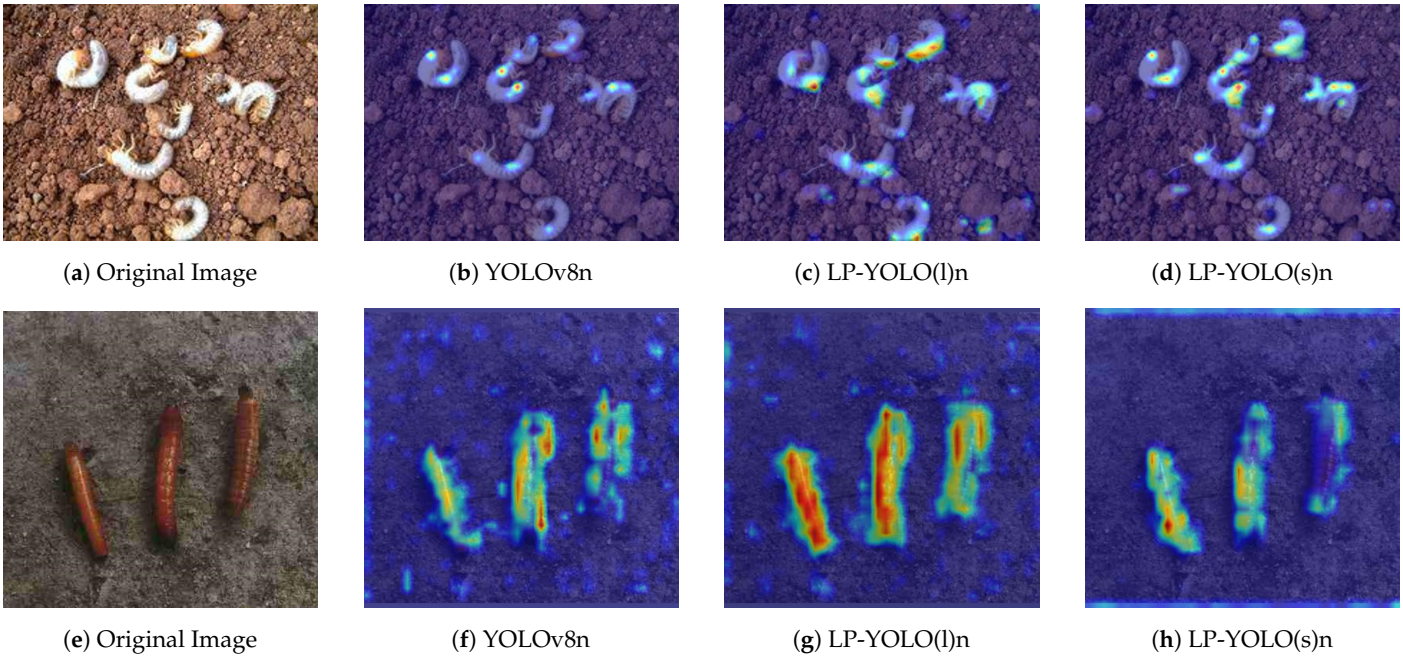


Figure 16. Detection results on LP-YOLO(s)n.

3.5. Additional Experimental Validation of LP-YOLO

3.5.1. Visual Comparison of Heatmaps

In this experiment, we conducted a comparative analysis of heatmaps generated by different models, including YOLOv8n, LP-YOLO(l)n, and LP-YOLO(s)n, to evaluate the effectiveness of the Efficient Channel and Spatial Attention (ECSA) mechanism integrated into LP-YOLO. We employed Grad-CAM to generate heatmaps for a set of images from the IP102 dataset. As shown in Figure 17, these heatmaps visually represent the regions within an image that the model deems important for making predictions. By comparing these heatmaps across the three models, we aimed to determine the degree of focus each model placed on the relevant features of the pests.



(**a**) Original Image (**b**) YOLOv8n (**c**) LP-YOLO(l)n (**d**) LP-YOLO(s)n

(**e**) Original Image (**f**) YOLOv8n (**g**) LP-YOLO(l)n (**h**) LP-YOLO(s)n

Figure 17. Comparison of heatmaps across two images. The first row displays the original image (**left**), YOLOv8n heatmap (**second**), LP-YOLO(l)n heatmap (**third**), and LP-YOLO(s)n heatmap (**fourth**) for the first image. The second row follows the same order for the second image.

The results showed that the heatmaps generated by LP-YOLO(l)n exhibited a significant improvement in attention focus compared to the baseline YOLOv8n model. Specifically, LP-YOLO(l)n demonstrated a more concentrated activation on the pest regions, whereas YOLOv8n's activations were more dispersed. While LP-YOLO(s)n was slightly less focused than LP-YOLO(l)n, it still outperformed YOLOv8n, indicating that the ECSA mechanism effectively enhances feature extraction, leading to more precise localization of pests. These findings suggest that the integration of the ECSA mechanism in LP-YOLO(l)n substantially improves the model's ability to focus on critical regions, thereby enhancing its detection performance and making it better suited for scenarios where accurate localization of small objects, such as pests, is crucial.

3.5.2. Robustness to Noise

To assess the robustness of the LP-YOLO(s)n model, we introduced various types of noise into the input images from the IP102 dataset and visually analyzed the model's performance under these degraded conditions. The noise types applied included Gaussian noise with sigma values of 25 and 50, as well as salt-and-pepper noise with salt_prob and pepper_prob set to 0.05 and 0.1, respectively. Figure 18 illustrates how these noise types affect the appearance of the input images.

(**a**) (**b**) (**c**) (**d**) (**e**)

Figure 18. Images illustrating the impact of various types of noise: (**a**) original image, (**b**) Gaussian noise with a variance of 25, (**c**) Gaussian noise with a variance of 50, (**d**) salt-and-pepper noise with salt_prob and pepper_prob set to 0.05, and (**e**) salt-and-pepper noise with salt_prob and pepper_prob set to 0.1.

Despite these visual degradations, the results presented in Table 7 demonstrate that LP-YOLO(s)n maintained a relatively stable performance. Specifically, with Gaussian noise (variance 25), the mAP50 dropped by only 1.2%, and with a variance of 50, the drop was 3.5%. For salt-and-pepper noise, the mAP50 decrease was 1.8% at the lower noise level and 8.8% at the higher level. These findings indicate that LP-YOLO(s)n is highly resilient to various types of noise, with only a slight degradation in performance even under more severe noise conditions. This robustness to noise, as quantified in Table 7, is a key advantage for practical applications, where environmental factors may introduce noise into the input data. The model's ability to maintain high accuracy despite these challenges underscores its reliability and makes it a valuable tool for pest detection in less-than-ideal conditions.

Table 7. Performance of LP-YOLO(s)n across different noise conditions.

Noise Type	mAP50(%)	mAP50-95(%)	FPS
No noise	62.2	39.8	370.4
Gaussian noise (25)	61.0	38.6	370.2
Gaussian noise (50)	58.7	36.2	360.3
Salt-and-pepper noise (0.05)	60.4	36.8	364.9
Salt-and-pepper noise (0.1)	53.4	31.6	354.3

3.5.3. Real-World Application: Predicting Insect Pests in Field-Captured Videos

In the final experiment of our study, we assessed the effectiveness of the LP-YOLO model in real-world agricultural environments by deploying it to detect and classify pests in videos captured under actual field conditions. These videos, featuring a high density of various pests, were processed using the LP-YOLO(s)n algorithm, segmented into frames for a detailed analysis.

Figure 19 illustrates pest detection in a group setting. Across multiple frames, LP-YOLO(s)n demonstrated robust detection capabilities, accurately identifying pests even in densely populated scenarios. Despite the proximity and potential overlap of subjects within these groups, the model assigned confidence scores that reflected high reliability in distinguishing individual pests. This precision is crucial for accurately quantifying pest populations, a key factor in effective pest management strategies.

Figure 19. Insects in a social state.

Figure 20 depicts pest detection during critical biological phases, such as reproduction and molting. The model effectively recognized and classified pests through these stages, with assigned confidence scores validating its ability to detect pests even when they exhibited less distinct visual features. Understanding these dynamics is also vital for targeted pest control interventions, particularly during vulnerable phases of the pest life cycle.

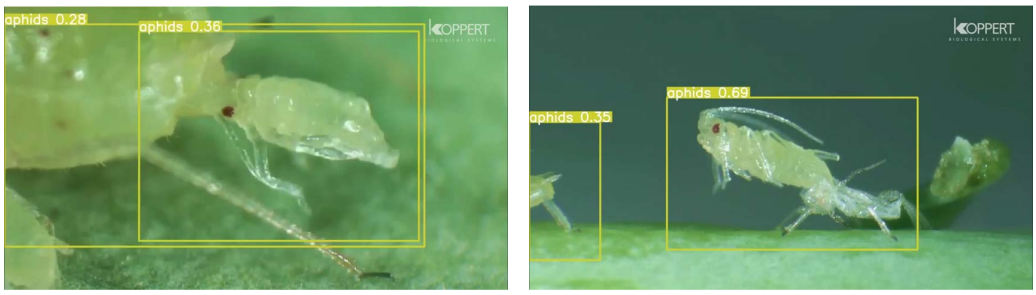


Figure 20. Individual insects at important life stages.

In this uncontrolled outdoor environment, the performance of LP-YOLO(s)n was consistent with observations made under controlled laboratory conditions, showcasing the model's adaptability to variable environmental factors such as changes in lighting, motion blur, and complex backgrounds, and proving its robustness in field applications. This field application of LP-YOLO(s)n highlights its practical applicability and effectiveness for pest detection and classification in real-world agricultural settings. The model's high detection rate and consistent performance across different pest stages and groupings confirm its suitability as a valuable tool in agricultural pest management. By facilitating accurate and timely pest identification, LP-YOLO(s)n supports enhanced decision-making for pest control strategies, ultimately contributing to improved crop protection and yield. We provide the generated video file with detection boxes, which can be accessed at https://drive.google.com/file/d/1cJxPEKRlRTMJpBvGnoYVyGLsuPIhsM8g/view?usp=sharing (accessed on 21 August 2024).

3.6. Discussion

We experimentally demonstrated the superiority of our proposed model in terms of accuracy and lightweight design. During the process of predicting images in the dataset with the trained model and comparing these predictions with the actual labels, we analyzed potential reasons for prediction errors. Next, we plan to compare our model with similar research results to comprehensively assess our model and address key issues in pest detection.

First, we compared our model with studies that also utilized the IP102 dataset, such as the work by Zhang et al. [14]. We were pleased to find that our model outperformed theirs in both mAP and lightweight design. Our analysis revealed that part of our model's accuracy advantage was due to our use of the YOLOv8 base model, which was superior to their YOLOX base model. Additionally, we applied fine-tuning during training, which they did not. In terms of lightweight design, we developed a lightweight model and performed network pruning, whereas their focus was not on model reduction. Nevertheless, their research provided valuable insights. Their Cross-Layer Transformer significantly enhanced the flow of information and accuracy, which could serve as a reference and improvement for future work in this field. The model by Wei et al. [31], which was also based on the IP102 dataset, achieved an mAP50 of 0.671, slightly higher than our 0.622. However, their primary focus was on improving accuracy, so their model was naturally less optimized for lightweight design compared to ours.

In Table 8, we compare the performance of LP-YOLO(s)n with other state-of-the-art models, specifically CLT-YOLOX [14], AEC-YOLOv8n [31], and C3M-YOLO [32], which

have also been evaluated on the IP102 dataset. Notably, AEC-YOLOv8n achieves the highest mAP50 at 67.1%, outperforming the other models in terms of detection accuracy. However, LP-YOLO(s)n demonstrates a significant advantage in terms of model efficiency. With a parameter count of just 1.00M, LP-YOLO(s)n is the most lightweight among the compared models, resulting in an impressive FPS of 83.2, which is higher than that of CLT-YOLOX and slightly lower than C3M-YOLO. This comparison highlights the trade-off between detection accuracy and model efficiency, where LP-YOLO(s)n strikes a balance by offering competitive accuracy while significantly reducing the computational burden.

Table 8. Performance comparison of LP-YOLO(s)n with other object detection models on the IP102 dataset.

	mAP50(%)	Param/M	FPS
CLT-YOLOX [14]	57.7	10.5	61.9
AEC-YOLOv8n [31]	67.1	7.8	-
C3M-YOLO [32]	57.2	7.1	97.0
LP-YOLO(s)n (ours)	62.2	1.00	83.2

Next, we compared our model with other studies in the pest detection field. For instance, Sun et al. [33] used a dataset of 1810 images they collected. Our model's parameter count was less than half of theirs, showcasing superior lightweight characteristics, which was a result of our proposed lightweight model and network pruning techniques. However, their mAP50 metric reached 0.939, which was higher than our 0.622. This difference was partly due to their proposed Universal Inverted Bottleneck (UIB) block, as well as the dataset differences. Their dataset had fewer images and pest species, with backgrounds consisting solely of spotted tomato leaves, making it easier to achieve a higher mAP.

Finally, we reviewed papers in the pest detection field that did not propose new network models. For example, Guo et al. [34] summarized widely used public datasets for pest detection and recognition and discussed various algorithms proposed in recent years. That paper provided a comprehensive overview and outlook on pest detection, offering valuable resources for researchers in the field. Additionally, the paper by Appiah et al. [35] introduced a novel mobile application powered by AI models that provided real-time pest and disease identification services. This was an excellent application-oriented work that effectively aided practical production efforts.

4. Conclusions

This study introduced LP-YOLO, an innovative pest detection methodology optimized for mobile platforms. To elevate detection capabilities, we integrated two sophisticated attention mechanisms: ECSA and CBAM. Further advancing the network's efficiency, the LP_Unit and LP_DownSample modules were developed to streamline the network architecture. Subsequent network pruning and meticulous fine-tuning were employed to significantly curtail the parameter volume. Comparative analyses against existing lightweight networks and diverse object detection algorithms, alongside methodical ablation studies, underscored LP-YOLO's enhanced performance metrics. Notably, LP-YOLO(s)n achieved a minimal reduction in mAP of only 0.8% relative to YOLOv8n, while concurrently facilitating a substantial 70.2% decrease in parameter count and a 40.7% improvement in FPS.

Additionally, we conducted experiments for a visual comparison of heatmaps to compare our model's focus on image features with other models, robustness to noise to test the model's robustness, and Predicting insect pests in field-captured videos to demonstrate the model's functionality and validate its applicability to real-world scenarios.

Looking ahead, we propose targeted avenues for deepening the research into deep learning-based pest detection. Given the pronounced morphological variations exhibited by insects throughout their life cycles and the considerable diversity within species, these factors can potentially compromise detection accuracy. Consequently, future dataset compilation ef-

forts should meticulously encompass these variances, ensuring the inclusion of comprehensive insect representations across all developmental stages and intra-species variations.

Author Contributions: Conceptualization, Y.Y. and H.W.; data curation, Y.Y., Q.Z., H.W. and K.L.; methodology, Y.Y., Q.Z., J.L. and D.L.; validation, Y.Y., Q.Z. and D.L.; visualization, Y.Y., Q.Z., H.W. and K.L.; writing—original draft, Y.Y., Q.Z. and H.W.; resources, K.L. and L.Z.; funding acquisition, L.Z.; supervision, L.Z., J.L. and D.L.; writing—review and editing, K.L., L.Z., J.L. and D.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (61806024, 62206257), the Jilin Province Science and Technology Development Plan Key Research and Development Project (20210204050YY), the Wuxi University Research Start-up Fund for Introduced Talents (2023r004, 2023r006), the National Training Program of Innovation and Entrepreneurship for Undergraduates (S202310698140), the Changchun Science and Technology Development Program (grant number 21ZGN26), and the Jilin Province Science and Technology Development Program (grant number 20230508026RC).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The IP102 dataset used to support the findings of this study was deposited in the PRCV2019 repository (DOI: 10.1109/CVPR.2019.00899). All the data mentioned in the paper are available through the corresponding author.

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

References

1. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
2. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
3. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. [CrossRef]
4. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016*; Proceedings Part I 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
5. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
6. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
7. Cui, L.; Ma, R.; Lv, P.; Jiang, X.; Gao, Z.; Zhou, B.; Xu, M. MDSSD: Multi-scale deconvolutional single shot detector for small objects. *arXiv* **2018**, arXiv:1805.07009.
8. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
9. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
10. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. Yolov10: Real-time end-to-end object detection. *arXiv* **2024**, arXiv:2405.14458.
11. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
12. Srinivas, A.; Lin, T.Y.; Parmar, N.; Shlens, J.; Abbeel, P.; Vaswani, A. Bottleneck transformers for visual recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 16519–16529.
13. Chen, C.F.R.; Fan, Q.; Panda, R. Crossvit: Cross-attention multi-scale vision transformer for image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Virtual, 11–17 October 2021; pp. 357–366.
14. Zhang, L.; Cui, H.; Sun, J.; Li, Z.; Wang, H.; Li, D. CLT-YOLOX: Improved YOLOX Based on Cross-Layer Transformer for Object Detection Method Regarding Insect Pest. *Agronomy* **2023**, *13*, 2091. [CrossRef]
15. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv* **2021**, arXiv:2107.08430.
16. Zhu, L.; Li, X.; Sun, H.; Han, Y. Research on CBF-YOLO detection model for common soybean pests in complex environment. *Comput. Electron. Agric.* **2024**, *216*, 108515.

17. Xu, W.; Xu, T.; Thomasson, J.A.; Chen, W.; Karthikeyan, R.; Tian, G.; Shi, Y.; Ji, C.; Su, Q. A lightweight SSV2-YOLO based model for detection of sugarcane aphids in unstructured natural environments. *Comput. Electron. Agric.* **2023**, *211*, 107961.
18. Wu, X.; Zhan, C.; Lai, Y.K.; Cheng, M.M.; Yang, J. IP102: A Large-Scale Benchmark Dataset for Insect Pest Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–17 June 2019; pp. 8787–8796.
19. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
20. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
21. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768.
22. Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both weights and connections for efficient neural network. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. [CrossRef]
23. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
24. 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, Seattle, WA, USA, 13–19 June 2020; pp. 1580–1589.
25. Koonce, B. MobileNetV3. In *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*; Apress: Berkeley, CA, USA, 2021; pp. 125–144.
26. 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, Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790.
27. Tan, M.; Le, Q. Efficientnetv2: Smaller models and faster training. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 10096–10106.
28. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
29. Mellor, J.; Turner, J.; Storkey, A.; Crowley, E.J. Neural architecture search without training. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 7588–7598.
30. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
31. Wei, J.; Gong, H.; Li, S.; You, M.; Zhu, H.; Ni, L.; Luo, L.; Chen, M.; Chao, H.; Hu, J.; et al. Improving the Accuracy of Agricultural Pest Identification: Application of AEC-YOLOv8n to Large-Scale Pest Datasets. *Agronomy* **2024**, *14*, 1640. [CrossRef]
32. Zhang, L.; Zhao, C.; Feng, Y.; Li, D. Pests Identification of IP102 by YOLOv5 Embedded with the Novel Lightweight Module. *Agronomy* **2023**, *13*, 1583. [CrossRef]
33. Sun, H.; Nicholaus, I.T.; Fu, R.; Kang, D.K. YOLO-FMDI: A Lightweight YOLOv8 Focusing on a Multi-Scale Feature Diffusion Interaction Neck for Tomato Pest and Disease Detection. *Electronics* **2024**, *13*, 2974. [CrossRef]
34. Guo, B.; Wang, J.; Guo, M.; Chen, M.; Chen, Y.; Miao, Y. Overview of Pest Detection and Recognition Algorithms. *Electronics* **2024**, *13*, 3008. [CrossRef]
35. Appiah, O.; Hackman, K.O.; Diallo, B.A.A.; Ogunjobi, K.O.; Diakalia, S.; Valentin, O.; Abdoul-Karim, D.; Dabire, G. PlanteSaine: An Artificial Intelligence Empowered Mobile Application for Pests and Disease Management for Maize, Tomato, and Onion Farmers in Burkina Faso. *Agriculture* **2024**, *14*, 1252. [CrossRef]