agronomy

MDPI

Article

Lightweight Mulberry Fruit Detection Method Based on Improved YOLOv8n for Automated Harvesting

Hong Qiu, Qinghui Zhang *, Junqiu Li, Jian Rong and Zongpeng Yang

College of Big Data and Intelligent Engineering, Southwest Forestry University, Kunming 650224, China; qiuh28@swfu.edu.cn (H.Q.); lijunqiu@swfu.edu.cn (J.L.); swordrong@swfu.edu.cn (J.R.); yzp888@swfu.edu.cn (Z.Y.)
* Correspondence: huizq@swfu.edu.cn

Abstract: Aiming at the difficulty of feature extraction in complex environments during mulberry detection and the need for embedded devices to lighten the model, this study carries out lightweight improvements on the basis of the YOLOv8n model. First, the CSPPC module incorporates lightweight partial convolution (PConv) within its bottleneck structure, replacing the C2f module to enhance feature extraction efficiency. Secondly, the ADown module is used to replace the traditional down-sampling module and the P-Head module is used to replace the traditional convolutional detector head with the partial convolutional detector head. Finally, a knowledge distillation technique is used to compensate for the loss of accuracy due to parameter reduction. Ablation experiments are conducted to evaluate the impact of each module on the model's performance. The experimental results show that the improved YOLOv8 model has a precision of 88.9%, a recall of 78.1%, and an average precision of 86.8%. The number of parameters is 1.29×10^6, the model size is 2.6 MB, the floating-point computation is 2.6 GFLOPs, and the frame rate reaches 19.84 FPS on the edge end. Therefore, this model provides theoretical and technical support for the deployment and application of mobile detection devices, such as automatic mulberry harvesting in practical scenarios.

Keywords: mulberry; deep learning; YOLOv8n; partial convolution; knowledge distillation; embedded devices

Citation: Qiu, H.; Zhang, Q.; Li, J.; Rong, J.; Yang, Z. Lightweight Mulberry Fruit Detection Method Based on Improved YOLOv8n for Automated Harvesting. *Agronomy* **2024**, *14*, 2861. https://doi.org/10.3390/agronomy14122861

Academic Editor: Jayme Garcia Arnal Barbedo

Received: 7 November 2024
Revised: 24 November 2024
Accepted: 27 November 2024
Published: 30 November 2024

1. Introduction

Mulberries are widely appreciated for their rich nutritional value and distinctive flavor [1]. Traditional mulberry picking mainly relies on manual operation, but due to the irregular distribution of mulberry fruit, picking workers need to repeatedly bend over and reach out to acquire the fruit, resulting in high labor intensity and low efficiency, making it difficult to meet the needs of large-scale planting [2]. Moreover, mulberry is usually planted in hilly areas, resulting in a complex picking environment and picking difficulties. Although automated picking equipment based on target detection technology has been gradually introduced, the agricultural equipment usually runs in a resource-constrained embedded environment, and the high computational demands of the existing model make it difficult to meet the real-time requirements [3], which further restricts its potential for application in actual production.

Object-detection technologies based on machine vision have made significant advancements in fruit recognition; however, compared to other fruits, mulberries present unique challenges due to their small size, irregular distribution, and complex backgrounds, making accurate detection particularly difficult.

To address these challenges, it is essential to develop a technical solution that can rapidly identify mulberry fruits and to integrate it effectively with automated harvesting systems.

Traditional machine-learning fruit-detection methods mainly investigate the color, texture, and shape of fruits and show better effectiveness in some specific environments.

For example, Miao et al. proposed an optimization algorithm combining noise filtering, an improved OTSU algorithm, and K-means clustering, which can effectively identify the boundaries of overlapping fruit in the natural environment [4]; equally, Zhuang et al. proposed a monocular-based vision-based citrus detection method using adaptive enhanced color mapping and local binary pattern techniques to effectively detect and locate ripe citrus in the orchard [5]. However, these methods primarily rely on the color, texture, and shape features of fruits, which are prone to occlusion or confusion in complex backgrounds, leading to reduced detection performance [6].

Although traditional machine learning methods have shown effectiveness in specific environments, they rely on manual feature engineering and rule-based algorithms, making it difficult to maintain robustness in complex environments and handle large-scale data. In contrast, deep convolutional neural networks have been widely applied in image segmentation and classification tasks in recent years, particularly in agriculture and plant phenotyping [7]. This is because CNNs can automatically learn robust discriminative features [8] and are equipped to handle complex variations such as changes in lighting, occlusion of fruits, and background interference.

Artificial intelligence and its deep learning methods have significantly improved crop recognition [9], leading to the development and practical application of many deep learning-based fruit detection algorithms due to their superior feature-learning capabilities. These algorithms are primarily categorized into two types: two-stage detection algorithms, such as Faster R-CNN [10]; and single-stage detection algorithms, such as You Only Look Once [11] and Single Shot MultiBox Detector [12].

In the field of deep learning, He et al. solved the problem of segmentation of mulberry images under complex lighting conditions by using a visual saliency approach and a Pulse Coupled Neural Network model, which greatly improved the accuracy of mulberry fruit recognition [13]. Ashtiani et al. combined CNN with transfer learning to achieve 98.65% accuracy in mulberry ripeness classification [14]. In addition, targeted model improvement for different objects can also produce superior performance. For example, Wang et al. added a CBAM module to the optimized YOLOv4-Tiny and achieved 97.3% accuracy in detecting blueberries in complex backgrounds, highlighting its efficacy in object detection tasks [15]. Similarly, Gai et al. proposed an improved YOLOv4 model using DenseNet, which effectively improved the accuracy of cherry detection [16]. Gao et al. proposed an enhanced binocular localization method based on YOLOv5x, which detects kiwifruit and aligns it with the calyx, thus significantly reducing localization errors [17]. Zhang et al. improved YOLOv5 for fast detection and pose classification of greenhouse tomatoes [18]. While these above methods have significantly improved accuracy, they significantly increase the model parameters and computational requirements, complicating the deployment of edge devices.

In order to cope with the problem of a large number of model parameters and high computational complexity, researchers have proposed lightweight network structures such as MobileNet [19], ShuffleNe [20], and GhostNet [21] in addition to techniques such as model pruning [22], model quantization [23], and knowledge distillation (KD) [24]. For instance, Zeng et al. restructured the YOLOv5 backbone network by integrating the MobileNetV3 bneck module and applied channel pruning and quantization, achieving efficient real-time detection of tomato fruits [25]. Additionally, Wu et al. proposed the TiGra-YOLOv8 model, which utilizes channel pruning to reduce model size, decrease parameter count, and improve detection speed, achieving a 52.19% reduction in parameters and a 51.72% decrease in computational demand [26]. Similarly, Zhao et al. replaced the YOLOv5 backbone with a lightweight ShuffleNetv2 network, using grouped convolution and channel shuffling to reduce computational load, enabling efficient real-time detection of pomegranates [27]. These experiments demonstrate that employing suitable lightweight architectures and techniques can significantly reduce model parameters and computational requirements while maintaining accuracy.

Existing models have made some progress in lightweighting, but in order to maintain detection accuracy, additional mechanisms often need to be introduced, which in turn increases the number of parameters in the mode [28]. To achieve real-time monitoring of mulberries, it is often necessary to rely on low-power embedded devices, requiring the deployment of algorithms on edge computing platforms for validation.

To address the aforementioned challenges and meet the practical needs of agriculture, this study focuses on the specific requirements of mulberry fruit detection and identification. By introducing lightweight modifications to the YOLOv8n network structure, the model's parameter count and computational complexity are reduced. Subsequently, knowledge distillation techniques are employed to compensate for the accuracy loss caused by parameter reduction. The proposed solution is then deployed on an embedded platform for testing.

This lightweight solution not only enables accurate detection of ripe and unripe mulberries but also operates efficiently on embedded platforms, providing technical support for real-world production. With this efficient object detection model, farmers can achieve automated monitoring and harvesting of mulberry fruits.

2. Materials and Methods

2.1. Dataset Production

The mulberry images below were taken on 30 April 2024 at Qinglong Street, Panlong District, Kunming City, Yunnan Province, China, with a Xiaomi 12 smartphone. A total of 844 images were collected, each with a resolution of 3072 × 3072 pixels, and saved in jpg format. Due to the complex natural environment, mulberry fruits are often obscured by leaves and branches, so different angles and distances were chosen when taking the pictures. As shown in Figure 1, this approach captured a variety of situations, including overlapping fruits, foliage occlusion, and individual or clustered fruits.

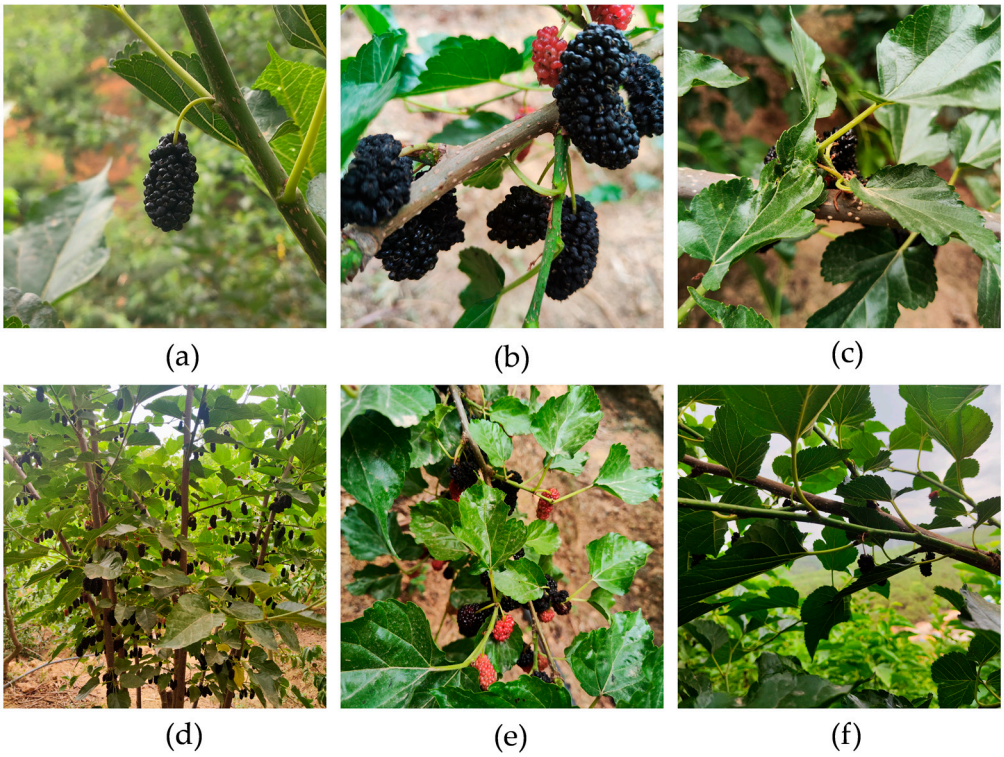

Figure 1. Example of mulberry fruit collection image: (**a**) simple target; (**b**) multiple target; (**c**) light occlusion; (**d**) heavy occlusion; (**e**) fairing; (**f**) backlighting.

To reduce model overfitting due to insufficient samples, Cut-Thumbnail [29] with a random combination of data enhancements was used. Cut-Thumbnail reduces shape bias

by randomly replacing portions of the image with resized thumbnails while retaining both detailed and global information.

The random-combination data-augmentation method integrates various image enhancement techniques to generate diversified training samples. These techniques include traditional image processing operations such as Gaussian noise and salt-and-pepper noise as well as spatial transformations such as horizontal and vertical flipping, non-uniform scaling, and random translation. These enhancement operations effectively increase the robustness and generalization of the model, as shown in Figure 2.

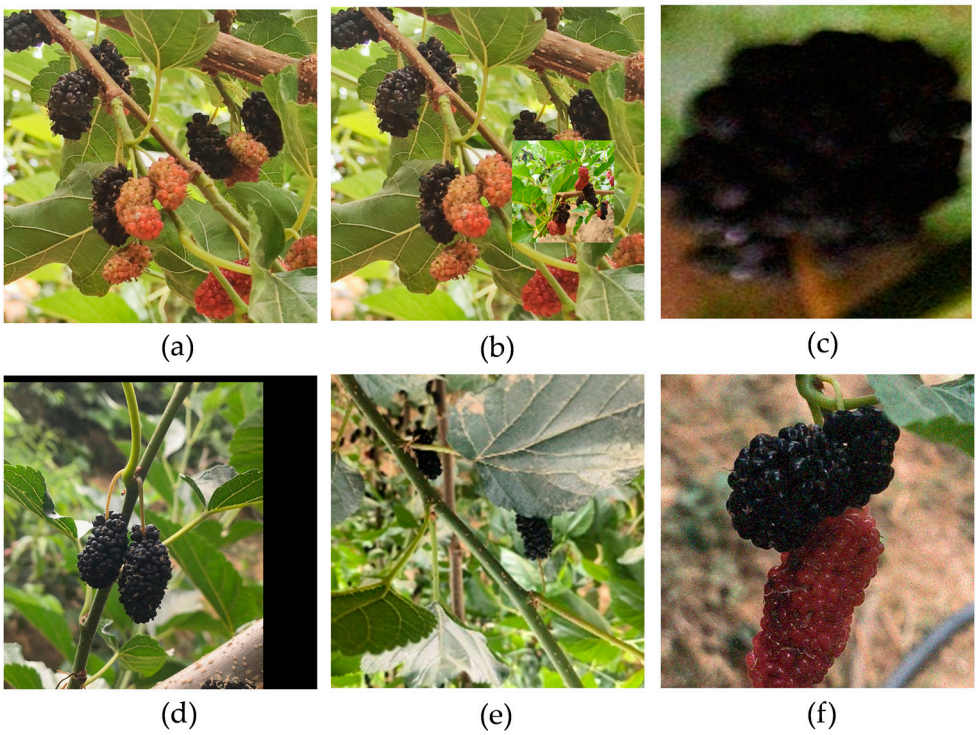

Figure 2. Data enhancement effect: (**a**) original image; (**b**) Cut-Thumbnail; (**c**) Gaussian noise; (**d**) non-uniform scaling; (**e**) transpose; (**f**) pretzel noise.

After the aforementioned offline data augmentation process, a total of 3088 mulberry images were obtained. These images were manually annotated using the LabelImg software, where mulberry targets were labeled, and bounding boxes were drawn to distinguish between immature and mature mulberries. The final dataset was randomly divided into training, validation, and test sets in a ratio of 8:1:1, comprising 2470, 309, and 309 images, respectively.

2.2. Model Architecture

2.2.1. YOLOv8 Model

YOLOv8 [30] is an advanced algorithmic model that excels in the field of object detection, inheriting and optimizing the core design concepts of the YOLO series. First of all, YOLOv8 provides a new state-of-the-art model for target detection tasks at different resolutions. The model can be divided into four main sections: input, backbone, neck, and head.

The input module is mainly responsible for the preprocessing of the raw image, including image resizing, pixel value normalization, data enhancement, and adding batch dimensions. These operations ensure that the image data conform to the format and dimensions required by the model, thus facilitating efficient object detection.

For the design of the backbone network, YOLOv8 draws inspiration from the CSP module and replaces the C3 module in YOLOv5 with the more lightweight C2f module.

This redesign draws on the principles of ELAN [31] to achieve greater model efficiency. In addition, YOLOv8 retains the SPPF module in YOLOv5, with careful parameterization for different model sizes.

In the neck design, YOLOv8 continues to adopt the PAN architecture but eliminates the 1 × 1 downsampling layer in YOLOv5, further optimizing the overall structure. In terms of the detection head, YOLOv8 has been significantly improved by adopting the widely used decoupled head architecture, which separates classification from detection and greatly improves detection accuracy and speed.

Finally, YOLOv8 utilizes BCELoss to compute the classification loss, while combining DFL Loss and CIOU Loss to optimize the regression loss.

2.2.2. Improved YOLOv8 for Mulberry Detection

In this study, targeted modifications were made to the YOLOv8 model to address the unique challenges faced by mulberry fruit detection. These challenges include a small target size, frequent occlusion by leaves and branches, complex ripeness recognition, and severe background interference. Given the perishability and short ripening period of mulberries, the detection model must achieve high accuracy while ensuring real-time performance and lightweight operation on embedded devices. The improved YOLOv8 network structure is shown in Figure 3.

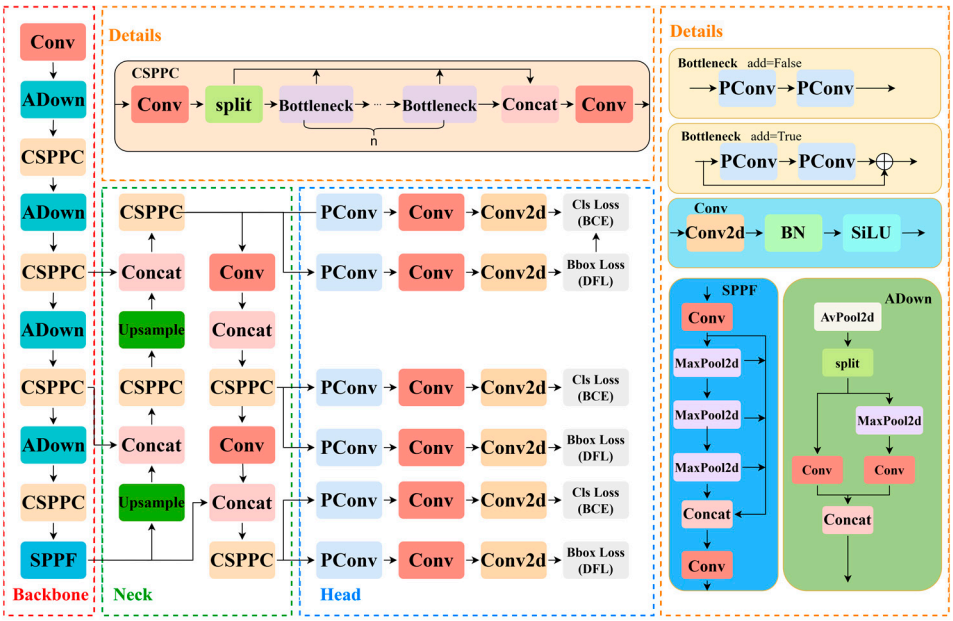

Figure 3. Diagram of the improved YOLOv8 network structure.

The C2f module in the network architecture was improved using partial convolution (PConv) [32]. PConv greatly improves the accuracy of mulberry small target detection by reducing information redundancy and focusing on critical information. In the convolutional computation, PConv only operates on some of the channels and leaves the other channels unchanged. This approach reduces redundant computation and focuses computational resources on key information, thus extracting the feature details of the mulberry small targets more efficiently. Compared with the traditional full-channel convolution, PConv uses more computational resources to preserve and enhance the detailed features of the mulberry small targets, thus improving the detection accuracy of the small targets.

PConv can also replace traditional convolutional operations within the network, especially in scenes where spatial resolution needs to be preserved. Detecting small mulberry targets relies on high-resolution features that are easily lost during the downsampling process. PConv helps to reduce parameters while preserving key details, thus improving the accuracy of small target localization and classification [33].

To ensure robust detection under complex background and illumination conditions, the ADown [34] module is used as the downsampling structure. By reducing repetitive redundant computations and enhancing multi-level feature representation, the ADown module effectively improves the fine-grained recognition of mulberry targets. In addition, the ADown module incorporates multiple feature representations to improve the overall detection accuracy while maintaining the lightweight nature of the model.

Compared with the standard convolutional module, the ADown module significantly reduces the computational cost in the downsampling process. By dividing the feature map into different regions and combining pooling and convolution operations, the ADown module reduces computational complexity while maintaining high detection accuracy for mulberry targets.

This design optimizes computational efficiency, making it particularly suitable for lightweight reasoning on embedded platforms without sacrificing detection accuracy. In the mulberry detection task, the improved P-Head detection head only convolves the critical channels, which further improves efficiency by significantly reducing the number of repetitive redundant computations and parameters in the network. This significantly reduces the overall size of the model. Compared to standard convolutional heads, the P-Head reduces computational overhead and memory consumption while maintaining efficient feature extraction. This lightweight design is especially beneficial for resource-constrained embedded devices. The introduction of P-Head greatly reduces the number of parameters and further optimizes the size of the mulberry detection model, enabling efficient, real-time detection even on limited hardware.

Finally, the Channel-wise Distillation (CWD) [35] technique was applied to the improved mulberry detection model to increase the detection accuracy. CWD efficiently transfers key features from the teacher model to the lightweight student model, thereby reducing unnecessary computations and parameters while retaining the ability to extract fine details of mulberries in complex backgrounds. This approach ensures accurate detection on resource-limited devices, significantly optimizing the parametric efficiency and performance of the model.

2.2.3. C2f Improvement

The C2f module in YOLOv8 employs standard convolution for feature extraction, which, although effective at capturing multi-level feature information, results in considerable computational overhead. This limitation reduces its suitability for lightweight and real-time applications, particularly when detecting small and easily occluded targets such as mulberry fruits. Detecting mulberry fruits necessitates not only maintaining high accuracy in complex backgrounds but also ensuring real-time performance to meet the practical demands of agricultural production.

In the intermediate layers of the mulberry detection model, certain channels are capable of identifying mulberries, as shown in Figure 4b,d,f,g,i, while other channels exhibit stronger responses to the background, as seen in Figure 4c,e,h,j. Overall, most feature map channels successfully capture critical features related to mulberry targets.

Therefore, due to the high similarity between channels in these feature maps, some channels may carry redundant information [36]. In such cases, reducing feature map redundancy can effectively decrease computational resource consumption.

In practical applications, convolutional layers in neural networks perform operations across all channels of the input feature map, including channels that contain redundant or less informative features. This could lead to an inefficient use of computational resources.

In this study, we propose a novel structure, CSPPC, to replace the standard convolution in the C2f module with PConv to achieve a lightweight design for the mulberry detection task. The structure of the PConv is shown in Figure 5. PConv performs convolution operations only on a subset of the input channels, significantly reducing computational overhead while preserving the network's feature extraction capacity. This approach is especially

critical when dealing with complex backgrounds and small targets as it allows the model to maintain detection accuracy while minimizing unnecessary computational overhead.

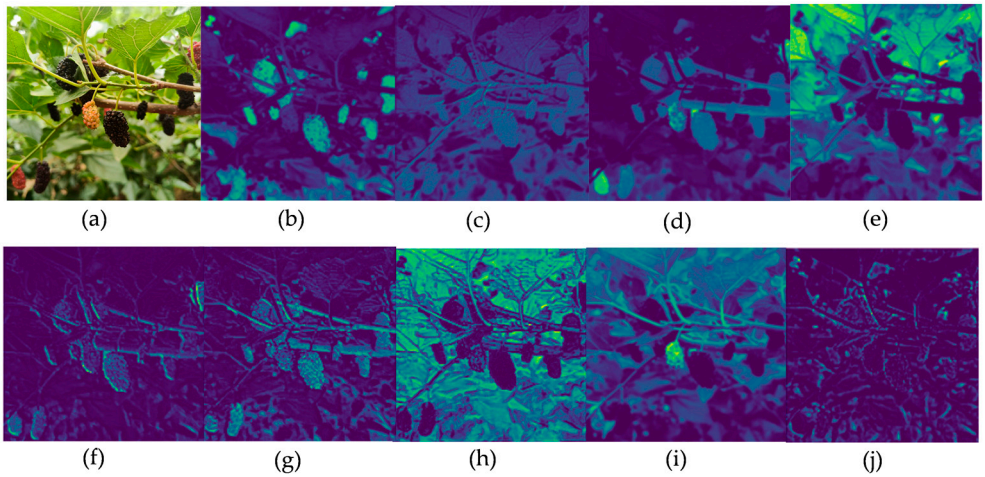

Figure 4. Visualization of feature maps: (**a**) original image; (**b**–**e**) first-layer convolutional feature map visualization; (**f**–**j**) third-layer C2f layer feature map visualization.

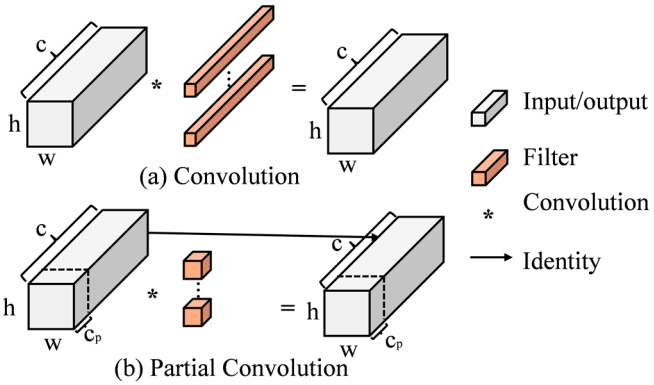

Figure 5. PConv structure diagram.

In standard convolution operations, all channels of the input feature map participate in the convolution calculation. The convolution applies filters to each input channel, generating output feature maps with the same number of channels as the input. Figure 6 illustrates the logical structure of the PConv module, where n_div represents the number of channel divisions used to control the proportion of channels processed by the convolution.

The floating point operations (*FLOPs*) of PConv are calculated as shown in Equation (1). In this equation, h and w represent the height and width of the input feature map, respectively; k is the size of the convolution kernel; c represents the number of channels; and c_p represents the number of channel divisions. The formula calculates the number of floating point operations required for performing a partial convolution on the input data.

$$FLOPs_{PConv} = h \cdot w \cdot k^2 \cdot c_p^2 \tag{1}$$

By comparing the computational load and memory access (*MA*) between standard convolution and partial convolution, it is evident from Equations (2) and (3) that with a typical partial ratio of r $= \frac{c_p}{c} = \frac{1}{4}$, the FLOPs of PConv are only 1/16 of those in standard convolution, and the memory access is approximately 1/4 of that of standard convolution.

$$\frac{FLOPs_{PConv}}{FLOPs_{Conv}} = \frac{(h \cdot w \cdot k^2 \cdot (c/4)^2)}{(h \cdot w \cdot k^2 \cdot c^2)} = \frac{1}{16} \tag{2}$$

$$\frac{MA_{PConv}}{MA_{Conv}} = \frac{(h \cdot w \cdot 2 \cdot c/4 + k^2 \cdot (c/4)^2)}{(h \cdot w \cdot 2c + k^2 \cdot c^2)} \approx \frac{1}{4} \quad (3)$$

By replacing the C2f module with the CSPPC module, the model greatly reduces the number of parameters and computational overhead. This improvement not only makes the model more lightweight but also greatly improves the operational efficiency in embedded environments, making mulberry fruit detection more efficient and reliable under resource-limited conditions. In addition, this improvement also enhances the robustness in complex environments, providing important technical support for agricultural automation and smart harvesting.

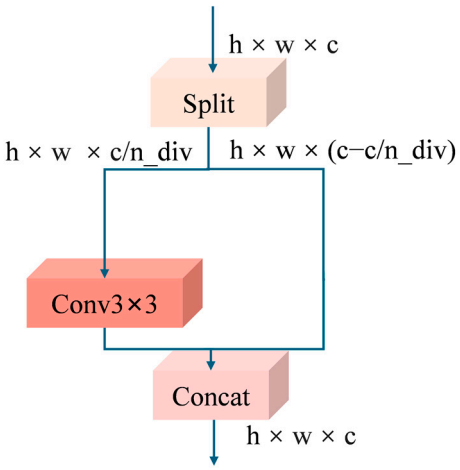

Figure 6. PConv logic structure diagram. Note: h is the feature map height, w is the width, and c is the number of channels.

2.2.4. Improvements in the Downsampling Module

Downsampling is one of the key steps in the mulberry inspection task. Downsampling not only reduces the size of the feature map, thus reducing the computational complexity and the number of parameters but also improves the inference efficiency of the model while retaining important feature information. However, when downsampling is performed using traditional convolutional methods, it often leads to the loss of fine-grained information, which negatively affects the detection accuracy of the model [37].

To address this challenge, this paper introduces the ADown module adapted from the YOLOv9 project to optimize the YOLOv8 network. The ADown module effectively reduces the size of the feature map of the detection target by applying multiple pooling and convolution operations. At the same time, the enhanced detection channel also accelerates the subsequent feature detection. The ADown module first preprocesses the input feature map by average pooling to reduce the sampling rate, and then divides the feature map into two parts along the channel dimension. The first part uses 3×3 convolution for feature extraction and dimensionality reduction, while the second part combines maximum pooling and 1×1 convolution to further enhance the nonlinear representation of features while reducing the dimensionality. Finally, the two parts are combined to generate the output of the ADown module. The logical structure of the ADown module is shown in Figure 7.

Compared to traditional convolutional downsampling, the ADown module offers significant advantages. It combines the strengths of both average pooling and max pooling, providing a more comprehensive extraction of multi-scale feature information for mulberries. By splitting the input feature map and applying pooling operations, the ADown module reduces the size of feature maps that directly participate in convolutional computation, significantly lowering the overall computational load. This improvement not only enhances the model's representational capacity but also boosts the detection accuracy of mulberry targets, effectively meeting the practical demands of complex environments.

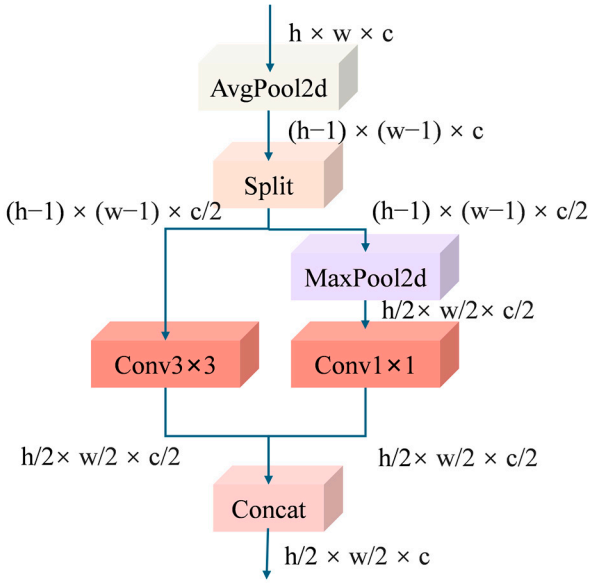

Figure 7. ADown logical structure diagram. Note: h is the feature map height, w is the width, and c is the number of channels.

2.2.5. Detection Head Module Improvements

In the mulberry detection task, the similarity between the fruit and the complex background, light variations, and the overlap between the mulberry fruit and the leaves make feature extraction more difficult. Although the traditional YOLOv8 detection head can extract rich features, it uses a standard convolutional layer for all channels, which leads to high computational overhead, especially in resource-constrained situations such as edge devices. To solve this problem, this study introduces an improved detection head structure, the P-Head, which reduces the computational overhead and the number of memory accesses by introducing PConv, a technique that performs convolutional operations only on some key channels. In the improved detection head, each PConv layer is followed by a standard convolutional layer, which finally outputs the bounding box regression and category classification through the Conv2d layer.

Compared with standard convolution, PConv can more effectively remove redundant features, effectively reducing the redundancy of the convolutional neural network channels, facilitating the learning of critical features, and lightening the weight without losing the ability of deep feature extraction. Eventually, the features are output through the Conv2d layer for the bounding box regression and category classification. With this approach, the improved PConv detection head not only significantly reduces the amount of computation but also enables efficient, real-time mulberry detection in resource-limited environments such as embedded devices.

2.2.6. Channel-Wise Knowledge Distillation

In order to improve the detection accuracy of the lightweight model in the mulberry fruit detection task, the KD technique was used in this study. Although KD does not reduce the number of parameters in the model itself, it effectively improves the accuracy of the model by transferring knowledge from more complex teacher models, thereby improving performance without significantly increasing the computational load. Given the challenges posed by the natural environment—such as background interference, occlusion, and mulberry fruit overlap—accurate feature extraction is critical to improving model accuracy. By efficiently transferring knowledge from the high-complexity teacher model to the lightweight student model, KD can significantly improve the performance of the student model in resource-constrained environments.

In the mulberry recognition task, the channel knowledge distillation method is introduced. This method aligns the channels of the teacher model and the student model

at specific feature layers and converts the feature activation values of each channel into probability distributions to minimize the Kullback–Leibler divergence between them. As shown in Figure 8, this process enables the student model to better learn the key feature extraction capabilities of the teacher model, effectively retaining the precise information related to mulberry fruit detection. As a result, both the localization and classification accuracy of the student model in complex background situations are improved.

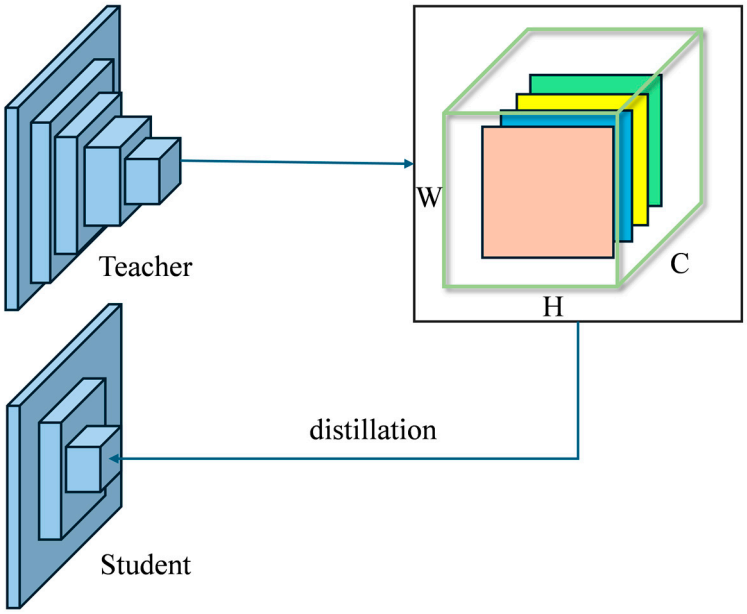

Figure 8. CWD distillation diagram. Different colors in the upper right corner represent different feature maps.

Specifically, it is necessary to convert the logits scores of the corresponding channels from both the teacher and student networks into probability distributions. This can be achieved by applying the softmax function, as expressed in Equation (4):

$$\phi(y_c) = \frac{\exp\left(\frac{y_{c,i}}{T}\right)}{\sum_{i=1}^{W\times H}\exp\left(\frac{y_{c,i}}{T}\right)} \tag{4}$$

where $y_{c,i}$ represents the logit score at the i-th spatial location within the channel c; T is the temperature parameter, which controls the smoothness of the probability distribution; W and H denote the width and height of the feature map, respectively; $\phi(y_c)$ represents the probability distribution of channel c. A higher value of T results in a smoother probability distribution, allowing the model to focus on a broader spatial region within the channel.

By summing the Kullback–Leibler divergence (*KL divergence*) across all channels in Equation (5), we obtain the Channel-wise Distillation Loss (L_{CWD}). Here, y_c^T and y_c^S represent the logits for the C-th channel from the teacher and student, respectively, and ϕ applies the softmax function to convert these logits into normalized probability distributions. *KL divergence* evaluates the similarity between these distributions, emphasizing regions where the teacher predicts with higher confidence, thus guiding the student to focus on critical spatial features. By summing over all channels and normalizing by C, the equation ensures comprehensive and balanced knowledge transfer, aligning the student's outputs with the teacher's fine-grained feature patterns across the entire feature map.

$$L_{CWD} = \frac{1}{C}\sum_{c=1}^{C} KL\left(\phi(y_c^T) \parallel \phi(y_c^S)\right) \tag{5}$$

Equation (6) defines the total loss function, combining the L_{CWD} with task loss (L_{task}) to derive the final training loss (L_{total}). In this equation, L_{total} refers to the combined loss and λ is a weight coefficient used to balance the task loss and the distillation loss.

$$L_{total} = L_{task} + \lambda \cdot L_{CWD} \tag{6}$$

2.3. Training Environment and Parameter Configuration

The model development was conducted on a computing platform with AlmaLinux as the operating system, an AMD EPYC 7402 CPU, an NVIDIA A100 GPU, and 94 GiB of RAM. The environment utilized Python version 3.11.5 and PyTorch version 2.1.0. The training parameters are detailed in Table 1.

Table 1. Training parameter configuration.

Parameter Categories	Parameter Settings
Optimizer	SGD
Batch size	16
Epochs	200
Input size	640 × 640
Initial learning rate	0.01
Momentum	0.937
Weight decay rate	0.0005

2.4. Evaluation Metrics

To comprehensively evaluate the model's performance in the mulberry recognition task, multiple evaluation metrics were employed, including precision, recall, mean average precision (mAP), the number of parameters, model memory usage, floating point operations (GFLOPs), and frames per second (FPS).

The precision measures the accuracy of the model's predictions, i.e., the proportion of samples predicted as positive that are positive; its formula is given in Equation (7):

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

where TP (True Positives) refers to the number of correctly predicted positive samples, and FP (False Positives) refers to the number of samples incorrectly predicted as positive.

Recall measures the model's ability to identify positive samples, i.e., the proportion of actual positive samples that are correctly predicted as positive; its formula is provided in Equation (8):

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

where FN (False Negatives) represents the number of positive samples that were incorrectly predicted as negative.

Mean Average Precision (mAP) is a widely used evaluation metric in object detection, which calculates the average precision across all categories. mAP50 represents the mean average precision at an intersection-over-union (IoU) threshold of 0.5. The formula is defined in Equation (9):

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

The number of parameters reflects the complexity of the model, while the memory usage indicates the memory resources required for the model during runtime. Giga Floating Point Operations (GFLOPs) represent the number of floating-point operations needed for a single inference, and frames per second (FPS) measure the inference speed of the model. These metrics collectively assess the model's level of lightweight design, computational resource requirements, and real-time detection capabilities.

3. Results and Discussion

3.1. Performance Evaluation and YOLOv8 Model Selection

The YOLOv8 algorithm is extremely flexible, allowing researchers to choose different model sizes according to actual needs. This study evaluated the detection performance of the four versions of our self-constructed mulberry dataset. The results are shown in Table 2. Although YOLOv8n has lower precision, recall, and average accuracy, its smaller parameter sizes and weights make it well-suited for resource-limited environments. In order to effectively utilize the memory and computational resources on embedded devices, we choose YOLOv8n as the baseline model for further lightweight improvements specifically for the mulberry recognition task. With further optimization, the model's performance in embedded applications is significantly improved, demonstrating its ability to accurately and quickly identify mulberry targets, especially in complex environments.

Table 2. Performance comparison of different YOLOv8 models.

Model	Precision (%)	Recall (%)	mAP@0.5 (%)	Parameter (M)	Model Size (MB)
YOLOv8l	91.9	85.2	91.4	4.36×10^7	85.6
YOLOv8m	89.7	84.2	90.1	2.59×10^7	50.8
YOLOv8s	88	82.3	88.4	1.11×10^7	22
YOLOv8n	82	78.5	84.6	3.01×10^6	6.3

3.2. CSPPC Module Ablation Results

Table 3 shows the ablation experiments using the CSPPC module to replace C2f at different locations. It can be seen that Experiment 4 has the smallest parameter count of 2.12×10^6 and lower weights, which makes it more advantageous in resource-constrained embedded applications. Meanwhile, Experiment 4 has the smallest floating-point computation of all experiments at 6.0 GFLOPs and a high frame rate, reflecting better computational efficiency; however, the accuracy rate of 83.2% and the average accuracy of 83.3% are lower than the other experiments, reflecting its smaller computational volume and parameter count. Therefore, Experiment 4 is selected for improvement.

Table 3. CSPPC module positional ablation experiments.

Model	Backbone	Neck	P (%)	R (%)	mAP@0.5 (%)	Parameter	Model Size (MB)	GFLOPs	FPS
1	×	×	83	78.3	84.7	3.01×10^6	6.3	8.2	253.9
2	✓	×	85.1	77.1	84.8	2.56×10^6	5.2	6.9	283
3	×	✓	85.2	75.5	84.3	2.58×10^6	5.4	7.3	267
4	✓	✓	83.2	75.8	83.3	2.12×10^6	4.3	6.0	280

3.3. Improved C2f Structure with Different Lightweight Methods

To identify the most effective lightweight improvement method, various techniques were compared regarding their impact on network performance when enhancing the C2f module. This study examines five distinct improvement strategies, including Heterogeneous Kernel-Based Convolutions [38], Dual Convolutional Kernels [39], a novel convolution method that integrates Ghost modules with dynamic convolution [40], and Spatial and Channel Reconstruction Convolution [41]. The results are shown in Table 4 and display that the CSPPC module achieves an average accuracy of 83.3% while maintaining the lowest number of parameters and model weights. In addition, its GFLOPs and FPS are kept at a relatively moderate level. Although the improved CSPPC model does not achieve the highest precision, recall, or average accuracy compared to other models, it has the smallest memory footprint and the highest computational efficiency. This suggests that the CSPPC module is not only suitable for deployment on embedded devices but also

effective in extracting features in complex environments while maintaining the accuracy of the model.

Table 4. Different lightweight methods to improve C2f comparison.

Model	P (%)	R (%)	mAP@0.5 (%)	Parameter	Model Size (MB)	GFLOPs	FPS
YOLOv8n	83	78.3	84.7	3.01×10^6	6.3	8.2	253.9
YOLOv8n + CSPHet	84.2	76.4	84.2	2.38×10^6	5.1	6.6	58
YOLOv8n + C2f_SCConv	85.5	75	84.3	2.71×10^6	5.7	7.5	76.1
YOLOv8n + C2f_Ghost	83.1	75.3	82.8	2.19×10^6	4.6	5.8	140.9
YOLOv8n + C2f_Dual	83.1	77.6	84.8	2.86×10^6	5.9	7.7	241.7
YOLOv8n + CSPPC	83.2	75.8	83.3	2.12×10^6	4.3	6.0	280

3.4. Ablation Experiment

In this ablation study, we systematically analyzed the impact of various modules on the mulberry detection task. The results demonstrate that lightweight improvements significantly enhance model performance. The results of the ablation experiments are shown in Table 5.

Table 5. Ablation experiment results.

	CSPPC	ADown	P-Head	KD	P (%)	R (%)	mAP (%)	Parameter	Model Size (MB)	GFLOPs	FPS
1	×	×	×	×	82	78.5	84.6	3.01×10^6	6.3	8.2	253.9
2	✓	×	×	×	83.2	75.8	83.3	2.12×10^6	4.3	6.0	280
3	×	✓	×	×	85.6	78	85.1	2.73×10^6	5.7	7.5	247.7
4	×	×	✓	×	82.9	75.5	83.4	2.46×10^6	5.1	5.7	297.2
5	×	×	×	✓	88.8	81.9	88.1	3.01×10^6	6.3	8.2	247
6	×	✓	×	✓	90.7	80.1	88.2	2.73×10^6	5.7	7.5	249
7	×	✓	✓	×	86.9	76	84.4	2.14×10^6	4.5	4.9	288.4
8	✓	×	×	✓	89.7	78.5	86.9	2.12×10^6	4.5	6.0	258.0
9	✓	✓	✓	×	84.5	74.3	82.8	1.29×10^6	2.6	2.7	254.8
10	✓	✓	✓	✓	88.9	78.1	86.8	1.29×10^6	2.6	2.6	260

In this experiment, all models were trained for 200 epochs, including both the teacher and student models in the KD process. When trained independently, both models achieved satisfactory convergence after 200 epochs; however, during the KD phase, the student model did not meet expectations with 200 epochs, showing slower convergence and limited accuracy improvement.

As a result, the training for distillation was extended to 300 epochs to allow the student model to better learn and absorb the features and knowledge from the teacher model. By adding 100 additional epochs, the student model showed a significant performance improvement after fully learning from the teacher model. The comparison of the Precision–Recall (PR) curves of different improved models is shown in Figure 9.

Firstly, Experiment 1 serves as a benchmark model without introducing any lightweighting module and exhibits low precision and recall. This is consistent with the findings of existing studies that the unoptimized YOLO model performs poorly in complex contexts. The model is limited by high computational complexity and limited feature extraction capability when dealing with small targets like mulberries.

The introduction of the CSPPC module in Experiment 2 resulted in a significant reduction in the number of parameters and FLOPs of the model to 70.43% and 73.17% of the baseline model, respectively. This result verifies the effectiveness of the CSPPC module in reducing computational complexity while maintaining a stable extraction capability for mulberry features.

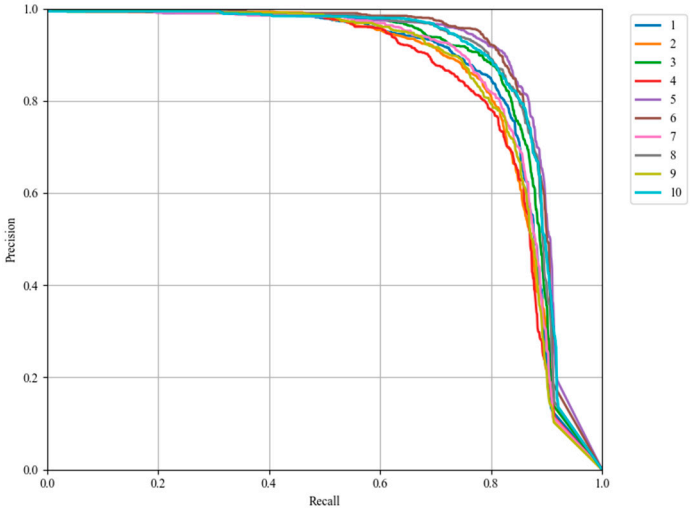

Figure 9. Comparison of PR curves for different models.

In Experiment 3, when the ADown module was added, the accuracy and average accuracy of the model were further improved to 85.6% and 85.1%, indicating that the module excelled in enhancing multi-scale feature extraction, effectively reducing the loss of fine-grained information during downsampling, especially playing a key role in the task of mulberry detection in complex backgrounds.

In Experiment 4, the introduction of the P-Head module significantly improves the frame rate, although the recall and average precision are reduced by 3% and 1.2% compared with the baseline model, which reflects the balance between the frame rate and the detection precision in the lightweight design; additionally, it embodies that P-Head is effective in lightweight applications pursuing real-time performance but further optimization may be required for tasks requiring higher precision.

Experiment 5 introduces knowledge distillation, and in the case of using only knowledge distillation, the precision and recall of the model without increasing the number of parameters and computation improves by 6.8% and 3.4% compared to the baseline model. The performance improvement was achieved without increasing the model complexity.

Experiment 6, using ADown and knowledge distillation, achieves the highest precision rate of 90.7% and improves the accuracy and recall by 5.1% and 2.1% compared to Experiment 3 using only the Adown model without increasing the computational complexity. It can fully reflect that knowledge distillation can improve accuracy without increasing the computational complexity.

Experiment 7 combines the ADown and P-Head modules, combining the advantages of the two modules to improve the efficiency and precision of downsampling. The ADown module effectively reduces the loss of fine-grained information by improving the downsampling process, thus enhancing the model's ability to extract multi-scale features, while the P-Head module optimizes the structure of the detection head to improve the inference speed and frame rate of the model. This combination aids the model not only to maintain a high precision and recall rate when dealing with the task of mulberry detection in complex backgrounds but also to significantly reduce the computational volume, achieving a good balance between precision and efficiency.

Experiment 9 combines three lightweight modules with a view to achieving better performance improvement. By integrating these three modules, the model achieves 84.5%, 74.3%, and 82.8% in terms of precision, recall, and average accuracy, respectively. Although these metrics decreased compared to Experiment 6, the number of parameters was significantly reduced to 1.29×10^6 and the number of floating-point computations to 2.6 GFLOPs, thus effectively reducing the computational complexity and memory footprint.

With the introduction of the knowledge distillation technique in Experiment 8 and 10, the models showed significant improvement in precision, recall, and average accuracy

without significantly increasing the computational cost, indicating that the knowledge distillation technique not only effectively transfers the knowledge of the teacher model but also significantly improves the overall performance of mulberry target detection in a lightweight model.

3.5. Contrast Experiment

The improved model based on YOLOv8 was compared with mainstream object detection networks such as SSD, YOLOv3-tiny, YOLOv5n, YOLOv6n, YOLOv7-tiny, YOLOv8n, YOLOv9t, and YOLOv9c. The results are shown in Table 6.

Table 6. Performance comparison of different models.

Model	Precision (%)	Recall (%)	mAP@0.5 (%)	Parameter	Model Size (MB)	FPS
SSD	80.6	59.6	71.5	2.63×10^7	91.1	86.02
YOLOv3-tiny	88.8	72.1	80.6	1.21×10^7	24.4	261
YOLOv5n	83.6	74.6	82.8	2.51×10^6	5.3	237
YOLOv6n	84.7	74.8	82.7	4.24×10^6	8.7	258.1
YOLOv7-tiny	86.9	83.1	88.1	6.02×10^6	12.3	258.6
YOLOv8n	82	78.5	84.6	3.01×10^6	6.3	253.9
YOLOv9t	87.4	76.3	84.7	2.01×10^6	4.7	111.9
YOLOv9c	88.3	78.1	85.9	2.55×10^7	51.6	62
Ours	88.9	78.1	86.8	1.29×10^6	2.6	260

The model selected in this study demonstrates exceptional performance and lightweight advantages across multiple metrics. First, the parameter count of the proposed model is only 1.29×10^6, with a weight size of just 2.6 MB, making it the smallest among all models. In contrast, YOLOv3-tiny has a parameter count of 1.21×10^7 and a weight size of 24.4 MB, while SSD reaches 2.63×10^7 parameters with a weight size of 91.1 MB. This clearly illustrates that the proposed model significantly reduces computational resource consumption, making it more suitable for embedded platforms or real-time applications.

The proposed model achieved the highest precision, surpassing SSD, YOLOv5n, YOLOv3-tiny, YOLOv6n, YOLOv7-tiny, YOLOv8n, YOLOv9t, and YOLOv9c by 8.3%, 0.1%, 5.3%, 4.2%, 2.0%, 6.9%, 2.5%, and 0.6%, respectively. This improvement in precision demonstrates the model's enhanced recognition capabilities in practical applications. Notably, in tasks that demand high accuracy, the proposed model consistently maintains a high level of correctness.

In terms of recall, the proposed model achieves 78.1%, which is on par with YOLOv9c. While this is 5.0% lower than YOLOv7-tiny, it exceeds YOLOv3-tiny, YOLOv5n, and YOLOv6n by 6.0%, 3.5%, and 3.3%, respectively. This indicates that the proposed model can effectively detect more targets while maintaining a low false detection rate.

For mAP, the proposed model achieves 86.8%, slightly below YOLOv7-tiny's 88.1% but surpassing YOLOv3-tiny, YOLOv5n, YOLOv6n, YOLOv8n, YOLOv9t, and YOLOv9c by 6.2%, 4.0%, 4.1%, 2.2%, 2.1%, and 0.9%, respectively. This demonstrates that the model delivers high overall performance in multi-class object detection, maintaining strong accuracy in complex scenarios.

Figure 10 is a radar chart comparing the performance metrics of our model and the other models. As can be seen from the figure, the YOLOv7-tiny model achieves the highest recall and mAP, while our model exhibits the highest precision. In addition, our model has the smallest number of parameters and the smallest model. Most of the models have an FPS of around 260, with YOLOv3-tiny leading with 261 FPS and our model following with 260 FPS.

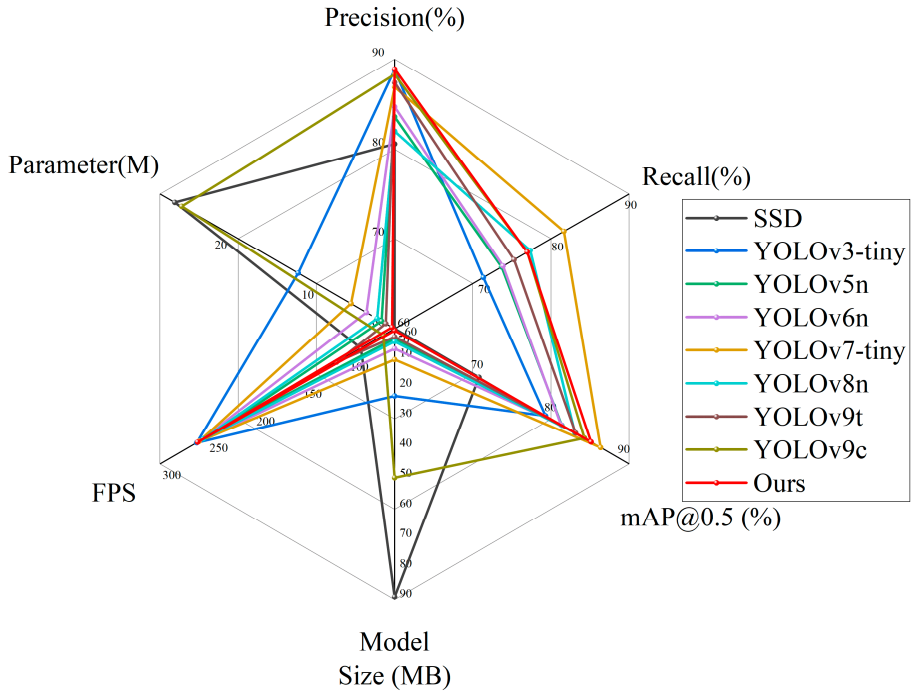

Figure 10. A radar chart comparing the metrics of our model with other mainstream object detection networks.

From Table 6, it can be observed that YOLOv7-tiny achieves the highest recall and mAP, while our proposed model achieves the highest precision. However, compared to our proposed model, YOLOv7-tiny exhibits significantly larger parameter counts and model size. The improved model presented in this study achieves the highest precision, the smallest parameter count, and the smallest model size, while maintaining relatively high recall and mAP. According to Equation (8), the slightly lower recall of our model indicates the occurrence of more false negatives; however, this trade-off is mitigated by its higher precision, which ensures fewer false positives.

Considering the limited computational power of embedded devices, the lower parameter count and smaller size of our model provide a superior performance–efficiency trade-off under constrained hardware conditions. By achieving real-time detection with minimal resource consumption, our model proves to be better suited for practical applications. This makes the proposed model a more practical choice when lightweight design and operational efficiency are prioritized.

3.6. Testing Our Model on Jetson

To validate the feasibility of the improved model, it was deployed onto an edge device for testing, specifically the Jetson Nano. Jetson Nano is an embedded development platform launched by NVIDIA, designed for AI and machine learning applications. It integrates powerful GPU computing capabilities, enabling real-time deep learning inference on small devices. The Jetson Nano features a quad-core ARM Cortex-A57 CPU and a 128-core GPU based on the Maxwell architecture, with 4 GB of LPDDR4 memory and 16 GB eMMC storage, supporting up to 472 GFLOPS of computational power, making it suitable for deep learning inference tasks on edge devices. Figure 11 shows a perspective of the Jetson Nano camera in operation.

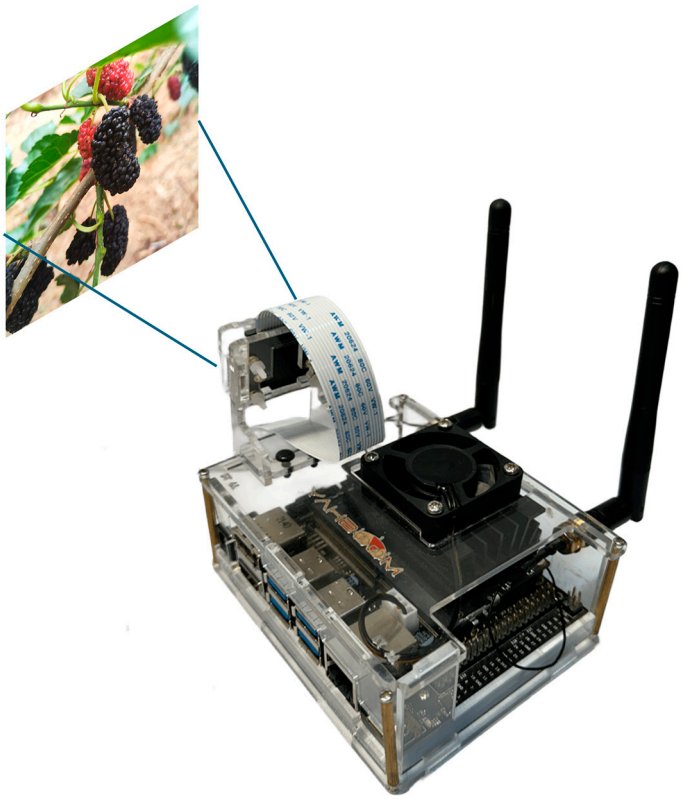

Figure 11. Work perspective diagram.

To further optimize inference performance on resource-constrained devices, this study employed NVIDIA's TensorRT acceleration tool. TensorRT is a high-performance deep learning inference library developed by NVIDIA, aimed at improving inference speed and efficiency through techniques such as network structure optimization, precision calibration, and layer fusion. These optimizations make it particularly suitable for embedded and edge devices.

Table 7 demonstrates that the performance of different models on a standard computer is very similar, with the proposed model achieving approximately 2.4% higher FPS compared to YOLOv8n. On the Jetson Nano platform, the FPS of the proposed model reaches 10, significantly outperforming other models. The lightweight optimization enables the proposed model to perform better on resource-constrained devices.

Table 7. Comparison of detection frame rates for device deployment.

Model	Computer FPS	Jetson Nano FPS	TensorRT FPS
SSD	86.02	1.19	3.79
YOLOv3-tiny	261	3.7	11.02
YOLOv5n	237	5.5	16.65
YOLOv6n	258.1	3.9	16.65
YOLOv7-tiny	258.6	4.58	9.95
YOLOv8n	253.9	5.2	16.11
YOLOv9t	111.9	3.6	14.02
Ours	260	10	19.84

After TensorRT acceleration, all models exhibit noticeable FPS improvements on the Jetson Nano. Specifically, SSD achieves the smallest improvement of 2.6 FPS, while YOLOv6n sees the largest improvement of 13.93 FPS. The FPS of the proposed model increases to 19.84, making it the model with the highest FPS among all tested models.

To further validate the improved model's detection performance on mulberries, various mulberry images were selected for testing. The figure presents a comparison of the detection results from the improved model and the YOLOv8n model on the Jetson Nano.

As shown in Figure 12, in close-range detection, the YOLOv8n model missed one target, while the improved model detected all targets without omissions. In long-range detection, the YOLOv8n model missed eight targets, whereas the improved model missed only five. Under occlusion conditions, neither model exhibited any missed detections. In backlighting conditions, the YOLOv8n model and the improved model each missed three targets. Overall, the improved model demonstrates better detection performance than the YOLOv8n model in complex scenarios.

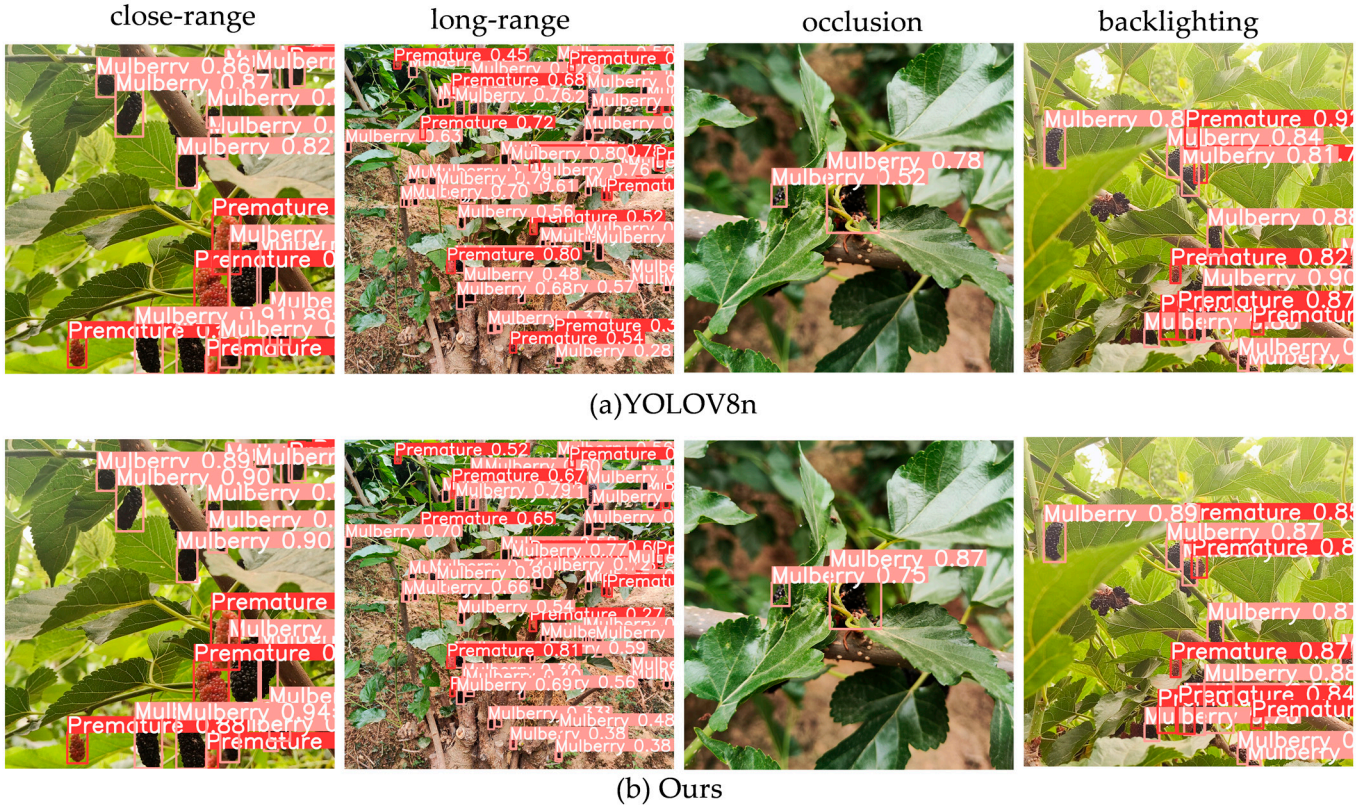

Figure 12. Detection results of (**a**) YOLOv8n; and (**b**) our model.

4. Conclusions

This study addresses the challenges of extensive model parameters and high computational complexity in mulberry detection by proposing an optimized algorithm based on YOLOv8. The main contributions are as follows: First, the lightweight CSPPC module replaces the original C2f module, reducing both computational load and memory usage. Second, the ADown module improves the downsampling process, enhancing feature extraction efficiency. Finally, the PConv detection head reduces the computational demand while incorporating KD techniques to enhance accuracy.

The effectiveness of the PConv method in mulberry detection was demonstrated by comparison with various C2f modification techniques. Ablation studies were also performed to evaluate the individual effects of each modification. The optimized model was compared with other models and deployed on the Jetson Nano edge computing device for testing. The experimental results show significant reductions in parameters and computational complexity, with the final model having 1.25 million parameters and a computational complexity of 2.6 GFLOPs. mAP reaches 86.9%, and the inference speed on the embedded device improves to 19.84 FPS with TensorRT acceleration. These improvements provide an

efficient and accurate solution for mulberry detection on embedded platforms and set the stage for agricultural automation.

While the model performed well in terms of accuracy and efficiency, there are still limitations, especially when dealing with images affected by heavy occlusion or extreme light variations. Future work will focus on enhancing robustness through better feature fusion mechanisms and more efficient loss functions. In addition, further evaluations on different hardware platforms are necessary to ensure broad applicability in the real world.

This study not only verifies the effectiveness of the proposed lightweight improvements but also provides new insights for optimizing future lightweight models. Future research could explore integrating multi-scale feature fusion and refining loss functions to enhance robustness and detection efficiency in complex environments, supporting broader applications in fruit detection.

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

Funding: This study was financially supported by the Agricultural Joint Project of Yunnan Province (Grant No. 202301BD070001-127); the Yunnan International Joint Laboratory of Natural Rubber Intelligent Monitor and Digital Applications (Grant No. 202403AP140001).

Data Availability Statement: The data presented in this study are available upon request from the corresponding author due to the fact that our research is part of an ongoing project, and in order to maintain the integrity of our continuing work, we are unable to publicly release the dataset at this time.

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

References

1. Yan, Z.; Xin, L.I.; Hao, D.; Lingling, L.I.; Yuxing, L.I.U.; Wanting, Y.; Shaobo, C.; Guogang, C. Optimization of the Production Process and Quality Evaluation of Mulberry-Purple Potato Compound Freeze-Dried Fruit Blocks. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2024**, *40*, 276–285. [CrossRef]
2. Ding, H.X.; Li, M.T.; Peng, J.; Liu, Z.J. Experimental Study on the Vibration Parameters of Mulberry Picking. *J. Agric. Mech. Res* **2016**, *10*, 183–186. [CrossRef]
3. Lu, J.; Chen, P.; Yu, C.; Lan, Y.; Yu, L.; Yang, R.; Niu, H.; Chang, H.; Yuan, J.; Wang, L. Lightweight Green Citrus Fruit Detection Method for Practical Environmental Applications. *Comput. Electron. Agric.* **2023**, *215*, 108205. [CrossRef]
4. Zhonghua, M.; Yichou, S.; Xiaohua, W.; Xiaofeng, Z.; Chengliang, L. Image Recognition Algorithm and Experiment of Overlapped Fruits in Natural Environment. *Nongye Jixie Xuebao/Trans. Chin. Soc. Agric. Mach.* **2016**, *47*, 21–26. [CrossRef]
5. Zhuang, J.J.; Luo, S.M.; Hou, C.J.; Tang, Y.; He, Y.; Xue, X.Y. Detection of Orchard Citrus Fruits Using a Monocular Machine Vision-Based Method for Automatic Fruit Picking Applications. *Comput. Electron. Agric.* **2018**, *152*, 64–73. [CrossRef]
6. Jia, W.; Meng, H.; Ma, X.; Zhao, Y.; Ji, Z.; Zheng, Y. Efficient Detection Model of Green Target Fruit Based on Optimized Transformer Network. *J. Agric. Eng.* **2021**, *37*, 163–170. [CrossRef]
7. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep Learning in Agriculture: A Survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [CrossRef]
8. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
9. Tang, Y.; Chen, M.; Wang, C.; Luo, L.; Li, J.; Lian, G.; Zou, X. Recognition and Localization Methods for Vision-Based Fruit Picking Robots: A Review. *Front. Plant Sci.* **2020**, *11*, 510. [CrossRef]
10. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]
11. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
12. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Proceedings of the Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 21–37.
13. He, F.; Guo, Y.; Gao, C.; Chen, J. Image Segmentation of Ripe Mulberries Based on Visual Saliency and Pulse Coupled Neural Network. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2017**, *33*, 148–155. [CrossRef]

14. Ashtiani, S.H.M.; Javanmardi, S.; Jahanbanifard, M.; Martynenko, A.; Verbeek, F.J. Verbeek Detection of Mulberry Ripeness Stages Using Deep Learning Models. *IEEE Access* **2021**, *9*, 100380–100394. [CrossRef]
15. Wang, L.; Qin, M.; Lei, J.; Wang, X.; Tan, K. Blueberry Maturity Recognition Method Based on Improved YOLOv4-Tiny. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2021**, *37*, 170–178. [CrossRef]
16. Gai, R.; Chen, N.; Yuan, H. A Detection Algorithm for Cherry Fruits Based on the Improved YOLO-v4 Model. *Neural Comput. Appl.* **2023**, *35*, 13895–13906. [CrossRef]
17. Gao, C.; Jiang, H.; Liu, X.; Li, H.; Wu, Z.; Sun, X.; He, L.; Mao, W.; Majeed, Y.; Li, R.; et al. Improved Binocular Localization of Kiwifruit in Orchard Based on Fruit and Calyx Detection Using YOLOv5x for Robotic Picking. *Comput. Electron. Agric.* **2024**, *217*, 108621. [CrossRef]
18. Zhang, J.; Xie, J.; Zhang, F.; Gao, J.; Yang, C.; Song, C.; Rao, W.; Zhang, Y. Greenhouse Tomato Detection and Pose Classification Algorithm Based on Improved YOLOv5. *Comput. Electron. Agric.* **2024**, *216*, 108519. [CrossRef]
19. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
20. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6848–6856.
21. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. GhostNet: More Features From Cheap Operations. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 1577–1586.
22. Vadera, S.; Ameen, S. Methods for Pruning Deep Neural Networks. *IEEE Access* **2022**, *10*, 63280–63300. [CrossRef]
23. Gholami, A.; Kim, S.; Zhen, D.; Yao, Z.; Mahoney, M.; Keutzer, K. A Survey of Quantization Methods for Efficient Neural Network Inference. In *Low-Power Computer Vision*; Chapman and Hall/CRC: New York, NY, USA, 2022; pp. 291–326, ISBN 978-1-00-316281-0.
24. Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge Distillation: A Survey. *Int. J. Comput. Vis.* **2021**, *129*, 1789–1819. [CrossRef]
25. Zeng, T.; Li, S.; Song, Q.; Zhong, F.; Wei, X. Lightweight Tomato Real-Time Detection Method Based on Improved YOLO and Mobile Deployment. *Comput. Electron. Agric.* **2023**, *205*, 107625. [CrossRef]
26. Wu, X.; Tang, R.; Mu, J.; Niu, Y.; Xu, Z.; Chen, Z. A Lightweight Grape Detection Model in Natural Environments Based on an Enhanced YOLOv8 Framework. *Front. Plant Sci.* **2024**, *15*, 1407839. [CrossRef] [PubMed]
27. Zhao, J.; Du, C.; Li, Y.; Mudhsh, M.; Guo, D.; Fan, Y.; Wu, X.; Wang, X.; Almodfer, R. YOLO-Granada: A Lightweight Attentioned Yolo for Pomegranates Fruit Detection. *Sci. Rep.* **2024**, *14*, 16848. [CrossRef]
28. Liu, Z.; Rasika, D.; Abeyrathna, R.M.; Mulya Sampurno, R.; Massaki Nakaguchi, V.; Ahamed, T. Faster-YOLO-AP: A Lightweight Apple Detection Algorithm Based on Improved YOLOv8 with a New Efficient PDWConv in Orchard. *Comput. Electron. Agric.* **2024**, *223*, 109118. [CrossRef]
29. Xie, T.; Cheng, X.; Wang, X.; Liu, M.; Deng, J.; Zhou, T.; Liu, M. Cut-Thumbnail: A Novel Data Augmentation for Convolutional Neural Network. In Proceedings of the 29th ACM International Conference on Multimedia, Online, 20–24 October 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 1627–1635.
30. Jocher, G.; Chaurasia, A.; Qiu, J. *YOLO by Ultralytics, v. 8.0.0*; Ultralytics: Los Angeles, CA, USA, 2023. Available online: https://github.com/ultralytics/ultralytics (accessed on 26 November 2024).
31. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 7464–7475.
32. Chen, J.; Kao, S.; He, H.; Zhuo, W.; Wen, S.; Lee, C.-H.; Chan, S.-H.G. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 12021–12031.
33. Duan, X.; Zhang, B.; Deng, Q.; Ma, H.; Yang, B. Research on Small Objects Detection Algorithm of UAV Photography Based on Improved YOLOv7. *Preprint* **2024**. [CrossRef]
34. Wang, C.-Y.; Yeh, I.-H.; Mark Liao, H.-Y. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. In *European Conference on Computer Vision*; Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G., Eds.; Springer Nature Switzerland: Cham, Switzerland, 2025; pp. 1–21.
35. Shu, C.; Liu, Y.; Gao, J.; Yan, Z.; Shen, C. Channel-Wise Knowledge Distillation for Dense Prediction. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 11–17 October 2021; pp. 5291–5300.
36. Lin, J.-D.; Wu, X.-Y.; Chai, Y.; Yin, H.-P. Structure Optimization of Convolutional Neural Networks: A Survey. *Acta Autom. Sin.* **2020**, *46*, 24–37. [CrossRef]
37. Keqi, C.; Zhiliang, Z.; Xiaoming, D.; Cuixia, M.; Hongan, W. Deep Learning for Multi-Scale Object Detection: A Survey. *J. Softw.* **2021**, *32*, 1201–1227. [CrossRef]
38. Singh, P.; Verma, V.K.; Rai, P.; Namboodiri, V.P. HetConv: Heterogeneous Kernel-Based Convolutions for Deep CNNs. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 4830–4839.
39. Zhong, J.; Chen, J.; Mian, A. DualConv: Dual Convolutional Kernels for Lightweight Deep Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 9528–9535. [CrossRef] [PubMed]

40. Han, K.; Wang, Y.; Guo, J.; Wu, E. ParameterNet: Parameters Are All You Need for Large-Scale Visual Pretraining of Mobile Networks. In Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–22 June 2024; pp. 15751–15761.
41. Li, J.; Wen, Y.; He, L. SCConv: Spatial and Channel Reconstruction Convolution for Feature Redundancy. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 6153–6162.