agriculture

MDPI

Article

Instance Segmentation of Tea Garden Roads Based on an Improved YOLOv8n-seg Model

Weibin Wu [1], Zhaokai He [1], Junlin Li [1], Tianci Chen [1], Qing Luo [1], Yuanqiang Luo [1], Weihui Wu [2] and Zhenbang Zhang [1,2,*]

[1] College of Engineering, South China Agricultural University, Guangzhou 510642, China; wuweibin@scau.edu.cn (W.W.); 20213163021@stu.scau.edu.cn (Z.H.); ljldragonquest@stu.scau.edu.cn (J.L.); chentianci@stu.scau.edu.cn (T.C.); luoqing39@stu.scau.edu.cn (Q.L.); luoyq@scau.edu.cn (Y.L.)

[2] College of Intelligent Engineering, Shaoguan University, Shaoguan 512005, China; wuweihui@sgu.edu.cn

* Correspondence: 744387896@scau.edu.cn

Abstract: In order to improve the efficiency of fine segmentation and obstacle removal in the road of tea plantation in hilly areas, a lightweight and high-precision DR-YOLO instance segmentation algorithm is proposed to realize environment awareness. Firstly, the road data of tea gardens in hilly areas were collected under different road conditions and light conditions, and data sets were generated. YOLOv8n-seg, which has the highest operating efficiency, was selected as the basic model. The MSDA-CBAM and DR-Neck feature fusion network were added to the YOLOv8-seg model to improve the feature extraction capability of the network and the feature fusion capability and efficiency of the model. Experimental results show that, compared with the YOLOv8-seg model, the DR-YOLO model proposed in this study has 2.0% improvement in *AP*@0.5 and 1.1% improvement in Precision. In this study, the DR-YOLO model is pruned and quantitatively compressed, which greatly improves the model inference speed with little reduction in *AP*. After deploying on Jetson, compared with the YOLOv8n-seg model, the *Precision* of DR-YOLO is increased by 0.6%, the *AP*@0.5 is increased by 1.6%, and the inference time is reduced by 17.1%, which can effectively improve the level of agricultural intelligent automation and realize the efficient operation of the instance segmentation model at the edge.

Keywords: road segmentation; YOLOv8-seg model; model structure optimization; model compression; edge–side deployment

check for updates

Citation: Wu, W.; He, Z.; Li, J.; Chen, T.; Luo, Q.; Luo, Y.; Wu, W.; Zhang, Z. Instance Segmentation of Tea Garden Roads Based on an Improved YOLOv8n-seg Model. *Agriculture* **2024**, *14*, 1163. https://doi.org/10.3390/agriculture14071163

Academic Editor: Takashi Okayasu

Received: 9 June 2024
Revised: 3 July 2024
Accepted: 13 July 2024
Published: 16 July 2024

Copyright: © 2024 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

Tea, *Camellia sinensis*, is a healthy and popular food worldwide. According to the National Bureau of Statistics, the total area of tea gardens in China will reach 3.3 million hectares in 2022, an increase of 2.7%. With the continuous expansion of the tea consumption market, the demand for high-quality tea is growing. However, traditional manual picking is time consuming, laborious and costly, so the development of automated picking robots is becoming an inevitable trend in the field of agricultural harvesting [1,2]. Presently, the picking machine still has defects of insufficient intelligence and automation in the inspection process, so it is particularly important to improve the efficient obstacle avoidance and automatic navigation ability of agricultural machinery [3]. For example, Ye et al. [4] proposed a mobile robot visual environment perception technology based on deep learning for obstacle recognition in various environments, and its obstacle avoidance success rate was about 93%. In addition, Wei et al. [5] developed a binocular vision perception system for agricultural machinery based on YOLOv3′s deep convolutional neural network to identify obstacles in the working environment. The average accuracy and recall rate of the system were 89.54% and 90.18%, respectively.

In order to improve the obstacle avoidance ability of agricultural machinery, it is necessary to improve the ability of environmental perception. At present, environmental

perception technology has become a research hotspot in the field of agricultural autonomous navigation [6], which is usually realized based on Light Detection And Ranging (LiDAR), a satellite remote sensing system and machine vision [7]. An object detection and terrain classification method proposed by Kragh et al. [8] can classify each point in the 3D point cloud obtained by a single multi-beam LiDAR scan. This method can classify each 3D point on a farm into three categories: ground, vegetation or object. Shi et al. [9] developed an orchard environment sensing technology based on LiDAR technology. The Vector Field Histogram (VFH) and Dynamic Window Approach (DWA) were adopted to determine the feasible direction of the mower and ensure smooth avoidance of obstacles during driving. LiDAR is highly accurate, but often more expensive.

In addition, based on the Global Navigation Satellite System (GNSS) [10], Chen et al. developed a field road classification algorithm [11] to record the track of agricultural machinery and an image segmentation model [12] specially used for field-road classification, which effectively improved the feature extraction capability of image segmentation model. Although the GNSS can provide road information and the location of agricultural machinery all day long, extreme weather or shelter can still affect its positioning signal [13].

In recent years, with the development of deep learning, environment perception methods with strong robustness and generalization ability based on deep learning have been gradually applied to the field of agricultural road recognition [14,15]. Kim et al. [16] proposed a new method for orchard path detection based on convolutional neural network (CNN) partial area classification. That is, by using CNN to segment a perspective path area based on partial area (patch) classification, the path is generated from RGB image, so that the agricultural machine can drive autonomously between the trees. Based on UNet, Li et al. [17] proposed a Faster-UNet deep learning model for the identification and prediction of furrows in maize crops. Similarly, Zhang et al. [18] also used the improved Fast-UNet model to identify navigation paths between fruit trees such as peaches, oranges and kiwis. Yang et al. [19] proposed a visual navigation path extraction method based on neural network and pixel scanning, which effectively improved the precision of orchard visual navigation path extraction by training Segnet and UNet semantic segmentation networks in orchard traffic data sets. Sun et al. [20] proposed a deep learning network model for semi-structured agricultural roads. This segmentation network with object-contextual representations (OCR) in this model is conducive to improving the detection accuracy of semi-structured drivable areas in agricultural scenes. Yu et al. [21] studied five deep learning-based visual recognition semantic segmentation methods, namely UNet, Deeplabv3+, BiseNetv1, BiseNetv2 and ENet networks, and extracted navigation lines on site in different scenarios.

Although there have been a considerable number of studies on the application of deep learning-based RGB cameras in the field of agricultural road recognition, as far as we know, the visual recognition methods in this relevant research only focus on common semantic segmentation. Until now, there has been no relevant research on instance segmentation of agricultural road images based on the YOLOv8 model. Therefore, this study aims to develop an instance segmentation method of agricultural road image based on YOLOv8-seg for the first time to fill the gap in this field. The algorithm studied in this work can be applied to the visual system of orchard machinery to improve its obstacle avoidance ability. Secondly, this method is not consistent with the traditional target detection method of environmental perception. It divides the driving area of agricultural machinery by examples and removes obstacles to avoid obstacles in hilly mountain tea gardens.

The main contribution of this study is to improve the defects and deficiencies of YOLOv8-seg. The improvement strategy mainly includes the following: first, based on the Convolutional Block Attention Module (CBAM) and Multi-Scale Dilated Attention (MSDA), the improvement of the enhanced feature extraction network and the improvement of the neck network was based on the DR-Neck result in the DR-YOLO model; then, the DR-YOLO model was compressed through model pruning and model quantization to realize efficient deployment and inference at the edge.

The rest of this paper is organized as follows: Section 2 introduces the composition of the image data set and the instance segmentation algorithm proposed in this study; Section 3 analyzes the experimental results and compares them with commonly used methods; Section 4 presents a summary of the work and future directions.

2. Materials and Methods

2.1. Data Acquisition

The tea garden roads in hilly areas are mainly gravel roads, dirt roads and concrete roads. The data sets used in this study were collected in mid-June 2023 and late May 2023 on the roads of hilly mountain tea plantations in Yangshan New Nature Tea Garden, Qingyuan City, Guangdong Province (24°42′ N, 112°40′ E), Huashun Agroforestry Fruit Tea Garden, Meizhou City, Guangdong Province (23°38′ N, 115°46′ E) and Guangzhou National Agricultural Science and Technology Park, Guangdong Province (23°35′ N, 113°37′ E). Data collection sites are shown in Figure 1. In this study, an Inter Realsense-D435i depth camera was used as an image acquisition device with a resolution of 1920 × 1080. Firstly, in order to enhance the elasticity and generalization performance of the visual recognition module, we collected images from two common roads in tea gardens as shown in Figure 1a,b. Secondly, considering the actual working time of the automated robot, we collected images from three periods of time, namely morning, afternoon and evening. As shown in Figure 1c, there is plenty of light in the morning and afternoon. As shown in Figure 1d, light intensity is low in the evening.

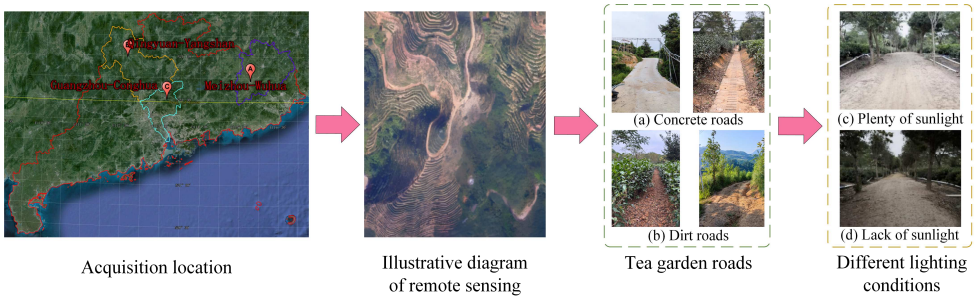


Figure 1. Image data collection locations. (**a**) Concrete roads; (**b**) dirt roads; (**c**) plenty of sunlight; (**d**) lack of sunlight.

The tea garden road image data set in this study consists of 3709 images, including paved (concrete) roads, unpaved (dirt) roads, plenty of sunlight and lack of sunlight. LabelMe5.2.1 software was used to label the data set and randomly divide it into a training set, validation set and test set according to the ratio of 8:1:1. The quantity classification of specific data samples is shown in Table 1, in which there are 2968 training sets, 370 validation sets and 371 test sets.

Table 1. Data samples from different roads and different sunlight conditions.

Category	Training Set	Validation Set	Test Set
Concrete road (plenty of sunlight)	740	90	91
Concrete road (lack of sunlight)	734	87	89
Dirt road (plenty of sunlight)	748	98	97
Dirt road (lack of sunlight)	746	95	94

2.2. Image Preprocessing

The image acquisition process is likely to blur or create noise in the image data set due to camera blur, sensor noise, image compression, etc. However, blurring and noise in image data sets may interfere with the deep learning model's learning of the real features of the data, resulting in a negative impact on the training and performance of the model. Therefore, the pre-processing of the image data set can improve image quality, enhance

image details and edges, make the model better capture the original features of the data, and improve the training efficiency and final performance of the model. At present, the commonly used image preprocessing methods include image sharpening, image noise reduction and image supergrading [22].

Furthermore, with the popularization and application of deep learning technology, deep learning technology has incomparable advantages in the aspects of image sharpening, image noise reduction and image supergrading. Taking image noise reduction as an example, the traditional smoothing filter can only deal with simple image noise, such as Gaussian noise and uniform noise. The noise reduction method based on deep learning can not only deal with complex noise, but also retain image details better.

In this study, the open source model Real-CUGAN was used to preprocess the images of a tea garden road, such as noise reduction, sharpening and supergrading. With a wide range of super-magnification and noise reduction intensities, Real-CUGAN can preserve more texture detail while sharpening the image. As can be clearly seen from the comparison between Figure 2c,d, after image preprocessing with Real-CUGAN, image resolution remains unchanged, image details are not only preserved, but also image detail edge contour becomes clearer and noise is removed.

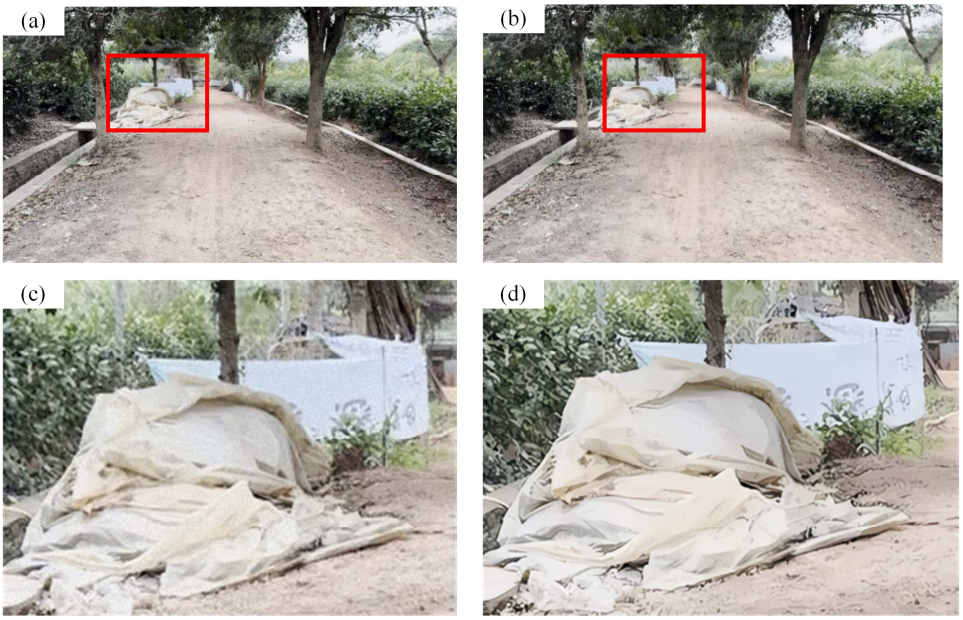


Figure 2. Image preprocessing. (**a**) Original image; (**b**) preprocessed image; (**c**) enlarged details of original image; (**d**) enlarged details of preprocessed image.

2.3. YOLOv8-seg Model Structure Optimisation

YOLOv8 is a major update version created by ultralytics on 10 January 2023, and currently supports tasks such as image classification, object detection, human pose estimation and instance segmentation. YOLOv8 is a new SOTA (state-of-the-art) model, which is currently recognized as the best and most advanced model in the field. YOLOv8-seg is the model for instance segmentation by YOLOv8. The head of YOLOV8-seg has an additional output module for an output mask coefficient, and an FCN layer called Proto module is added for the output mask to achieve instance segmentation [23].

YOLOv8-seg has models of YOLOv8n-seg, YOLOv8s-seg, YOLOv8m-seg and other scales, but the parameters of these models are getting larger and larger, the inference accuracy is getting higher and longer and the inference time is getting longer and longer, so it is not conducive to deployment at the edge. Therefore, in this study, the model structure was optimized based on the YOLOv8n-seg model as follows:

(1) In this study, CBAM was combined with MSDA to form MSDA-CBAM, which was added to backbone to improve the feature extraction ability of the model and improve the precision of the model.
(2) This study innovatively proposes a new neck network, DR-Neck, and replaces the original neck network. In this way, the feature fusion of the model is strengthened and the precision of the model is improved.

2.3.1. MSDA-CBAM

Woo et al. proposed the CBAM in 2018, which focuses on the network's learning of important features in the feature channel dimension and spatial dimension [24]. In this study, the weight of important features was increased and passed into the network to carry out accurate learning classification in a deeper level. Especially, CBAM is a lightweight, general-purpose module that can be seamlessly integrated into any CNN architecture, with negligible additional overhead, and can be trained end-to-end alongside the basic CNN. CBAM was constructed by the fusion of channel attention mechanism and spatial attention mechanism and its structure is shown in Figure 3.

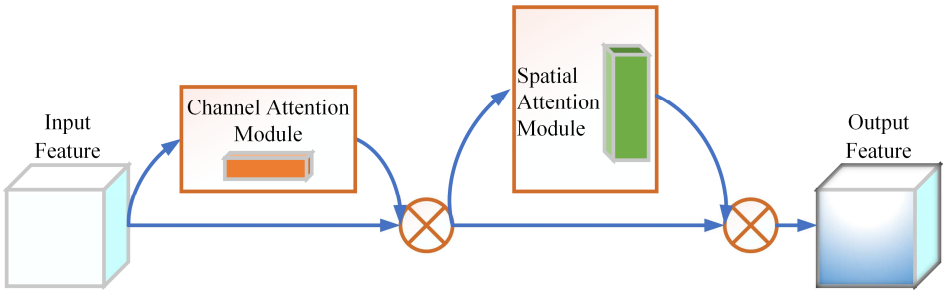


Figure 3. CBAM structure.

The channel attention mechanism performs global average pooling and global maximum pooling operations on the input feature layer, and then the pooling results of the two are processed and added using the shared full connection layer, and finally the channel attention diagram is obtained using the Sigmoid activation function. Through the above operations, the weights of each channel can be generated to highlight important feature channels.

The spatial attention mechanism takes the maximum and average value of the channels of each feature point and stack the results of the two, then adjusts the number of channels by using the convolution of the number of channels as 1, and finally obtains the spatial attention diagram by using the Sigmoid activation function. Through the above operations, the weights of each spatial position can be generated to highlight important feature space regions.

For the road segmentation task of tea garden in hilly areas, the segmentation area is often in the middle position of the image, so the spatial attention mechanism can effectively improve the segmentation accuracy of the model.

MSDA, derived from the study of Jiao et al. on Transformer, is an attention mechanism that can simulate the interaction of local and sparse image blocks in a small range. They found that Vision Transformer (ViT) at a shallow level, the attention matrix has locality and sparseness. This shows that in shallow semantic modeling, the parts far from the query block are largely irrelevant, so there is a lot of redundancy in the global attention module. On the basis of this research, DilateFormer was further constructed, which stacks MSDA blocks in the bottom stage and uses global multi-head self-attention blocks in the high stage. This design allows the model to take advantage of locality and sparsity when dealing with low-level information, while simulating distant dependencies when dealing with high-level information [25]. The basic structure of the MSDA module is shown in Figure 4.

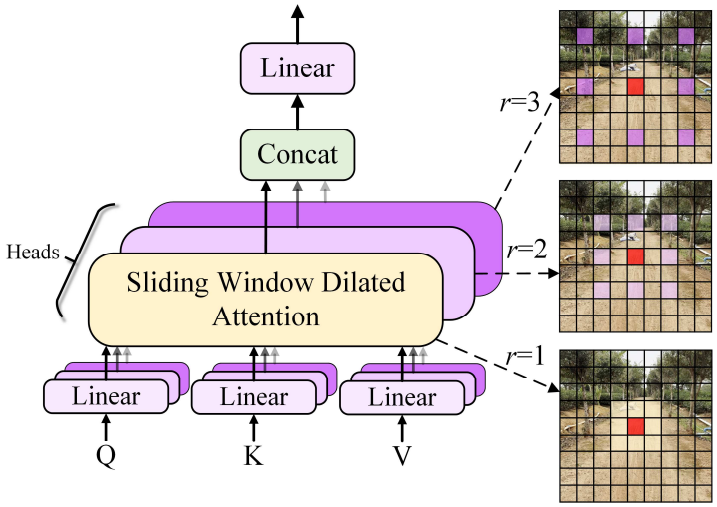


Figure 4. MSDA module structure.

The MSDA module also adopts a multi-head design, dividing the channel of the feature graph into different heads, and performing Strong–Weak Distribution Alignment (SWDA) at different heads using different cavity rates. In this way, semantic information can be aggregated at various scales within the sensory field of attention, and the redundancy of self-attention mechanisms can be effectively reduced without complex operations and additional computational costs. The specific operations are as follows.

For each head, there will be an independent expansion rate. When it is 1, 2 and 3, the receptive fields are 3×3, 5×5 and 7×7, respectively.

Obtain the slice from the feature map, and execute SWDA to obtain the output. The *QKV* parameter is a method used to calculate the correlation between different positions in the input sequence. They stand for query, key and value, all of which are derived from different linear transformations of the embedded vectors of the input sequence. Query and key are used to calculate the similarity score of each position with the others, and then the weight matrix is normalized by SoftMax. Value is used to represent the information of the input sequence, which is multiplied by the weight matrix to obtain the weighted output sequence.

The outputs of all the heads are joined together and then feature aggregated through a linear layer.

Since the small model with depth scaling was used for this paper, the depth of the model was relatively shallow, so MSDA could be applied to achieve efficient feature extraction.

Based on the CBAM, this study replaces the channel attention module in CBAM with the MSDA module and generates the MSDA-CBAM through this operation. The MSDA-CBAM combines the advantages of CBAM and MSDA to achieve multi-scale channel attention capabilities, as well as spatial attention capabilities, and is also a combination of CNN and Transformer. Finally, the improved attention mechanism module is added to the SPPF pooling layer of backbone to improve the feature extraction capability of the backbone network. The structure of the improved attention module is shown in Figure 5.

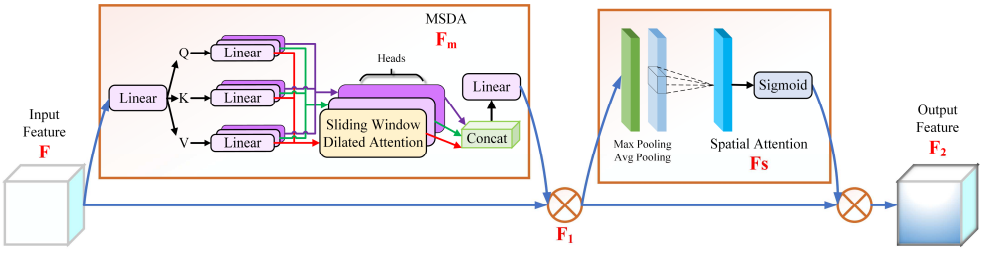


Figure 5. MSDA-CBAM structure.

For MSDA-CBAM, if the input eigenmatrix is $F \in R^{C \times H \times W}$, then the output matrix F_2 satisfies the following:

$$\begin{gathered} F_2 = F_s \otimes F_1 \\ F_1 = F_m \otimes F \\ F_s = Sigmoid(conv(AvgPool(F_1),\ \ MaxPool(F_1))) \\ F_m = Linear(Concat[h_1, \ldots, h_n]) \\ h_i = SWDA(Q_i, K_i, V_i, r_i), 1 \leqq i \leqq n \\ Q_i = F_i W^Q \\ K_i = F_i W^K \\ V_i = F_i W^V \end{gathered} \tag{1}$$

As shown in Formula (1) above, F_s is the weight transformation matrix of spatial attention and F_m is the weight transformation matrix of MSDA mechanism. W^Q, W^K and W^V is a linear transformation matrix after model training. $\otimes$ represents multiply elements. *Sigmoid* is the activation function, *conv* is 7×7 convolution, *AvgPool* is average pooling operation, *MaxPool* is the maximum pooling operation, *Linear* is linear transformation and *Concat* is concatenation operation. SWDA is the sliding window expansion attention operator in Transformer. n means that the feature map is sliced n times according to the *QKV* matrix of the self-attention mechanism. r_i is the expansion rate of the corresponding head in the i-th slice.

2.3.2. DR-Neck Feature Extraction Network

The DR-Neck network is proposed by referring to several excellent feature fusion network structures and combining them with practical application requirements. The network pays full attention to the deep network features from the structure, adopts multiple up-sampling to enrich the edge information of segmentation, uses CBAM to correct the original feature distortion caused by the up-sampling operator and adopts a Semantics and Detail Infusion (SDI) multi-level fusion module to achieve efficient cross-scale feature fusion.

SDI, meaning the fusion of semantics and details, is a multi-level fusion module derived from UNet v2 network [26]. SDI strengthens feature extraction through attention mechanism, preprocesses cross-scale information through up-sampling, down-sampling and other operations, and fuses features through Hadamard product operations to enhance semantic and detailed information in images.

The attention mechanism is added to YOLOv8-seg, and the feature fusion network is modified to DR-Neck, thus forming the DR-YOLO instance segmentation algorithm. The network structure is shown in Figure 6. Lines of different colors indicate that the feature information does not intersect and does not fuse.

The basic idea of DR-Neck is that the feature information of C2f_1 and C2f_2, C2f_2 and C2f_3 is firstly fused through the SDI module for cross-scale feature fusion, and then the feature information of C2f_4 is twice up-sampled, and then cross-scale feature fusion is carried out through the SDI module. Both CBS_1 and CBS_2 convolution modules are 3×3 convolution, and their functions are to keep the number of feature channels unchanged and down-sample the feature information. The function of up-sampling is to improve the resolution of the feature map, so as to improve the precision of the segmentation edge, but the feature loss problem is caused by the up-sampling interpolation operation. Therefore, this study adopts CBAM to optimize the feature information after up-sampling. As an attention mechanism based on CNN, CBAM has the advantages of being lightweight and easily portable. DR-Neck improves the segmentation of large targets by repeatedly referencing the feature information generated by C2f_4.

The original YOLOv8-seg feature fusion network does not directly use the feature information extracted by the C2f_1 module, but the feature map of the module has a higher resolution and a clearer edge, which can be used to improve the accuracy of the segmentation edge of the instance. Therefore, the C2f_1 module is extra invoked by DR Neck.

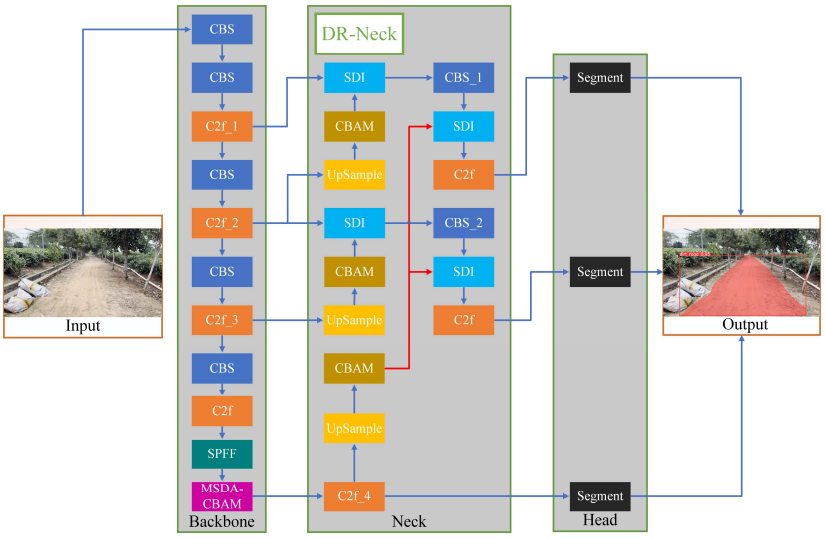


Figure 6. DR-YOLO network structure.

After model structure optimization, the DR-YOLO model is obtained, and its network structure is shown in Table 2.

Table 2. Network input–output tensor.

Tab	Module Names	Input Tensor	Output Tensor
0	Conv	[640,640,3]	[320,320,16]
1	Conv	[320,320,16]	[160,160,32]
2	C2f	[160,160,32]	[160,160,32]
3	Conv	[160,160,32]	[80,80,64]
4	C2f	[80,80,64]	[80,80,64]
5	Conv	[80,80,64]	[40,40,128]
6	C2f	[40,40,128]	[40,40,128]
7	Conv	[40,40,128]	[20,20,256]
8	C2f	[20,20,256]	[20,20,256]
9	SPPF	[20,20,256]	[20,20,256]
10	MSDA-CBAM	[20,20,256]	[20,20,256]
11	C2f	[20,20,256]	[20,20,256]
12	UpSample	[20,20,256]	[40,40,256]
13	CBAM	[40,40,256]	[40,40,256]
14	UpSample	[40,40,128]	[80,80,128]
15	CBAM	[80,80,128]	[80,80,128]
16	SDI	[80,80,128], [80,80,64]	[80,80,128]
17	Conv	[80,80,128]	[40,40,128]
18	SDI	[40,40,128], [40,40,256]	[40,40,128]
19	C2f	[40,40,128]	[40,40,128]
20	UpSample	[80,80,64]	[160,160,64]
21	CBAM	[160,160,64]	[160,160,64]
22	SDI	[160,160,64], [160,160,32]	[160,160,64]
23	Conv	[160,160,64]	[80,80,64]
24	SDI	[80,80,64], [40,40,256]	[80,80,64]
25	C2f	[80,80,64]	[80,80,64]
26	Segment	[80,80,64], [40,40,128], [20,20,256]	[640,640,3]

2.4. DR-YOLO Model Compression

Although the DR-YOLO obtained after model structure optimization has improved its feature extraction and fusion capabilities, model accuracy and other performance, the structural complexity of the model has increased, and the calculation amount of the model has also increased, resulting in a weakening of the efficiency of the model deployed at the

actual edge. Therefore, in order to improve the inference efficiency of the model, the study uses pruning and quantization to compress the model.

2.4.1. Model Pruning

Model pruning is a model compression technology. Its purpose is to delete unimportant or redundant parts in the model, such as neurons, weights, convolutional cores and filters on the premise of maintaining the model performance, so as to reduce the number of parameters and calculation amount of the model [27], and thus reduce the storage space and inference time of the model. Model pruning is conducive to model deployment on the edge [28]. Its principle is shown in Figure 7.

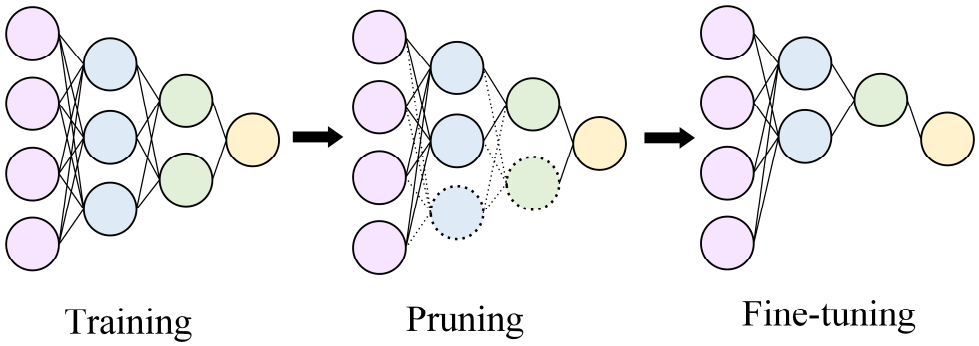


Figure 7. CNN structure pruning diagram.

Fang et al. [29] propose a generic approach, the Dependency Graph algorithm, to automate the search for coupling layers to enable general structural pruning of arbitrary architectures such as CNNS, RNNS, GNNS and Transformers. The biggest difficulty in general structural pruning is structural coupling, that is, when structural pruning is carried out, parameters are automatically grouped through the construction of a network dependency graph, so as to achieve group-level pruning, so as to ensure that the clipped part is almost a useless part of the network and avoid structural problems and significant performance degradation after pruning. This above pruner was named GroupNormPruner by Fang et al. [29] They combined pruners such as GroupNormPruner and MagnitudePruner to form the torch pruning tool. On the basis of previous studies, torch pruning was used to implement model pruning in this study, and a comparative test of multiple pruners was carried out.

2.4.2. Model Quantification and Accelerated Deployment

At the heart of NVIDIA TensorRT is a C++ library that facilitates high-performance reasoning on NVIDIA graphics processing units (GPU). TensorRT takes a trained network and generates a highly optimized real-time running engine to perform efficient inference. TensorRT provides API via C++ and Python to help express deep learning models through network definition API or load predefined models through parsers, enabling TensorRT to optimize and run these models on NVIDIA GPU. TensorRT applies optimization such as graph optimization and layer fusion, while also using highly optimized kernel instruction set to achieve the highest operating efficiency of the model. TensorRT can run on all NVIDIA GPU after the Kepler core.

TensorRT has a variety of network parameter formats, including FP32, FP16, INT8 and TF32. FP32 is a full-precision floating point number, occupying 32 bits, with 1 sign bit, 8 exponent bits, and 23 decimal places. TensorRT is a common precision for training deep learning models, which can guarantee a high accuracy and dynamic range, but the calculation speed is slow and the memory footprint is large. FP16 is a half-precision floating point number that takes up 16 bits, has 1 sign bit, 5 exponent bits and 10 decimal places. It is a low-precision floating-point number that can improve reasoning speed and memory efficiency, but may lose some precision and dynamic range.

The trained YOLO model file is saved in FP32 format by default, and converted to an FP16 format model file by TensorRT which is actually a model quantization operation, that is, the floating point conversion of the neural network to a more condensed format. Although this method may lead to the loss of model accuracy, it can effectively reduce the model volume and memory occupancy, and effectively improve the model inference speed.

In this study, Nvidia TensorRT was used to transform the model (model quantization) and accelerate the deployment, so as to improve the Frames Per Second (FPS) while maintaining the accuracy of the model.

2.5. Experimental Platform

The model training in this paper was carried out on Nvidia RTX 4080 GPU using Pytorch, a deep learning framework, and CUDA and cuDNN acceleration environments were used to accelerate the training. The installed hardware and environment configurations are shown in Table 3.

Table 3. Experimental configuration.

Configuration	Parameters (Training Terminal)
CPU	I7-13700K
GPU	NVIDIA RTX 4080
Operating system	Windows 11
Accelerated environment	CUDA 11.8+ cuDNN 8.7.0
Library	Pytorch 2.0.0

In terms of parameters, taking into account the limitations of memory and video memory, the training batch size was set to 24, and a total of 500 epochs were iterated. The optimizer for training was the SGD optimizer with a maximum learning rate of 1×10^{-2} and minimum learning rate of 1×10^{-4}. The model input size was selected as $640 \times 640 \times 3$. In the training process, it was found that the training of the model tends to converge at about 350 epochs. After the training, the optimal network model structure and parameters were saved and used to evaluate the model performance. Table 4 describes the environment configuration of the deployment end. The deployment end is manufactured by Nvidia, China.

Table 4. Jetson environment configuration.

Index	Parameters (Deploy Phase)
Type	Jetson Orin Nano 4 GB
Operating system	Ubuntu 20.04LTS
Accelerated environment	CUDA 11.4+ cuDNN 8.6.0
Library	Pytorch 1.12
SDK	JetPack 5.1
TensorRT version	TensorRT 8.5.2

2.6. Evaluation Metrics

This study used *Precision*, *Recall*, *GFlops*, Parameters (*Params*), *Latency* and *AP*@0.5 as indicators for evaluating an instance segmentation model. *Precision* is the proportion of True Positives (*TP*) to all predicted positives (*TP* and False Positives (*FP*)), that is, the proportion of all targets predicted correctly by the model. *Recall* is the proportion of the number of pixels of *TP* to the total number of pixels of *TP* and False Negatives (*FN*), that is, the proportion of all real targets that the model predicts correctly. *GFlops* (Giga floating point operations) is a measure of a model's complexity and performance. Generally, the smaller the *GFlops*, the lighter and faster the model, but the *Precision* may also be reduced. *F1* Score is used in statistics to measure the *Precision* of binary (or multi-task binary) models. *F1* Score can be seen as a kind of weighted average of model *Precision* and *Recall*, with a

maximum value of 1 and a minimum value of 0, and a larger value means a better model. *F1* Score can be used to analyze the model specificity, which reflects the performance of the model under different confidence levels, considering the *Precision* and *Recall* rate.

Parameters is the total number of all trainable variables in the model. Parameters reflects the complexity and storage space of the model. *Latency* refers to the time it takes the model to segment an image, reflecting the model's inference speed. *AP* (Average Precision) is a common metric used to evaluate instance segmentation performance, which is a combination of *Precision* and *Recall* that reflects the precision and completeness of the model across different IoU thresholds and categories, so *AP* is the area of the *Precision-Recall* curve, which is used to measure the comprehensive index of detection performance for this category. IoU (Intersection over Union) is a measure of how much the predicted box or mask overlaps with the real box or mask. The higher the value of IoU, the more precise the prediction. *AP*@0.5 indicates the *AP* value when the IoU threshold is set to 0.5. In particular, for the multi-class instance segmentation model, *mAP* (Mean Average Precision) is usually used as an important reference index. *mAP* can be seen as the average of each type of *AP*, which is used to measure the comprehensive performance of the model to predict all targets. It is also the most important and intuitive indicator to comprehensively judge the quality of a network. Since there is only one category "*dirt road*" in this study, the *mAP* values and *AP* are numerically equal.

For YOLOv8-seg, *Precision*, *Recall* and *AP*@0.5 are divided into two groups: box and mask. Box is used to evaluate the precision of the target detection box, and mask is used to evaluate the segmentation precision. In this study, only the mask group data were recorded and analyzed. The preceding indicators are shown in Formulas (2) to (7).

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$AP = \int_0^1 Precison(Recall)dR \tag{4}$$

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N} \tag{5}$$

$$GFlops \sim O\left(\sum_{i=1}^{n} K_i^2 \cdot C_{i-1}^2 \cdot C_i + \sum_{i}^{n} M^2 \cdot C_i\right) \tag{6}$$

$$Params \sim O\left(\sum_{i=1}^{n} M_i^2 \cdot K_i^2 \cdot C_{i-1} \cdot C_i\right) \tag{7}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{8}$$

TP is the number of pixels whose IoU is greater than 0.5, and *FP* is the number of pixels whose IoU is less than 0.5 or the number of pixels that detect the same Ground Truth. *FN* is the number of undetected Ground Truth. *O* is the constant order, *K* is the convolution kernel size, *C* is the number of channels, *M* is the input image size, and i is the number of iterations.

Latency is related to the environment configuration and actual hardware performance, and cannot be calculated by an explicit formula. It requires deployment models and inference. Model inference can be performed using either GPU or CPU, GPU inference is usually much faster than CPU inference because GPU can operate in parallel whereas CPU can only operate serially. Furthermore, GPU also has many designs and optimizations for deep learning accelerated inference. CPU inference can reflect the inference speed of the model with limited hardware performance, and GPU inference can reflect the limiting inference speed of the model with sufficient hardware support. The GPU-accelerated

environment configuration between different types of models is often inconsistent, so it is fairer to apply the CPU inference on the training side to make a comparison. For example, it is not rigorous to use GPU inference as a performance comparison for deep learning models based on TensorFlow and PyTorch frameworks. In addition, different object detection algorithms, different instance segmentation algorithms and different semantic segmentation algorithms are also different types of models. This study uses the training side CPU inference and the actual inference time on the Jetson's GPU as the data source for *Latency*.

3. Results

3.1. Model Structure Optimization Test

3.1.1. Comparative Test of Attention Mechanism

In this study, the CoordAttention and Multi-Scale Convolutional Attention (MSCA) module are respectively given as a comparison, and the additional position is consistent with that of the MSDA-CBAM. CoordAttention can be used to enhance channel representation in lightweight networks. It imparts location information into channel attention by breaking it down into two one-dimensional feature coding processes that aggregate features along different directions. This allows the capture of long-range spatial dependencies while retaining precise location information [30]. MSCA is a component of convolutional network architecture and an important unit in SegNeXt. MSCA uses a multi-branch depth strip convolutional layer to capture multi-scale context information. The output of each branch and the output of the first part are added to form an attention feature map. Finally, multi-scale feature fusion is carried out to obtain the output of the MSCA mechanism structure [31].

The results of adding different attention mechanisms are shown in Table 5. With the addition of the attention mechanism, the complexity and computation amount of the model are improved to some extent. After adding the MSDA mechanism, the model's *GFlops* increased by 2.0%, *Params* increased by 8.2%, *Latency* (CPU) increased by 3.3% and *Latency* (Edge) increased by 13.6%. After the CBAM was added, *GFlops* of the model remained basically unchanged, *Params* increased by 2.0%, *Latency* (CPU) increased by 1.1% and *Latency* (Edge) increased by 3.0%. After adding the MSDA-CBAM, the *GFlops* of the model increased by 2.0%, *Params* increased by 8.2%, *Latency* (CPU) increased by 8.2% and *Latency* (Edge) increased by 14.7%.

Table 5. Attention mechanism improvement experiment.

Models	Precision	Recall	AP@0.5	GFlops	Params	Latency (CPU)	Latency (Edge)
YOLOv8n-seg	0.9806	0.9191	0.9383	10.6362	3.2251M	70.12 ms	31.84 ms
With MSDA	0.9895	0.9234	0.9469	10.8467	3.4881M	72.42 ms	36.17 ms
With MSDA-CBAM	0.9904	0.9263	0.9501	10.8468	3.4882M	75.87 ms	36.52 ms
With MSCA	0.9896	0.9277	0.9473	10.7097	3.3191M	71.78 ms	33.39 ms
With CBAM	0.9883	0.9021	0.9438	10.6366	3.2911M	70.89 ms	32.81 ms
With Coord	0.9890	0.8979	0.9454	10.6438	3.2378M	73.33 ms	33.13 ms

Although the complexity, computation and delay time of the model increased, the *Precision* of the model was generally improved. After adding the MSDA mechanism, *Precision*, *Recall* and *AP*@0.5 increased by 0.9%, 0.5% and 0.9% respectively. After the CBAM was added, the Precision of the model increased by 0.1% and *AP*@0.5 by 0.6%. After adding MSDA-CBAM, *Precision* increased by 1.0%, *Recall* increased by 0.8%, and *AP*@0.5 increased by 1.3%.

It was found that MSDA-CBAM increased the calculation amount and complexity of the model the most, but it improved the *Precision* of the model the most. Compared with MSDA by integrating CBAM, *GFlops* and *Parmas* were basically unchanged, although *Latency* (CPU) and *Latency* (Edge) increased by 4.8% and 1.0%, respectively. However, *Precision*, *Recall* and *AP*@0.5 were improved by 0.1%, 0.3% and 0.3%, respectively. Since

the inference speed can be improved through the following ways, and the model inference speed is still in the optimal range, this method is worthwhile in exchange for the accuracy improvement.

3.1.2. Neck Network Replacement Test

In addition to the modified DR-Neck proposed in this study, there were also attempts to modify the Neck into Gold-YOLO-neck, ASF-FPN and HS-FPN. The test data are shown in Table 6.

Table 6. Feature fusion network improvement experiment.

Models	Precision	Recall	AP@0.5	GFlops	Params	Latency (CPU)	Latency (Edge)
Include MSDA-CBAM	0.9904	0.9263	0.9501	10.8468	3.4882M	75.87 ms	36.52 ms
With DR-Neck	0.9912	0.9249	0.9573	10.8293	3.2754M	116.26 ms	51.71 ms
With Gold-YOLO	0.9881	0.9277	0.9519	13.1077	6.4790M	124.35 ms	55.84 ms
With ASF-FPN	0.9884	0.9228	0.9475	11.3182	3.5329M	91.74 ms	42.09 ms
With HS-FPN	0.9775	0.9256	0.9486	9.5281	2.3710M	70.98 ms	40.43 ms

Compared with the segmentation model with the added attention mechanism, the *Latency* (CPU) and *Latency* (Edge) of the model after the neck network was replaced by DR-Neck increased by 53.2% and 41.6%, but *GFlops* decreased by 1.6% and *Params* decreased by 6.1%. Because many lightweight operations in models trade space (*Params*, *GFlops*) for time (*Latency*), such as depthwise separable convolutions, the final model *AP*@0.5 is increased by 0.8%. Similar to the attention mechanism above, the reasoning time is worth the precision. Compared with Gold-YOLO and ASF-FPN, DR-Neck has higher precision and lower model complexity. HS-FPN network can improve the running speed of the model by reducing the convolutional kernel of the feature fusion network, and its model complexity is also the lowest, but its *Precision* is also the worst. In particular, Gold-YOLO, ASF-FPN and HS-FPN are not effective in improving precision because these networks are often designed for small target detection and segmentation, and their recognition effectiveness for large objects may not be good.

3.2. Ablation Experiments

In order to better reflect the influence of different improvement measures on the parameters and performance of the model, this study trained YOLOv8-seg models of different scales for ablation experiments.

As shown in Table 7, compared with the YOLOv8n-seg model, the *Latency* (CPU) and *Latency* (Edge) of DR-YOLO increased by 65.8% and 62.4%, but the *AP*@0.5 and *Precision* increased by 2.0% and 1.1% respectively. The DR-YOLO model is comprehensively superior to YOLOv8s-seg, which is reflected in the higher precision and faster inference speed of DR-YOLO. DR-YOLO precision is similar to YOLOv8m-seg, but its *GFlops* and *Params* are significantly lower than YOLOv8m-seg, and its inference speed is significantly higher than YOLOv8m-seg. This demonstrates that DR-YOLO's model improvement work is effective and shows that DR-YOLO is more suitable for edge–side development reasoning.

Table 7. Ablation experiment.

Basic Models	MSDA-CBAM	DR-Neck	Precision	AP@0.5	GFlops	Params	Latency (CPU)	Latency (Edge)
YOLOv8m-seg	-	-	0.9939	0.9572	96.396	26.885M	248.74 ms	113.77 ms
YOLOv8s-seg	-	-	0.9917	0.9488	37.142	11.649M	135.54 ms	55.16 ms
YOLOv8n-seg	-	-	0.9806	0.9383	10.636	3.225M	70.12 ms	31.84 ms
YOLOv8n-seg	√	-	0.9904	0.9501	10.847	3.488M	75.87 ms	36.52 ms
YOLOv8n-seg	-	√	0.9823	0.9457	10.198	3.002M	108.27 ms	39.07 ms
YOLOv8n-seg	√	√	0.9912	0.9573	10.829	3.275M	116.26 ms	51.71 ms

3.3. Specificity of the Model

Due to the larger convolution number and deeper network of YOLOv8s-seg and YOLOv8m-seg, *F1* Scores of both YOLOv8n-seg and YOLOv8n-seg are better than those of YOLOv8n-seg at both low- and high-confidence levels. After the model structure optimization, the *F1* Score of DR-YOLO is significantly better than that of YOLOv8n-seg in the wide confidence interval.

As is shown in Figure 8, within the confidence range of 0.5–0.9, the *F1* Score of DR-YOLO is better than that of the YOLOv8s-seg model, that is, the performance of the DR-YOLO model is better than that of YOLOv8s-seg. Within the confidence range of 0.55–0.75, the *F1* Score of DR-YOLO is very close to that of the YOLOv8m-seg model, and the performance of the two models is very close.

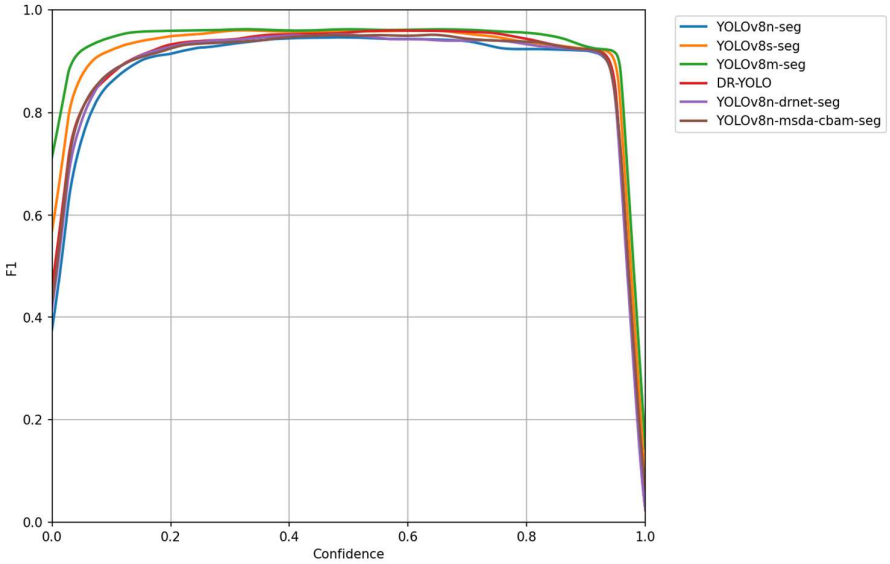


Figure 8. Specificity of the model.

3.4. Model Visualization Analysis

In order to verify the high efficiency and precision of DR-YOLO deployment, various model visualization tests were carried out on Jetson and the training terminal, including an instance segmentation test of the model in complex scenes and a class activation mapping (CAM) test in complex scenes.

3.4.1. Segmentation Result Analysis

As shown in Figure 9a(I),b(I), the road of tea garden in hilly areas has a boundary between concrete road and dirt road, and human occlusion. During the actual deployment of the original YOLOv8n-seg model, its *Recall* and *Precision* were insufficient, resulting in detection frame errors and mask errors.

Furthermore, as shown in Figure 9c(I), due to insufficient robustness, the confidence of the original YOLOv8n-seg model will decline in the unfamiliar and environmentally disturbed tea garden path scene. Similarly, the confidence will decrease in the case of insufficient light, as shown in Figure 9d(I). Figure 9e(I) is a complex scene with multiple obstacles, and the model has the defect of edge distortion when it is segmented.

After the model structure optimization, DR-YOLO has a comprehensive improvement in precision compared with the YOLOv8n-seg model. Figure 9 clearly shows that the DR-YOLO model can effectively solve problems such as detection frame error, mask error, confidence decline and edge distortion caused in the YOLOv8n-seg model.

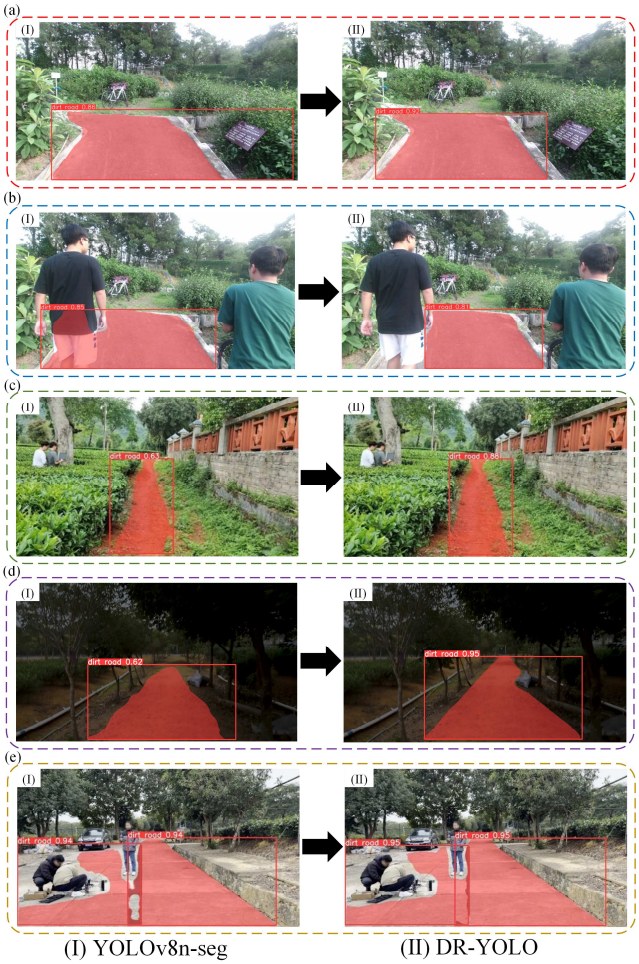


Figure 9. Segmentation results of road examples in complex scenarios. (**a**) A mix of concrete and dirt roads; (**b**) human occlusion; (**c**) tea garden path scene; (**d**) insufficient light tea garden roads; (**e**) complex scenes with multiple obstacles.

3.4.2. Class Activation Mapping Test

CNN can only obtain the final result when dealing with the problem of instance segmentation, its interpretability in the process of network processing is not strong. Therefore, in this study, the visualization effects of YOLOv8n-seg and the improved DR-YOLO model were compared by using a class activation map (CAM), and the features extracted by road instance segmentation after the last convolution were visualized.

The model visualization method adopted in this paper is EigenCAM. EigenCAM is an interpretable method for computer vision proposed by Muhammad and Yeasin [32]. EigenCAM performs Principal Component Analysis (PCA) on the two-dimensional activation matrix of the convolution layer, and takes the first principal component as the CAM. EigenCAM does not require class-related information and does not rely on gradients, activation values or other weights to show which areas of the image contribute most to the model's predictions.

The darker the color of the high-temperature region, that is, the red region in the CAM, the more attention the network pays to the features of this place, as shown in Figure 10. The feature extraction ability of DR-YOLO is generally stronger than that of the unimproved YOLOv8n-seg. The MSDA-CBAM and DR-Neck network can significantly improve the model's attention weight to the road features of the tea garden in hilly areas, and better integrate high-level semantic information and low-level semantic information, which is conducive to the accurate segmentation of roads. The instance segmentation model, DR-

YOLO, is more suitable to be deployed to the edge computing side for segmentation of tea garden roads in hilly areas.

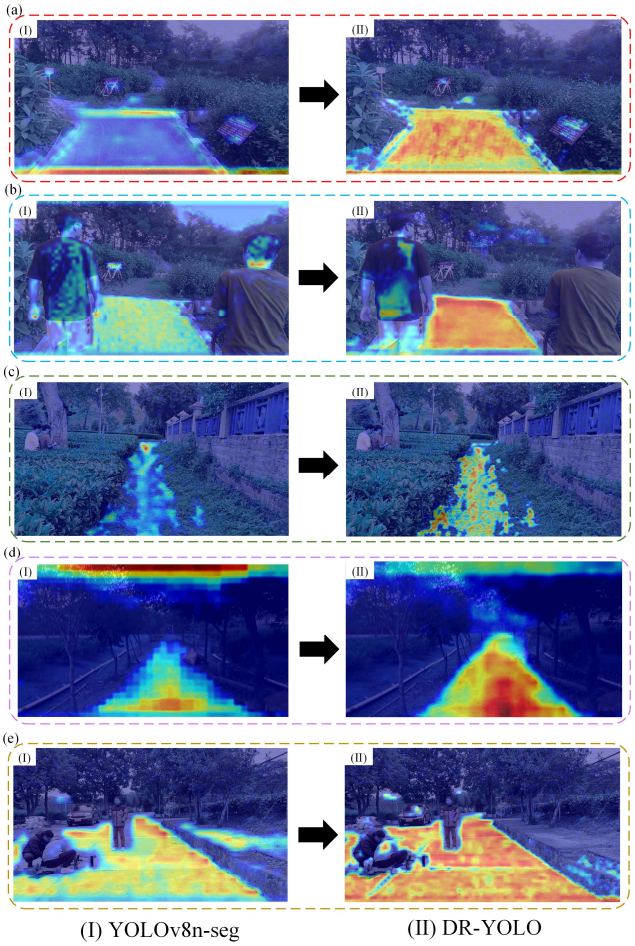


Figure 10. Class activation map. (**a**) A mix of concrete and dirt roads; (**b**) human occlusion; (**c**) tea garden path scene; (**d**) insufficient light tea garden roads; (**e**) complex scenes with multiple obstacles.

3.5. Comparative Tests of Similar Models

This study compares the performance of DR-YOLO with other common instance segmentation and semantic segmentation models, including the following models: YOLACT, a one-stage instance segmentation algorithm designed for real-time inference [33]; Mask-RCNN, a classical two-stage instance segmentation algorithm that divides the instance segmentation into two steps; Faster R-CNN, which generates candidate frames and category predictions, and then generates masks for each candidate frame using a full convolutional network (FCN) [34]; DeepLab V3+, a semantic segmentation model that uses Atrous Convolution and ASPP (Atrous Spatial Pyramid Pooling) to enhance feature extraction and fusion at different scales, and at the same time, a global average pooling layer is added to introduce image-level features [35]; SegFormer, a neural network model for semantic segmentation that combines a Transformer encoder with a lightweight multi-layer perceptron (MLP) decoder to achieve high segmentation accuracy while increasing inference rates [36]; in particular, IoU is introduced as an index to evaluate the accuracy of the semantic segmentation model, where IoU refers to the intersection of the true value and the predicted value of the pixel divided by the union of the real value and the predicted value of the pixel.

In this study, the input resolution of different models is set at 640 × 640, which is consistent with YOLOv8-seg. However, the training hyperparameters and configuration environment of different neural network models are different, so variable unification cannot

be achieved. This study makes a performance comparison only after ensuring that the trained models are close to or have converged. Moreover, there may also be small errors between the evaluation procedures of different models. The test results are shown in Table 8 and Figure 11.

Table 8. Model performance comparison experiment.

Models	Precision	AP@0.5	mIoU	GFlops	Params	Latency (CPU)
DR-YOLO	0.9912	0.9573	-	10.408	3.265M	116.26 ms
YOLACT	-	0.9352	-	81.423	34.727M	517.85 ms
Mask-RCNN	-	0.9680	-	142.43	63.617M	992.81 ms
DeepLab V3+	0.9476	-	0.9059	17.627	2.754M	150.92 ms
SegFormer_b0	0.9653	-	0.9438	21.152	3.715M	354.90 ms

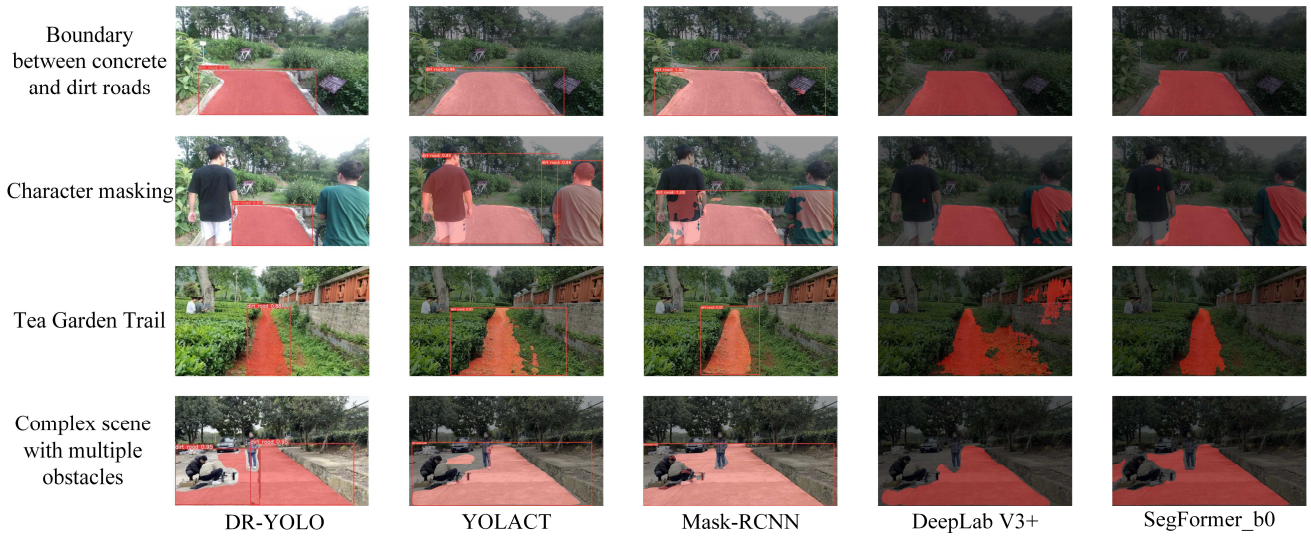


Figure 11. Performance comparison experiment of DR-YOLO, YOLACT, Mask-RCNN, DeepLab V3+ and SegFormer_b0 in four scenarios.

In this study, the backbone network of Mask-RCNN and YOLACT is ResNet-50. As the lightest model, DR-YOLO also has good precision compared to other instance segmentation models, and is comprehensively superior to YOLACT algorithm. Mask-RCNN has the highest *AP*@0.5, but its inference speed is too slow for deployment at the edge. Compared with other semantic segmentation models, DR-YOLO has advantages in *Precision*. Although mIoU and *AP*@0.5 cannot be directly compared, they are of certain reference value.

The backbone network of DeepLab V3+ trained in this study adopts MobileNetv3, which has a faster inference speed but slightly lower precision. The backbone network codename of SegFormer trained in this study is b0, which is the lightest backbone network of SegFormer. Although the precision of SegFormer is greatly improved compared with DeepLab V3+, SegFormer is based on Transformer architecture, which is not efficient in practical reasoning.

In conclusion, DR-YOLO has the fastest inference speed and high precision, and is more suitable for deployment at the edge.

3.6. Model Compression

3.6.1. Pruning Tests

In this study, an advanced pruning device in a torch pruning project was used to conduct experiments on three pruning methods: GroupNorm (group normalization), Lamp [37] and L1 [38]. The three pruning methods do not require sparse training, but require more fine-tuning epochs to restore model precision.

In this study, the target sparsity was set at 0.49, that is, after pruning, the model MACs (Multiply-Accumulate Operations) was about 0.49 times of the original. In order to reduce the precision loss caused by pruning, the method of iterative pruning was adopted in this study, and the sparsity was set to 0.7 for each pruning and 0.49 after two pruning cycles. Fine-tuning training was carried out after pruning, and the parameters of fine-tuning training were consistent with the basic parameters of normal model training. Only 300 epochs were set for each fine-tuning round, and the total fine-tuning round was 600 epochs. In addition, the parts that affect the structure or feature expression of the network, such as the output layer of the split head, were skipped (without pruning). The pruning test results are shown in Table 9, and the changes of some channels in the pruning test are shown in Figure 12.

Table 9. Pruning results.

Models	Pruning Methods	Precision	Recall	AP@0.5	GFlops	Params	Latency (CPU)	Latency (Edge)
YOLOv8n-seg	-	0.9806	0.9191	0.9383	10.636	3.225M	70.12 ms	31.84 ms
DR-YOLO	-	0.9912	0.9249	0.9573	10.829	3.275M	116.26 ms	51.71 ms
DR-YOLO	GN	0.9789	0.9149	0.9518	5.009	1.427M	99.63 ms	49.77 ms
DR-YOLO	Lamp	0.9897	0.9191	0.9538	6.530	1.301M	102.12 ms	50.51 ms
DR-YOLO	L1	0.9808	0.9181	0.9521	5.202	1.421M	101.34 ms	49.86 ms

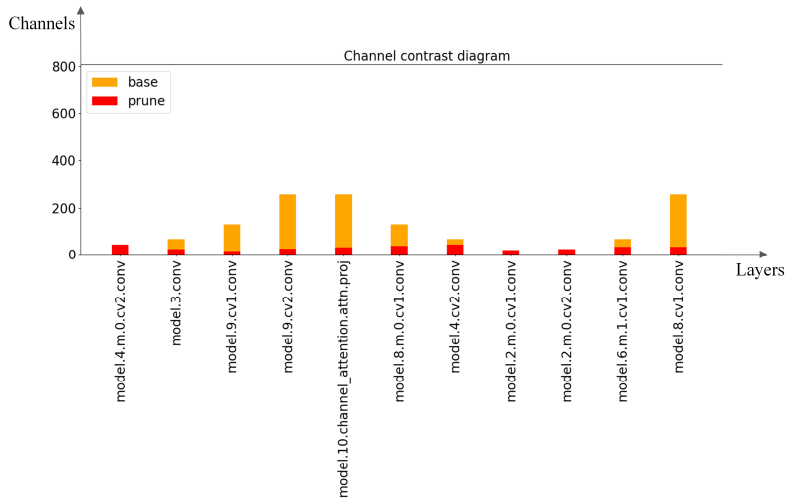


Figure 12. The changes of some channels in the pruning test.

As shown in Table 9, after different pruning methods, the performance of the model had different changes. After the Lamp pruning test, *AP*@0.5 decreased by 0.4%, but *GFlops* decreased by 39.7%, *Params* decreased by 60.3%, *Latency* (CPU) decreased by 12.2% and *Latency* (Edge) decreased by 2.3%. The pruned model is more conducive to reducing memory space consumption and more conducive to deployment at the edge.

At the edge, pruning is not effective for accelerating model inference, because pruning does not reduce the number of network layers, that is, it does not reduce the total number of various modules, and also shows that even if some channels are reduced, the memory cache performance at the edge is still a performance bottleneck. GroupNorm and L1 pruning can obtain a slightly better acceleration effect, but the precision loss is too large, so it is not adopted in this study. Finally, the Lamp pruning model was used for follow-up experiments.

After pruning test, the channels of the model tended to be at the same level, that is, pruning could optimize the problem of excessive channel number of the model, which is conducive to the transmission efficiency of the model feature information.

3.6.2. Quantitative Tests

In this study, TensorRT and YOLOv8 environments were deployed on Jetson, and the .pt files, trained by Pytorch, were converted into engine files through TensorRT conversion scripts, so as to realize the deployment of TensorRT accelerated reasoning. The results of DR-YOLO, YOLOv8-seg and other models after TensorRT conversion (quantization) are shown in Table 10 and Figure 13.

Table 10. TensorRT conversion test.

Models	Precision	Recall	AP@0.5	Latency (Edge)
YOLOv8n-seg	0.9806	0.9191	0.9383	31.84 ms
YOLOv8s-seg	0.9917	0.9268	0.9488	55.16 ms
YOLOv8m-seg	0.9939	0.9362	0.9572	113.77 ms
YOLOv8n-seg-FP16	0.9803	0.9064	0.9349	22.32 ms
YOLOv8s-seg-FP16	0.9911	0.9177	0.9476	31.64 ms
YOLOv8m-seg-FP16	0.9933	0.9237	0.9562	59.54 ms
DR-YOLO-P	0.9897	0.9191	0.9538	50.51 ms
DR-YOLO-P-FP32	0.9867	0.9188	0.9532	39.44 ms
DR-YOLO-P-FP16	0.9863	0.9190	0.9530	26.41 ms

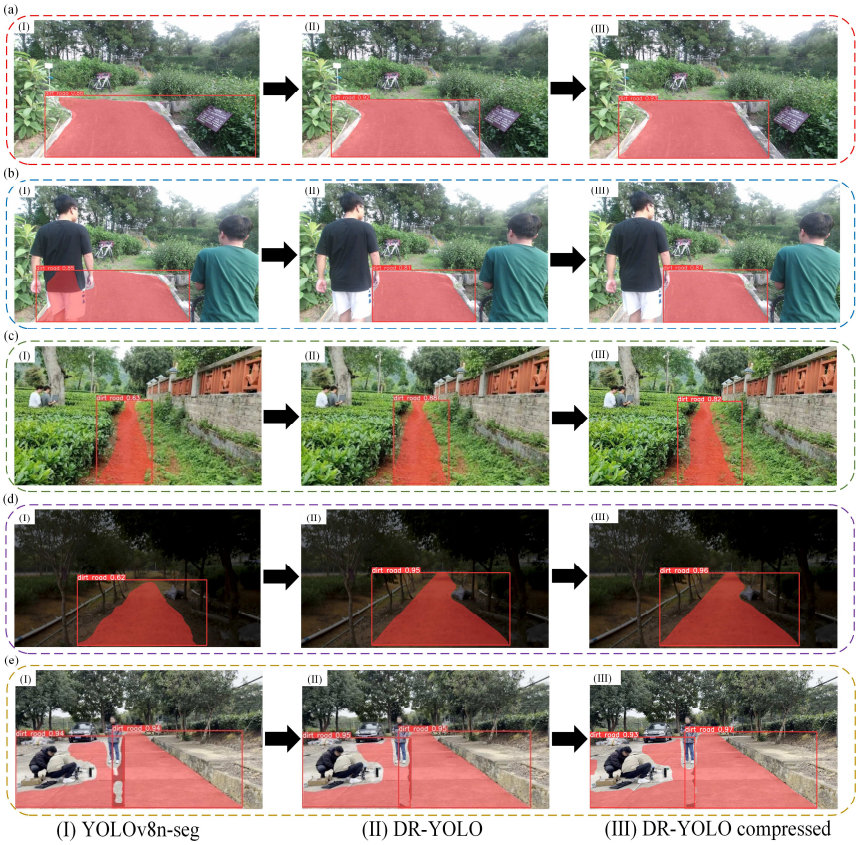


Figure 13. Comparison of segmentation results after model compression. (**a**) A mix of concrete and dirt roads; (**b**) human occlusion; (**c**) tea garden path scene; (**d**) insufficient light tea garden roads; (**e**) complex scenes with multiple obstacles.

After TensorRT conversion, the precision of the model slightly decreased but the deployment efficiency of the model improved. The inference time after YOLOv8n-seg conversion reduced by 29.9%, the inference time after YOLOv8s-seg conversion reduced by 42.6%, and the inference time after YOLOv8m-seg conversion reduced by 47.7%. This shows that the acceleration effect of TensorRT is better for large models and less effective

for small models, which is due to performance limitations that cause small models to not greatly improve the inference speed.

After the pruned DR-YOLO model was converted into FP16, the DR-YOLO-FP16 model *Recall* was reduced, *Precision* and *AP*@0.5 were reduced by 0.3% and 0.1%, but the inference time was reduced by 47.7%. Compared with the YOLOv8m-seg model converted to FP16, the DR-YOLO-FP16 model had similar performance to *AP*@0.5, but the inference speed was about 2.25 times faster. Compared with YOLOv8n-seg, the *Precision* of DR-YOLO-FP16 increased by 0.6%, *AP*@0.5 increased by 1.6% and the inference time reduced by 17.1%. Therefore, the DR-YOLO-FP16 (referred to as DR-YOLO) model was finally adopted in this study for edge deployment.

4. Discussion

Aiming at the automation and intellectualization of auxiliary operation machinery in tea gardens in hilly areas, in this study, we proposed DR-YOLO algorithm based on YOLOv8n-seg, an instance segmentation algorithm for tea gardens in hilly areas through model structure optimization and model compression technology. Compared with the basic model, the algorithm improves the precision by optimizing the model structure, and greatly improves the reasoning speed after the model compression, so as to realize the lightweight network under high precision. The algorithm can be easily deployed on the edge computing end to achieve accurate segmentation of fruit and tea gardens in hilly areas and accurate removal of obstacles, so as to improve the perception ability of auxiliary operation machinery to obstacles on the road of tea gardens in hilly areas. The specific improvements are summarized as follows:

(1) Firstly, we optimized the model of YOLOv8n-seg; the feature extraction network based on the CBAM and MSDA was enhanced to improve the feature extraction capability of the network.
(2) Secondly, we modified the Neck network into the DR-Neck feature fusion network to improve the feature fusion capability of the model. The DR-YOLO model was obtained through model optimization, compared with the YOLOv8n-seg model, the *Precision* increased by 1.108% and the *AP*@0.5 increased by 2.025%.
(3) Then, we carried out pruning and quantitative model compression on the DR-YOLO model to improve the model inference speed with little reduction in *AP*. After model compression, compared with the YOLOv8n-seg model, the *Precision* of DR-YOLO increased by 0.581%, the *AP*@0.5 increased by 1.57% and the inference time reduced by 17.05%. The final test results also show that the model compression has almost no loss of precision, but brings objective inference speed.

5. Conclusions

Agricultural terrain and obstacle sensing technology can enable agricultural machinery to sense, detect and identify obstacles only through its own sensors in complex environments, providing a reliable basis for reasonable planning of local self-walking paths. Our improved and optimized algorithm plays an excellent role in assisting the operation of tea gardens in real hilly areas. It has the advantages of lightweight structure and strong real-time performance, and can be easily deployed on agricultural inspection robots, light transport aircraft and traditional non-intelligent agricultural machines to realize obstacle perception and assist operations. However, there are some limitations in this study. This study collected the road data of tea gardens in Guangzhou, Qingyuan and Meizhou, Guangdong Province. Based on this data set, we can further expand the data set to make the feature information contained in the data set more complex and rich in the future. In addition, we will also collect data on different weather, for example, rainy weather, to expand the scope of the model. These improved methods can improve the robustness and generalization of the model after training. In the construction process of the DR-YOLO algorithm, we will continue to add ablation experiments of different attention mechanisms and Neck networks to obtain models with better performance. The model can be deployed

at the edge. And the segmentation results can be transmitted to the ROS system to achieve the precise segmentation of hilly and mountainous fruit gardens, as well as precise removal of obstacles. In addition, this algorithm can be applied to the ROS system of lychee orchard robots to achieve intelligent operations in lychee orchards. For example, the improved YOLOv8n-seg algorithm can be deployed to the vision system of lychee picking robots to achieve intelligent obstacle avoidance on lychee orchard roads.

Author Contributions: Data curation, Z.H., J.L., Z.Z. and T.C.; funding acquisition, Z.Z. and W.W. (Weibin Wu); project administration, W.W. (Weibin Wu); formal analysis, Z.H., J.L., Z.Z., T.C. and Q.L.; investigation, Y.L., W.W. (Weibin Wu), W.W. (Weihui Wu) and Z.Z.; methodology, Z.H., J.L., T.C., Z.Z. and Y.L.; software, Z.H., J.L., Z.Z., T.C. and W.W. (Weihui Wu); supervision, W.W. (Weibin Wu), Y.L., W.W. (Weihui Wu) and Z.Z.; resources, T.C., Z.H. and Z.Z.; validation, W.W. (Weibin Wu), J.L., Q.L. and Z.H.; writing—original draft, Z.H., J.L., Z.Z., Q.L., T.C. and W.W. (Weibin Wu); writing—review and editing, W.W. (Weibin Wu), J.L., Y.L. and Q.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the 2024 Rural Revitalization Strategy Special Funds Provincial Project (2023LZ04), Guangdong Province (Shenzhen) Digital and Intelligent Agricultural Service Industrial Park (FNXM012022020-1), Construction of Smart Agricultural Machinery and Control Technology Research and Development, and the 2023 Guangdong Provincial Special Fund For Modern Agriculture Industry Technology Innovation Teams (2023KJ120).

Data Availability Statement: The data presented in this study are available in the article.

Acknowledgments: The authors acknowledge the editors and reviewers for their constructive comments and all the support in this work.

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

References

1. Chen, T.; Li, H.; Chen, J.; Zeng, Z.; Han, C.; Wu, W. Detection network for multi-size and multi-target tea bud leaves in the field of view via improved YOLOv7. *Comput. Electron. Agric.* **2024**, *218*, 108700. [CrossRef]
2. Li, J.; Li, J.; Zhao, X.; Su, X.; Wu, W. Lightweight detection networks for tea bud on complex agricultural environment via improved YOLO v4. *Comput. Electron. Agric.* **2023**, *211*, 107955. [CrossRef]
3. Huo, H.; Zou, X.; Chen, Y.; Zhou, X.; Chen, M.; Li, C.; Pan, Y.; Tang, Y. Visual Robot Obstacle Avoidance Planning and Simulation Using Mapped Point Clouds. *J. Syst. Simul.* 2024. Available online: http://kns.cnki.net/kcms/detail/11.3092.V.20230823.0932.002.html (accessed on 12 June 2024).
4. Ye, Z.; Chen, Y.; Huang, P.; Zhong, Y. Design of intelligent obstacle-avoiding mobile robot based on ROS and deep learning. *Internet Things Technol.* **2021**, *11*, 62–64.
5. Wei, J.; Pan, S.; Tian, G.; Gao, W.; Sun, Y. Design and experiments of the binocular visual obstacle perception system for agricultural vehicles. *Trans. Chin. Soc. Agric. Eng. (Trans. CSAE)* **2021**, *37*, 55–63.
6. Meng, Z.; Wang, H.; Fu, W.; Liu, M.; Yin, Y.; Zhao, C. Research Status and Prospects of Agricultural Machinery Autonomous Driving. *Trans. Agric. Mach.* **2023**, *54*, 1–24.
7. Oliveira, L.F.P.; Moreira, A.P.; Silva, M.F. Advances in agriculture robotics: A state-of-the-art review and challenges ahead. *Robotics* **2021**, *10*, 52. [CrossRef]
8. Kragh, M.; Jørgensen, R.N.; Pedersen, H. Object detection and terrain classification in agricultural fields using 3D lidar data. In Proceedings of the International Conference on Computer Vision Systems, ICVS 2015, Copenhagen, Denmark, 6–9 July 2015; Springer International Publishing: Cham, Switzerland, 2015; pp. 188–197.
9. Shi, L.; Liu, H.; Liu, W.; Yang, X.; Li, J.; Wang, P. Research on Obstacle Avoidance Method of Orchard Mower Based on Lidar. *Agric. Mech. Res.* **2023**, *45*, 62–66.
10. Chen, Y.; Zhang, X.; Wu, C.; Li, G. Field-road trajectory segmentation for agricultural machinery based on direction distribution. *Comput. Electron. Agric.* **2021**, *186*, 106180. [CrossRef]
11. Chen, Y.; Li, G.; Zhang, X.; Jia, J.; Zhou, K.; Wu, C. Identifying field and road modes of agricultural Machinery based on GNSS Recordings: A graph convolutional neural network approach. *Comput. Electron. Agric.* **2022**, *198*, 107082. [CrossRef]
12. Chen, Y.; Li, G.; Zhou, K.; Wu, C. Field–Road Operation Classification of Agricultural Machine GNSS Trajectories Using Spatio-Temporal Neural Network. *Agronomy* **2023**, *13*, 1415. [CrossRef]
13. Higuti, V.A.; Velasquez, A.E.; Magalhaes, D.V.; Becker, M.; Chowdhary, G. Under canopy light detection and ranging-based autonomous navigation. *J. Field Robot.* **2019**, *36*, 547–567. [CrossRef]
14. Wan-zhi, Z.; Zeng-cai, W. Rural road detection of color image in complicated environment. *Int. J. Signal Process. Image Process. Pattern Recognit.* **2013**, *6*, 161–168.

15. Mao, J.; Cao, Z.; Wang, H.; Zhang, B.; Guo, Z.; Niu, W. Agricultural robot navigation path recognition based on k-means algorithm for large-scale image segmentation. In Proceedings of the 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), Xi'an, China, 19–21 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1233–1237.
16. Kim, W.S.; Lee, D.H.; Kim, Y.J.; Kim, T.; Lee, H.J. Path detection for autonomous traveling in orchards using patch-based CNN. *Comput. Electron. Agric.* **2020**, *175*, 105620. [CrossRef]
17. Li, X.; Su, J.; Yue, Z.; Duan, F. Adaptive multi-ROI agricultural robot navigation line extraction based on image semantic segmentation. *Sensors* **2022**, *22*, 7707. [CrossRef]
18. Zhang, L.; Li, M.; Zhu, X.; Chen, Y.; Huang, J.; Wang, Z.; Hu, T.; Wang, Z.; Fang, K. Navigation path recognition between rows of fruit trees based on semantic segmentation. *Comput. Electron. Agric.* **2024**, *216*, 108511. [CrossRef]
19. Yang, Z.; Ouyang, L.; Zhang, Z.; Duan, J.; Yu, J.; Wang, H. Visual navigation path extraction of orchard hard pavement based on scanning method and neural network. *Comput. Electron. Agric.* **2022**, *197*, 106964. [CrossRef]
20. Sun, Y.; Gong, L.; Zhang, W.; Gao, B.; Li, Y.; Liu, C. Drivable Agricultural Road Region Detection Based on Pixel-Level Segmentation with Contextual Representation Augmentation. *Agriculture* **2023**, *13*, 1736. [CrossRef]
21. Yu, J.; Zhang, J.; Shu, A.; Chen, Y.; Chen, J.; Yang, Y.; Tang, W.; Zhang, Y. Study of convolutional neural network-based semantic segmentation methods on edge intelligence devices for field agricultural robot navigation line extraction. *Comput. Electron. Agric.* **2023**, *209*, 107811. [CrossRef]
22. Zhang, Z.; Han, C.; Wang, X.; Li, H.; Li, J.; Zeng, J.; Sun, S.; Wu, W. Large field-of-view pine wilt disease tree detection based on improved YOLO v4 model with UAV images. *Front. Plant Sci.* **2024**, *15*, 1381367. [CrossRef]
23. Zhao, X.; Ding, W.; An, Y.; Du, Y.; Yu, T.; Li, M.; Tang, M.; Wang, J. Fast segment anything. *arXiv* **2023**, arXiv:2306.12156.
24. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
25. Jiao, J.; Tang, Y.M.; Lin, K.Y.; Gao, Y.; Ma, A.J.; Wang, Y.; Zheng, W.S. Dilateformer: Multi-scale dilated transformer for visual recognition. *IEEE Trans. Multimed.* **2023**, *25*, 8906–8919. [CrossRef]
26. Peng, Y.; Sonka, M.; Chen, D.Z. U-Net v2: Rethinking the Skip Connections of U-Net for Medical Image Segmentation. *arXiv* **2023**, arXiv:2311.17791.
27. Ning, X.; Zhao, W.; Zong, Y.; Zhang, Y.; Chen, H.; Zhou, Q.; Ma, J. An overview of the joint optimization method for neural network compression. *CAAI Trans. Intell. Syst.* **2024**, *19*, 36–57.
28. Zhang, Y.; Wu, H.; Lin, F.; Huang, F.; Liu, Y. Deep learning model pruning technology in image recognition. *J. Nanjing Univ. Sci. Technol.* **2023**, *47*, 699–707.
29. Fang, G.; Ma, X.; Song, M.; Mi, M.B.; Wang, X. Depgraph: Towards any structural pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 16091–16101.
30. Hou, Q.; Zhou, D.; Feng, J. Coordinate attention for efficient mobile network design. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 13713–13722.
31. Guo, M.H.; Lu, C.Z.; Hou, Q.; Liu, Z.; Cheng, M.M.; Hu, S.M. Segnext: Rethinking convolutional attention design for semantic segmentation. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 1140–1156.
32. Muhammad, M.B.; Yeasin, M. Eigen-cam: Class activation map using principal components. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–7.
33. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. Yolact: Real-time instance segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9157–9166.
34. He, K.; Gkioxari, G.; Dollár, P. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 2980–2988.
35. 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 European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 801–818.
36. Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J.M.; Luo, P. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12077–12090.
37. Lee, J.; Park, S.; Mo, S.; Ahn, S.; Shin, J. Layer-adaptive sparsity for the magnitude-based pruning. *arXiv* **2020**, arXiv:2010.07611.
38. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning filters for efficient convnets. *arXiv* **2016**, arXiv:1608.08710.

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.