agriculture

MDPI

Article

Lightweight-Improved YOLOv5s Model for Grape Fruit and Stem Recognition

Junhong Zhao [1,2], Xingzhi Yao [1], Yu Wang [1,*], Zhenfeng Yi [1], Yuming Xie [2] and Xingxing Zhou [2]

[1] College of Engineering, South China Agricultural University, Guangzhou 510642, China; zhaojunhong@gdaas.cn (J.Z.); yaoxingzhi@stu.scau.edu.cn (X.Y.); yizhenfeng@stu.scau.edu.cn (Z.Y.)
[2] Institute of Facility Agriculture, Guangdong Academy of Agricultural Sciences, Guangzhou 510640, China; xieyuming@gdaas.cn (Y.X.); zhouxingxing@gdaas.cn (X.Z.)
* Correspondence: yu-wang@scau.edu.cn

Abstract: Mechanized harvesting is the key technology to solving the high cost and low efficiency of manual harvesting, and the key to realizing mechanized harvesting lies in the accurate and fast identification and localization of targets. In this paper, a lightweight YOLOv5s model is improved for efficiently identifying grape fruits and stems. On the one hand, it improves the CSP module in YOLOv5s using the Ghost module, reducing model parameters through ghost feature maps and cost-effective linear operations. On the other hand, it replaces traditional convolutions with deep convolutions to further reduce the model's computational load. The model is trained on datasets under different environments (normal light, low light, strong light, noise) to enhance the model's generalization and robustness. The model is applied to the recognition of grape fruits and stems, and the experimental results show that the overall accuracy, recall rate, mAP, and F1 score of the model are 96.8%, 97.7%, 98.6%, and 97.2% respectively. The average detection time on a GPU is 4.5 ms, with a frame rate of 221 FPS, and the weight size generated during training is 5.8 MB. Compared to the original YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x models under the specific orchard environment of a grape greenhouse, the proposed model improves accuracy by 1%, decreases the recall rate by 0.2%, increases the F1 score by 0.4%, and maintains the same mAP. In terms of weight size, it is reduced by 61.1% compared to the original model, and is only 1.8% and 5.5% of the Faster-RCNN and SSD models, respectively. The FPS is increased by 43.5% compared to the original model, and is 11.05 times and 8.84 times that of the Faster-RCNN and SSD models, respectively. On a CPU, the average detection time is 23.9 ms, with a frame rate of 41.9 FPS, representing a 31% improvement over the original model. The test results demonstrate that the lightweight-improved YOLOv5s model proposed in the study, while maintaining accuracy, significantly reduces the model size, enhances recognition speed, and can provide fast and accurate identification and localization for robotic harvesting.

Keywords: YOLOv5s; lightweight; target detection; mechanized picking; grape fruits and stems

Citation: Zhao, J.; Yao, X.; Wang, Y.; Yi, Z.; Xie, Y.; Zhou, X. Lightweight-Improved YOLOv5s Model for Grape Fruit and Stem Recognition. *Agriculture* **2024**, *14*, 774. https://doi.org/10.3390/agriculture14050774

Academic Editor: Massimo Cecchini

Received: 27 March 2024
Revised: 8 May 2024
Accepted: 12 May 2024
Published: 17 May 2024

1. Introduction

With the increase in the scale and yield of grape cultivation, mechanized harvesting has become crucial in addressing issues such as high labor costs and low efficiency. Grapes, as an important economic fruit, primarily rely on manual labor for mechanical and repetitive harvesting, which not only demands a substantial workforce but also results in rising labor costs. Currently, the development of smart agriculture has made mechanized fruit harvesting a key means to address the shortage of agricultural labor and high labor costs, with the application of machine vision and related algorithms enhancing the efficiency, intelligence, and remote interaction of harvesting robots in agricultural environments. However, one of the major bottlenecks in the widespread adoption of mechanized harvesting technology is the low target detection and recognition rate of grape fruits and stems during

the mechanical harvesting process [1]. Achieving rapid and accurate recognition of the target being harvested is crucial for improving harvesting efficiency. In recent years, with the development of computer vision technology, research in the field of fruit-picking robot vision can mainly be categorized into target detection based on digital image processing and deep learning methods [2].

Digital image processing-based target detection methods primarily achieve target recognition by integrating various algorithms for image segmentation, enhancement, and feature extraction. They utilize important features such as colors and edge information of the target in the image to perform object detection [3]. Additionally, when clustering a specific color component in different color spaces (e.g., RGB, HSV, HIS, YUV, etc.), target segmentation can be performed, followed by post processing using morphological methods to achieve target detection [4–7]. However, traditional digital image processing methods are limited in their feature extraction capabilities, and they cannot extract deep features from images, making them inadequate for accurate identification of grape fruits and stems in complex agricultural scenarios.

In contrast to traditional image processing methods, deep learning methods employ deep neural networks with strong feature extraction and self-learning capabilities, and they are widely used for fruit detection in agricultural production environments, including grapes [6,7], apples [8,9], mangoes [10], tomatoes [11], and more. Deep neural networks mainly involve two types: two-stage object detection networks focused on detection accuracy, such as Faster-RCNN [8,12–14] and Mask-RCNN [15], and one-stage object detection networks emphasizing detection speed, such as YOLOv3 [16], YOLOv4 [10,11,17–19], YOLOv5 [9,20–23], and SSD [24,25]. One-stage object detection networks can directly generate prediction boxes in images, significantly boosting detection speed. With the application of attention mechanisms in the field of computer vision, one-stage object detection networks, like the YOLO series, have widely incorporated channel attention (CA) [9,18,21], spatial attention (SA) [17], and convolutional block attention module (CBAM) [22,23] in fruit target detection. Moreover, combining one-stage detection networks with traditional image processing methods, such as K-means clustering [11,20], image enhancement [16], and HSV color space [11,19], can result in better feature extraction. For instance, Chen et al. [20] used YOLOv5 with K-means++ clustering for automatic selection of target datasets and introduced coordinated attention modules to enhance feature extraction for occluded camellia oleifera fruit, achieving a mean average precision (mAP) of 94.10% and a frame rate of 74.8 FPS with a model size of 27.1 MB. Li et al. [19] combined the YOLOv4 and HSV methods for recognizing ripe tomatoes, achieving a recognition accuracy of 94.77%. Additionally, Zhang et al. [11] used YOLOv4 to identify tomato clusters and extract stem images by converting them to the HSV color space using the K-means clustering algorithm. Chen et al. [24] introduced the MobileNetV3 lightweight module in SSD to enable the recognition of tomato flowers and fruits with an average recognition rate of 92.57% and a recognition speed of 0.079 s per frame.

However, the majority of the aforementioned studies predominantly focus on improving recognition accuracy. An increase in accuracy often implies an escalation in model parameters and computational load, leading to a decrease in detection speed. In order to achieve a balance between detection speed and accuracy, some scholars have delved into the field of model lightweighting. Yu Sun et al. [26] employed partial convolution (PConv) to reduce floating-point operations and create a lightweight network. By building upon YOLOVX-Tiny, Ji et al. [27] utilized Shufflenetv2 as the backbone network for apple detection. Zeng et al. [28] replaced CSPDarknet with Mobilebetv3 to lightweightly enhance YOLOv5s, reducing model computational load. Bin Zhang [29], Rui Ren [30], and others achieved YOLO model lightweighting by introducing the Ghost module, resulting in improved models with sizes of 11.5 MB and 13.93 MB, respectively. Jiqing Chen [31] designed a novel backbone feature extraction network, SE-CSPGhostnet, significantly reducing model parameters, with the improved weight size being 32.5 MB. While these lightweighting studies effectively reduce model size, they also, to some extent, sacrifice

model accuracy. Furthermore, the post-improvement model memory footprint remains relatively large, and corresponding validations on pure CPU devices are lacking. There is limited exploration of the issue of model lightweighting, which can reduce computational load during recognition, accelerate detection speed, and simultaneously decrease model size.

In summary, the YOLO model, owing to its fast detection speed, represents the primary direction for target detection. However, existing YOLO model research primarily considers the trade-off between real-time detection speed and accuracy, making it challenging to meet the multifaceted requirements of mechanized harvesting for detection accuracy, speed, and model size. For mechanized harvesting, it is essential to maintain detection accuracy while achieving fast detection speed and simultaneously reducing the model's size for deployment on edge devices. Therefore, this paper proposes a lightweight-improved YOLOv5s model, focusing on lightweighting the CSP module and convolutional layers within YOLOv5s. The CSP module in YOLOv5s is improved by using the Ghost module, and the model parameters are reduced by ghost feature maps and less costly linear operations in the Ghost module. Moreover, the traditional convolution in the original network is replaced by deep convolution to further reduce the computations of the model, and the computations of different environments (normal light, low light, strong light, noise) to improve the generalization ability and robustness of the model. Finally, the proposed model is experimentally validated for target detection performance by identifying grape fruits and stems on both GPU and CPU platforms.

2. Materials and Methods

2.1. Image Acquisition and Augmentation

The grape fruit images used in this study were captured in the standardized grape orchard shown in Figure 1, located at Baiyun Base in Guangdong Academy of Agricultural Sciences, Guangzhou City, Guangdong Province, 23°23′ N, 113°26′ E. The image acquisition equipment used was an Olympus TG-4 digital camera (resolution 3200 × 2400 pixels, focal length 4.5–18 mm, shutter speed 1/2-1/2000 s, ISO100–6400), on 24 May 2022, the collection time was from 9:00 to 15:00, and the shooting distance was approximately 500–1000 mm. A total of 564 images of Shine Muscat in the greenhouse environment were collected under clear weather conditions.

Figure 1. Standardized orchard environment.

To improve the model's robustness and its ability to recognize grape fruits in different environments and prevent overfitting due to a lack of data, Gaussian noise was added to the collected images, and brightness levels were adjusted to simulate different weather like foggy environment (noise interference), overcast (low-light), and high-light conditions. Under different lighting conditions, images may contain varying feature information, which contributes to the model learning richer and more robust feature representations. By simulating real-world lighting variations, this enhances recognition accuracy. Gaussian noise is a common image degradation phenomenon. By introducing noise during training, the model can learn how to recover clear image features from noise, thus better handling noisy images in practical applications. After data augmentation, there were 2256 images, and the dataset was divided into a training set and a validation set in an 8:2 ratio.

2.2. Image Annotation

The collected images comprised only clear grape fruit clusters in the foreground, while distant, blurry, and smaller images were disregarded. The gathered images were imported into PaddlePaddle EasyDL to label a part of the images manually (about 100–200 images), the ground truth contained the identified targets as comprehensively as possible to ensure the quality of the dataset. EasyDL learns from the manually labeled images in the initial stage to automatically annotate the remaining images. Subsequently, the labeled images were subjected to scrutiny using the image labeling software, LabelImg (App version: 1.8.6), in which any inaccuracies in the automatic annotation results were rectified through manual modification. Compared to fully manual labeling, automatic labeling achieves better accuracy and efficiency. The accuracy is significantly improved, resulting in fewer images requiring manual correction after automatic labeling. Consequently, an XML file conforming to the YOLO data format was generated. This file included the name of the image, the count of labels within the image, the label names, and the corresponding x and y coordinates for each label, representing the ground truth. The label names were divided into two categories: "grapes" for grape fruit clusters and "stem" for grape stems. The numbers of annotated labels for grapes and stems were 8169 and 6933, respectively. The annotated label files were also divided into training and validation sets in an 8:2 ratio, following the format of the PASCAL VOC dataset to create the dataset for this study.

2.3. Lightweight Improvement of YOLOv5s

2.3.1. YOLOv5s Model

YOLOv5 is a one-stage object detection model released by Ultralytics on June 9, 2020. It is characterized by smaller average weight files, shorter training times, and faster inference speeds, with a relatively small decrease in average detection accuracy. To date, YOLOv5 has been updated to version 7.0. This study used YOLOv5 version 6.0, which consists of four main parts: input, backbone, neck, and prediction. Mosaic data augmentation is applied to the input to enhance model generalization and robustness. YOLOv5 also introduces an adaptive image scaling module to add minimal padding during resizing to the standard size while preserving the original aspect ratio. The backbone uses the CSPDarknet53 structure to enhance feature extraction capacity, reduce computation, and memory usage while maintaining accuracy. The neck employs an FPN + PAN structure [32], with FPN extracting strong semantic features and PAN extracting strong positional features. These features are integrated for predicting objects at three different scales. CSP2 is used within the neck to improve feature fusion capability. Prediction uses CIoU_Loss [33] as the bounding box loss function to enhance speed and accuracy of regression prediction.

YOLOv5 version 6.0 includes the YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x models. Although YOLOv5n has the smallest number of parameters, its accuracy is lower than other models. This study focused on improving the lightweight YOLOv5s model with fewer parameters while maintaining model accuracy. In existing lightweight improvement schemes for detection models, there is often a trade-off between detection speed and accuracy. However, the lightweight solution proposed in this study

can significantly improve the model's detection speed while ensuring an increase in P and maintaining mAP50 unchanged. The improvements are made to the CSP module and convolutional layers using GhostBottleneck for CSP and deep convolution to replace traditional convolution.

2.3.2. Ghost Module for CSP Improvement

The Ghost module [34] can generate more feature maps at a low cost by applying a series of low-cost linear transformations to a set of inherent feature maps. During the convolution operation, many feature maps are generated, and some of these feature maps are similar. These similarities can be exploited through simple linear operations in the Ghost module, which significantly reduce computational costs compared to convolution operations. In the CSP model, the underlying structure still consists of stacked convolutional layers and residual connections, and convolutional operations generate a large number of redundant feature maps. The Ghost module can reduce the generation of redundant feature maps through linear operations, thereby reducing computational costs.

Deep convolutional neural networks often consist of a large number of convolutions, increasing computational costs. Although there are studies using Shuffle operations [35] to construct efficient neural networks by smaller convolutional kernels, convolutional layers with a convolutional kernel size of 1×1 still incur significant memory usage and FLOPs. In mainstream neural network computations, there are similar features present in the intermediate feature maps. Reducing the convolutional kernels used to generate intermediate feature maps with similar features can decrease the required resources. Given input data $X \in \mathbb{R}^{c \times h \times w}$, where c is the number of input channels, and h and w are the height and width of the input data, respectively, the arbitrary convolutional layer used to generate n feature maps can be expressed as:

$$Y = X * f + b \tag{1}$$

where * represents the convolution operation; b represents the bias term; $f \in \mathbb{R}^{c \times k \times k \times n}$ is the convolutional filter in this layer; $Y \in \mathbb{R}^{h' \times w' \times n}$ is the output feature map with n channels; h' and w' are the height and width of the output data, respectively; and $k \times k$ is the kernel size of the convolutional filter. During the convolution process, the required FLOPs can be calculated as $n \cdot h' \cdot w' \cdot c \cdot k \cdot k$.

As shown in Equation (1), the dimensions of the input and output feature maps determine the number of parameters (in f and b) to be optimized.

It can be seen in Figure 2, the output feature maps of convolutional layers typically contain a large amount of redundancy, some of which may be similar to each other. The redundant feature maps can be obtained through low-cost transformations, which typically have smaller sizes and are generated by ordinary convolutional kernels. Specifically, for input data $X \in \mathbb{R}^{c \times h \times w}$, when using basic convolution to generate m intrinsic feature maps $Y' \in \mathbb{R}^{h' \times w' \times m}$,

$$Y' = X' * f' + b \tag{2}$$

where $f' \in \mathbb{R}^{c \times k \times k \times m}$ is the convolutional filter used to obtain the desired n feature maps, in which $m \leq n$, and a series of low-cost linear operations are performed on each feature map according to Equation (3) to generate s ghost features.

$$y_{ij} = \Phi_{i,j}\left(y_i'\right), \forall i = 1, ..., m, j = 1, ..., s \tag{3}$$

where y_i' is the ith intrinsic feature in Y'; and $\Phi_{i,j}$ is the jth (excluding the last one) linear operation that generates the jth ghost feature map y_{ij}, that is, y_i' can be one or more ghost feature maps $\left\{y_{ij}\right\}_{j=1}^{s}$, and the last one, $\Phi_{i,s}$, is an identity map used to maintain the intrinsic feature map. As shown in Figure 2, there are $n = m \cdot s$ feature maps $Y = [y_{11}, y_{12}, \cdots, y_{ms}]$ obtained via Equation (2) as the output data of the Ghost module, and the computational cost of linear operations Φ on each channel is much lower than that of ordinary convolutions.

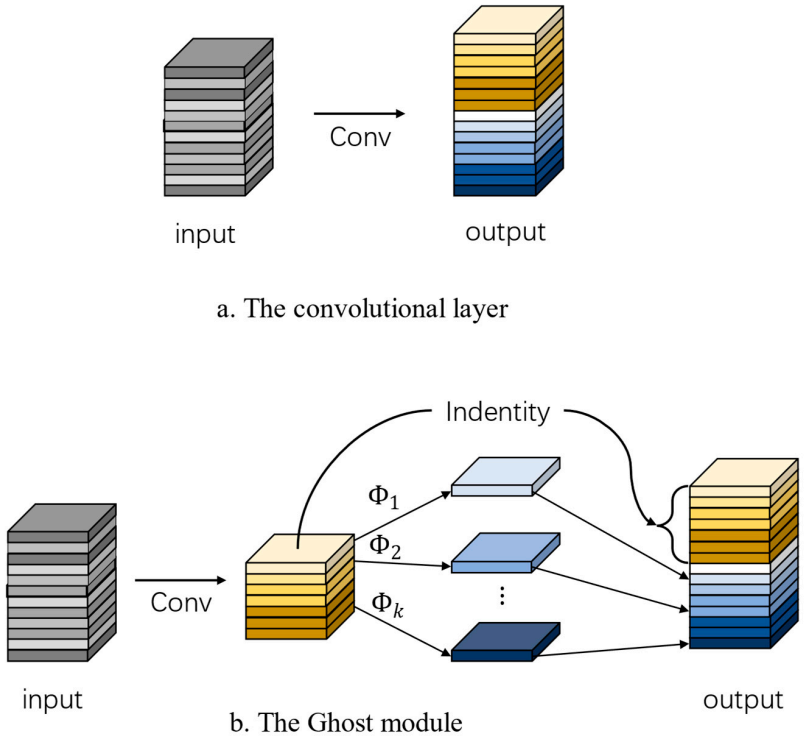


Figure 2. Ghost module.

To utilize the advantages of the Ghost module, GhostBottleneck stacks two Ghost modules, similar to the basic residual blocks in ResNet [36]. The first Ghost module serves as an expansion layer, increasing the number of channels, while the second Ghost module reduces the number of channels for shortcut connections between the input and output. Batch normalization (BN) [37] and ReLU non-linearity are applied after each layer, according to the suggestion from MobileNetV2 [38] to avoid ReLU after the second Ghost module.

The stride of GhostBottleneck is set to 1, and when the stride is 2, shortcut connections are implemented through downsampling layers. Meanwhile, a deep direction convolution (DWConv) [39] is inserted between two Ghost modules with a stripe of 2. In practice, the Ghost module predominantly uses depthwise convolution for pointwise convolution to improve efficiency. Multiple GhostBottlenecks are stacked to form GhostCSP_n, as shown in Figure 3.

2.3.3. Depthwise Separable Convolution for Improved Convolutional Layers

Depthwise separable convolution (DSC) is constructed by combing depthwise convolution (DWC) and pointwise convolution (PWC). This structure is illustrated in Figure 4. DSC is similar to regular convolution in feature extraction but differs in that DWC focuses on the spatial dimensions, while PWC focuses on the channel dimensions.

In regular convolution operations, a 3×3 kernel, for instance, performs convolutions on both channel and spatial dimensions. For a three-channel color image, a $3 \times 3 \times 3 \times 4$ kernel is required to generate four feature maps. However, DSC splits the channel and spatial dimensions into two separate steps. DWC operates on the channel dimension and uses a 3×3 kernel on each channel individually. This produces three feature maps, each generated by convolving the corresponding RGB channel. PWC then operates on the spatial dimension and uses $1 \times 1 \times N \times M$ kernels, where N is the number of channels from the previous convolution, and M is the number of output feature maps. PWC combines the depthwise feature maps, weighted by the depthwise direction, to generate new feature maps. The combination of these lightweight strategies for improving the YOLOv5s model results in the network structure shown in Figure 5.

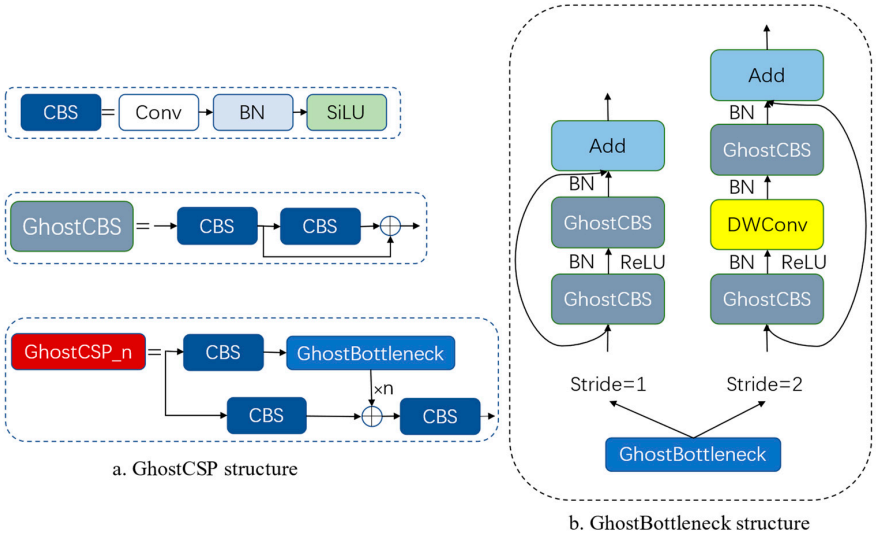


Figure 3. GhostCSP and GhostBottleneck structure.

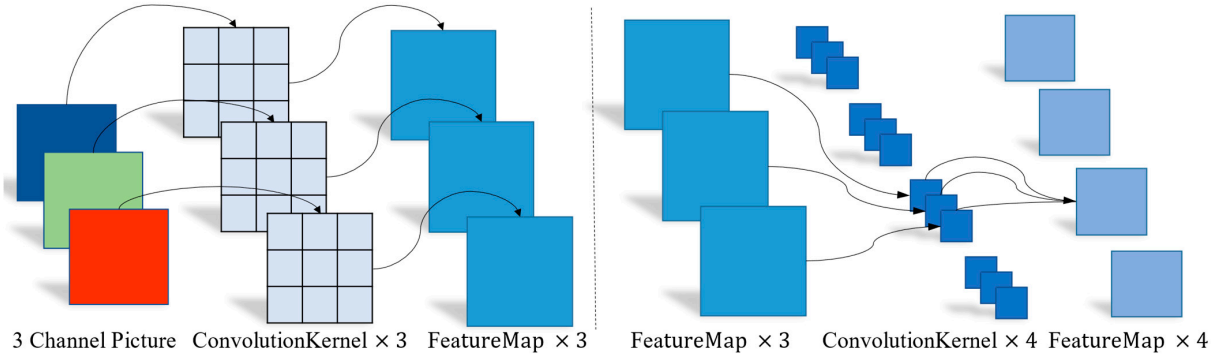


Figure 4. Depth separable convolution Left: Depthwise convolution Right: Pointwise convolution.

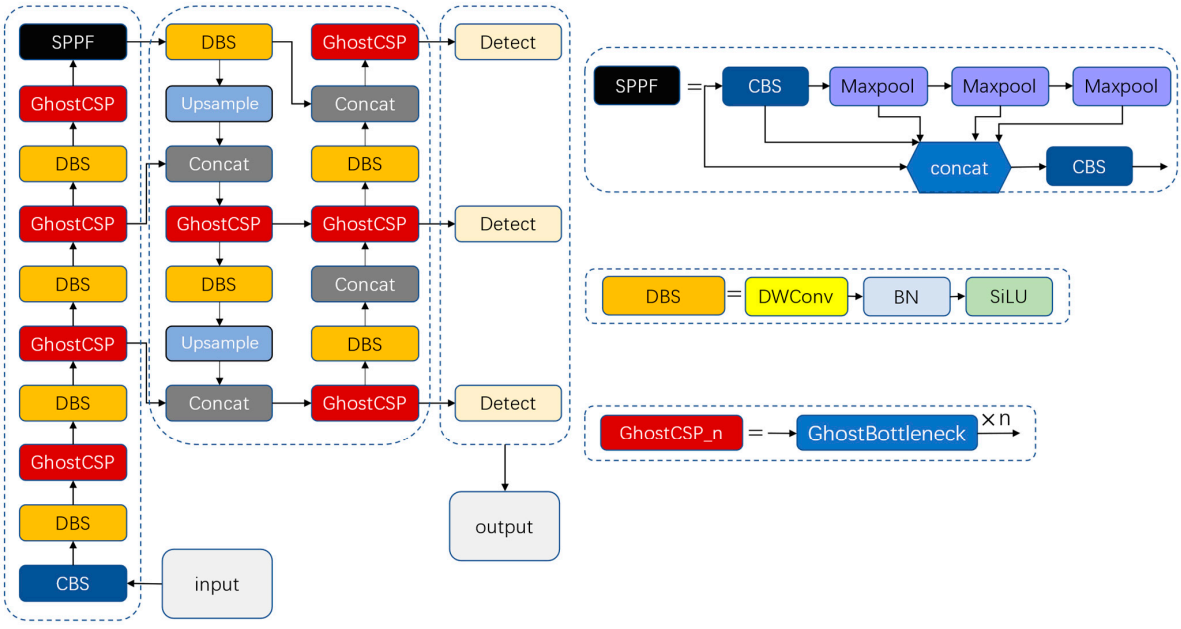


Figure 5. Improved YOLOv5s network structure.

2.4. Training and Evaluation

2.4.1. Training Platform

Both the training and testing of this research were conducted on the same computer with the following hardware and software specifications: an Intel(R) Core (TM) i7-11700K @3.6 GHz desktop computer with 16 cores, 64 GB of RAM, an NVIDIA GeForce RTX 3090 GPU with 24 GB of VRAM, running a 64-bit Ubuntu 18.04 operating system. The deep learning environment used PyTorch, Python version 3.8.16, CUDA version 11.1, OpenCV version 4.7.0.68, and the Python IDE was PyCharm Community Edition 2022.

2.4.2. Model Training

The improved YOLOv5s model was trained using the Adam optimization algorithm. Pretrained weights of YOLOv5s on the COCO dataset were utilized, in which the batch size for model training was set to 32, and batch normalization (BN) was applied at each training step for weight regularization. The initial learning rate was 0.01, weight decay was 0.0005, momentum was set to 0.937, and mosaic augmentation was used during training. The hue (H), saturation (S), and brightness (V) augmentation coefficients were set to 0.015, 0.7, and 0.4, respectively. The number of training epochs was 200, the IoU threshold was set to 0.2, and the input image size was 1280 × 1280. After training, the trained model's weights were saved, and the model's precision (P), recall (R), mean average precision (mAP), and frame rate were evaluated. The changes in various loss values during training are shown in Figure 6, and the changes in the P-R and F1 score are illustrated in Figure 7.

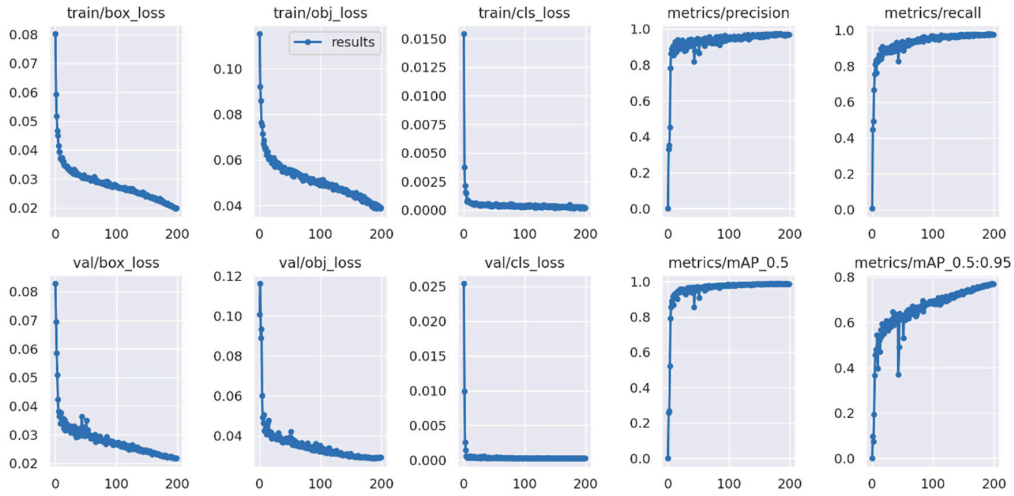


Figure 6. Improved YOLOv5 training results curve.

2.4.3. Model Validation and Testing

The trained grape fruit and stem recognition model was evaluated using metrics such as precision, recall, mean average precision (mAP), and F1 score. The evaluation was performed on both the fruit and stem recognition model using the validation dataset. The evaluation metrics were calculated as follows:

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

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

$$mAP = \frac{1}{C}\sum_{k=i}^{N} P(k)\Delta R(k) \tag{6}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

In Equation (4), *TP* represents the number of correctly identified targets, and *FP* represents the number of false positives. In Equation (5), *FN* represents the number of false negatives. In Equation (6), *C* is the number of object categories, *N* is the number of IoU thresholds, *k* is an IoU threshold, *P*(*k*) is precision, and *R*(*k*) is recall. In Equation (7), *Precision* and *Recall* are derived from Equation (4) and Equation (5), respectively.

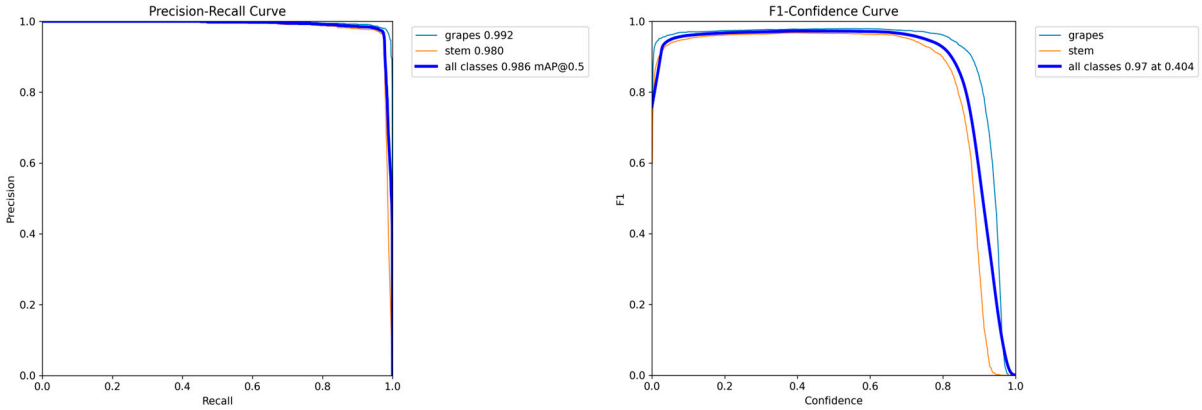


Figure 7. Left—P-R Graph, **Right**—F1 Graph.

3. Results

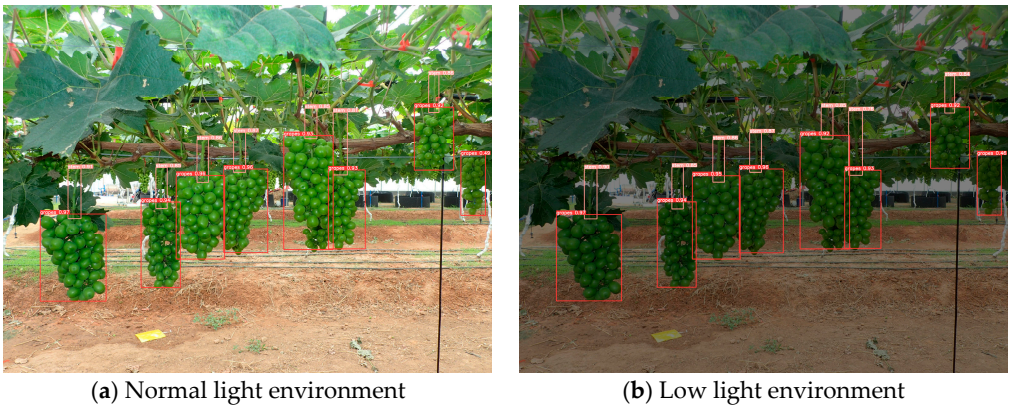
3.1. The Performance of Lightweight-Improved YOLOv5s

The performance of the lightweight-improved YOLOv5s object detection model was validated on a dataset of grape fruits and stems under various lighting conditions, including normal light, low light, strong light, and noisy environments. The validation results for the model on a set of 452 validation images revealed a total of 3129 recognized objects, with 1737 grape fruit (grapes) targets and 1392 stem (stem) targets. The specific recognition results are shown in Table 1, which demonstrate that, for grape fruits, the model achieved precision, recall, mAP, and F1 score of 97.1%, 98.5%, 99.2%, and 97.8%, respectively. For stems, the model achieved precision, recall, mAP, and F1 score of 96.5%, 96.8%, 98.0%, and 96.6%, respectively. The overall recognition accuracy, recall, mAP, and F1 score were 96.8%, 97.7%, 98.6%, and 97.2%, respectively. The average processing time for one image on a GPU was 4.5 ms, with a frame rate of 221 FPS. The weight size of the trained model was 5.8 MB. In terms of resource utilization, the parameters of the original model were 7,025,023, and the FLOPs were 16.0 GFLOPs. After improvement, the overall parameters of the model were 2,444,583, with FLOPs of 5.7 GFLOPs.

Table 1. Results of grape fruit and stem recognition by lightweight-improved YOLOv5s.

Class	Number of Pictures	Number of Targets	P (%)	R (%)	Map (%)	F1 (%)
Grapes	452	1737	97.1	98.5	99.2	97.8
Stems	452	1392	96.5	96.8	98.0	96.6
Total	452	3129	96.8	97.7	98.6	97.2

The performance of the model was evaluated using trained weights on the test set, and the test results showed that the position boxes and corresponding label names of grape clusters and fruit stems were identified in the images. Figure 8 shows the recognition results of the proposed model under normal lighting and simulated cloudy weak light, sunny strong light, and noise interference environments. The red box represents the fruits, and the pink box represents the stems. From the figure, it can be seen that the proposed model could not only achieve good recognition in cloudy weak light and sunny strong light environments, but it also recognized targets well in severe noise interference situations.



(**a**) Normal light environment (**b**) Low light environment

(**c**) Strong light environment (**d**) Noise environment

Figure 8. Recognition results under different environments.

3.2. Comparison of Recognition Results via Different Object Detection Models

To further verify the object recognition performance of the proposed model, the lightweight-improved YOLOv5s model was trained and validated on the same dataset as the original YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x, Faster R-CNN [40], and SSD [41] models. The input image size was uniformly scaled to 1280 × 1280 for all four models. Both the Faster R-CNN and SSD models utilize a ResNet50 and FPN network as their backbone. The four networks were compared based on metrics such as precision, recall, F1 score, mAP, weight size, and frame rate. The validation results on the validation dataset for each metric are shown in Table 2.

Table 2. Performance comparison of different target detection models.

Model	P (%)	R (%)	F1 (%)	mAP (%)	Weight (MB)	Frame Rate (FPS)
YOLOv5s	95.8	97.9	96.8	98.6	14.9	154
YOLOv5m	94.8	96.7	95.7	97.0	42.2	80
YOLOv5l	94.5	96.5	95.5	96.8	92.8	55
YOLOv5x	95.4	96.5	96.0	97.3	173.1	31
YOLOv7-tiny	94.9	95.5	95.2	97.8	12.3	161
Faster-RCNN	97.6	97.4	97.5	97.6	330.2	20
SSD	88.7	95.0	91.7	88.7	106.0	25
Our Model	96.8	97.7	97.2	98.6	5.8	221

According to Table 2, the lightweight-improved YOLOv5s model achieved a 1% increase in precision (P) compared to the original YOLOv5s, a 0.2% decrease in recall (R), a 0.4% increase in F1 score, and the same mAP. The weight size of the improved model was 5.8 MB, which was a 61.1% reduction compared to the original model. The frame rate was 221 FPS, which was a 43.5% improvement over the original model. Compared with YOLOv5's original model of different sizes, the improved model was better than

the large model in terms of precision, recall, and mAP. Furthermore, the improved model outperformed the Faster R-CNN and SSD models by 11.05 and 8.84 times in terms of frame rate, respectively. The results demonstrated that the proposed network achieved lightweight model deployment while effectively enhancing object detection efficiency without sacrificing recognition accuracy.

In order to more intuitively evaluate the recognition ability of different models in orchard environments, recognition tests needed to be conducted in orchard environments. The Azure Kinect DK (RGBD camera used by future grape harvesting robots) was used to capture images of the orchard environment and different models were used to recognize the collected images, and the recognition results are shown in Figure 9.

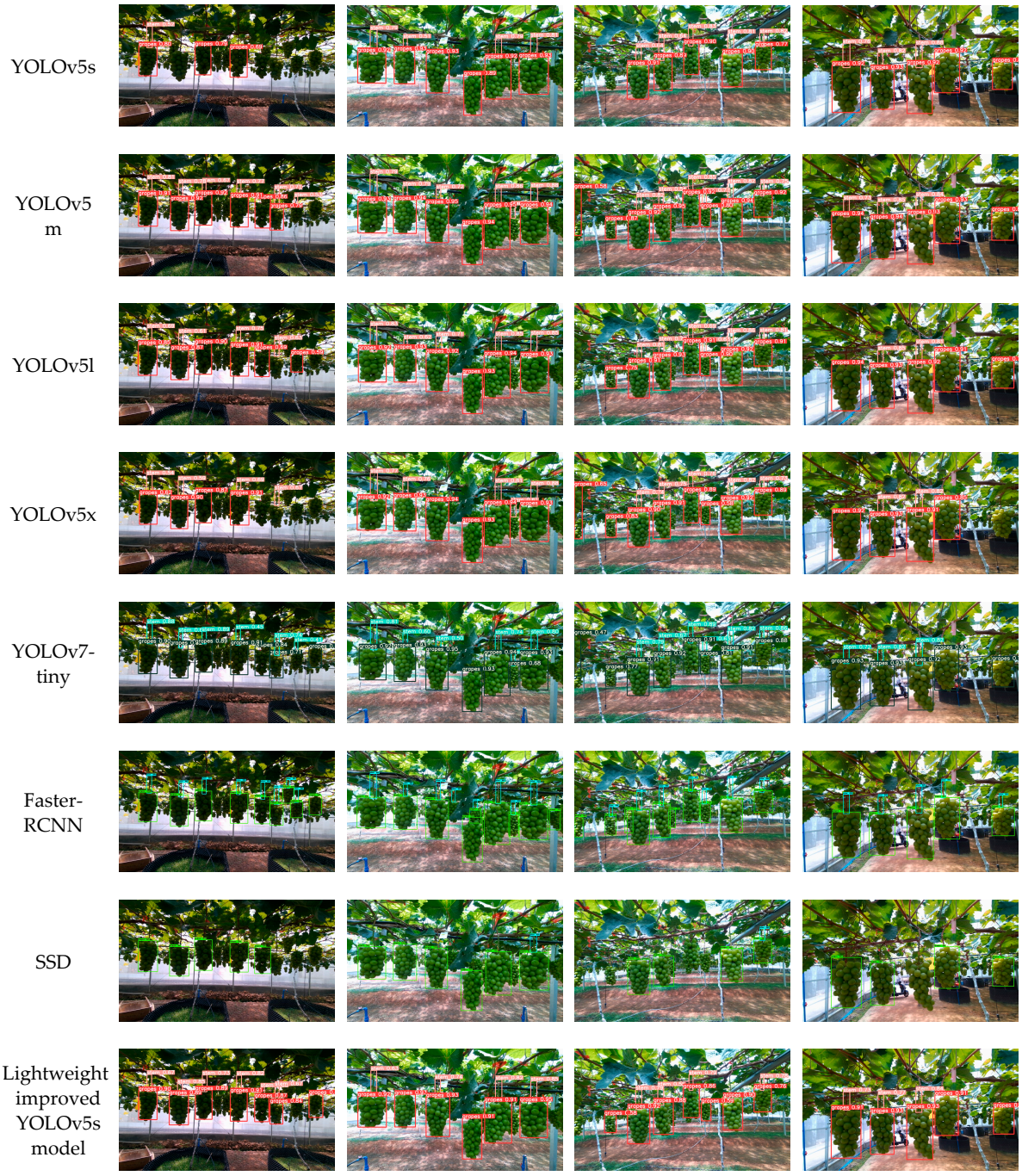


Figure 9. Recognition results of different models in orchard environments.

The recognition results of different models indicated that the two-stage object detection model Faster RCNN, which focuses on accuracy, had high confidence. However, two-stage models often require excessive computation and have slow detection speed, which cannot meet the requirements of real-time detection well. In a single-stage model, the SSD model could not recognize grape fruits and stems well, and the recognition results were relatively poor in different environments. The recognition results of the YOLOv5 series models were generally satisfactory, but existing lightweight improvements often tended to lower their detection accuracy. The lightweight-improved YOLOv5s model proposed in this paper demonstrated higher accuracy and completeness in detection compared to the original model (YOLOv5s). Moreover, its recognition results even surpassed those of larger models (YOLOv5m, YOLOv5l, YOLOv5x) and YOLOv7-tiny in certain scenarios.

3.3. Speed Comparison on Different CPUs

The goal of a lightweight model is to be applied in real-world scenarios. In the experiments described above, the frame rate was calculated on GPUs, but in practical applications, large and bulky computers are unlikely to be used in real-world scenarios. Additionally, the cost of GPUs is relatively high. The purpose of network lightweighting is to enable deployment on hardware with lower specifications, such as small computers (e.g., industrial PCs), portable laptops without GPUs, and other terminal devices. To validate the detection speed of the proposed model on different CPUs, the model was tested on CPUs without GPU acceleration. The performance on different CPUs is shown in Table 3.

Table 3. Parameters of four CPUs and results of model runs.

CPU	Number of Cores	Threads	Basic Frequency (GHz)	Detection Time (ms)	Frame Rate (FPS)	Original Model Frame Rate (FPS)
Intel® Core TM i7-11700K	8	16	3.60	23.9	41.9	32.0
AMD RyzenTM7 5800H	8	16	3.20	94.9	10.5	9.0
Intel® Core TM i5-8500	6	6	3.00	100.7	9.9	8.2
Intel® Core TM i5-10210U	4	8	1.60	155.6	6.4	4.4

On each computer, the same environment as the research model was set up. During testing, the default input image size was set to 640 × 640, IoU threshold to 0.45, and confidence threshold to 0.5. The trained weights were used for object detection. The time required to process each image was measured, and the frame rate was calculated. The frame rate for the improved model was compared to the original YOLOv5s model for each CPU. The results in Table 3 indicate that the Intel® Core TM i7-11700K, with the most cores and highest clock speed, achieved the fastest detection speed with an average processing time of 0.024 s and a frame rate of 41.9 FPS. The Intel® Core TM i5-10210U, with the fewest cores and lowest clock speed, had an average processing time of 0.156 s and a frame rate of 6.4 FPS. Compared to the original YOLOv5s model, the improved model achieved an increased frame rate of 31.0%, 16.7%, 20.7%, and 45.5% for the four different CPUs.

4. Discussion

The experimental comparison results of detection accuracy and detection speed show that the improved YOLOv5s model achieves a high level of accuracy and speed. In terms of accuracy, compared with other large volume models in the same YOLO family and SSD model, the improved YOLOv5s with small volume can achieve or even surpass the accuracy of large volume models. Compared with the two-stage object detection algorithm that pursues accuracy, its accuracy is also better than Faster-RCNN. In terms of speed, its detection speed is also better than mainstream lightweight models (YOLOv7-tiny). The analysis of its reasons mainly depend on the Ghost module. Although Ghost is a lightweight module, it does not involve discarding features or other operations. Instead, it cleverly uses linear operations to integrate similar feature maps generated during the convolution

process in a simpler and faster way, and fuses them with the features extracted from the convolution. By combining residual connections, shallow and deep features are combined. For convolution operations, DWConv reduces the computational complexity by three times compared to traditional convolution, further reducing computational complexity and improving detection speed.

Of course, the improvement of recognition accuracy should not only rely on the superiority of the model itself, but also be supported by a good dataset. A large number of research results have shown that establishing an image dataset in multiple environments can improve the recognition accuracy and robustness of deep learning models on the one hand, and on the other hand, the trained model can be deployed on mobile harvesting robots to achieve stable and reliable work. In real orchard scenes, the environment faced is often much more complex than the scenes in the trained images, often due to different lighting intensities. The similarity between the background and the color of the detected object affects the accuracy of recognition. Therefore, based on the pursuit of lightweight detection speed, establishing a dataset that is more similar to the environment can further improve detection accuracy.

In terms of cost, the total cost of a mobile harvesting robot is approximately RMB 60,000 to 80,000 (calculated at RMB 70,000). Based on labor costs, a farm worker's monthly salary is about RMB 6500 (according to the wage standard of the Institute of Facility Agriculture, Guangdong Academy of Agricultural Sciences). During the harvest period after grape ripening (about 2–3 months, calculated at 2.5 months), two workers are often required to work together. The annual labor cost is RMB 94,250, so cost control can be reduced by about 26%. In terms of picking time, current robots take about 20 s to pick a fruit, while manual picking relies on subjective judgment, cutting, and collection by workers. On average, collecting a bunch of grapes takes about 30 s, and robot harvesting can reduce time consumption by about 33%.

Furthermore, in our current work, we only detected ripe grape fruits. In reality, to achieve full cycle and automatic harvesting of fruits by robots, it is necessary to combine relevant technologies such as fruit maturity detection. In our previous work [42], we have achieved deep learning image semantic segmentation combined with hyper-spectral images for grape fruit maturity detection. The plan is to equip robots with hyperspectral cameras for collaborative operations in the future, achieving fully automatic fruit detection and maturity discrimination in picking ripe fruits.

5. Conclusions

To address the need on an efficient, accurate, and easily deployable grape fruit and stem recognition model for mechanized harvesting, the study builds a grape dataset under different lighting conditions, including normal light, low light, strong light, and noisy interference, and a lightweight YOLOv5s model is improved for identifying grape fruits and stems. To achieve lightweighting, the network architecture improves the original CSP module by replacing it with the GhostCSP module and upgrades traditional convolutions to DWC to further reduce parameters and computations. When applied to grape fruit and stem recognition under different environments, the experimental results show that the improved model effectively recognizes targets in various environments, improves detection speed without compromising detection accuracy, and reduces model weight size. The model's precision, recall, mAP, and F1 score are 96.8%, 97.7%, 98.6%, and 97.2%, respectively, with a model size of 5.8 MB, an average detection speed of 4.5 ms, and a frame rate of 221 FPS.

When compared to the original YOLOv5s, the improved model increases precision by 1%, decreases recall by 0.2%, improves F1 score by 0.4%, maintains the same mAP, reduces weight size by 61%, and increases the frame rate by 43.5%. When comparing the YOLOv5m, YOLOv5l, YOLOv5x, YOLOv7 tiny, Faster R-CNN, and SSD models, it was found that except for the precision and F1 score of Faster R-CNN, all other indicators were better than the above models. The detection speed of the improved model on four different

CPUs is compared to the original YOLOv5s model, showing improvements of 31.0%, 16.7%, 20.7%, and 45.5%, respectively. These results further validate the significant efficiency improvements of the model for object detection on different CPU platforms. The improved model proposed in this article has been validated for feasibility on harvesting robots, and in the future, multiple varieties of grapes can be considered as identification targets, providing a universal and reliable recognition model for the mechanized harvesting of grapes that will help to improve harvesting efficiency and alleviate the shortage of agricultural labor and high labor costs.

Author Contributions: Conceptualization, J.Z.; methodology, J.Z. and Y.W.; software, X.Y., Z.Y., Y.X. and X.Z.; writing—original draft, X.Y. and Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key-Area Research and Development Program of Guangdong Province (2023B0202090001), the National Natural Science Foundation of China (32372002), The Project of Collaborative Innovation Center of Guangdong Academy of Agricultural Sciences (XTXM202201), the Guangzhou Science and Technology Plan Project (2023A04J0830), and the Academic Team Construction Project of Guangdong Academy of Agricultural Sciences (202130TD).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data are available from the corresponding author.

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

References

1. Li, Y.; Feng, Q.; Li, T.; Xie, F.; Liu, C.; Xiong, Z. Advance of target visual information acquisition technology for fresh fruit robotic harvesting: A review. *Agronomy* **2022**, *12*, 1336. [CrossRef]
2. 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] [PubMed]
3. Lin, Y.; Lv, Z.; Yang, C.; Lin, P.; Chen, F.; Hong, J. Identification and experimentation of overlapping honeydew in natural scene images. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2021**, *37*, 158–167.
4. Chen, M.; Tang, Y.; Zou, X.; Huang, K.; Huang, Z.; Zhou, H.; Wang, C.; Lian, G. Three-dimensional perception of orchard banana central stock enhanced by adaptive multi-vision technology. *Comput. Electron. Agric.* **2020**, *174*, 105508. [CrossRef]
5. Syazwani, R.W.N.; Asraf, H.M.; Amin, M.A.M.S.; Dalila, N.K.A. Automated image identification, detection and fruit counting of top-view pineapple crown using machine learning. *Alex. Eng. J.* **2022**, *61*, 1265–1276. [CrossRef]
6. Lei, W.; Lu, J. Visual localization of picking points in grape picking robots. *J. Jiangsu Agric.* **2020**, *36*, 1015–1021.
7. Luo, L.; Zou, X.; Xiong, J.; Zhang, Y.; Peng, H.; Lin, G. Automatic positioning of grape picking robot picking points under natural environment. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2015**, *31*, 14–21.
8. Gao, F.; Fu, L.; Zhang, X.; Majeed, Y.; Li, R.; Zhang, Q. Multi-class fruit-on-plant detection for apple in SNAP system using Faster R-CNN. *Comput. Electron. Agric.* **2020**, *176*, 105634. [CrossRef]
9. Yan, B.; Fan, P.; Lei, X.; Liu, Z.; Yang, F. A real-time apple targets detection method for picking robot based on improved YOLOv5. *Remote Sens.* **2021**, *13*, 1619. [CrossRef]
10. Roy, A.M.; Bhaduri, J. Real-time growth stage detection model for high degree of occultation using DenseNet-fused YOLOv4. *Comput. Electron. Agric.* **2022**, *193*, 106694. [CrossRef]
11. Zhang, Q.; Chen, J.; Li, B.; Xu, C. Localization method of tomato bunch picking point identification based on RGB-D information fusion and target detection. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2021**, *37*, 143–152.
12. Sa, I.; Ge, Z.; Dayoub, F.; Upcroft, B.; Perez, T.; McCool, C. Deepfruits: A fruit detection system using deep neural networks. *Sensors* **2016**, *16*, 1222. [CrossRef]
13. Fu, L.; Majeed, Y.; Zhang, X.; Karkee, M.; Zhang, Q. Faster R–CNN–based apple detection in dense-foliage fruiting-wall trees using RGB and depth features for robotic harvesting. *Biosyst. Eng.* **2020**, *197*, 245–256. [CrossRef]
14. Yan, J.; Zhao, Y.; Zhang, L.; Su, X.; Liu, H.; Zhang, F.; Fan, W.; He, L. Improved Faster-RCNN for identifying prickly pear fruits under natural environment. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2019**, *35*, 144–151.
15. Yu, Y.; Zhang, K.; Yang, L.; Zhang, D. Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. *Comput. Electron. Agric.* **2019**, *163*, 104846. [CrossRef]
16. Xu, Z.; Jia, R.; Sun, H.; Liu, Q.; Cui, Z. Light-YOLOv3: Fast method for detecting green mangoes in complex scenes using picking robots. *Appl. Intell.* **2020**, *50*, 4670–4687. [CrossRef]
17. Qiu, C.; Tian, G.; Zhao, J.; Liu, Q.; Xie, S.; Zheng, K. Grape Maturity Detection and Visual Pre-Positioning Based on Improved Yolov4. *Electronics* **2022**, *11*, 2677. [CrossRef]

18. Huang, M.; Wu, Y. GCS-YOLOV4-Tiny: A lightweight group convolution network for multi-stage fruit detection. *Math. Biosci. Eng.* **2023**, *20*, 241–268. [CrossRef] [PubMed]
19. Li, T.; Sun, M.; Ding, X.; Li, Y.; Zhang, G.; Shi, G.; Li, W. A method based on YOLOv4+HSV for tomato identification at maturity. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2021**, *37*, 183–190.
20. Chen, S.; Zou, X.; Zhou, X.; Xiang, Y.; Wu, M. Study on fusion clustering and improved YOLOv5 algorithm based on multiple occlusion of Camellia oleifera fruit. *Comput. Electron. Agric.* **2023**, *206*, 107706. [CrossRef]
21. Duan, J.; Wang, Z.; Zou, X.; Yuan, H.; Huang, G.; Yang, Z. Identification of banana spikes and their bottom fruit axis positioning using improved YOLOv5. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2022**, *38*, 122–130.
22. Li, K.; Wang, J.; Jalil, H.; Wang, H. A fast and lightweight detection algorithm for passion fruit pests based on improved YOLOv5. *Comput. Electron. Agric.* **2023**, *204*, 107534. [CrossRef]
23. Sun, H.; Wang, B.; Xue, J. YOLO-P: An efficient method for pear fast detection in complex orchard picking environment. *Front. Plant Sci.* **2023**, *13*, 1089454. [CrossRef] [PubMed]
24. Chen, X.; Wu, P.; Zu, S.; Xu, D.; Zhang, Y.; Dong, Z. A farming recognition method for tomato flower and fruit thinning based on improved SSD lightweight neural network. *China Cucurbits Veg.* **2021**, *34*, 38–44.
25. Li, S.; Hu, D.; Gao, S.; Lin, J.; An, X.; Zhu, M. Real-time classification detection of citrus based on improved SSD. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2019**, *35*, 307–313.
26. Sun, Y.; Zhang, D.; Guo, X.; Yang, H. Lightweight algorithm for apple detection based on an improved YOLOv5 model. *Plants* **2023**, *12*, 3032. [CrossRef]
27. Ji, W.; Pan, Y.; Xu, B.; Wang, J. A real-time apple targets detection method for picking robot based on ShufflenetV2-YOLOX. *Agriculture* **2022**, *12*, 856. [CrossRef]
28. 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]
29. Zhang, B.; Wang, R.; Zhang, H.; Yin, C.; Xia, Y.; Fu, M.; Fu, W. Dragon fruit detection in natural orchard environment by integrating lightweight network and attention mechanism. *Front. Plant Sci.* **2022**, *13*, 1040923. [CrossRef]
30. Ren, R.; Sun, H.; Zhang, S.; Wang, N.; Lu, X.; Jing, J.; Xin, M.; Cui, T. Intelligent Detection of lightweight "Yuluxiang" pear in non-structural environment based on YOLO-GEW. *Agronomy* **2023**, *13*, 2418. [CrossRef]
31. Chen, J.; Ma, A.; Huang, L.; Su, Y.; Li, W.; Zhang, H.; Wang, Z. GA-YOLO: A lightweight YOLO model for dense and occluded grape target detection. *Horticulturae* **2023**, *9*, 443. [CrossRef]
32. 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, 21–25 July 2017; pp. 2117–2125.
33. Zheng, Z.; Wang, P.; Ren, D.; Liu, W.; Ye, R.; Hu, Q.; Zou, W. Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Trans. Cybern.* **2021**, *52*, 8574–8586. [CrossRef] [PubMed]
34. 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, virtually, 14–19 June 2020; pp. 1580–1589.
35. 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, 19–21 June 2018; pp. 6848–6856.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
37. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6–11 July 2015; pp. 448–456.
38. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 19–21 June 2018; pp. 4510–4520.
39. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–25 July 2017; pp. 1251–1258.
40. 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.* **2016**, *39*, 1137–1149. [CrossRef] [PubMed]
41. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.; Berg, A. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: Proceedings of the 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016, Proceedings, Part I 14*; Springer International Publishing: New York, NY, USA, 2016; pp. 21–37.
42. Zhao, J.; Hu, Q.; Li, B.; Xie, Y.; Lu, H.; Xu, S. Research on an Improved Non-Destructive Detection Method for the Soluble Solids Content in Bunch-Harvested Grapes Based on Deep Learning and Hyperspectral Imaging. *Appl. Sci.* **2023**, *13*, 6776. [CrossRef]