


Article

Lightweight Strawberry Instance Segmentation on Low-Power Devices for Picking Robots

Leilei Cao ¹, Yaoran Chen ² and Qiangguo Jin ^{1,*}¹ School of Software, Northwestern Polytechnical University, Xi'an 710072, China; caoleilei@nwpu.edu.cn² School of Artificial Intelligence, Shanghai University, Shanghai 200444, China; ychen169@shu.edu.cn

* Correspondence: qgking@nwpu.edu.cn

Abstract: Machine vision plays a great role in localizing strawberries in a complex orchard or greenhouse for picking robots. Due to the variety of each strawberry (shape, size, and color) and occlusions of strawberries by leaves and stems, precisely locating each strawberry brings a great challenge to the vision system of picking robots. Several methods have been developed for localizing strawberries, based on the well-known Mask R-CNN network, which, however, are not efficient running on the picking robots. In this paper, we propose a simple and highly efficient framework for strawberry instance segmentation running on low-power devices for picking robots, termed StrawSeg. Instead of using the common paradigm of “detection-then-segment”, we directly segment each strawberry in a single-shot manner without relying on object detection. In our model, we design a novel feature aggregation network to merge features with different scales, which employs a pixel shuffle operation to increase the resolution and reduce the channels of features. Experiments on the open-source dataset StrawDI_Db1 demonstrate that our model can achieve a good trade-off between accuracy and inference speed on a low-power device.

Keywords: computer vision; image segmentation; fruit localization; lightweight network; mobile robots; vision system



Citation: Cao, L.; Chen, Y.; Jin, Q. Lightweight Strawberry Instance Segmentation on Low-Power Devices for Picking Robots. *Electronics* **2023**, *12*, 3145. <https://doi.org/10.3390/electronics12143145>

Academic Editor: Krzysztof Okarma

Received: 3 July 2023
Revised: 14 July 2023
Accepted: 17 July 2023
Published: 20 July 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Due to socioeconomic changes of the current society, fewer people are willing to be engaged in agricultural production [1]. In order to solve the challenge of labor shortage in the agricultural industry, various robots have been developed for agricultural activities, e.g., sowing of seeds, irrigating, spraying pesticides, weeding, and harvesting [2]. Among these activities, harvesting is the most time-consuming and labor-intensive task [2,3]. Consequently, a few commercial harvesting robots have been used to pick fruits and vegetables in orchards or greenhouses, e.g., apples [4], strawberries [5], grapes [6], tomatoes [7], and sweet peppers [8]. Strawberries, one of the profitable fruits, are widely cultivated in the world [3]. Picking a strawberry requires skilled operations, since it is easily bruised. Training a new picker to have the same skill as an experienced one needs at least one year [1]. This motivates us to develop an autonomous strawberry-picking robot to reduce the demand for human labor and improve picking efficiency.

A few companies have developed strawberry-picking robots, e.g., Berry 5 designed by Harvest CROO, SW 6010 from Agrobot, Dogtooth from Cambridge, and Rubion made by Octinion [1]. Berry 5 and SW 6010 are designed for picking strawberries cultivated in large-scale and open-field orchards, which are equipped with large machines with high costs. Dogtooth and Rubion are small robots designed for picking strawberries in greenhouses. Designing a picking robot refers to many technologies, e.g., machine vision, mechanical design, kinematics, path planning, control, and navigation. Among these, machine vision plays a great role in localizing strawberries in a complex orchard or greenhouse environment, in which each strawberry needs to be precisely located [1,2]. Due

to the variety in shape and scale of strawberries and occlusions of strawberries by leaves and stems, precisely locating each strawberry brings a great challenge to the vision system of picking robots.

Compared with the bounding box provided by object detection, the segmentation mask by the instance segmentation technology can provide better localization accuracy, which avoids the impact of a complex background in the bounding box. Recently, a few methods and datasets for strawberry instance segmentation have been proposed [5,9,10]. Borrero et al. [9] released a large-scale and high-resolution dataset of strawberry images, along with the corresponding manually labeled instance segmentation mask images. In addition, they proposed a strawberry instance segmentation network based on the framework of Mask R-CNN [11], which, however, required a large processing power for vision systems of picking robots. Therefore, they designed a new network based on U-Net [12] to segment each strawberry in an image with better accuracy and faster inference speed [10]. However, these methods are still too heavy to run on the vision system of picking robots, which usually are equipped with energy-constrained supplies and low-compute devices. This motivates us to develop a novel lightweight network for strawberry instance segmentation with low latency running on low-power devices of picking robots.

In this paper, we present a simple and highly efficient framework for strawberry instance segmentation running on low-power devices for picking robots, termed StrawSeg. Instead of using the common paradigm of “detection-then-segment”, we directly segment each strawberry in a single-shot manner without relying on object detection. Our network consists of three parts: backbone, neck, and head. Given an image containing strawberries, MobileNetV2 [13] is adopted as the backbone to extract multiscale and multilevel features from the input image, and the multiscale features are aggregated by the neck module. We design a novel feature aggregation network termed FAN to merge these features with different scales. Instead of implementing by interpolation or deconvolution layer, we employ a pixel shuffle operation to increase the resolution and reduce the channels of features, which can avoid the use of convolutional layers to reduce channels. The head module directly predicts a fixed-size set of segmentation masks wherein each mask indicates a target strawberry or background. During training, the predicted masks are matched to the ground truth by using the bipartite matching strategy. At the inference, we compute an average pixel value along the spatial dimension for each mask to be its classification score, and some low-confidence predictions can be dropped. Experiments on the open-source dataset StrawDI_Db1 [9] demonstrate that our model can achieve a good trade-off between accuracy and inference speed on the low-power device.

Our contributions are summarized as follows:

- (1) We present a lightweight yet effective framework for strawberry instance segmentation running on low-power devices for picking robots, which can directly segment each strawberry without relying on object detection.
- (2) We design a novel feature aggregation network to aggregate features with different scales extracted from different levels of the backbone network, which can increase the resolution and reduce the channels of features.
- (3) Experimental results demonstrate that our model achieves a good trade-off between accuracy and inference speed running on the low-power device.

2. Related Work

2.1. Instance Segmentation

Instance segmentation aims to produce a pixel-wise segmentation mask for the object of interest in an image. It has been significantly improved with the advancement of CNNs and Transformers. The conventional methods for instance segmentation follow the “detection-then-segment” paradigm, which first generates bounding boxes by detectors and predicts masks by ROIAlign [11] or dynamic convolutions [14]. Mask R-CNN [11], YOLACT [15], and MEInst [16] are the representative methods. Instead of relying on the object detectors, SOLO [17,18] directly segmented objects according to the object’s location

and size. PolarMask [19] employed polar coordinates to represent mask contours. Instead of directly predicting masks, a few methods try to predict mask embeddings. SOLQ [20] encoded the spatial binary mask into embeddings, and the network is trained to predict the embedding for the mask. ISTR [21] predicted low-dimensional mask embeddings. Cheng et al. [22] proposed a sparse set of instance activation maps as an object representation to highlight informative regions for each object, which achieves a good trade-off between accuracy and inference speed.

2.2. Fruit Localization

Recently, a few methods have been developed to improve the performance of machine vision for fruit or vegetable localization. Yu et al. [5] proposed a method for strawberry detection and segmentation based on Mask R-CNN. Jia et al. [4] designed a model for the recognition and segmentation of overlapped apples based on Mask R-CNN. Santos et al. [6] used Mask R-CNN [11] and YOLO [23–27] to segment and detect wine grapes, respectively. Instead of using the heavy Mask R-CNN, Borrero et al. [10] designed a new network based on U-Net to segment each strawberry in an image with better accuracy and faster inference speed. Ning et al. [8] proposed to combine the convolutional block attention module with YOLOv4 to recognize and localize sweet peppers. Liu et al. [28] proposed a detection and segmentation method for obscured green fruit based on a FCOS [29] object detection model. Zeng et al. [7] proposed a lightweight network based on YOLOv5 to achieve real-time localization and ripeness detection of tomatoes. Liu et al. [30] proposed a method for localizing pineapples based on binocular stereo vision and an improved YOLOv3 model. Kang et al. [31] introduced a LiDAR-camera fusion-based instance segmentation method for the localization of apples.

2.3. Lightweight Detection and Segmentation

Real-time object detection or instance segmentation is necessary for a model running on edge devices. Recently, real-time detection and segmentation methods are still being developed. YOLO series [23–27] have been continuously advanced for faster and stronger object detection based on efficient architectures and bag-of-freebies. CSL-YOLO [32] proposed a cross-stage lightweight module to generate redundant features from cheap operations, and the module was combined with YOLO. Cui et al. [33] proposed a lightweight pinecone detection algorithm based on the improved YOLOv4-Tiny network, wherein ShuffleNet [34] was used as a backbone to extract features. Gui et al. [35] proposed a lightweight tea bud detection model based on the YOLOv5 network, wherein the Ghost_conv [36] module was applied to reduce the computational complexity and model size. Li et al. [37] designed a fast and lightweight detection algorithm based on YOLOv5 for passion fruit pest detection, wherein the attention module was added to improve accuracy.

3. Methods

Our model, StrawSeg, aims to directly segment instance-level strawberries without relying on object detection. To this end, we first design a lightweight network to extract features from the input image, and predict all target masks at once. The model is trained end to end with a set loss function, which performs bipartite matching between the predicted masks and ground truth. Finally, a simple inference process is described to acquire final segmentation masks for strawberries. A flowchart of our method is shown in Figure 1.

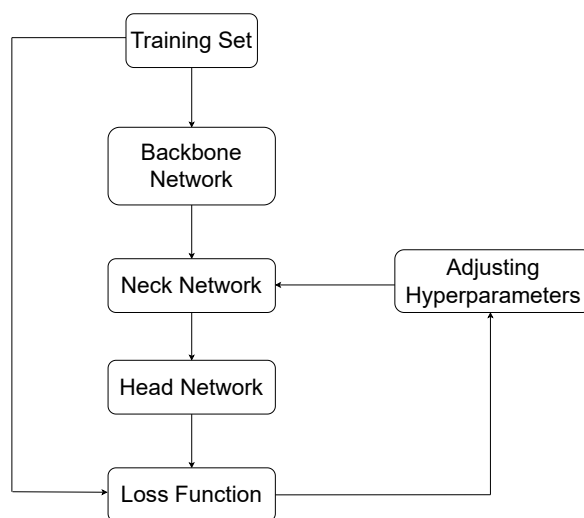


Figure 1. The flowchart of our method.

3.1. StrawSeg Architecture

The overall framework of StrawSeg is shown in Figure 2. This simple network consists of three parts: backbone, neck, and head modules. The backbone module extracts multilevel and multiscale features from a given image, and the neck module aggregates features from the backbone. Finally, the head module directly predicts a set of segmentation masks.

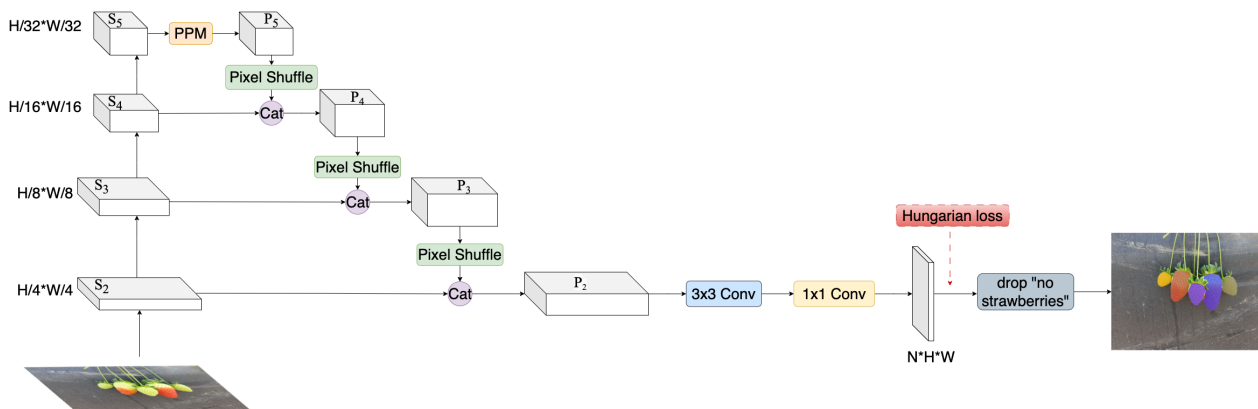


Figure 2. Overall framework of StrawSeg. In the figure, *Cat* represents the concatenate operation, and *PPM* represents the pyramid pooling module [38].

3.1.1. Backbone

To reduce the latency of our network running on low-power devices, we use MobileNetV2 [13] as the backbone network to extract multilevel and multiscale features from the input image. Given an image $I \in \mathbb{R}^{3 \times H \times W}$, the backbone extracts the multiscale image features from the shallow to deep layers of the backbone network, i.e., $\{S_2, S_3, S_4, S_5\}$, where S_i has a resolution of $\frac{H}{2^i} \times \frac{W}{2^i}$, $i = 2, 3, 4, 5$.

3.1.2. Neck

To enhance the feature representations, the neck module is employed to aggregate the multiscale and multilevel features. To further reduce computational complexity and model parameters, we design a novel feature aggregation network (FAN) to aggregate features extracted from the backbone. To enlarge the receptive field of the network, we first apply a pyramid pooling module (PPM) [38] on the feature map S_5 to acquire a feature map P_5 with global prior representations. For further details on PPM, we refer the reader to [38].

We then upscale P_5 to the same resolution as the feature map S_4 . Instead of implementing by interpolation or deconvolution layer [39], we employ a pixel shuffle operation [40] to increase the resolution and reduce the channels of P_5 . Pixel shuffle is an operation used in super-resolution models to implement efficient subpixel convolutions with a stride of r . Specifically, it rearranges elements in a tensor of shape $(*, C \times r^2, H, W)$ to a tensor of shape $(*, C, H \times r, W \times r)$. Suppose P_5 has C_5 channels; thus, the pixel shuffle rearranges the feature map P_5 of shape $\frac{H}{32} \times \frac{W}{32} \times C_5$ to a higher-resolution feature map of shape $\frac{H}{16} \times \frac{W}{16} \times \frac{C_5}{4}$. This higher-resolution feature map is concatenated with the feature map S_4 (of shape $\frac{H}{16} \times \frac{W}{16} \times C_4$) along the channel dimension to form a mixed feature map P_4 . Similarly, the feature map P_4 is upsampled by the pixel shuffle and concatenated with S_3 (of shape $\frac{H}{8} \times \frac{W}{8} \times C_3$) to acquire a feature map P_3 . Additionally, P_3 is also upsampled by the pixel shuffle and concatenated with S_2 (of shape $\frac{H}{4} \times \frac{W}{4} \times C_2$) to acquire a feature map P_2 of shape $\frac{H}{4} \times \frac{W}{4} \times (\frac{C_5}{64} + \frac{C_4}{16} + \frac{C_3}{4} + C_2)$. Let us take MobileNetV2_0.5 as a backbone network, and $C_5 = 160, C_4 = 48, C_3 = 16, C_2 = 16$; thus, the feature map P_2 has a channel number of 26. Finally, P_2 is attached by a 3×3 convolution layer to generate a merged feature map, which aggregates the multilevel and multiscale feature maps. The input channel and output channel numbers are the same with P_2 .

3.1.3. Head

The segmentation head directly predicts N masks by a single 1×1 convolution layer on the fused feature map from the neck module, which are rescaled to the original resolution of an input image through interpolation: $y = \{m_i | m_i \in [0, 1]^{H \times W}\}_{i=1}^N$, where N is set to be significantly larger than the typical number of strawberries in an image.

3.2. Label Assignment and Training Loss

To train our model, a label assignment strategy is needed. The ground truth binary masks of strawberries in an image are denoted as $y^{gt} = \{m_i^{gt} | m_i^{gt} \in [0, 1]^{H \times W}\}_{i=1}^{N^{gt}}$, where N^{gt} is the number of strawberries in the image. Since N is different from N^{gt} and $N \geq N^{gt}$, we pad the set of ground truth labels with all-zero masks to allow one-to-one matching. A bipartite matching-based assignment is employed between the predicted masks and ground truth labels, which searches for a permutation of N elements $\sigma \in \{1, 2, \dots, N\}$ with the lowest cost [41,42]:

$$\hat{\sigma} = \arg \min_{\sigma} \sum_i^N \mathcal{L}_{\text{match}}(y_i^{gt}, y_{\sigma(i)}), \quad (1)$$

where $\mathcal{L}_{\text{match}}(y_i^{gt}, y_{\sigma(i)})$ is a pairwise matching cost between ground truth y_i^{gt} and a prediction with index $\sigma(i)$, which is defined as

$$\mathcal{L}_{\text{match}} = \lambda_{\text{dice}} \left(1 - \mathcal{L}_{\text{dice}}(m_i^{gt}, m_{\sigma(i)})\right) + \lambda_{\text{focal}} \mathcal{L}_{\text{focal}}(m_i^{gt}, m_{\sigma(i)}), \quad (2)$$

where λ_{dice} and λ_{focal} are hyperparameters, and $\mathcal{L}_{\text{dice}}$ and $\mathcal{L}_{\text{focal}}$ denote dice loss and focal loss, respectively. This optimal assignment is computed with the Hungarian algorithm [41].

Given the optimal assignment $\hat{\sigma}$, we define N^{gt} matched predicted masks and $N - N^{gt}$ nonmatched predictions as positive pairs and negative pairs, respectively. The matched predictions tend to predict the ground truth masks, and the nonmatched predictions aim to output all-zeros. To this end, we use the Hungarian loss to optimize our network, which is defined as

$$\mathcal{L}_{\text{Hung}} = \sum_{i=1}^{N^{gt}} \left[\lambda_{\text{dice}} \left(1 - \mathcal{L}_{\text{dice}}(m_i^{gt}, m_{\hat{\sigma}(i)})\right) + \lambda_{\text{focal}} \mathcal{L}_{\text{focal}}(m_i^{gt}, m_{\hat{\sigma}(i)}) \right], \quad (3)$$

where λ_{dice} and λ_{focal} are hyperparameters, and denote dice loss and focal loss, respectively. For our experiments, we set $\lambda_{\text{dice}} = 1, \lambda_{\text{focal}} = 20$.

3.3. Inference

The segmentation head of our network directly outputs N masks $\{m_i\}^N$, and we can compute an average pixel value along the spatial dimension for each mask to be its classification score of a strawberry. Thus, some low-confidence predictions can be dropped. Finally, we obtain the final binary masks by thresholding (we set it as 0.5). Specifically, to achieve better accuracy, we remove some binary masks that have a few parts occluded by other masks through nonmaximum suppression (NMS) [43].

4. Experiments

4.1. Dataset and Metrics

4.1.1. Dataset

Our experiments are conducted on the StrawDI_Db1 dataset [9] containing 3100 images taken in strawberry plantations in the province of Huelva (Spain) at different times during a full picking campaign. The images were taken with a smartphone and rescaled to 1008×756 pixels in PNG format. The dataset is divided into 2800 images for training, 100 images for validation, and 200 images for testing. Each image contains a few strawberries with the number ranging from 1 to 21. Straw DI_Db1 is the only open-source dataset for strawberry instance segmentation, in which variety in shape and scale of strawberries exists, as well as occlusions of strawberries by leaves and stems.

4.1.2. Metrics

We evaluate models on accuracy and inference speed on devices. Following the commonly used metric in the MS-COCO [44] competition of instance segmentation, the average precision (AP) metric is used to evaluate the accuracy of predicted masks. Specifically, the mean average precision (mAP) is computed using 10 IoU thresholds from 0.5 to 0.95. In addition, we also report the mean average precision for small (mAP_S), medium (mAP_M), and large sizes (mAP_L) of strawberries as the same criteria as COCO. The value of AP for IoU = 0.50 (AP₅₀) and 0.75 (AP₇₅) is also reported. For measuring the inference speed, we report the frames per second (FPS) of the network on a single NVIDIA RTX 3090 GPU and an edge device, NVIDIA Jetson Nano 2G(Made by NVIDIA Corporate, Santa Clara, CA, USA). TensorRT or FP16 is not used for acceleration.

4.1.3. Implementation Details

We implement our model in PyTorch and train over one NVIDIA RTX 3090 GPU with 32 images per minibatch and 200 epochs. We adopt an AdamW optimizer with an initial learning rate of 5×10^{-3} with a weight decay of 0.0005. The backbone is initialized with the ImageNet-pretrained weights, and other layers are randomly initialized. The standard random scale jittering between 0.8 and 1.5, random horizontal flipping, random rotating between -30° and 30° , random cropping, and random color jittering are used as data augmentation. We use a crop size of 640×640 as input for training. We adopt $N = 21$ for each image. We report the performance of the original scale inference without horizontal flip or multiple scales.

4.2. Comparison with State-of-the-Art Methods

Table 1 compares our model, StrawSeg, with some state-of-the-art methods with respect to accuracy and inference speed. We set a baseline model wherein FPN [39] is adopted to replace our designed FAN and the other modules are the same. SparseInst [22] achieves good accuracy and fast inference speed on the MS-COCO dataset for real-time instance segmentation, which can be applied for strawberry instance segmentation. We use MobileNetV2 [13] with different ratios as backbones to achieve the trade-off between accuracy and inference speed. All models are only trained on the StrawDI_Db1 dataset and evaluated on the testing set. The results show that our model is superior to the baseline and SparseInst with better accuracy and faster inference speed under the same backbone. Specifically, our model with MobileNetV2_0.25 has achieved 72.9% mAP, which improves

the baseline by 4.7% mAP and 15 FPS on RTX 3090 and 4 FPS on Jetson Nano. This verifies that our proposed FAN can bring improvement on accuracy and reduction on inference time compared with the commonly used FPN. It is worth noting that our model with MobileNetV2_0.25 achieves 20 FPS on the edge device NVIDIA Jetson Nano 2G, which has the right balance of low power and affordability for a picking robot. The inference speed of our model can be accelerated if using TensorRT or running on more powerful devices (e.g., Jetson TX2, Jetson Xavier NX, Jetson Orin NX). Using a heavier backbone does not bring large improvement on accuracy yet reduces the speed.

Table 1. Performance comparison of our model with state-of-the-art methods on the Straw DI_Db1 testing set. Numbers in bold indicate the best performance.

Backbone	Method	<i>mAP</i>	<i>AP</i> ₅₀	<i>AP</i> ₇₅	<i>mAP</i> _S	<i>mAP</i> _M	<i>mAP</i> _L	Params	FPS (3090)	FPS (Jetson)
MobileNetV2_0.25	Baseline	68.2	82.4	74.0	26.8	68.1	94.0	0.18 M	140	16
	SparseInst	65.0	80.2	69.0	24.6	66.4	87.2	0.20 M	121	13
	Ours	72.9	86.4	78.5	29.1	74.8	94.4	0.15 M	155	20
MobileNetV2_0.5	Baseline	76.2	86.7	79.3	30.1	80.6	96.0	0.65 M	119	14
	SparseInst	77.9	89.9	83.0	33.5	81.2	97.3	0.75 M	97	12
	Ours	79.7	90.6	84.3	41.0	82.7	96.4	0.53 M	139	19
MobileNetV2_1.0	Baseline	68.2	80.3	71.2	28.9	71.0	88.7	2.50 M	114	12
	SparseInst	79.6	89.7	83.8	38.3	83.4	96.0	2.86 M	89	10
	Ours	80.0	89.8	83.8	40.9	83.3	97.1	1.97 M	131	17

There are only several published methods based on Mask R-CNN that have been applied to strawberry instance segmentation on the Straw DI_Db1 dataset. Table 2 compares our model, StrawSeg, with a few existing methods that have been evaluated on the Straw DI_Db1 testing set. The results show that our model surpasses the existing methods with great superiority.

Table 2. Performance comparison of our model with a few existing models that have been evaluated on the Straw DI_Db1 testing set. Numbers in bold indicate the best performance.

Methods	<i>mAP</i>	<i>AP</i> ₅₀	<i>AP</i> ₇₅	<i>mAP</i> _S	<i>mAP</i> _M	<i>mAP</i> _L
Yu et al. [5]	45.4	76.6	47.1	07.4	50.0	78.3
Perez-Borrero et al. [9]	43.8	74.2	45.1	07.5	51.8	75.9
Perez-Borrero et al. [10]	52.6	69.4	57.8	17.0	65.3	53.3
Ours	80.0	89.8	83.8	40.9	83.3	97.1

Figure 3 shows visualization comparisons of different methods on some images from the Straw DI_Db1 testing set, wherein we denote the inaccurate predictions by the red arrows. For the first column, the baseline model mistakenly predicts the leaf as the strawberry, which occurs at SparseInst. For the second column, the baseline model and SparseInst miss two and one strawberries, respectively, and SparseInst predicts an inaccurate mask. For the third column, the baseline model and SparseInst mistakenly predict the leaf as the strawberry, and SparseInst misses one strawberry in the corner of the image. The visualization results demonstrate the superiority of our model.

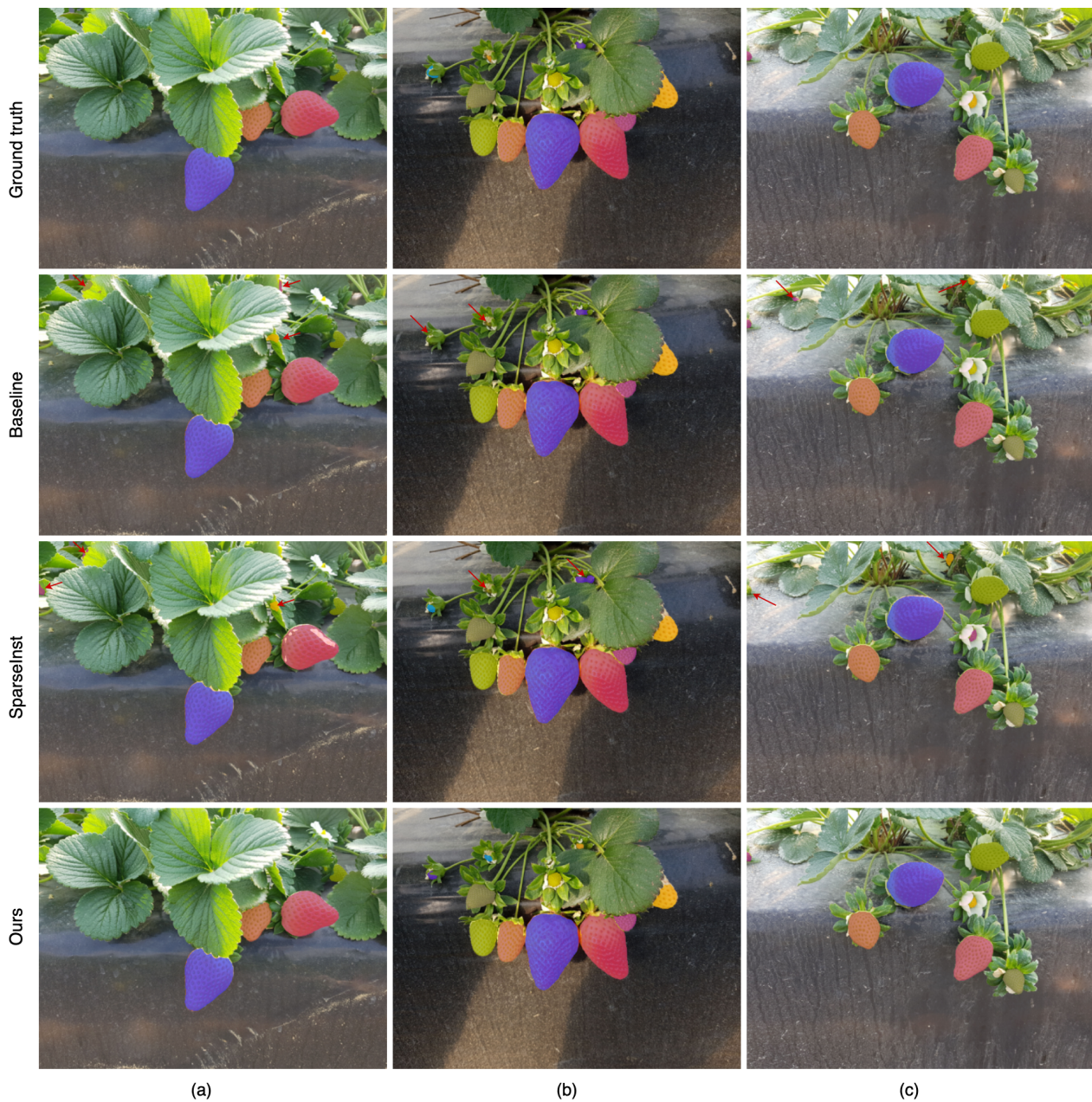


Figure 3. Visualization comparison of different methods on some images (a–c) from the Straw DI_Db1 testing set. The inaccurate predictions are denoted by the red arrows.

4.3. Ablation Studies

We investigate the effectiveness of our designs through a few ablation studies, including the neck module, the feature aggregation network, the scale of an input image, the number of convolution layers in the neck module, the number of predicting masks by the head network, the hyperparameters of loss function, and the usage of NMS in the inference stage. Without losing generality, we use MobileNetV2_0.5 as the backbone and evaluate on the testing set.

4.3.1. Structure of the Neck Module

The neck module consists of two parts: PPM and FAN. To further analyze the importance of each component in the neck, PPM and FAN are progressively added into the neck module to verify their effectiveness. We first set a baseline wherein S_5 of the backbone is directly appended to the head. Table 3 summarizes the results of the investigation on each

component. It shows that PPM and FAN improve the baseline by 6.2% and 21.5% mAP, respectively. The combination of PPM and FAN can achieve 79.7% mAP. It is worth noting that the ASPP [45] module is commonly adopted to enlarge and acquire different scales of receptive fields for semantic information. The result shows that adding ASPP even drops 0.2% mAP compared with adding PPM and FAN, which only improves 1% mAP_M and 0.6% mAP_L for medium and large sizes, respectively.

Table 3. Ablation study on the structure of the neck module. Numbers in bold indicate the best performance.

Module	mAP	AP_{50}	AP_{75}	mAP_S	mAP_M	mAP_L	FPS (3090)
Backbone only	52.5	76.5	55.2	7.3	50.8	82.0	194
+PPM	58.7	80.1	62.4	10.5	58.3	89.1	145
+FAN	74.0	86.8	78.1	37.4	75.8	93.0	152
+PPM+FAN	79.7	90.6	84.3	41.0	82.7	96.4	139
+PPM+FAN+ASPP	79.5	89.8	83.4	37.4	83.7	97.0	108

4.3.2. Stage of Output Feature Maps

In FAN, features with different scales are aggregated progressively, and the feature map P_2 is appended to the head module to predict masks. We investigate the accuracy and inference speed when using features from different levels, as shown in Table 4. If the head module directly appends to P_5 , which means that the scale of the output feature map is only $\frac{H}{32} \times \frac{W}{32}$ and the predicted masks are rescaled to the original resolution of the input image through interpolation, then the model can only achieve 58.7% mAP. Adopting P_4 can improve the model by 13.4% mAP yet reduce the speed by 4 FPS. P_3 does not further improve the model compared with P_4 . The feature map P_2 achieves a good trade-off between accuracy and speed, which improves to 79.7% mAP and with 139 FPS.

Table 4. Ablation study on the output feature maps. Numbers in bold indicate the best performance.

Stage	mAP	AP_{50}	AP_{75}	mAP_S	mAP_M	mAP_L	FPS (3090)
P_5	58.7	80.1	62.4	10.5	58.3	89.1	145
P_4	72.3	86.9	76.5	27.4	75.2	94.2	141
P_3	72.2	85.4	75.9	28.6	75.0	93.2	140
P_2	79.7	90.6	84.3	41.0	82.7	96.4	139

4.3.3. Scale of Input Image

The default input image size is set to 640×640 ; we further analyze the influence of an input image size. Table 5 summarizes the results of models trained with different input image sizes. It shows that increasing the input image size does not bring an improvement of accuracy yet reduces the speed. Decreasing the input image size also reduces the accuracy. The results verify that an input image size of 640×640 is appropriate.

Table 5. Ablation study on the scale of an input image. Numbers in bold indicate the best performance.

Scale	mAP	AP_{50}	AP_{75}	mAP_S	mAP_M	mAP_L	FPS (3090)
704	70.1	83.4	74.9	31.5	71.0	92.3	119
640	79.7	90.6	84.3	41.0	82.7	96.4	139
512	74.9	87.1	80.0	32.9	78.2	93.6	141

4.3.4. Number of Convolution Layers in the Neck

In the neck module, we use a single 3×3 convolution layer to generate a merged feature map from P_2 ; now we investigate the influence of the number of convolution layers. Table 6 summarizes the results of models with different numbers of convolution layers in

the neck module. It shows that removing or increasing the number of convolution layers will reduce the accuracy. A single 3×3 convolution layer has achieved a good trade-off between accuracy and inference speed.

Table 6. Ablation study on the number of convolutional layers in the neck module. Numbers in bold indicate the best performance.

Number of Conv	<i>mAP</i>	<i>AP</i> ₅₀	<i>AP</i> ₇₅	<i>mAP</i> _S	<i>mAP</i> _M	<i>mAP</i> _L	FPS (3090)
w/o	71.4	85.9	75.4	33.4	71.4	92.0	144
1	79.7	90.6	84.3	41.0	82.7	96.4	139
2	75.4	84.7	78.8	37.4	77.9	93.9	134

4.3.5. Number of Predicting Masks

In the above experiments, we set the number of predicting masks by the head network as 21, which is the maximum number of strawberries in the Straw DI_Db1 dataset. Thus, the model would lose some targets if the testing image contains more than 21 strawberries. Could we set this number to be larger? We then set $N = 30$ to investigate how this number affects the performance of StrawSeg. Table 7 shows that increasing the number of predicting masks greatly reduces the performance of StrawSeg. According to our statistics on Straw DI_Db1, the average number of strawberries in an image is only 5.8. Thus, predicting too many masks causes excessive negative samples when training, which makes the model hard to optimize the parameters.

Table 7. Ablation study on the number of predicting masks by the network. Numbers in bold indicate the best performance.

Number of Masks	<i>mAP</i>	<i>AP</i> ₅₀	<i>AP</i> ₇₅	<i>mAP</i> _S	<i>mAP</i> _M	<i>mAP</i> _L
30	68.4	79.9	73.3	33.9	76.8	83.5
21(Ours)	79.7	90.6	84.3	41.0	82.7	96.4

4.3.6. Hyperparameters of Loss Functions

In our experiments, we choose $\lambda_{dice} = 1$, $\lambda_{focal} = 20$ in the loss function by evaluating on the validation set. Table 8 shows the performance variation of StrawSeg on the testing set when varying the hyperparameters in the loss function. It is obvious that increasing λ_{focal} or λ_{dice} can improve the performance of StrawSeg, and a larger λ_{focal} brings better performance. However, $\lambda_{focal} = 30$ achieves a lower result, which illustrates that setting $\lambda_{dice} = 1$, $\lambda_{focal} = 20$ is appropriate for training StrawSeg on this dataset.

Table 8. Ablation study on varying hyperparameters in the loss function. Numbers in bold indicate the best performance.

λ_{dice}	λ_{focal}	<i>mAP</i>	<i>AP</i> ₅₀	<i>AP</i> ₇₅	<i>mAP</i> _S	<i>mAP</i> _M	<i>mAP</i> _L
1	1	67.6	81.0	70.7	26.5	69.3	89.8
10	1	72.5	85.1	76.5	31.4	75.7	91.1
1	10	75.1	87.4	79.6	36.8	76.1	95.8
1	20	79.7	90.6	84.3	41.0	82.7	96.4
1	30	77.8	89.6	82.9	36.2	81.2	95.5

4.3.7. Usage of NMS

During the inference stage, we use the NMS process to remove a few binary masks that have a few parts occluded by other masks. We explore the effectiveness of NMS. Table 9 shows that dropping the NMS process at the inference stage only reduces the accuracy by 3.4% *mAP*. This demonstrates that NMS is not necessary for our model, yet an effective trick for improving accuracy.

Table 9. Ablation study on the usage of NMS at the inference stage. Numbers in bold indicate the best performance.

Postprocessing	<i>mAP</i>	<i>AP</i> ₅₀	<i>AP</i> ₇₅	<i>mAP</i> _S	<i>mAP</i> _M	<i>mAP</i> _L
w/o NMS	76.3	86.4	80.5	37.7	79.5	94.5
Ours	79.7	90.6	84.3	41.0	82.7	96.4

4.4. Discussion

We develop StrawSeg to segment each strawberry in an image, and this model performs well on the Straw DI_Db1 dataset compared with some state-of-the-art methods. Theoretically, our model is available for any one-class instance segmentation task. To investigate how well our StrawSeg generalizes to other more larger-scale datasets, we train and evaluate models on a person instance segmentation dataset, CIHP [46], which is an instance-level human-parsing dataset. This dataset includes 28,280 images for training, 5000 for validation, and 5000 for testing. The average and maximum number of persons in an image are 3.4 and 12, respectively. Thus, we set $N = 12$ for StrawSeg when training on CIHP, and MobileNetV2_0.5 is utilized as the backbone. We train models with 50 epochs, and the other settings are the same with training on Straw DI_Db1. Table 10 shows a performance comparison of our model with the baseline and SparseInst. The results verify that our model, StrawSeg, still has superiority over the baseline and SparseInst.

Table 10. Performance comparison of our model with state-of-the-art methods on the CIHP testing set. Numbers in bold indicate the best performance.

Methods	<i>mAP</i>	<i>AP</i> ₅₀	<i>AP</i> ₇₅	<i>mAP</i> _S	<i>mAP</i> _M	<i>mAP</i> _L
Baseline	44.6	74.7	46.5	2.7	26.5	54.9
SparseInst	44.1	72.9	46.5	2.9	27.2	56.2
Ours	47.7	76.4	50.9	4.4	28.8	58.3

It is worth noting that the head network of StrawSeg only directly predicts masks without a classification output; thus, StrawSeg can only adapt to the one-class instance segmentation task. It may be applied to multiple-class instance segmentation by adding a classification head to represent the probability of belonging to the target class. This can be our future work to investigate the performance of StrawSeg on the multiple-class instance segmentation task.

5. Conclusions

In this paper, we present a novel and highly efficient method for strawberry instance segmentation on low-power devices for picking robots. Our network uses MobileNetV2 as the backbone to extract multiscale and multilevel features from the input image, and the multiscale features are aggregated by the neck module. We design a novel feature aggregation network termed FAN to merge these features with different scales. Instead of implementing by interpolation or deconvolution layer, we employ a pixel shuffle operation to increase the resolution and reduce the channels of features. The aggregated features directly output a fixed number of masks to represent strawberries of the input image. Experimental results demonstrate that our model can achieve a good trade-off between accuracy and inference speed on a low-power device (NVIDIA Jetson Nano 2G), in which our model with MobileNetV2_0.50 achieves 79.7% mAP and 19 FPS. In a future work, we will explore the application of this model to the other fruit or vegetable localization on different edge devices.

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

Funding: This research was funded by Fundamental Research Funds for the Central Universities.

Institutional Review Board Statement: This article does not contain any studies with human participants or animals performed by any of the authors.

Data Availability Statement: Data are available on this website: <https://strawdi.github.io/> (accessed on 10 January 2023).

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Preter, A.D.; Anthonis, J.; Baerdemaeker, J.D. Development of a Robot for Harvesting Strawberries. *IFAC-PapersOnLine* **2018**, *51*, 14–19. [CrossRef]
2. Charania, I.; Li, X. Smart farming: Agriculture’s shift from a labor intensive to technology native industry. *Internet Things* **2020**, *9*, 100142. [CrossRef]
3. Hernandez-Martinez, N.R.; Blanchard, C.; Wells, D.; Salazar-Gutierrez, M.R. Current state and future perspectives of commercial strawberry production: A review. *Sci. Hortic.* **2023**, *312*, 111893. [CrossRef]
4. Jia, W.; Tian, Y.; Luo, R.; Zhang, Z.; Lian, J.; Zheng, Y. Detection and segmentation of overlapped fruits based on optimized mask R-CNN application in apple harvesting robot. *Comput. Electron. Agric.* **2020**, *172*, 105380. [CrossRef]
5. 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]
6. Santos, T.T.; Souza, L.L.d.; Santos, A.A.d.; Avila, S. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Comput. Electron. Agric.* **2020**, *170*, 105247. [CrossRef]
7. 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]
8. Ning, Z.; Luo, L.; Ding, X.; Dong, Z.; Yang, B.; Cai, J.; Chen, W.; Lu, Q. Recognition of sweet peppers and planning the robotic picking sequence in high-density orchards. *Comput. Electron. Agric.* **2022**, *196*, 106878. [CrossRef]
9. Borrero, I.P.; Santos, D.M.; Arias, M.E.G.; Ancos, E.C. A fast and accurate deep learning method for strawberry instance segmentation. *Comput. Electron. Agric.* **2020**, *178*, 105736. [CrossRef]
10. Borrero, I.P.; Santos, D.M.; Vazquez, M.J.V.; Arias, M.E.G. A new deep-learning strawberry instance segmentation methodology based on a fully convolutional neural network. *Neural Comput. Appl.* **2021**, *33*, 15059–15071. [CrossRef]
11. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.
12. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the MICCAI, Munich, Germany, 5–9 October 2015; pp. 234–241.
13. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the CVPR, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
14. Tian, Z.; Shen, C.; Chen, H. Conditional convolutions for instance segmentation. In Proceedings of the ECCV, Glasgow, UK, 23–28 August 2020; pp. 282–298.
15. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT: Real-Time Instance Segmentation. In Proceedings of the ICCV, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9156–9165.
16. Zhang, R.; Tian, Z.; Shen, C.; You, M.; Yan, Y. Mask Encoding for Single Shot Instance Segmentation. In Proceedings of the CVPR, Seattle, WA, USA, 13–19 June 2020; pp. 10223–10232.
17. Wang, X.; Kong, T.; Shen, C.; Jiang, Y.; Li, L. SOLO: Segmenting Objects by Locations. In Proceedings of the ECCV, Glasgow, UK, 23–28 August 2020; pp. 649–665.
18. Wang, X.; Zhang, R.; Kong, T.; Li, L.; Shen, C. SOLOv2: Dynamic and Fast Instance Segmentation. In Proceedings of the NeurIPS, Virtual, 6–12 December 2020.
19. Xie, E.; Sun, P.; Song, X.; Wang, W.; Liu, X.; Liang, D.; Shen, C.; Luo, P. PolarMask: Single Shot Instance Segmentation With Polar Representation. In Proceedings of the CVPR, Seattle, WA, USA, 13–19 June 2020; pp. 12190–12199.
20. Dong, B.; Zeng, F.; Wang, T.; Zhang, X.; Wei, Y. SOLQ: Segmenting Objects by Learning Queries. In Proceedings of the NeurIPS, Virtual, 6–14 December 2021.
21. Hu, J.; Cao, L.; Lu, Y.; Zhang, S.; Wang, Y.; Li, K.; Huang, F.; Shao, L.; Ji, R. ISTR: End-to-End Instance Segmentation with Transformers. In Proceedings of the CVPR, Virtual, 19–25 June 2021; pp. 8737–8746.

22. Cheng, T.; Wang, X.; Chen, S.; Zhang, W.; Zhang, Q.; Huang, C.; Zhang, Z.; Liu, W. Sparse Instance Activation for Real-Time Instance Segmentation. In Proceedings of the CVPR, New Orleans, LA, USA, 18–24 June 2022; pp. 4423–4432.
23. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the CVPR, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
24. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the CVPR, Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
25. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
26. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
27. 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. *arXiv* **2022**, arXiv:2207.02696.
28. Liu, M.; Jia, W.; Wang, Z.; Niu, Y.; Yang, X.; Ruan, C. An accurate detection and segmentation model of obscured green fruits. *Comput. Electron. Agric.* **2022**, *197*, 106984. [[CrossRef](#)]
29. Tian, Z.; Shen, C.; Chen, H.; He, T. FCOS: Fully Convolutional One-Stage Object Detection. In Proceedings of the ICCV, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9626–9635.
30. Liu, T.H.; Nie, X.N.; Wu, J.M.; Zhang, D.; Liu, W.; Cheng, Y.F.; Zheng, Y.; Qiu, J.; Qi, L. Pineapple (*Ananas comosus*) fruit detection and localization in natural environment based on binocular stereo vision and improved YOLOv3 model. *Precis. Agric.* **2023**, *24*, 139–160. [[CrossRef](#)]
31. Kang, H.; Wang, X.; Chen, C. Accurate fruit localisation using high resolution LiDAR-camera fusion and instance segmentation. *Comput. Electron. Agric.* **2022**, *203*, 107450. [[CrossRef](#)]
32. Zhang, Y.M.; Lee, C.C.; Hsieh, J.W.; kuo Chin, F. CSL-YOLO: A new lightweight object detection system for edge computing. *arXiv* **2021**, arXiv:2107.04829.
33. Cui, M.; Lou, Y.; Ge, y.; Wang, K. LES-YOLO: A lightweight pinecone detection algorithm based on improved YOLOv4-Tiny network. *Comput. Electron. Agric.* **2023**, *205*, 107613. [[CrossRef](#)]
34. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In Proceedings of the ECCV, Munich, Germany, 8–14 September 2018; pp. 122–138.
35. Gui, Z.; Chen, J.; Li, Y.; Chen, Z.; Wu, C.; Dong, C. A lightweight tea bud detection model based on Yolov5. *Comput. Electron. Agric.* **2023**, *205*, 107636. [[CrossRef](#)]
36. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. GhostNet: More Features From Cheap Operations. In Proceedings of the CVPR, Seattle, WA, USA, 13–19 June 2020; pp. 1577–1586.
37. 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](#)]
38. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid Scene Parsing Network. In Proceedings of the CVPR, Honolulu, HI, USA, 21–26 July 2017; pp. 6230–6239.
39. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the CVPR, Honolulu, HI, USA, 21–26 July 2017; pp. 936–944.
40. Shi, W.; Caballero, J.; Huszár, F.; Totz, J.; Aitken, A.P.; Bishop, R.; Rueckert, D.; Wang, Z. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In Proceedings of the CVPR, Las Vegas, NV, USA, 27–30 June 2016; pp. 1874–1883.
41. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In Proceedings of the ECCV, Glasgow, UK, 23–28 August 2020; pp. 213–229.
42. Cheng, B.; Schwing, A.G.; Kirillov, A. Per-Pixel Classification is Not All You Need for Semantic Segmentation. In Proceedings of the NeurIPS, Virtual, 6–14 December 2021.
43. Neubeck, A.; Van Gool, L. Efficient Non-Maximum Suppression. In Proceedings of the ICPR, Hong Kong, China, 20–24 August 2006; Volume 3, pp. 850–855.
44. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the ECCV, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
45. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Proceedings of the ECCV, Munich, Germany, 8–14 September 2018; pp. 801–818.
46. Gong, K.; Liang, X.; Li, Y.; Chen, Y.; Yang, M.; Lin, L. Instance-Level Human Parsing via Part Grouping Network. In Proceedings of the ECCV, Munich, Germany, 8–14 September 2018; pp. 770–785.

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