YOLOv8np-RCW: A multi-task deep learning model for comprehensive visual information in tomato harvesting robot

Xinyi Ai[1], Tianxue Zhang[2,3*], Ting Yuan[1*], Xiajun Zheng[1], Ziming Xiong[1], Jiace Yuan[1]

(1. *College of Engineering, China Agricultural University, Beijing 100083, China*;
2. *School of Mechanical Engineering and Automation, Beihang University, Beijing 100191, China*;
3. *Institute of Medical Robotics, Shanghai Jiaotong University, Shanghai 200240, China*)

Abstract: In greenhouse environments, using automated machines for tomato harvesting to reduce labor consumption is a future development trend. Accurate and effective visual recognition is essential to accomplish harvesting tasks. However, most current studies use various models to gain harvesting information in multiple steps, resulting in heavy calculation costs, poor real-time availability, and weak recognition precision. In this study, an improved YOLOv8np-RCW end-to-end model based on YOLOv8n pose is proposed to simultaneously detect tomato bunches, maturity, and keypoints using a decoupled-head structure. The model integrates a ResNet-enhanced RepVGG architecture for a balance of accuracy and speed, employs the CARAFE upsampling algorithm for a larger receptive field with lightweight design, and optimizes the loss function with WIoU loss to enhance bounding box prediction, maturity detection, and keypoint extraction. Experimental results indicate that mAP50 of YOLOv8np-RCW model for bounding box and keypoints is 87.3% and 86.8% respectively, which is 6.2% and 5.5% higher than YOLOv8n pose model. Completing the tasks of bunch detection, maturity assessment, and keypoint localization requires only 9.8 ms. Euclidean distance error is less than 20 pixels in detecting keypoints. Based on this model, a method is proposed to quickly determine the orientation of tomato bunches using geometric cross-product and cross-multiplication calculations from keypoint 2D information, providing guidance for the motion planning of the end-effector. In field experiments, the robot achieved a harvesting success rate of 68%, with an average time of 10.8366 seconds per tomato bunch.

Keywords: tomato bunch detection, maturity detection, keypoint detection, harvesting robots
DOI: 10.25165/j.ijabe.20251805.9719

Citation: Ai X Y, Zhang T X, Yuan T, Zheng X J, Xiong Z M, Yuan J C. YOLOv8np-RCW: A multi-task deep learning model for comprehensive visual information in tomato harvesting robot. Int J Agric & Biol Eng, 2025; 18(5): 246–258.

1 Introduction

As a widely consumed vegetable, tomatoes are in great demand globally[1,2]. To meet people's daily needs and facilitate management and production, most tomatoes are grown in greenhouse mode. Open-field farming is slowly being supplanted by facility agriculture[3]. However, with fewer people working in farming, there is a shortage in agriculture labor force resources[4,5]. Manual harvesting of tomatoes in greenhouses is inefficient and highly expensive[6,7]. Therefore, machine technology is essential to solve these concerns[8].

In the working process of agricultural harvesting machine, deep learning technology has been applied to obtain fruit information. Xiong et al.[9] determined obstacles around strawberries with 3D point cloud information and the surrounding environment to calculate the harvesting path. Kim et al.[10] obtained tomato maturity and 6D pose estimation through Deep-ToMaToS model combined with transformation loss. Li et al.[11] improved obscured grape bunches recognition accuracy via improved Yolov4. Li et al.[12] identified the maturity of tomatoes via modified YOLOv5s, then carried out distortion removal and the detected region of interest (ROI) clipping on the obtained images to the position. Zhang et al.[13] conducted tomato bunch and occlusion detection by using the Yolov5 model and identified the maturity and pose of tomatoes by using the improved model.

In addition to detecting target fruits, the key of fruit harvesting is to accurately determine the location of the harvesting point. People have also carried out further research to solve the harvesting point location problem. For example, Yoshida et al.[14] obtained the cherry tomato peduncle point cloud, and after voxel filtering and clustering generation, used the Region Growing method to create a directed acyclic graph, comparing two image processing methods to find the longest path, then selected the one with a small Mahalanobis distance as the appropriate harvesting point on the peduncle. Qi et al.[15] detected the main stem of litchi using modified YOLOv5, extracted the ROI from the main stem, segmented the ROI by PSPNet model, and combined it with traditional image processing methods to obtain harvesting points on the original image. Rong et al.[16] segmented the fruits, calyxes, and pedicles in tomato images via modified Swin Transformer V2, and integrated multiple image processing algorithms to generate harvesting points located at tomato pedicles in steps.

The aforementioned research methods have achieved significant progress in harvesting point localization but also exhibit certain limitations. These approaches typically divide fruit detection

Received date: 2025-02-04 **Accepted date:** 2025-07-09
Biographies: Xinyi Ai, ME, research interest: target recognition and localization algorithm, Email: axy@cau.edu.cn; **Xiajun Zheng**, ME, research interest: target recognition and localization algorithm, Email: zhengxiajun2000@163.com; **Ziming Xiong**, ME, research interest: deep learning algorithm, Email: 15377218508@163.com; **Jiace Yuan**, ME, research interest: deep learning algorithm, Email: thefronty@163.com.
***Corresponding author: Tianxue Zhang**, Professor, research interest: intelligent agricultural machinery equipment. School of Mechanical Engineering and Automation, Beihang University, Beijing 100191, China. Tel: +86-13161680797, Email: tianxuezhang@buaa.edu.cn; **Ting Yuan**, Professor, research interest: intelligent agricultural machinery equipment. College of Engineering, China Agricultural University, Beijing 100083, China. Tel: +86-13810592356, Email: yuanting122@cau.edu.cn.

and harvesting point extraction into multiple steps, with the determination of harvesting points relying on additional image processing operations. In some existing systems, fruit detection, maturity classification, and harvesting point extraction are executed sequentially using different models or separate processing pipelines. This modular design leads to increased computational load, slower response times, and potential inconsistencies between intermediate outputs. In particular, cumulative errors may propagate from the fruit detection stage to the final harvesting point localization, significantly reducing overall accuracy. This stepwise processing not only increases computation time but also introduces potential errors and cumulative inaccuracies, thereby affecting the overall efficiency and precision of the system. To address these limitations, there is a critical need for an integrated framework that unifies multiple perception tasks in a single-stage model. Such integration minimizes latency and error accumulation while improving robustness, which is crucial for real-time deployment in greenhouse environments. Therefore, to accelerate detection speed while enhancing the accuracy of harvesting point localization, it is particularly necessary to develop an efficient model that directly integrates fruit detection and harvesting point extraction. Fu et al.[17] detected banana bundle and stalk as two targets via YOLOv4 network. The detection accuracy for banana bundles and stalks reached 99.55% and 87.82%, respectively, with an average processing time of 44.96 ms. Because of the obvious characteristics of the banana stalk, the center of the recognized peduncle image can be used as the harvesting point. However, it does not apply to the extraction of fruit harvesting points with curving peduncles. Zhu et al.[18] identified grape bunches and fruit peduncles by Yolov5s-CFD model. The mAP50 of grape bunches and fruit peduncles were 96.8% and 72.9% respectively, and the average detection time was 28.9 ms. The bending condition was determined according to the ratio of the length to the width of the recognition fruit peduncle image. When the grape peduncle was upright without bending, the midpoint of its image was designated as the picking point. In cases where the peduncle was bent, the harvesting point was determined as the location with the highest gray value directly above the image center. However, this method of point extraction depends on the gray value and geometric shape of the image. When the environmental light changes (such as shadow, reflection, etc.), the gray value will change, consequently impacting the precision of the harvesting point. Chen et al.[19] introduced the enhanced YOLOv8-GP model for concurrently detecting grapes with their harvesting points. Harvesting points were marked exactly at the center of the peduncle, but in cases where peduncles were curved, obscured, or growing in clusters, it was difficult to determine the 'true center' of the peduncle, resulting in inaccurate marking and thus affecting the quality of the model training.

Summarizing prior studies on fruit harvesting information, certain issues still require resolution: Firstly, the majority of visual inspection methods focus on only one of the tasks of fruit detection, occlusion, bunch maturity, or harvesting point detection. Secondly, determination of harvesting points location is a complicated process using a multi-step detection method. Finally, although general fruits are well detected due to their obvious characteristics, the accuracy of picking point location needs to be further improved.

Cherry tomatoes in greenhouses are predominantly harvested in bunches. This paper proposes a method leveraging keypoint detection technology, where the end-effector moves linearly from a keypoint at the base of the tomato bunch to the cutting point for harvesting. This approach aims to minimize picking damage. To efficiently accomplish the task of tomato bunch harvesting, this paper presents an efficient YOLOv8np-RCW model that performs both object detection and keypoint detection, enabling the recognition of tomato bunches, fruit bunch maturity, and picking points. Keypoint information is extracted for the tomato bunches detected as ripe, successfully obtaining the fruit bunch's pose. The motion path of the end effector is then planned to facilitate effective harvesting.

2 Methodologies

2.1 Data acquisition and processing

Cherry tomato images were captured in the greenhouse of Hongfu Industrial Park located in Daxing, Beijing. Cherry tomato was planted and managed following Dutch standards, with plants evenly spaced and pruned regularly. The Realsense D435i camera was bracketed on the robotic arm end, placed at a horizontal distance of 500-600 mm from the tomatoes and a vertical distance of 1200-1500 mm from the ground, shown in Figure 1. Data collection was carried out at three distinct time periods to ensure variability in lighting conditions: Morning (08:00-10:00): Natural light entering the greenhouse with mild shadowing; Noon (12:00-13:00): Strong sunlight with potential highlights and reflections on fruit surfaces; Afternoon (15:00-17:00): Soft indirect light with more stable illumination and fewer shadows. Throughout all sessions, the ambient lighting relied primarily on natural sunlight filtered through the greenhouse roof, with no artificial lighting used. This setup simulated real-world conditions for robot operation in agricultural environments.

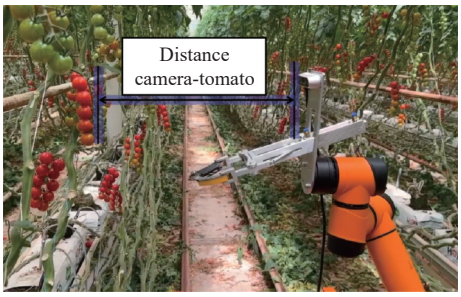


Figure 1 Image data acquisition scene

Images are labeled using Labelme, and the process of labeling the dataset is carried out according to the following principles, as shown in Figure 2: 1) Bounding Boxes: Each box enclosed a tomato bunch and included a small segment of the main stem that is physically connected to the fruit peduncle. This inclusion ensures sufficient visual cues for harvesting point detection; 2) Maturity Classification: Tomato bunches are classified into two categories—ripe and unripe. A bunch is labeled as ripe if it contains no more than two green fruits, based on visual inspection and practical picking thresholds used in greenhouse farming. If the number of green fruits exceeds two, the bunch is labeled unripe. This threshold is widely adopted by greenhouse workers to determine harvesting readiness; 3) Two keypoints are defined: point *a*, precisely positioned at the junction between fruit peduncles and main stems, and point *b*, located at the centroid of the bottommost fruit in the bunch, which serves as a reference for determining the vertical orientation and motion planning. Label format is Equation (1):

$$Label_{\text{Ripe-point}}(class,\ x_{gt},\ y_{gt},\ w_{gt},\ h_{gt},\ x_{pa},\ y_{pa},\ visible_{pa},\ldots,\ x_{pb},\ y_{pb},\ visible_{pb}) \tag{1}$$

In Equation (1), class means the maturity classification of tomato bunches; (x_{gt},y_{gt}) means bounding box center coordinates;

w_{gt},h_{gt} means the width and height of the bounding box; (x_{pa},y_{pa}), (x_{pb},y_{pb}) means the coordinates of points *a* and *b*, respectively; $visible_{pa}$, $visible_{pb}$ represents the visibility state of points *a* and *b*, respectively.

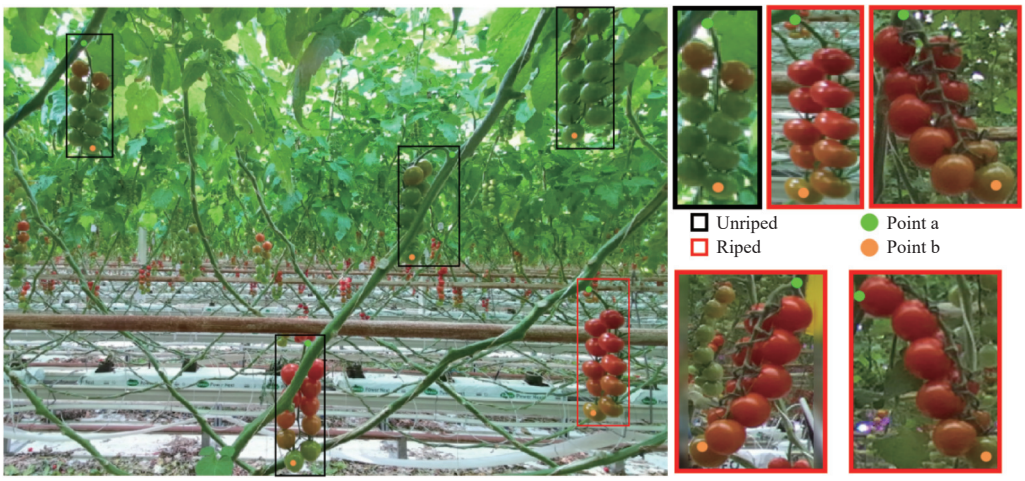


Figure 2 Image annotation interface

The dataset comprises 756 labeled images which are split into the training set (604 images), validation set (76 images), and test set (76 images) at a ratio of 8:1:1. To enhance dataset diversity and improve the model's generalization capability, data augmentation techniques are applied to the training process. Details of these augmentation methods can be found in Table 1.

2.2 YOLOv8 network structure

Compared with the algorithm based on region proposal, the one-stage algorithm model is more compatible with real-time detection in complex environments[20,21]. The YOLO (you only look once) algorithm is noted for the effective one-stage detection method[22–24]. YOLOv8 model is composed of five variables, which share a consensus network structure across models, as shown in Figure 3. Decreasing the depth and width of the network can shorten training and detection times. YOLOv8n network is selected to satisfy the real-time detection requirements for tomatoes. An additional branch head is added at the head layer for keypoint detection, as shown in Figure 4.

Table 1 Data augmentation parameter settings

Dataset	Scale	Mosaic	Translation	Hue, saturation, value (HSV)
Dataset_A	0.5	1.0	0.1	hsv_h:0.015, hsv_s:0.7, hsv_v:0.4

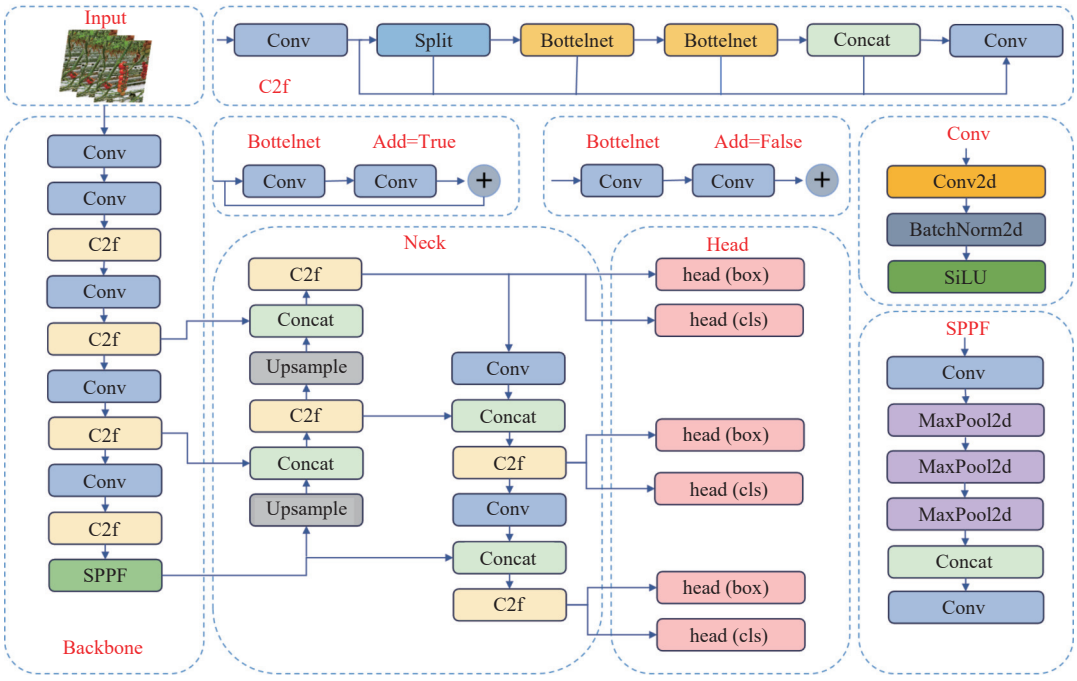


Figure 3 YOLOv8 model structure

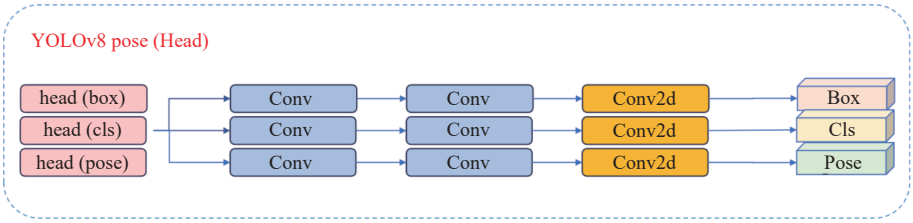


Figure 4 YOLOv8 pose head model structure

2.3 YOLOv8 pose improvement

YOLOv8 demonstrates remarkable overall behavior; however, harvesting tomato bunches requires high precision in keypoint detection. Therefore, the RepVGG is introduced in the Backbone, the CARAFE is used in the Neck, and the IoU metrics are replaced with WIoU. The improved network framework is shown in Figure 5. RepVGG is a straightforward and effective CNN architecture, consisting only of conv and ReLU. Additionally, the model exhibits a multi-branching structure during training stages, while maintaining simplicity during inference stages by re-parameterizing into a single-branch structure, as shown in Figure 6. CARAFE is a method for feature fusion, which can effectively utilize the feature information after upsampling and fuse with the original feature. Consequently, the expression capacity and transmission efficacy of features can be enhanced. WIoU is a dynamic non-monotonic focusing mechanism that presents an intelligent gradient gain assignment method, thus diminishing competitive advantages conferred by high-quality bounding boxes, allowing WIoU to focus on average-quality bounding boxes and improving the detector's overall performance.

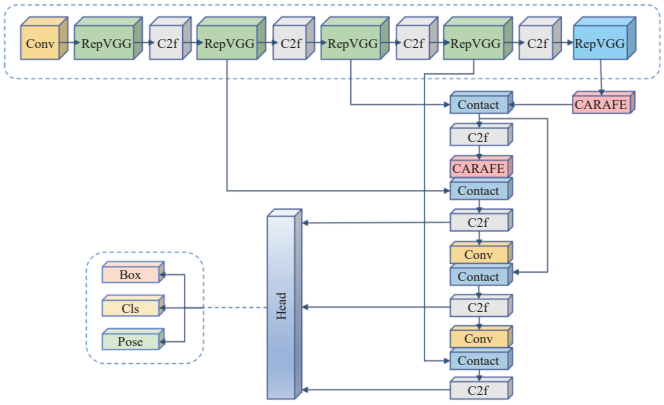


Figure 5 Improved YOLOv8 pose network structure

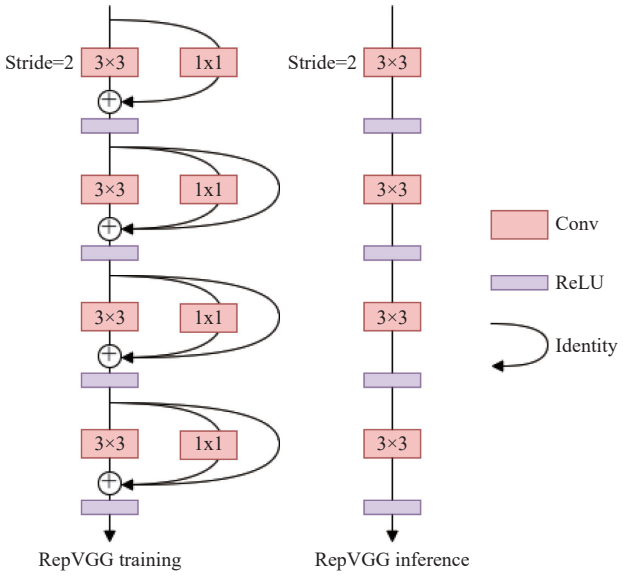


Figure 6 RepVGG frame

2.3.1 RepVGG

Many complex convolutional neural networks (CNN) achieve higher accuracy than simpler ones; however, their disadvantages are outstanding. These increase the difficulty of implementation and customization, and bring slow inference, low memory utilization, high memory access costs, and limited compatibility between various devices. RepVGG brings an ideal balance between accuracy and speed compared to existing technologies[25]. Inspired by ResNet[26], the architecture employs both identity and 1×1 branches to explicitly construct a shortcut branch. To ensure most members remain shallow or simple, ResNet identity and a 1×1 branch are adopted. By stacking multiple blocks, a model suitable for training is constructed.

The transformation between the multi-branch structure during training and the plain structure in inference is realized through architectural re-parameterization. For instance, an identity branch can be regarded as a reduced form of a 1×1 conv, which can be further simplified into a reduced form of a 3×3 conv. This enables the construction of a unified 3×3 kernel by merging the parameters of the initial 3×3 kernel, identity branch, 1×1 branch, and Batch Normalization (BN)[27] layer. Specifically, the network architecture is integrated with a corresponding group of parameters during training. At inference time, the trained blocks are converted into a unified 3×3 conv, thus obtaining a 3×3 kernel, two 1×1 kernels, and three bias vectors. The ultimate bias is computed by summing the three bias vectors, while the eventual 3×3 kernel is derived by incorporating the 1×1 kernel to the center of the 3×3 kernel. This is trivially achieved via zero-padding the 1×1 kernel to 3×3 and then summing all three kernels, as illustrated in Figure 7. Consequently, the RepVGG model at inference time consists of one type of operator: ReLU following conv. This design enables achievement of high operating speed on generic computing devices such as GPUs for efficient testing and deployment.

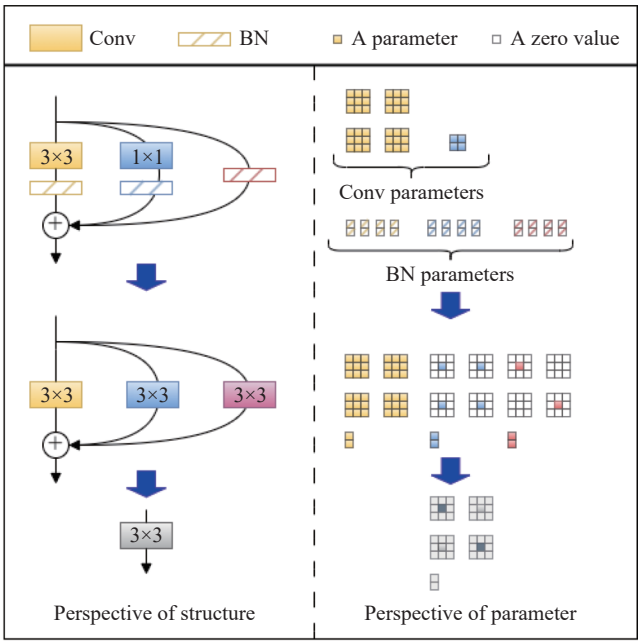


Figure 7 RepVGG module of the parametric structure

2.3.2 CARAFE

Feature upsampling is a crucial structure in many network architectures, with the composition playing a pivotal role in prediction performance[28]. In YOLOv8, an upsampling method utilized is nearest-neighbor interpolation, which replicates the nearest pixel values, resulting in an upsampling image that lacks smooth gradient transitions, which in turn results in missing image detail. During the keypoint detection task, insufficient precision of details caused by nearest-neighbor interpolation can adversely impact detection performance, thus the efficient CARAFE structure is introduced to enhance the upsampling algorithm in YOLOv8.

CARAFE offers a wide receptive field and dynamically produces fitness kernels in response to incoming information rather than importing significant computational overhead or additional parameters, making it lightweight while effectively leveraging surrounding information[28]. CARAFE comprises two components: A kernel prediction and a content-aware reassembly module, functioning through upsampling kernel prediction and feature reassembly. The process involves two main stages: First, a reassembly kernel is generated based on the content at each target location. Second, the features are recombined with the predicted ones. CARAFE will generate a feature map χ' based on χ with dimensions $C\times H\times W$, using an upsample ratio σ, as shown in Figure 8.

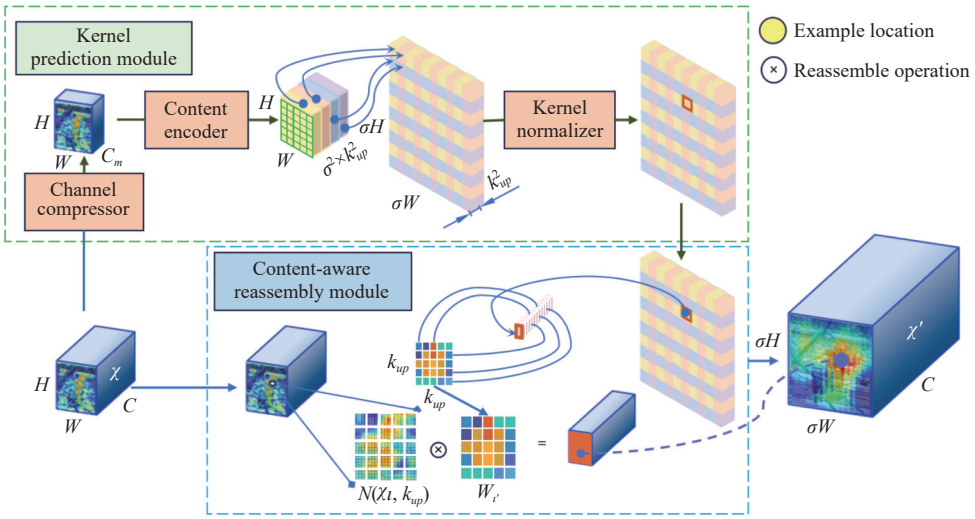


Figure 8 CARAFE's overall framework

In the first stage, feature channels are compressed by the kernel prediction module via the conv layer to decrease parameters and improve feature extraction efficiency. Compressed images of dimensions $C_m\times H\times W$ are encoded employing a conv layer with encoder kernel, output a reassembled kernel of dimensions $\sigma^2 k_{up}^2 \times H\times W$, then the generated $k_{up} \times k_{up}$ kernel is normalized with softmax.

In the second stage, for reassembled kernel $W_{l'}(k_{up}\times k_{up})$, the content-aware reassembly module reorganizes features over the selective area. Under the action of a reassembled kernel, each point in the $N(\chi_l, k_{up})$ area has a stronger contribution to the upsampled area on account of the content of the feature. More attention is paid to pertinent points in the selective area, improving the semantics of the reassembled feature map.

2.3.3 WIoU

Loss calculation in YOLOv8 pose comprises classification, bounding boxes, and keypoints for the decoupled-head structure. Bounding box loss influences keypoint predictions because the target region is delineated by bounding boxes, within which precise regression of keypoints is performed. Therefore, if the bounding box prediction is inaccurate, the model may search for keypoints in the incorrect region, resulting in inaccurate keypoint predictions. While the bounding box and keypoint loss are computed separately, they are jointly optimized. If the weight of the bounding box loss is excessively high, it will suppress keypoint regression. Bounding box loss in YOLOv8npose is computed via CIoU. However, CIoU does not capture practical variations between width and height, for training images unavoidably contain low-quality data. Variables of distance and length-width ratio aggravate penalties on low-quality data, diminishing the model's ability to generalize. To minimize the above impairments, WIoU is drawn to optimize the network[29].

The prediction process of CIoU and WIoU loss between different iterations is shown in Figure 9. The orange area is the ground truth, and the blue area is the initial detection area. Areas of green and red are the forecasting processes of CIoU and WIoU, respectively. CIoU's width and height cannot be simultaneously increased or decreased, whereas WIoU focuses on the differences in width and height, as shown in Figure 9. During the early iterations, WIoU achieves faster shape matching of the bounding area. With further iterations, WIoU effectively reduces the area of non-overlap between the predicted and the ground truth while optimizing the shape of the borders, allowing the predicted area to converge toward the ground truth more rapidly than CIoU. In later stages, WIoU demonstrates superior performance. Compared to CIoU, the bounding area predictions under WIoU align almost perfectly with the ground truth, achieving significantly higher prediction accuracy.

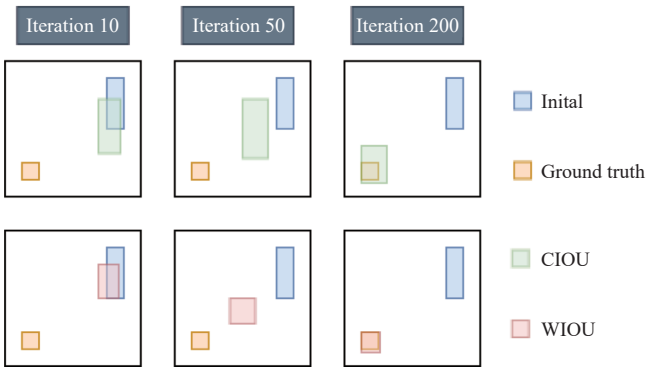


Figure 9 Forecasting process for CIoU and WIoU

WIoU regression loss is shown in Figure 10. The green box is the prediction box, whose center point is (x,y); width and height are w and h, respectively. The yellow box is the ground truth, whose center point is (x_{gt},y_{gt}); width and height are w_{gt} and h_{gt}, respectively. The red box is the minimum vertical rectangular box of the predicted and ground truth; the width and height are W_g and H_g, respectively. The blue box is the overlap between the predicted and ground truth; width and height are W_i and H_i, respectively. WIoU losses are calculated as follows in Equations (2-4):

$$\mathcal{L}_{IoU} = 1 - IoU = 1 - \frac{W_i H_i}{wh + w_{gt}h_{gt} - W_i H_i} \quad (2)$$

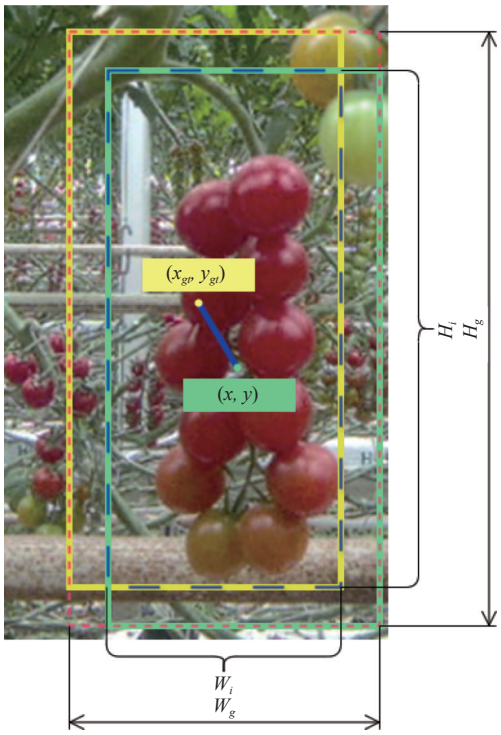


Figure 10 WIoU loss for bounding box regression

$$R_{\mathrm{WIoU}} = \exp\left(\frac{(x-x_{\mathrm{gt}})^2+(y-y_{\mathrm{gt}})^2}{(W_g^2+H_g^2)^*}\right) \tag{3}$$

$$\mathcal{L}_{\mathrm{WIoU}} = rR_{\mathrm{WIoU}}\mathcal{L}_{\mathrm{IoU}}, \quad r=\frac{\beta}{\delta\alpha^{\beta}-\delta}, \quad \beta=\frac{\mathcal{L}_{\mathrm{IoU}}^*}{\mathcal{L}_{\mathrm{IoU}}} \tag{4}$$

In the Equation (4), $\mathcal{L}_{\mathrm{IoU}}$ represents the intersection over union (IoU) loss; $\mathcal{L}_{\mathrm{WIoU}}$ represents WIoU loss; β represents the outlier factor; $r$represents the gradient gain; α and δ are hyper-parameters controlling the mapping between β and r; the superscripted * indicates detachment from the computation graph.

2.4 Tomato bunch pose acquisition

The improved keypoint algorithm enables precise acquisition of tomato bunch information, providing technical support for efficient harvesting. When extracting and identifying the keypoint information of mature tomato bunches, as mentioned earlier, the keypoints include two points, namely point *a* and point *b*, where the line connecting these two points represents the growth orientation of the tomato bunch. This information can guide the motion path of the end-effector and optimize the harvesting operation. However, due to the natural growth of tomato bunches, their orientations vary, and there are different degrees of inclination, as shown in Figure 11a. If these inclination angles are not considered, it may lead to unsuccessful grasping or fruit damage caused by angular deviations. Therefore, the end-effector must adjust in real time according to the inclination of the tomato bunch.

To determine the inclination angle of the tomato bunch, the angle between the line connecting the two keypoints and the *Y*-axis of the 2D image pixel coordinate system is calculated. Using the 2D pixel coordinates of the keypoints, the vector between the two points is computed, and the inclination angle of the tomato bunch is obtained using the vector angle formula, as illustrated in the corresponding Equation (5). This angle not only provides crucial parameters for adjusting the end-effector but also helps predict potential risks during the harvesting process.

a. Original image b. Posec alculation c. Pose acquisition

Figure 11 Tomato bunch pose calculation

In addition to the inclination angle, the direction of the angle must also be determined to ensure that the end-effector can be adjusted to the correct working orientation, as shown in Figure 11b. In a 2D plane, the relative direction can be calculated using the cross-product of two vectors, thereby determining the direction of the adjustment angle, as illustrated in the corresponding Equation (6). The cross-product formula is simple and efficient, making it suitable for real-time applications in automated control systems.

$$\mathrm{angle} = \cos^{-1}\left(\frac{\vec{v}_{\mathrm{ab}}\cdot\vec{v}_Y}{|\vec{v}_{\mathrm{ab}}||\vec{v}_Y|}\right) \tag{5}$$

$$\mathrm{cross\ product} = x_{\mathrm{ab}}y_Y - y_{\mathrm{ab}}x_Y \tag{6}$$

where, $\vec{v}_{\mathrm{ab}}$ is a vector from point *a* to point *b*, $\vec{v}_Y$ is a unit vector parallel to the *Y*-axis, $\vec{v}_{\mathrm{ab}}=(x_{\mathrm{ab}},y_{\mathrm{ab}})$, $\vec{v}_Y=(x_Y,y_Y)$. If the cross product>0, it indicates that $\vec{v}_Y$ is in the counterclockwise direction relative to $\vec{v}_{\mathrm{ab}}$. The end-effector needs to rotate counterclockwise by the corresponding angle.

3 Experiments and results

3.1 Experimental details

Model training hardware configuration: Windows 10 Pro OS, 16 GB RAM, 11th Gen Intel(R) Core(TM) i5-11400@2.60 GHz NVIDIA GeForce RTX 3060. The coding language is Python 3.7.16, and the open-source learning framework is PyTorch 1.11.0. In addition, CUDA 11.3 is updated to boost the system's computing performance with 16 images trained in each epoch. AdamW optimizer is utilized, setting initial and termination learning rates to 0.01. The weight decay coefficient is 0.0005, and the momentum parameter reaches 0.937. The training process is configured for 300 epochs. However, to avoid overfitting and reduce unnecessary computation, an early stopping strategy is applied based on the validation loss. If the validation loss does not improve for 20 consecutive epochs, the training is stopped automatically. The model weights corresponding to the lowest validation loss are saved as the final model for evaluation.

3.2 Evaluation indicators

To assess the precision of tomatoes multi-task assay, Precision (P), Recall (R), Average Precision (AP), and mean Average Precision (mAP) are used. mAP indicator is used to evaluate the model performance. Parameters and giga floating-point operations per second (Gflops) are used to assess the complexity of models. The time indicator is measured by the predicted time of each image. The definition is shown in Equations (7-10):

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{7}$$

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{8}$$

$$\mathrm{AP} = \int_0^1 P(R)\mathrm{d}R \tag{9}$$

$$\mathrm{mAP} = \frac{1}{2}\sum_{i=1}^{2} AP_i \tag{10}$$

where, $P(R)$ represents the P corresponding to R.

mAP of keypoint detection is calculated based on object keypoint similarity (OKS), which computes the matching degree between predicted and ground truth. The definition is shown in Equation (11):

$$\mathrm{OKS} = \frac{\sum_i \exp\left(-\frac{d_i^2}{2s^2k_i^2}\right).1}{\sum_i 1} \tag{11}$$

where, d_i is the Euclidean distance between the predicted keypoint and ground truth; s represents the scale factor; k_i represents the weight factor of the keypoint. When OKS exceeds a certain threshold, the prediction is considered correct. Then AP and mAP are calculated by the formula of target detection.

3.3 Ablation experiment

To comprehensively evaluate the performance contributions of each module in the proposed model, this paper conducts systematic ablation experiments from the following three aspects: (1) Comparison of the backbone network structure, (2) improvement of the upsampling module, and (3) replacement of the bounding box regression loss function.

Firstly, to verify the effectiveness of RepVGG as the backbone network, under the same training settings, comparative experiments are conducted with various mainstream lightweight network structures (including MobileNetV2, EfficientNetV2, and ShuffleNetV2). The experimental results are listed in Table 2 as follows. Secondly, in order to evaluate the influence of different regression loss functions on the model performance, CIoU is replaced with WIoU, DIoU, SIoU, and ShapeIoU respectively for comparative experiments, as listed in Table 3. Finally, under the YOLOv8n-pose framework, this paper conducts combination experiments on each enhancement module (RepVGG module R, CARAFE upsampling module C, and WIoU loss function W), and the corresponding relationship of algorithm combinations is listed in Table 4. Finally, the model formed by integrating the three improvements is named YOLOv8np-RCW. The performance comparison results of each combined model are listed in Table 5.

The RepVGG backbone significantly improved detection performance while maintaining a relatively small model size of only 3.08 million parameters, as listed in Table 2. Compared to the original CSPDarkNet backbone, the precision, recall, and mAP50 of the bounding box detection increased by 1.3%, 5.8%, and 2.6%, respectively, while the precision, recall, and mAP50 of keypoint detection improved by 1.7%, 4.7%, and 3.2%. In addition, the inference time was reduced from 9.5 ms to 8.2 ms, representing a 13.7% improvement in real-time performance. Although EfficientNetV2 achieved higher detection accuracy, with mAP50 values of 89.5% for bounding boxes and 90.4% for keypoints, its model complexity is significantly higher, with 21.8 million parameters, 55.3 Gflops, and an inference time of 28.2 ms. Such computational demands are unsuitable for deployment on resource-constrained agricultural robotic platforms. In contrast, RepVGG provides a more favorable trade-off among detection accuracy, inference speed, and model complexity, making it the optimal backbone choice for the proposed model.

Table 2 Comparison of lightweight backbone networks

Name	Box			Pose			Parameters/ M	Gflops	Time/ ms
	P/ %	*R*/ %	mAP50/ %	*P*/ %	*R*/ %	mAP50/ %			
CSPDarkNet (Initial)	80.3	73.5	81.1	81.3	75.3	81.3	3.08	8.3	9.5
MobileNetV2	86.3	78.4	85.8	85.1	77.2	85.1	3.83	10.3	10.2
EfficientNetV2	87.4	84.4	89.5	88.8	85.3	90.4	21.8	55.3	28.2
ShuffleNetV2	78.3	85.2	84.4	78.7	85.8	83.9	2.86	7.7	9.9
RepVGG	81.6	79.3	83.7	83.0	80.0	84.5	3.08	8.3	8.2

Table 3 Comparison of regression loss functions

Name	Box			Pose			Parameters/ M	Gflops	Time/ ms
	P/ %	*R*/ %	mAP50/ %	*P*/ %	*R*/ %	mAP50/ %			
CIoU (Initial)	80.3	73.5	81.1	81.3	75.3	81.3	3.08	8.3	9.5
DIoU	72.1	85.1	84.9	72.9	86.2	85.2	3.08	8.3	6.2
SIoU	81.0	75.3	83.6	82.0	76.2	83.3	3.08	8.3	6.5
WIoU	89.0	78.0	86.1	87.6	76.6	85.1	3.08	8.3	9.1
ShapeIoU	86.5	81.5	86.3	85.3	80.3	85.2	3.08	8.3	6.1

Table 4 Algorithm improvement and corresponding name

Initial	Improvement	Name
YOLOv8n pose	RepVGG	YOLOv8np-R
YOLOv8n pose	CARAFE	YOLOv8np-C
YOLOv8n pose	WIoU	YOLOv8np-W
YOLOv8n pose	RepVGG+CARAFE+WIoU	YOLOv8np-RCW

Table 5 Comparison of algorithmic indicators

Name	Box			Pose			Parameters/ M	Gflops	Time/ ms
	P/ %	*R*/ %	mAP50/ %	*P*/ %	*R*/ %	mAP50/ %			
YOLOv8n pose	80.3	73.5	81.1	81.3	75.3	81.3	3.08	8.3	9.5
YOLOv8np-R	81.6	79.3	83.7	83.0	80.0	84.5	3.08	8.3	8.2
YOLOv8np-C	83.8	85.3	86.4	82.9	86.2	86.3	3.22	8.6	11.3
YOLOv8np-W	89.0	78.0	86.1	87.6	76.6	85.1	3.08	8.3	9.1
YOLOv8np-RCW	84.1	86.3	87.3	83.6	85.9	86.8	3.22	8.6	9.8

As listed in Table 3, the WIoU loss function exhibited overall superiority in both bounding box and keypoint detection tasks, with mAP50 values reaching 86.1% and 85.1%, respectively, significantly outperforming CIoU, DIoU, and SIoU. Compared to CIoU, WIoU not only provided notable accuracy gains but also reduced inference time from 9.5 ms to 9.1 ms, without any increase in model parameters or computational cost, thus achieving a

balanced improvement in efficiency and precision. While ShapeIoU achieved relatively high recall, with *R* values of 81.5% for bounding boxes and 80.3% for keypoints, its overall detection accuracy remained lower than that of WIoU. Therefore, WIoU is better suited for complex scenarios involving occlusion and variation in tomato bunch shapes, enhancing both model robustness and localization accuracy.

As demonstrated in Table 5, the YOLOv8np-R model, which incorporates only the RepVGG backbone, improved the detection precision for both bounding boxes and keypoints while reducing the inference time to 8.2 ms, making it well-suited for applications requiring real-time performance. The YOLOv8np-C model, which integrates the CARAFE upsampling module, achieved the highest recall rates—85.3% for bounding boxes and 86.2% for keypoints—by enhancing local feature extraction, making it more effective in greenhouse environments with significant occlusion. The YOLOv8np-W model, which uses the WIoU loss function, achieved steady gains in detection accuracy without increasing the model's complexity, demonstrating superior bounding box regression performance. The final integrated model, YOLOv8np-RCW, which combines RepVGG, CARAFE, and WIoU, achieved the best overall performance. Compared to the baseline, it improved the precision, recall, and mAP50 of bounding box detection by 3.8%, 12.8%, and 6.2%, respectively, and enhanced keypoint detection by 2.3%, 10.6%, and 5.5%. These results demonstrate the effectiveness of the combined optimization strategy in improving both detection accuracy and inference efficiency.

The maturity confusion matrix from the ablation experiments is shown in Figure 12. The accuracy of YOLOv8np-R, YOLOv8np-C, YOLOv8np-W, and YOLOv8np-RCW models in recognizing maturity has improved compared to YOLOv8n pose, with YOLOv8np-RCW having the fittest maturity category recognition.

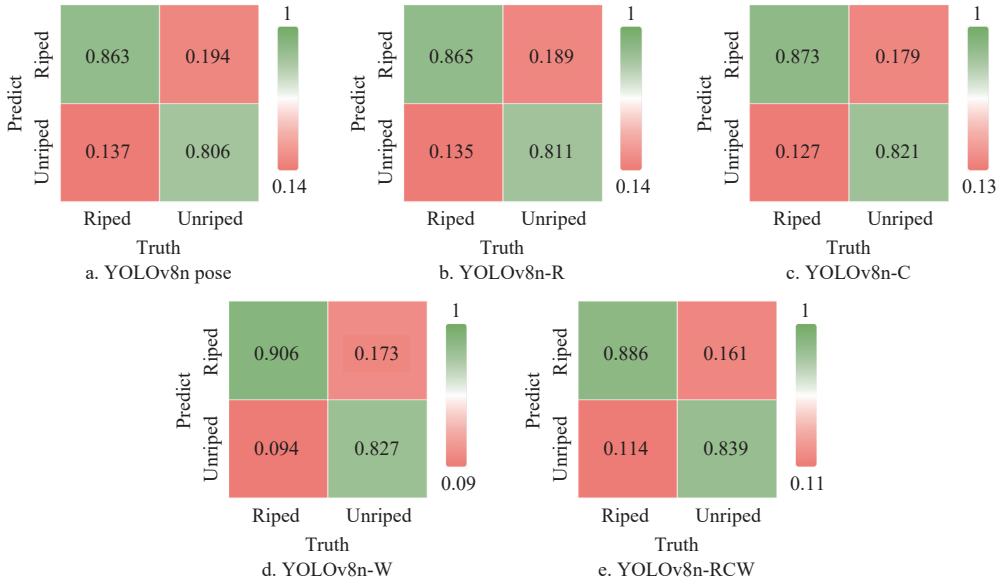


Figure 12 Maturity confusion matrices

The bounding box loss curve from the ablation experiments is shown in Figure 13. The YOLOv8np-W and YOLOv8np-RCW converge faster due to the addition of WIoU, which speeds up the convergence of the original model, and the loss of the YOLOv8np-RCW model finally converges to 0.482 15, which is 55.76% lower than the YOLOv8n pose model. It effectively decreases the loss value and improves the detection performance.

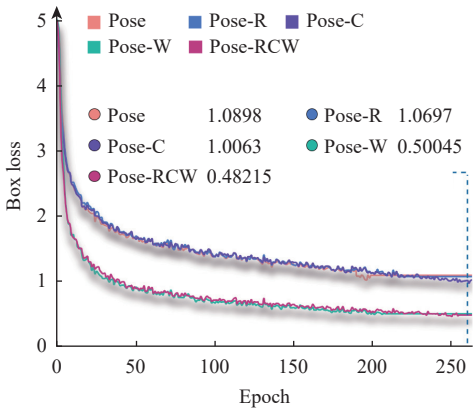


Figure 13 Box loss curve in ablation experiment

3.4 Comparison of detection performance with other models

To prove YOLOv8np-RCW model's comprehensive performance, compared with other algorithms fusing target detection and keypoint detection, the outcomes are presented in Figure 14. Since the parameters, calculation volume, and detection time of multi-model fusion are higher than those of a single model, the box mAP50 and pose mAP50 of the models are evaluated for comparison. The box mAP50 and pose mAP50 of RTMDet-RTMPose are 2.9% and 2.8% higher than those of YOLOv8n pose. The box mAP50 and pose mAP50 of Fasterrcnn-RTMPose are 3.3% and 2.8% higher than those of YOLOv8n pose, but their detection time is longer than that of YOLOv8n pose. The detection time of YOLOv8np-RCW and YOLOv8n pose model is comparable, and the box mAP50 and pose mAP50 values are the highest; thus, comprehensively considered, YOLOv8np-RCW has the best performance.

To compare the maturity and keypoints classification detection effects of different algorithms, experiments are conducted on the test dataset by calculating the number of identified ripe tomato bunches, unripe tomato bunches, point *a* and point *b*. When the confidence score threshold is 0.5, the result is shown in Figure 15. YOLOv8np-RCW has the largest total number of detections, among which the number of maturity and keypoint detections also exceeds other models. To compare the performance of YOLOv8np-RCW and other models in extracting two keypoints in detail, the difference in pixel distance between the predicted point and ground

truth on the 2D image is taken as an assessment metric. The average error is calculated for the coordinates of point *a* and point *b* detected by each model. Table 4 shows the average distance error statistics between predicted keypoints of the different models and ground truth in *X*, *Y*, and Euclidean directions. The distribution of point *a* and point *b* in the *X*-axis and *Y*-axis is presented in Figure 16.

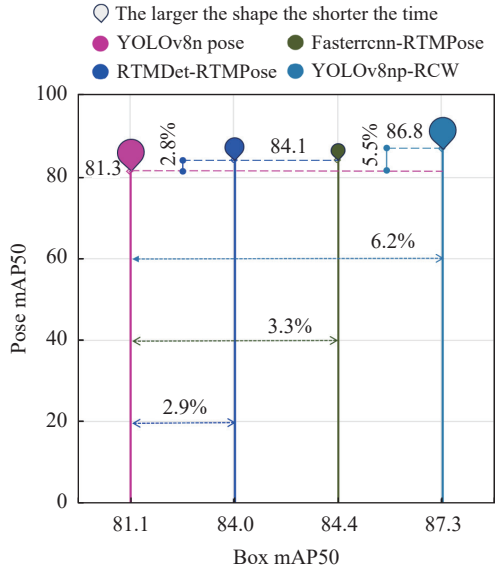
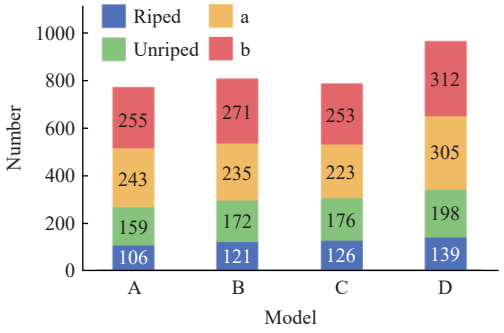


Figure 14 Performance comparison of different models

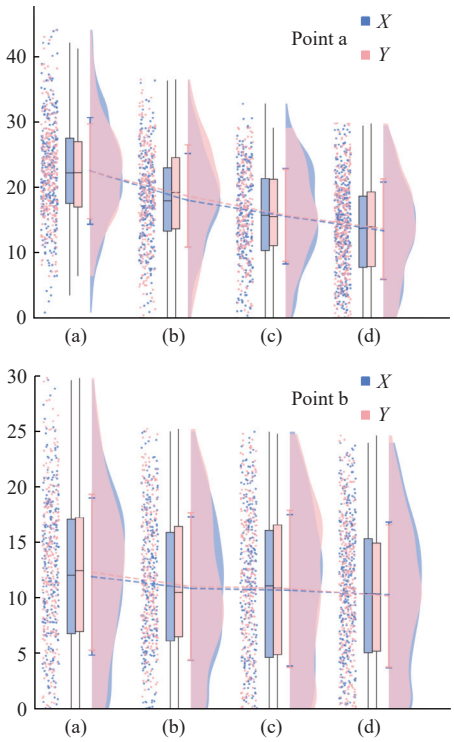


Note: A. YOLOv8n pose B. RTMDet-RTMPose C. Fasterrcnn-RTMPose D. YOLOv8np-RCW

Figure 15 Comparison of outcomes from different models

Table 6 shows that the accuracy of YOLOv8np-RCW in recognizing the points exceeds that of the other models. The average Euclidean distance errors for the two keypoints are 19.638 pixels and 14.708 pixels, with standard deviations of 9.578 pixels and 8.728 pixels, respectively. This level of precision in locating the keypoints makes it suitable for use in tomato bunch picking. As presented in Figure 16, the YOLOv8np-RCW model's detection error in the *X*-axis and *Y*-axis is more concentrated near the average value, and the maximum distance error is less than 50 pixels, with a small error fluctuation range, which is better than the detection results of other models. In summary, YOLOv8np-RCW has superior performance and accurate prediction, making it sufficient to fulfill the real-time detection needs of tomato harvesting robots.

The visualization results of different models are shown in Figures 17 and 18. The bounding box and maturity detection visualization results of tomato bunches in complex scenarios are illustrated in Figure 17. In the images detected by the YOLOv8n pose model, missed detections of tomato bunches and imprecise bounding boxes are observed, leading to errors in maturity recognition, with maturity confidence falling below 90%, as shown in Figure 17b. The RTMDet-RTMPose accurately detects bounding boxes compared with the YOLOv8n pose; however, misclassifications occur, with the same tomato bunch being identified as both ripe and unripe, as shown in Figure 17c. The Fasterrcnn-RTMPose model exhibits high confidence in maturity recognition; however, the bounding boxes fail to fully enclose the tomato bunches in some cases, as shown in Figure 17d. Although the YOLOv8np-RCW model exhibits lower maturity confidence compared to the Fasterrcnn-RTMPose model, it performs better in detection, successfully identifying tomato bunches missed by the other three models. Additionally, the detected bounding boxes effectively enclose the tomato bunches, as shown in Figure 17e.



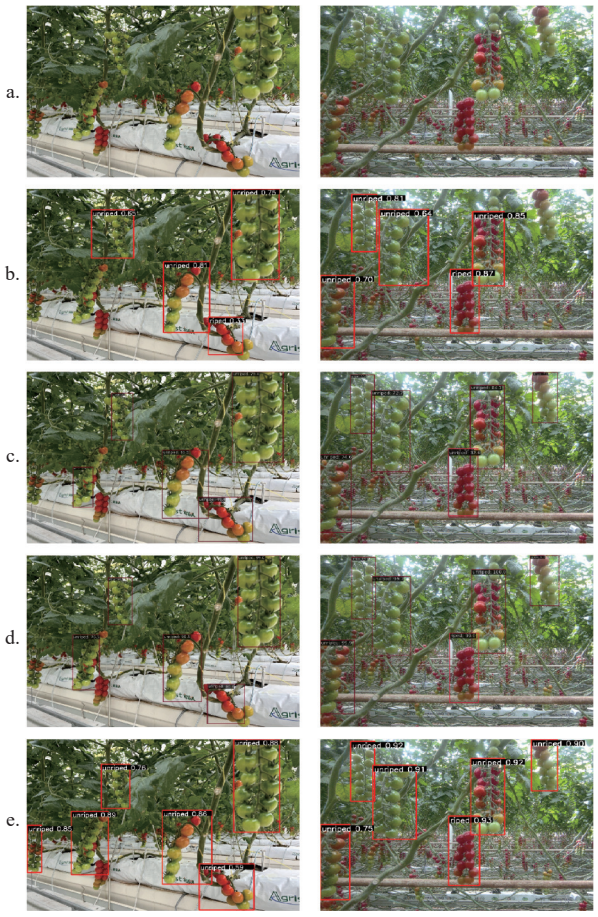
Note: (a) YOLOv8n pose (b) RTMDet-RTMPose (c) Fasterrcnn-RTMPose (d) YOLOv8np-RCW

Figure 16 Distribution of data at keypoints in different models

Table 6 Comparison of pixel errors at keypoints of different models

Keypoints	Model	Average distance (pixel)			Standard deviation (pixel)
		X	*Y*	Euclidean distance	
Point *a*	YOLOv8n pose	21.265	22.359	32.057	9.932
	RTMDet-RTMPose	18.691	20.267	26.296	9.548
	Fasterrcnn-RTMPose	15.304	16.858	22.535	9.098
	YOLOv8np-RCW	14.267	14.762	19.638	9.578
Point *b*	YOLOv8n pose	12.297	14.338	17.313	9.595
	RTMDet-RTMPose	11.895	12.581	15.655	8.896
	Fasterrcnn-RTMPose	11.032	11.826	15.479	9.347
	YOLOv8np-RCW	10.638	10.465	14.708	8.728

Keypoint detection is a critical step for the successful harvesting of tomato bunches. In greenhouse environments, the peduncles of tomato bunches are often curved rather than growing vertically, posing challenges in identifying the keypoints. The YOLOv8n pose model exhibits large prediction errors for keypoints within the inaccurately detected bounding boxes, with point *a*,



Note: a. Original image b.YOLOv8n pose c. RTMDet-RTMPose d. Fasterrcnn-RTMPose e. YOLOv8np-RCW

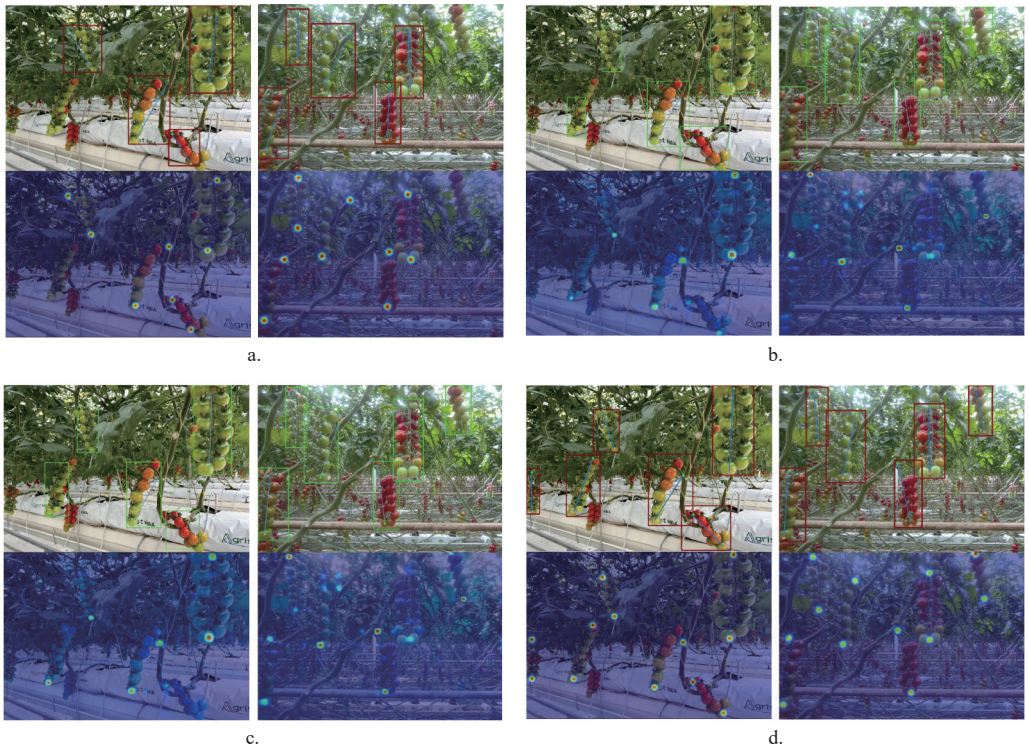
Figure 17 Visualization of tomato bunch and maturity recognition

located at the connection between the main stem and fruit peduncle, being incorrectly identified and deviating onto the main stem presented in Figure 18a. The RTMDet-RTMPose model successfully detects more keypoints than the YOLOv8n pose model, with higher accuracy in keypoint identification. However, there is still some deviation in the position of point *a* presented in Figure 18b. Compared to the RTMDet-RTMPose model, the Fasterrcnn-RTMPose model detects the tomato bunch bounding boxes in different positions, leading to different results when using the same keypoint detection model. Although fewer keypoints are identified, the results are more accurate compared to the first two models, as shown in Figure 18c. The YOLOv8np-RCW model demonstrates more accurate positioning of the two keypoints. Although there is a slight deviation at some keypoints, the number and accuracy of the detected keypoints are superior to those of the other three models, as shown in Figure 18d. From the above analysis, it can be seen that both the bounding box and keypoint recognition are more accurate, highlighting the model's strong feature extraction ability.

3.5 Field experiment

To verify whether the improved algorithm meets the efficiency and accuracy requirements of tomato harvesting robots for recognition and localization, as well as the effectiveness of the harvesting method, the algorithm is deployed on a harvesting robot for field testing. The robot is equipped with a six-degree-of-freedom industrial robotic arm, a Realsense D435i depth camera, an integrated clip-and-cut end-effector, and a mobile platform, offering high flexibility and adaptability to environmental conditions, as shown in Figure 19. The main steps of the robot's harvesting process include recognition, sleeving, and cutting, forming a complete harvesting workflow, as illustrated in Figure 20.

The specific operation process is as follows: First, the robot starts the depth camera according to the standard posture set during the dataset collection and captures image data of the current environment. The improved algorithm is then used to identify mature tomato bunches in the image and extract keypoint



Note: a. YOLOv8n pose b. RTMDet-RTMPose c. Fasterrcnn-RTMPose d. YOLOv8np-RCW

Figure 18 Visualization of tomato keypoint recognition

Figure 19 Tomato harvesting robot

information. Using the 2D pixel coordinates of the keypoints, the inclination angle and direction of the tomato bunch are calculated. Simultaneously, the depth camera retrieves the depth information of the keypoints, converting the 2D coordinates into 3D spatial coordinates, which provide precise data support for the subsequent motion path planning of the robotic arm. Next, the robot controls the robotic arm to perform the harvesting operation according to the following steps:

Initial Positioning: The robotic arm moves to point *b* based on the calculated 3D coordinates, ensuring the end-effector is close to the tomato bunch.

Angle Adjustment: The end-effector's orientation is adjusted according to the inclination angle of the tomato bunch, aligning it with the bunch's growth direction.

Precise Grasping: The end-effector moves along the line connecting the tomato bunch to point *a*, ensuring the blade is accurately aligned with the fruit stem.

Cutting Operation: The end-effector executes the cutting action, precisely cutting and securely holding the tomato bunch, completing the harvesting operation.

Three sets of experiments are conducted. The first set of experiments evaluates the tomato bunch recognition rate and the maturity recognition accuracy. A conveyor channel is randomly selected, and the maturity of tomato bunches on both sides is determined based on a manually calibrated dataset standard. The number of ripe tomato bunches, T_{riped}, and unripe tomato bunches, T_{unriped}, are recorded. Using the improved algorithm, the detected tomato clusters within the field of view are identified, and the number of ripe tomato bunches, P_{riped}, and unripe tomato bunches, P_{unriped}, are recorded. The tomato bunch recognition rate and maturity recognition accuracy are calculated using Equations (12-13). The experimental results are listed in Table 7.

a. Recognition b. Sleeving c. Cutting

Figure 20 Harvesting process

Table 7 First set of experiments

Times	T_{riped}	T_{unriped}	P_{riped}	P_{unriped}
1	22	32	19	33
2	16	28	13	30

$$R_1 = 1 - \frac{\left|(T_{\text{riped}} + T_{\text{unriped}}) - (P_{\text{riped}} + P_{\text{unriped}})\right|}{(T_{\text{riped}} + T_{\text{unriped}})} \quad (12)$$

$$R_2 = 1 - \frac{\frac{|T_{\text{riped}} - P_{\text{riped}}|}{T_{\text{riped}}} + \frac{|T_{\text{unriped}} - P_{\text{unriped}}|}{T_{\text{unriped}}}}{2} \quad (13)$$

The second set of experiments aims to evaluate the response time of the model, defined as the duration from the start of recognition to the issuance of the picking command. To conduct the experiment, five bunches of ripe tomatoes are selected, and each bunch is tested five times. The response time for each test is measured using the time.time() function, with all measurements recorded. The average response time is calculated based on the collected data. The results of this evaluation are presented in Table 8.

The third set of experiments test the ability of keypoint positioning and end-effector sleeving, and harvesting experiments on 50 bunches of ripe tomatoes are carried out. The harvesting results are analyzed through three groups of data, as shown in Table 9.

In this field experiment, the tomato bunch recognition rate is 96.9%, the maturity recognition accuracy is 89.6%, the average visual response time is 0.1499s, the harvesting success rate is 68%, and the average time to pick each bunch of tomatoes is 10.8366 s. These results indicate that the improved algorithm and harvesting method meet the performance requirements for the tomato harvesting robot.

Table 8 Second set of experiments

Number of repetitions	Response time of model/s				
	Bunch 1	Bunch 2	Bunch 3	Bunch 4	Bunch 5
1	0.1426	0.1468	0.1546	0.1393	0.1539
2	0.1495	0.1505	0.1495	0.1460	0.1466
3	0.1449	0.1475	0.1606	0.1625	0.1447
4	0.1390	0.1456	0.1575	0.1595	0.1515
5	0.1524	0.1464	0.1536	0.1458	0.1555

Table 9 Third set of experiments

Number of bunches	Keypoint location failure	Sleeving failure	Average harvesting time/s
22	4	3	11.6585
12	2	3	10.5241
16	3	1	10.3273

4 Discussion

An innovative method is proposed, integrating tomato bunch detection, maturity assessment, and keypoint detection with a single model. Compared to cascaded approaches integrating target and keypoint detection, YOLOv8np-RCW is more lightweight, more rapid, and simpler to implement on embedded devices. Besides, the inclusion of maturity evaluation enables precise picking, thereby improving both picking efficiency and fruit quality. In keypoint detection, two keypoints are extracted for each tomato bunch: point *a*, the connection between the peduncle and the main stem, and point *b*, the centroid of the lowest fruit in the bunch. The end-effector adopts a straight-line motion from the bottom keypoint of the tomato bunch to the point for cutting, minimizing potential damage during harvesting.

The improved YOLOv8np-RCW model presents significant enhancements compared with the YOLOv8n pose: *P*, *R*, and mAP50 of the bounding boxes increased by 3.8%, 12.8%, and 6.2% respectively, while keypoints' corresponding indicators increased by 2.3%, 10.6%, and 5.5% respectively. The bounding box loss decreased by 55.76%, and the maturity detection accuracy improved by 2.3%. Despite these performance improvements, the model's parameters, Gflops, and inference time only slightly increased, maintaining fast computation and detection capability. The model processes individual images quickly, with minimal pixel Euclidean distance error, facilitating real-time bunch harvesting tasks for harvesting robots.

While the proposed YOLOv8np-RCW model demonstrates high detection accuracy and efficiency under greenhouse conditions, its current training and evaluation are limited to images captured in a controlled environment. This constraint raises important considerations regarding the model's generalization to more diverse and challenging real-world settings.

To ensure broader applicability, future work will focus on evaluating the model's performance under varying illumination conditions, including intense sunlight, shadows, and artificial night-time lighting. These factors can significantly alter the appearance of tomato fruits and peduncles, thus affecting detection and keypoint localization. Moreover, the presence of more severe occlusions—caused by overlapping fruits, leaves, or support structures—remains a common challenge in real harvesting scenarios. Incorporating dynamic occlusion-aware mechanisms, such as multi-frame information fusion or temporal consistency modules, could enhance model robustness under such constraints.

Another crucial aspect concerns the adaptability of the model to different tomato cultivars, which may vary in shape, size, clustering pattern, and color distribution. To improve the model's cultivar-invariance, a more diversified training dataset will be collected, encompassing multiple growth stages, environmental backgrounds, and tomato varieties.

Furthermore, to extend the application scope beyond greenhouses, future research will explore adapting the model for open-field environments. Open-field farming introduces additional complexity due to unpredictable weather, background clutter, and less-structured plant arrangements. Domain adaptation techniques, transfer learning, and real-time image enhancement algorithms may offer promising solutions to bridge the performance gap between greenhouse and open-field applications.

In the long term, the proposed model can be integrated into autonomous harvesting and inspection platforms operating across a wider spectrum of agricultural environments. Its multi-task capabilities—combining object detection, maturity assessment, and keypoint localization—make it an ideal candidate for deployment in complex field conditions, contributing to intelligent yield estimation, labor planning, and post-harvest quality control in precision agriculture.

5 Conclusions

In this paper, for effective tomato bunch harvesting, an improved YOLOv8np-RCW model based on YOLOv8n pose is proposed, which is an end-to-end multitasking model that recognizes tomato bunches and bunches maturity and location of keypoints. The RepVGG architecture, CARAFE upsampling module, and WIoU loss are incorporated to improve the model's performance. Ablation and comparative experiments indicate that *P*, *R*, and mAP50 of YOLOv8np-RCW for detection boxes are 84.1%, 86.3%, and 87.3% respectively. For keypoints, the *P*, *R*, and mAP50 are 83.6%, 85.9%, and 86.8% respectively. Compared to the YOLOv8n pose for detection boxes, there have been increases of 3.8%, 12.8%, and 6.2% in *P*, *R*, and mAP50 respectively; for keypoints, improvements of 2.3%, 10.6%, and 5.5% are observed in *P*, *R*, and mAP50 respectively. The model's parameters, Gflops, and inference time remain essentially unchanged. The bounding box loss decreased by 55.76%, and the maturity detection accuracy improved. Besides, the Euclidean distance error in pixels between predicted and ground truth is maintained within 20 pixels. Completing the tasks of bunch detection, maturity assessment, and keypoint localization requires only 9.8 ms. Compared to RTMDet-RTMPose and Fasterrcnn-RTMPose algorithms, the proposed algorithm demonstrates superior performance in both detection accuracy and speed.

This paper proposes a method that combines keypoint 2D information to control the end effector's motion path based on an improved model, and applies it to a harvesting robot. Field experiments show that the harvesting success rate is 68%, and the average time to harvest each bunch of tomatoes is 10.8366 s.

These outcomes suggest that the improved YOLOv8np-RCW demonstrates both robustness and suitability regarding detection precision and equipment deployment. Looking ahead, there is a plan to further optimize the algorithm and explore potential applications in detecting other crops.

Acknowledgements

This work received funding support from the National Key Research and Development Program of China (Grant No. 2022YFD2000500).

[References]

[1] Liu J Z, Peng Y, Faheem M. Experimental and theoretical analysis of fruit plucking patterns for robotic tomato harvesting. Comput Electron Agric, 2020; 173: 105330.

[2] Zhang F, Gao J, Zhou H, Zhang J X, Zou K L, Yuan T. Three-dimensional pose detection method based on keypoints detection network for tomato bunch. Comput Electron Agric, 2022; 195: 106824.

[3] Maureira F, Rajagopalan K, Stöckle C O. Evaluating tomato production in open-field and high-tech greenhouse systems. J Clean Prod, 2022; 337: 130459.

[4] Zhou H Y, Wang X, Au W, Kang H W, Chen C. Intelligent robots for fruit harvesting: recent developments and future challenges. Precis Agric, 2022; 23: 1856–1907.

[5] Zheng X J, Rong J C, Zhang Z Q, Yang Y, Li W, Yuan T. Fruit growing direction recognition and nesting grasping strategies for tomato harvesting robots. J Field Robot, 2024; 41: 300–313.

[6] Wu J Q, Fan S Z, Gong L, Yuan J, Zhou Q, Liu C L. Research status and development direction of design and control technology of fruit and vegetable picking robot system. Smart Agric, 2020; 2(4): 17–40.
[7] Gao J, Zhang J X, Zhang F, Gao J F. LACTA: A lightweight and accurate algorithm for cherry tomato detection in unstructured environments. Expert Syst Appl, 2024; 238: 122073.
[8] Rapado-Rincón D, van Henten E J, Kootstra G. Development and evaluation of automated localisation and reconstruction of all fruits on tomato plants in a greenhouse based on multi-view perception and 3D multi-object tracking. Biosyst Eng, 2023; 231: 78–91.
[9] Xiong Y, Ge Y, From P J. An obstacle separation method for robotic picking of fruits in clusters. Comput Electron Agric, 2020; 175: 105397.
[10] Kim J, Pyo H, Jang I, Kang J, Ju B, Ko K. Tomato harvesting robotic system based on Deep-ToMaToS: Deep learning network using transformation loss for 6D pose estimation of maturity classified tomatoes with side-stem. Comput Electron Agric, 2022; 201: 107300.
[11] Li H P, Li C Y, Li G B, Chen L X. A real-time table grape detection method based on improved YOLOv4-tiny network in complex background. Biosyst Eng, 2021; 212: 347–359.
[12] Li T H, Sun M, He Q H, Zhang G S, Shi G Y, Ding X M, et al. Tomato recognition and location algorithm based on improved YOLOv5. Comput Electron Agric, 2023; 208: 107759.
[13] Zhang J X, Xie J Y, Zhang F, Gao J, Yang C, Song C Y, et al. Greenhouse tomato detection and pose classification algorithm based on improved YOLOv5. Comput Electron Agric, 2024; 216: 108519.
[14] Yoshida T, Fukao T, Hasegawa T. Cutting point detection using a robot with point clouds for tomato harvesting. J Robot Mechatron, 2020; 32(2): 437–444.
[15] Qi J T, Liu X N, Liu K, Xu F R, Guo H, Tian X L, et al. An improved YOLOv5 model based on visual attention mechanism: Application to recognition of tomato virus disease. Comput Electron Agric, 2022; 194: 106780.
[16] Rong Q J, Hu C H, Hu X D, Xu M X. Picking point recognition for ripe tomatoes using semantic segmentation and morphological processing. Comput Electron Agric, 2023; 210: 107923.
[17] Fu L H, Wu F Y, Zou X J, Jiang Y L, Lin J Q, Yang Z, et al. Fast detection of banana bunches and stalks in the natural environment based on deep learning. Comput Electron Agric, 2022; 194: 106800.
[18] Zhu Y J, Li S S, Du W S, Du Y P, Liu P, Li X. Identification of table grapes in the natural environment based on an improved YOLOv5 and localization of picking points. Precis Agric, 2023; 24: 1333–1354.
[19] Chen J Q, Ma A Q, Huang L X, Li H W, Zhang H Y, Huang Y, et al. Efficient and lightweight grape and picking point synchronous detection model based on key point detection. Comput Electron Agric, 2024; 217: 108612.
[20] Ukwuoma C C, Zhiguang Q, Bin Heyat M B, Ali L, Almaspoor Z, Monday H N. Recent advancements in fruit detection and classification using deep learning techniques. Math Probl Eng, 2022; 2022(1): 9210947.
[21] Koirala A, Walsh K B, Wang Z, McCarthy C. Deep learning – Method overview and review of use for fruit detection and yield estimation. Comput Electron Agric, 2019; 162: 219–234.
[22] Redmon J, Farhadi A. YOLO9000: Better, faster, stronger. 2016; Available: https://doi.org/10.48550/arXiv.1612.08242. Accessed on [2024-11-17].
[23] Bochkovskiy A, Wang C-Y, Liao H-Y M. YOLOv4: Optimal speed and accuracy of object detection. 2020; Available: https://doi.org/10.48550/arXiv.2004.10934. Accessed on [2024-11-17].
[24] 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. 2022; Available: https://doi.org/10.48550/arXiv.2207.02696. Accessed on [2024-11-17].
[25] Ding X H, Zhang X Y, Ma N N, Han J G, Ding G G, Sun J. RepVGG: making VGG-style ConvNets great again. 2021. Available: https://doi.org/10.48550/arXiv.2101.03697. Accessed on [2024-09-23].
[26] He K M, Zhang X Y, Ren S Q, Sun J. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA: IEEE, 2016; pp.770–778. doi: 10.1109/CVPR.2016.90.
[27] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015; Available: https://doi.org/10.48550/arXiv.1502.03167. Accessed on [2025-01-23].
[28] Wang J Q, Chen K, Xu R, Liu Z W, Loy C C, Lin D H. CARAFE: Content-aware reassembly of FEatures. 2019; Available: https://doi.org/10.48550/arXiv.1905.02188. Accessed on [2024-11-03].
[29] Tong Z J, Chen Y H, Xu Z W, Yu R. Wise-IoU: Bounding box regression loss with dynamic focusing mechanism. 2023; Available: https://doi.org/10.48550/arXiv.2301.10051. Accessed on [2024-12-12].