Alexandria Engineering Journal 128 (2025) 992–1014

Contents lists available at ScienceDirect

Alexandria Engineering Journal

journal homepage: www.elsevier.com/locate/aej

Original article

VBP-YOLO-prune: Robust apple detection under variable weather via feature-adaptive fusion and efficient YOLO pruning

Haohai You [a], Hao Wang [b], Zhanchen Wei [a], Chunguang Bi [a], Lijuan Zhang [a,c], Xuefang Li [a], Yingying Yin [a,*]

[a] College of Information Technology, Jilin Agricultural University, Changchun 130118, China
[b] Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[c] College of Internet of Things Engineering, Wuxi University, Wuxi 214105, China

ARTICLE INFO

Keywords:
apple detection
YOLOv8
deep learning
lightweight deployment

ABSTRACT

Apple-picking robots are increasingly applied in smart agriculture, but their performance is limited by complex orchard conditions such as unstable lighting, occlusion, and weather variations. This study proposes an optimized lightweight detection model, VBP-YOLO-prune, based on YOLOv8n, to enhance detection accuracy and deployment efficiency on edge devices. The model incorporates a V7 downsampling module, BiFPN feature fusion, and an improved PIOUv2 loss function, aiming to improve multi-scale representation and bounding box regression. A custom apple dataset was augmented with diverse lighting and weather conditions to improve generalization. Experimental results using 10-fold cross-validation show that VBP-YOLO-prune achieves 89.0 % mAP50 and 66.26 % mAP50–95, with Precision of 84.01 % and Recall of 80.52 %. Additionally, it reduces parameters by 79.7 %, FLOPs by 60.9 %, and increases FPS by 29.2 % compared to YOLOv8n.The final model contains only 0.61 M parameters, 3.2 GFLOPs, and runs at 102.6 FPS on NVIDIA Jetson Orin Nano. These results demonstrate that VBP-YOLO-prune provides a practical and efficient solution for real-time fruit detection in complex environments. Future research may extend this approach to other crop types and explore full integration into autonomous harvesting systems.

1. Introduction

Apples are one of the most significant fruit crops globally, predominantly cultivated in temperate regions. In recent years, China has emerged as the leading apple producer, contributing to nearly half of the global output [14]. Renowned for their distinct flavor and high nutritional value, apples offer numerous health benefits, including the potential to reduce the risk of colds and certain cancers [43,45]. Despite these advantages, one of the primary challenges faced during the apple ripening period is ensuring efficient and timely harvesting [42]. Historically, orchard managers have depended on a substantial temporary workforce to meet this demand. However, with an increasing scarcity of labor, the adoption of automated harvesting technologies, particularly robotics, has become a critical focus in modern agricultural practices [15,38].

The advancement of computer vision technology has significantly enhanced target recognition in fruit and vegetable inspection [21,26,34]. Machine vision [19], empowered by deep learning, is increasingly adopted in agriculture for its efficiency and non-destructive nature [28]. Notably, deep learning-based visual feature extraction methods—ranging from CNNs and DenseNet to hybrid GNNs—have demonstrated strong cross-domain generalization in areas such as medical imaging [2,25,36], facial recognition [13,27], and plant disease identification, providing a robust technical foundation for their adaptation to precision agriculture ([30]; Y et al., 2025).[31]. proposed a Hybrid Vision Transformer and CNN model for detecting Overripe Coffee Berry Disease, achieving 96.7 % accuracy and outperforming traditional CNN-based methods. Jia et al., [11]. proposed a visual detector model

List of Abbreviations: AI, Artificial Intelligence; AP, Average Precision; BN, Batch Normalization; CNN, Convolutional Neural Network; FLOPs, Floating Point Operations; FPS, Frames Per Second; GPU, Graphics Processing Unit; IoU, Intersection over Union; MAP, mean Average Precision; NAS, Neural Architecture Search; NMS, Non-Maximum Suppression; ReLU, Rectified Linear Unit; SiLU, Sigmoid Linear Unit; SSD, Single Shot MultiBox Detector; YOLOv8, You Only Look Once version 8.

* Corresponding author.
E-mail addresses: zhanglijuan@cwxu.edu.cn (L. Zhang), 2222210687@stu.xjtu.edu.cn, yyy@jlau.edu.cn (Y. Yin).

https://doi.org/10.1016/j.aej.2025.08.013
Received 17 April 2025; Received in revised form 24 June 2025; Accepted 8 August 2025
Available online 12 August 2025

based on the Mask R-CNN for harvesting robots, which detects and segments overlapping green apples. Zhu et al., [51]. developed an improved YOLOv7 model, SDF-YOLO, designed for oleander fruit detection in complex orchard environments, achieving a remarkable mAP of 96.65 % and high adaptability to variables such as fruit traits, varietal differences, environmental changes, and lighting conditions. Chen et al. [3]. introduced an efficient, real-time cabbage seedling counting method using the YOLOv8n model, improving mAP50–95 by 14.5–90.3 % and reducing counting time to one-seventh that of manual methods. Chen et al., [4]. proposed an improved YOLOv7 multi-task deep convolutional neural network (MTD-YOLOv7) for automated detection of tomato fruit and ripeness, expanding dataset labels, adding decoders, and optimizing the loss function. The model demonstrated excellent accuracy and fast inference speed on the cherry tomato dataset. Bai et al., [1]. improved the YOLO real-time recognition algorithm, successfully addressing challenges such as small size, similar color, and overlapping occlusion of strawberry flowers and fruits. By introducing the Swin Transformer prediction head and GS-ELAN network neck optimization, the model significantly improved detection accuracy and real-time performance, surpassing the original YOLOv7 model in both robustness and detection capability. Deep learning-based research has brought a variety of effective solutions for dealing with complex external environments. Despite the positive efforts in improving detection accuracy and speed, the models are usually large in size, with numerous parameters and high computational overheads, which limit their further application in the field, especially for agricultural users seeking portability and cost-effectiveness.

Therefore, deploying deep learning models on edge devices has become an important hot topic in recent years. Many studies are turning to more lightweight target detection algorithms. Wang, He [40]. proposed a fast and accurate method for detecting apple fruits before fruit thinning, using channel pruning on the YOLOv5s model, reducing the model size to 1.4 MB and achieving an F1 score of 91.5 %. This serves as a reference for portable mobile fruit-thinning devices. Yan et al., [44]. developed a lightweight apple detection method for picking robots, improving YOLOv5s to efficiently identify graspable and ungraspable apples under varying degrees of obstruction. Ma et al., [24]. proposed a lightweight detection algorithm based on an upgraded YOLOv7-tiny, achieving 80.4 % mAP and a loss rate of 0.0316 in complex weather conditions by enhancing the network structure and attention mechanism. Liu et al., [17]. introduced Faster-YOLO-AP, a lightweight apple detection algorithm based on YOLOv8, reducing parameters to 0.66 M and FLOPs to 2.29 G, achieving a mAP of 84.12 % on edge devices. By optimizing the network with PDWConv and DWSConv and using the EIoU loss, this model significantly improved speed and accuracy. Wang et al., [41]. developed the PG-YOLO method, which drastically reduced model complexity and enhanced the speed and accuracy of pomegranate detection. The model's weight is only 2.2 MB, 89.9 % smaller than YOLOv8s, with a 74.1 % improvement in detection speed. In apple orchards, the detection of small targets and the complex natural environment—impacted by weather variations, cluttered backgrounds, and fruit-leaf shadows—can significantly hinder accurate yield estimation and fruit picking. As picking robots are often equipped with vision modules on mobile devices that have limited computational resources, lightweight models are crucial to overcoming these challenges. This research proposes the VBP-YOLO-prune model, a super-lightweight solution designed specifically for fruit detection in apple orchards under complex weather conditions. The aim of this study is to develop efficient, real-time apple detection models that are suitable for deployment on mobile devices with constrained computational power. Fruit detection, as a core technology in agricultural robotics, plays a vital role in orchard monitoring, automated picking, and yield counting, which are essential for enabling intelligent orchard management.

The main contributions of this paper are:

(1) By self-constructing the apple dataset and using data enhancement techniques to construct an apple dataset with complex scenes, we diversify the image data and improve the anti-interference ability of the model under complex conditions. (2) The V7SP module reduces spatial redundancy through efficient downsampling, improving semantic information extraction, which is particularly useful for detecting occluded or small-scale targets. The BiFPN module enhances the model's target perception across different scales and complex backgrounds by fusing features at multiple levels. Together, these modules optimize detection performance and control model complexity. (3) PIoUv2 decouples localization and confidence in bounding box regression, improving accuracy for overlapping or occluded targets, while pruning reduces model size and computational complexity by removing unnecessary network structures, optimizing performance and accelerating inference without sacrificing accuracy. (4) Model deployment on NVIDIA Jetson Orin Nano verifies real-time inference capability, highlighting its accuracy, efficiency, and applicability for smart agriculture.

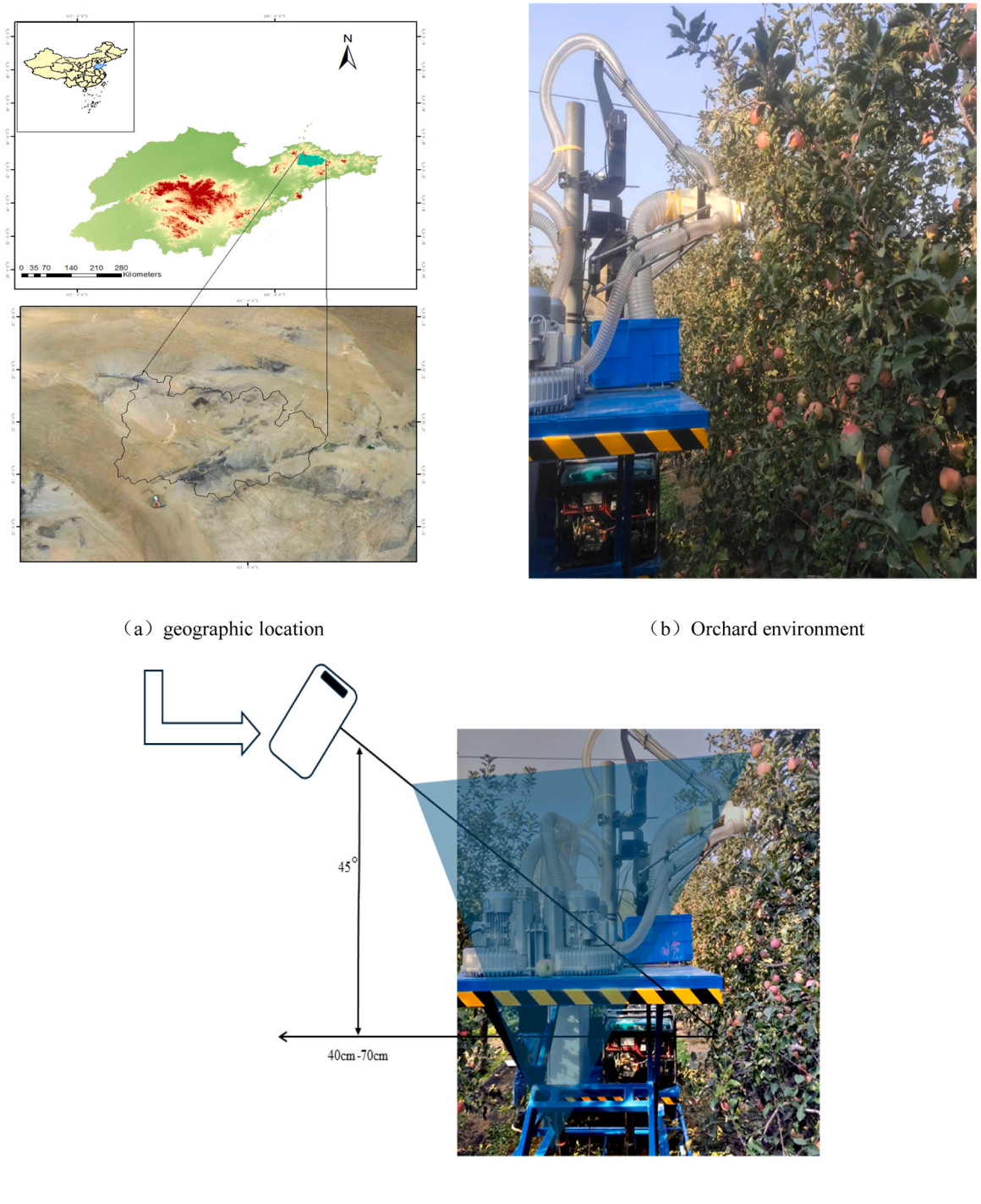
2. Materials and methods

2.1. Data collection

The dataset used in this study was manually collected in an apple orchard in Yantai City, Shandong Province, China, between October 20 and 31, 2023, from 8:00 am to 5:00 pm. Fuji apples were the primary object of interest, and images were captured using an iPhone 15 Pro Max (Apple, Cupertino, CA, USA) to ensure high-resolution data. To simulate the movement of a harvesting robot, the camera was positioned approximately 30–70 cm away from the apples, a distance range chosen based on typical robotic harvesting parameters. Images were taken from various angles and under different lighting conditions (e.g., direct sunlight, overcast skies) to enhance the model's adaptability. A total of 1550 images were collected and manually annotated using the online tool "makesense.ai" (accessed 20 July 2024), with each apple labeled with a rectangular bounding box under the category "apple." The annotations were exported in VOC format and converted to YOLO format for model training. While the iPhone 15 Pro Max's camera specifications (e.g., 48 MP resolution, automatic adjustments) may differ from those of industrial harvesting robot cameras, post-processing techniques were applied to standardize the images and minimize variability. Preliminary tests indicate that the model can detect apples effectively within the 30–70 cm range and up to 1 m, though further validation with robotic camera data is planned for future work. The specific geographical location of the collection, the growing environment of the orchard and the angle of the collected data set are shown in Fig. 1.

2.2. Data augmentation

To evaluate the robustness and generalization capability of the proposed model under complex orchard conditions, we constructed a weather-augmented dataset based on real-world apple images[33]. The dataset initially included 32,465 labeled apple instances. Before augmentation, the original dataset was first split into training, testing, and validation sets in a 7:2:1 ratio, ensuring that augmentation would not introduce data leakage and that class and environmental distributions remained balanced across subsets. Subsequent augmentation was performed independently within each subset using the imgaug library and the RoboFlow platform (https://app.roboflow.com) to simulate various complex weather scenarios, including: (1) 90-degree image rotation to enhance geometric diversity; (2) Saturation adjustment (saturation = 45) and exposure increase (+40 %) to simulate strong lighting during morning or evening hours; (3) Gaussian blur (blur = 1.5) to simulate light fog conditions; (4) Addition of salt-and-pepper noise combined with imgaug transformations to mimic rainy and noisy environments. After augmentation, the dataset was expanded to 4900 images containing a total of 110,766 apple instances, and the dataset is divided into training set (3430 images), test set (979 images), and validation set (491 images). In addition, the proportion of simulated



（a）geographic location

（b）Orchard environment

（c）camera position

Fig. 1. Data collection location and environment.

weather conditions and natural weather conditions captured in the original dataset was kept nearly identical across all subsets. This ensured that no subset was overly dominated by any specific weather type, effectively preventing overfitting to a particular environment and enhancing the model's adaptability and generalization to diverse real-world orchard conditions Fig. 2.

2.3. YOLOv8

YOLOv8, developed based on YOLOv5, represents the latest advancement in the YOLO series[12].It supports detection, classification, and segmentation tasks with a modular architecture consisting of a backbone, neck, and detection head. As a single-stage algorithm[35], YOLO offers faster detection than traditional two-stage methods, making it well-suited for real-time applications in agriculture, such as identifying crops, weeds, and pests to optimize management and increase productivity.YOLOv8 improves performance through architectural and training optimizations, including replacing 6×6 convolutions with more efficient 3×3 convolutions, using the C2f module instead of C3, and enhancing multi-scale detection through spatial pyramid pooling (SPPF). The neck aggregates multi-scale features, and a new Mosaic data augmentation technique is employed

Original dataset dataset after data enhancement

（a）Enhanced exposure to bright light

（b）Weak light intensified in the evening

（c）Light fog intensifies in complex weather

（d）Complex weather and rainy days intensified

Fig. 2. Data augmentation for complex environments.

during training. These enhancements improve both flexibility and efficiency. To further boost prediction accuracy, YOLOv8 employs an optimized loss function that combines bounding box loss, classification loss, and distribution focal loss (DFL).

2.4. Proposed modelling framework

In designing the YOLOv8.0 y model, we optimised both the Backbone and Head sections to enhance feature extraction and fusion capabilities while reducing model complexity. Specifically, the Backbone employs V7 downsampling instead of the traditional convolutional structure, improving feature extraction efficiency. In the Neck, we introduced a feature pyramid to enhance multi-scale feature fusion, thereby improving detection performance across different target scales. Additionally, we integrated PIOUv2 as an improved loss function, significantly increasing the model's convergence speed and robustness. These optimizations not only enable YOLOv8.0 n to maintain lightweight characteristics while achieving efficient target detection but also demonstrate significant performance improvements in experiments, as shown in Fig. 3.

2.4.1. YOLOv8y

YOLOv8 provides five scaled variants—YOLOv8n, s, m, l, and x—designed for diverse application needs. These variants differ in width (widen factor, adjusting channel numbers), depth (deepen factor, controlling repeated modules), and output feature resolution (ratio, tuning channels of high-level feature maps). However, for our single-category apple detection task, even YOLOv8n introduces unnecessary complexity, as it is designed for general-purpose detection with robustness exceeding our actual requirements. To address this, we propose a customized lightweight variant, YOLOv8y ("y" denoting an even smaller scale), which further simplifies the model structure. By setting the scaling factors to 0.16 (width), 0.33 (depth), and 2.0 (output ratio), YOLOv8y significantly reduces computational cost while maintaining detection accuracy. Table 1 summarizes the scaling factors for all YOLOv8 variants, including YOLOv8y, to illustrate their relative

Table 1
Details of scaling factors of YOLOv8.

Network	widen factor	deepen factor	ratio
YOLOv8y	0.16	0.33	2.0
YOLOv8n	0.25	0.33	2.0
YOLOv8s	0.50	0.33	2.0
YOLOv8m	0.75	0.67	1.5
YOLOv8l	1.0	1,00	1.0

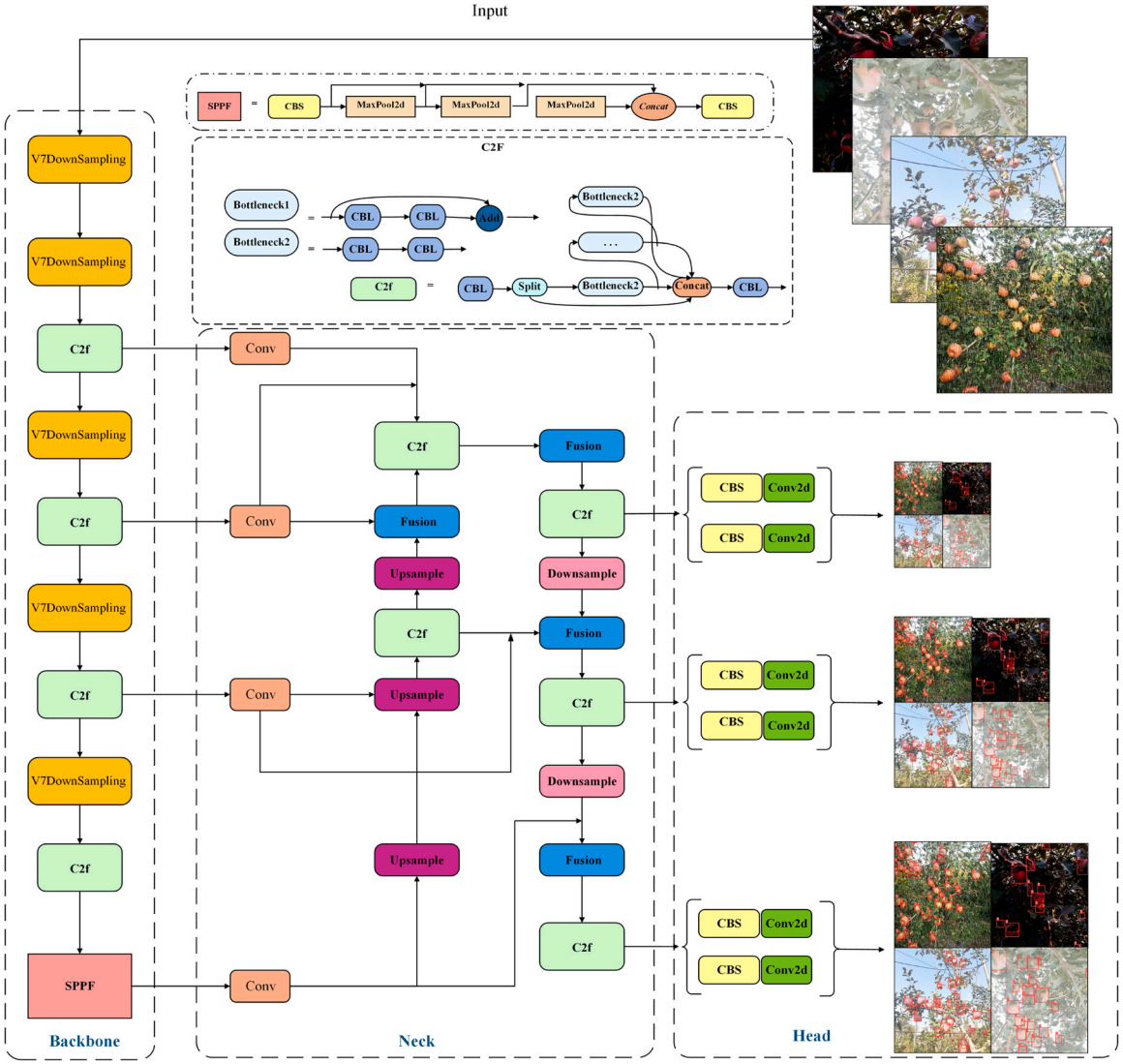


Fig. 3. Network structure of VBP-YOLO-prune.

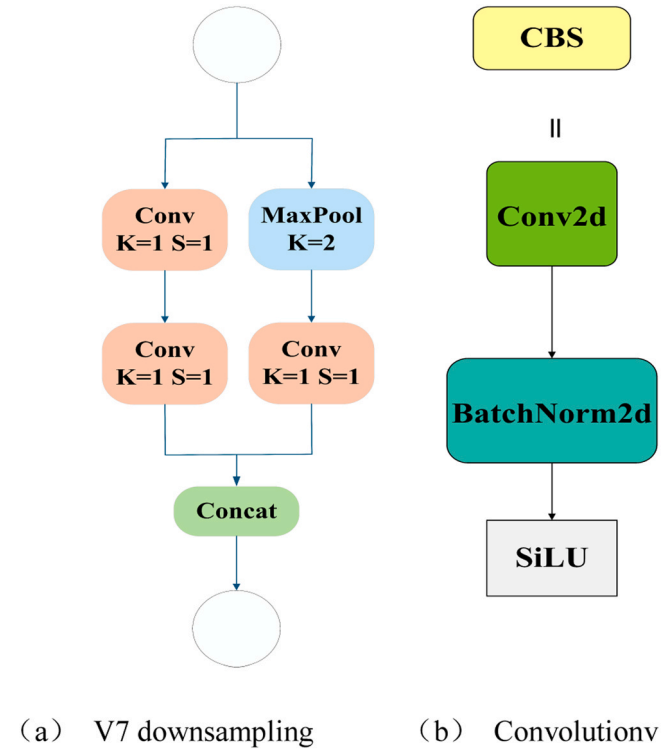


Fig. 4. Convolution comparison between the V7 downsampling structure and the YOLOv8 model.

complexity and computational demands.

2.4.2. Downsampling (V7Downsampling)

Downsampling is essential in Convolutional Neural Networks (CNNs) to reduce the spatial dimensions of feature maps while retaining key features[49]. YOLOv8 uses a 3×3 convolution with stride 2, but in complex scenarios with many targets, this may cause loss of fine details, affecting accuracy. To improve this, we adopt YOLOv7's downsampling module, which combines max-pooling and convolution to enhance feature extraction and reduce computational complexity. The module applies 2×2 max-pooling (stride 2) to reduce spatial size, followed by a 1×1 convolution to adjust channel numbers, and a 3×3 convolution (stride 2) for deeper downsampling. The resulting feature maps from both paths are concatenated, integrating multiple feature sets. This approach retains more features and improves efficiency compared to YOLOv8's original downsampling method. Figs. 4 and 5 shows the structure of YOLOv7's downsampling module, where the feature maps from max-pooling and convolution paths are concatenated to form the final output.

Algorithm 1. na

Input: x

Output: y

1: $ouc \leftarrow ouc // 2$

2: $maxpool\_branch \leftarrow MaxPool2d\ (kernel = 2, stride = 2) \rightarrow Conv(inc, ouc, k = 1)$

3: $conv\_branch \leftarrow Conv\ (inc, ouc, k = 1) \rightarrow Conv\ (ouc, ouc, k = 3, s = 2)$

4: $feature1 \leftarrow maxpool\_branch(x)$

5: $feature2 \leftarrow conv\_branch(x)$

6: $y \leftarrow Concat(feature1, feature2, dim = 1)$

7: $return\ y$

2.4.3. BIFPN (Bidirectional Feature Pyramid Network)

BiFPN (Bidirectional Weighted Feature Pyramid Network) improves upon the traditional FPN (Feature Pyramid Network) by addressing its limitations in multi-scale object detection. FPN uses top-down unidirectional feature transfer with fixed-weight fusion, which struggles to preserve fine-grained information, especially for small objects[5,8]. BiFPN introduces a bidirectional feature fusion mechanism that enables feature transfer in both directions across network layers, improving efficient computations without sacrificing performance. Experimental results demonstrate that VBP-YOLO-prune, with BiFPN, achieves superior performance in multi-scale target detection, significantly improving detection accuracy and computational efficiency compared to traditional models.

Algorithm 2. na

Input: P1, P2, P3, P4, P5

Output: P1_out, P2_out, P3_out, P4_out, P5_out

$P1\_in \leftarrow P1$

$P2\_in \leftarrow P2$

$P3\_in \leftarrow P3$

$P4\_in \leftarrow P4$

$P5\_in \leftarrow P5$

$P5\_td \leftarrow P5\_in$

$P4\_td \leftarrow fuse(P4\_in, Upsample(P5\_td))$

$P3\_td \leftarrow fuse(P3\_in, Upsample(P4\_td))$

$P2\_td \leftarrow fuse(P2\_in, Upsample(P3\_td))$

$P1\_td \leftarrow fuse(P1\_in, Upsample(P2\_td))$

$P1\_out \leftarrow P1\_td$

$P2\_out \leftarrow fuse(P2\_in, P2\_td, Downsample(P1\_out))$

$P3\_out \leftarrow fuse(P3\_in, P3\_td, Downsample(P2\_out))$

$P4\_out \leftarrow fuse(P4\_in, P4\_td, Downsample(P3\_out))$

$P5\_out \leftarrow fuse(P5\_in, P5\_td, Downsample(P4\_out))$

$return\ P1\_out, P2\_out, P3\_out, P4\_out, P5\_out$

multi-scale feature sharing. This allows for more efficient integration of both low-level and high-level features, enhancing detection accuracy, especially for small objects. Additionally, BiFPN uses learnable weights to dynamically adjust the importance of feature maps at different resolutions, optimizing fusion and reducing computational complexity. These optimizations simplify the process, aligning resolutions and reducing floating-point operations (FLOPs), which leads to more

2.4.4. IoU-based loss function (PIoUv2)

PIoUv1 addresses the limitations of traditional IoU-based loss functions. It introduces a penalty factor that adjusts for the relative sizes of anchor and target boxes, reducing unnecessary anchor box expansion and improving regression efficiency. Additionally, PIoUv1 includes a gradient adjustment function that aligns anchor boxes more precisely with target boxes, helping the model converge faster. The calculation

formula is (1) and (2).

$$P = \frac{\frac{dw_1}{w_{gt}} + \frac{dw_2}{w_{gt}} + \frac{dh_1}{h_{gt}} + \frac{dh_2}{h_{gt}}}{4} \# \tag{1}$$

$$R_{PIoU} = f(P) \# \tag{2}$$

PIoUv2 builds upon the strengths of PIoUv1, incorporating features from EIoU, SIoU, and WIoU[18]. A key innovation in PIoUv2 is the introduction of a non-monotonic attention mechanism, controlled by a single hyperparameter. This mechanism allows the model to differentiate more effectively between high-quality and low-quality anchor boxes, improving detection accuracy. PIoUv2 also optimizes regression by focusing on medium-quality anchor boxes, enhancing the model's performance further. The calculation formula is (3).

$$L_{PIoU_v2} = u(\lambda q) L_{PIoU_v1} \# \tag{3}$$

In summary, PIoUv2 reduces anchor box expansion, improves the model's ability to select high-quality anchors, and significantly enhances both detection performance and model robustness.

Algorithm 3. na

Input: pred_box, gt_box // (cx, cy, w, h)

Output: L_p2

$dx \leftarrow pred\_box.cx - gt\_box.cx$

$dy \leftarrow pred\_box.cy - gt\_box.cy$

$dw \leftarrow log\ (pred\_box.w\ /\ gt\_box.w)$

$dh \leftarrow log\ (pred\_box.h\ /\ gt\_box.h)$

$P \leftarrow compute\ the\ square\ root\ of\ (dx\ squared\ +\ dy\ squared\ +\ dw\ squared\ +\ dh\ squared\ +\ epsilon)$

$RP \leftarrow 1 - exp\ (-P\ squared)$

$q \leftarrow exp(-P)$

$u \leftarrow 3\ multiplied\ by\ q\ multiplied\ by\ exp\ (-q\ squared)$

$iou \leftarrow IOU\ (pred\_box, gt\_box)$

$L\_p1 \leftarrow 1 - iou + RP$

$L\_p2 \leftarrow u\ multiplied\ by\ L\_p1$

$return\ L\_p2$

2.5. Experimental environment

In the experiments, the experimental platform was built on Ubuntu 18.04 LTS as the operating system. The deep learning framework used was PyTorch 2.0.0, running with CUDA 11.8 and cuDNN 8.6.0 for GPU acceleration. Python 3.8.10 was used as the programming language. The hardware configuration includes an Intel(R) Xeon(R) Silver 4214 R CPU with 24 cores, and an NVIDIA GeForce RTX 3090 GPU with 24 GB of VRAM. The system was equipped with 128 GB of DDR4 ECC memory and a 2 TB SSD. All experiments were conducted in a virtual environment created using Conda, and code development was performed using PyCharm2024.1.3. The detailed hyperparameters of the experiment are shown in Table 2.

2.6. Evaluation

Evaluation metrics are key tools for measuring model performance. To comprehensively evaluate detection accuracy, we used precision (P), recall (R), and mean Average Precision (mAP) as core indicators. Precision is defined as the ratio of correctly detected objects to the total number of detected objects, and recall is defined as the ratio of correctly detected objects to the total number of ground truth objects, to provide a more comprehensive measure across different IoU thresholds, we used mAP@0.50:0.95, which averages the AP scores calculated at IoU thresholds ranging from 0.50 to 0.95 in increments of 0.05. As given in Eqs. (4) and (7):

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

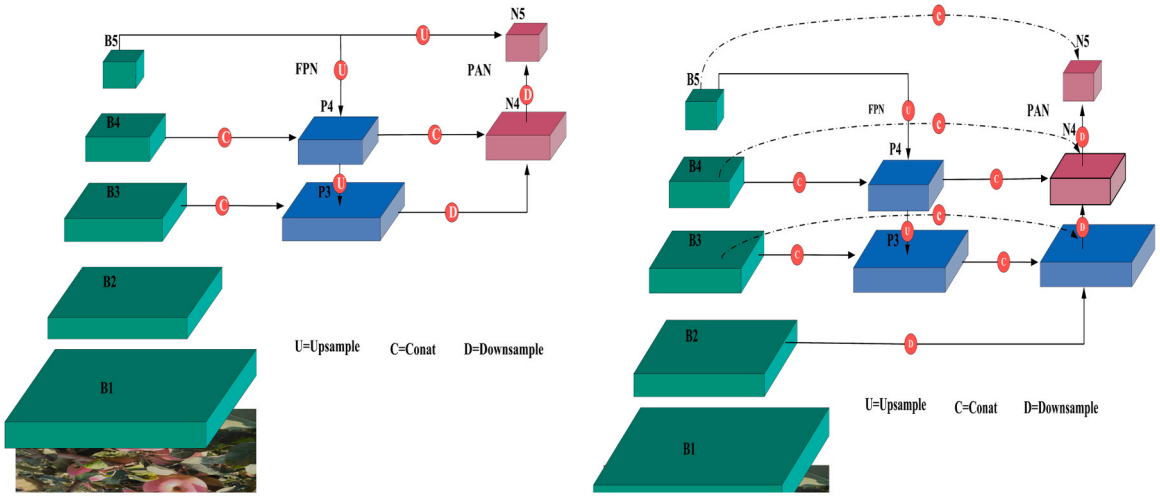


Fig. 5. Structure diagram of FPN and BIFPN.

Table 2
Detailed hyperparameters of the experiment.

Parameters	Setup
Epochs	100
Batch Size	16
Optimizer	SGD
Initial Learning Rate	0.01
Final Learning Rate	0.01
Momentum	0.937
Weight-Decay	5×10^{-4}
Images	640

$$Recall = \frac{TP}{TP + FN} \# \quad (5)$$
$$\#$$

$$AP = \int P(R)dR \times 100\% \# \quad (6)$$

$$mAP = \frac{1}{k}\sum_{i=1}^{k} AP_i \times 100\% \# \quad (7)$$

The correlation between model counts and true counts is evaluated by the coefficient of determination R^2 in Equation X, where y_i denotes the manually observed true value. $\hat{y}_i$ denotes the model count result. $\hat{y}_i$ denotes the meaning. The calculation formula is (8).

$$R^2 = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (y_i - y_i)^2} \quad (8)$$

The Root Mean Square Error (RMSE) is used to assess the degree of discrepancy between the model counts and the true counts and is defined as the formula X. n is the number of images, y_i is the true count, and $\hat{y}_i$ denotes the model count results. The calculation formula is (9).

$$RMSE = \sqrt{\frac{\sum_i^n (\hat{y}_i - y_i)^2}{n}} \quad (9)$$

Where true positives (TP) indicate the number of instances that are Apple and are also predicted to be Apple. false positives (FP) indicate the number of instances that are not actually Apple but are predicted to be Apple. False Negatives (FN) indicate the number of instances that are Apple but not predicted to be Apple. In terms of efficiency and speed, we evaluate the computation and complexity of the model using FLOP (Giga), model size (Megabyte) and parameters (Mega).

3. Experimental results

3.1. Scale factor selection

A scale factor of 0.16 was chosen as it provides an optimal balance between detection performance and computational cost. With this configuration, the model achieved 87.89 % mAP50 and 64.78 % mAP50–95, along with a precision of 83.65 % and recall of 80.11 %. While its FLOPs and parameter count are slightly higher than those of ultra-lightweight models, the increase is moderate and still well below the thresholds of heavier models. This makes the 0.16 scale factor an effective choice, delivering strong accuracy while maintaining efficiency, and is well-suited for real-world applications with resource constraints. Specific results are summarized in Table 3.

From the results of (a) and (b) in Fig. 6, it is evident that different expansion factors have a significant impact on model performance. Among the tested configurations, 0.16 provides the best trade-off between mAP50 and mAP95. Compared to smaller factors (e.g., 0.12 and 0.14), it notably improves accuracy—especially mAP95—where lower factors fall short of practical thresholds. While 0.20 offers marginally better accuracy, it incurs substantial computational and resource costs, limiting its practical applicability. Thus, 0.16 emerges as the optimal compromise, boosting accuracy while maintaining computational efficiency.

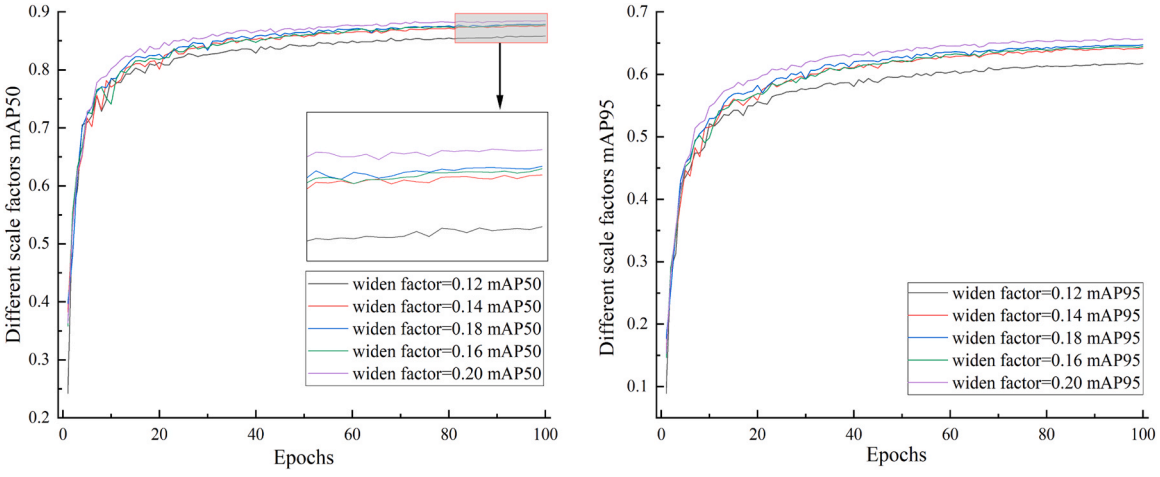
3.2. Ablation experiments

To evaluate the contribution of key modules to apple detection performance, we conducted an ablation study. The experiment started with a customized lightweight baseline model, YOLOv8y, derived from the YOLOv8 series, and progressively added optimization modules to observe their impact on performance. The results are shown in Table 4.

YOLOv8y serves as the baseline model, simplified using the Scaling Factors mechanism to reduce depth and width while maintaining high detection accuracy. This optimization makes YOLOv8y suitable for real-time apple detection. Compared to YOLOv8n, YOLOv8y reduces parameters by 49.2 % and FLOPs by 46.3 %, with only a minor decrease in performance (−0.88 % in mAP50 and −1.23 % in mAP50–95), making it a lightweight model suitable for deployment in environments with

Table 3
Selection of different scaling factors.

Scaling Factors	P (%)	R (%)	mAP50 (%)	mAP50-95 (%)	Parameters(M)	FLOPs(G)	Weigh (M)
0.12	82.33	79.34	86.13	61.86	0.91	2.8	2.0
0.14	82.45	79.48	87.36	64.09	1.14	3.8	2.5
0.16	83.65	80.11	87.89	64.78	1.53	4.4	3.1
0.18	83.57	80.13	87.88	64.5	1.80	5.2	3.7
0.20	83.72	80.36	88.47	65.51	2.12	6.4	4.4



（a）mAP50 Training Variations （b）mAP95 Training Variations

Fig. 6. Curve results for mAP50 and mAP95 for different scaling factors.

Table 4
Results of ablation experiments.

Model	Scaling Factors	Bifpn	V7SP	PIOUv2	Prune	P (%)	R (%)	mAP50 (%)	mAP50-95 (%)	Parameters (M)	FLOPs (G)	Weight (M)
YOLOv8n						84.01	79.8	88.68	66.01	3.01	8.2	6.3
YOLOv8y	√					83.65	80.11	87.89	64.78	1.53	4.4	3.1
Model 1	√	√				82.94	80.17	88.07	64.96	1.06	4.4	2.2
Model 2	√	√	√			83.78	79.96	88.22	65.16	0.98	4.4	2.1
Model 3	√	√	√	√		84.10	79.53	88.67	65.59	0.98	4.4	2.1
Model 4	√	√	√	√	√	84.01	80.52	89.0	66.26	0.61	3.2	1.5

Note: √ indicates that the module is used.

limited computational resources. YOLOv8y achieves a precision (P) of 83.65 % and recall (R) of 80.11 %. In Model 1, we incorporated the BiFPN module, which optimizes multi-scale feature fusion. This is particularly important for apple detection, as apples can vary in size and may be partially occluded in orchard settings. BiFPN allows for more efficient fusion of low and high-resolution features, improving the model's ability to detect apples at different scales and under varying conditions. As a result, BiFPN boosts mAP50 by 0.20 % and mAP50–95 by 0.18 %. Despite adding 1.06 M parameters and 4.4 G FLOPs, the accuracy gains confirm the module's effectiveness. Model 1 achieves a precision (P) of 82.94 % and recall (R) of 80.17 %. Model 2 builds upon Model 1 by adding the V7SP downsampling module. V7SP improves feature extraction by efficiently reducing spatial resolution while preserving key semantic information. This is crucial for detecting small and partially occluded apples in orchard environments. V7SP's integration increases mAP50 by 0.15 % and mAP50–95 by 0.20 %. It reduces parameters to 0.98 M while keeping the FLOPs at 4.4 G, balancing speed and accuracy. Model 2 achieves a precision (P) of 83.78 % and recall (R) of 79.96 %. In Model 3, we introduced the PIoUv2 module to optimize bounding box localization, especially in scenarios where apples are overlapping or partially obscured. PIoUv2 enhances the accuracy of bounding box regression by decoupling localization and confidence, leading to improved detection precision. In apple detection, PIoUv2 helps accurately locate apples, even when they are clustered or blocked by other apples. The inclusion of PIoUv2 raises mAP50 to 88.67 % and mAP50–95–65.59 %, without increasing parameters or FLOPs. Model 3 achieves a precision (P) of 84.10 % and recall (R) of 79.53 %. Finally, Model 4 applies the VBP-YOLO-prune strategy to prune redundant parameters, reducing the model size and computational load. This optimization is particularly valuable for deploying models on edge devices, where resource constraints are critical. Pruning reduces parameters by 79.7 % and FLOPs by 60.9 %, while still improving mAP50 to 89.0 % and mAP50–95–66.26 %. This demonstrates that pruning not only reduces model size but also slightly improves detection performance. Model 4 achieves a precision (P) of 84.01 % and recall (R) of 80.52 %. Each module progressively enhances the model's performance from YOLOv8y to Model 4. The final model achieves optimal performance: mAP50 of 89.0 % and mAP50–95 of 66.26 %, with precision (P) of 84.01 % and recall (R) of 80.52 %.

The confusion matrix for the test set, shown in Fig. 7, provides a comparative analysis of the classification performance across various models. Model 4 outperforms all others, achieving the highest True

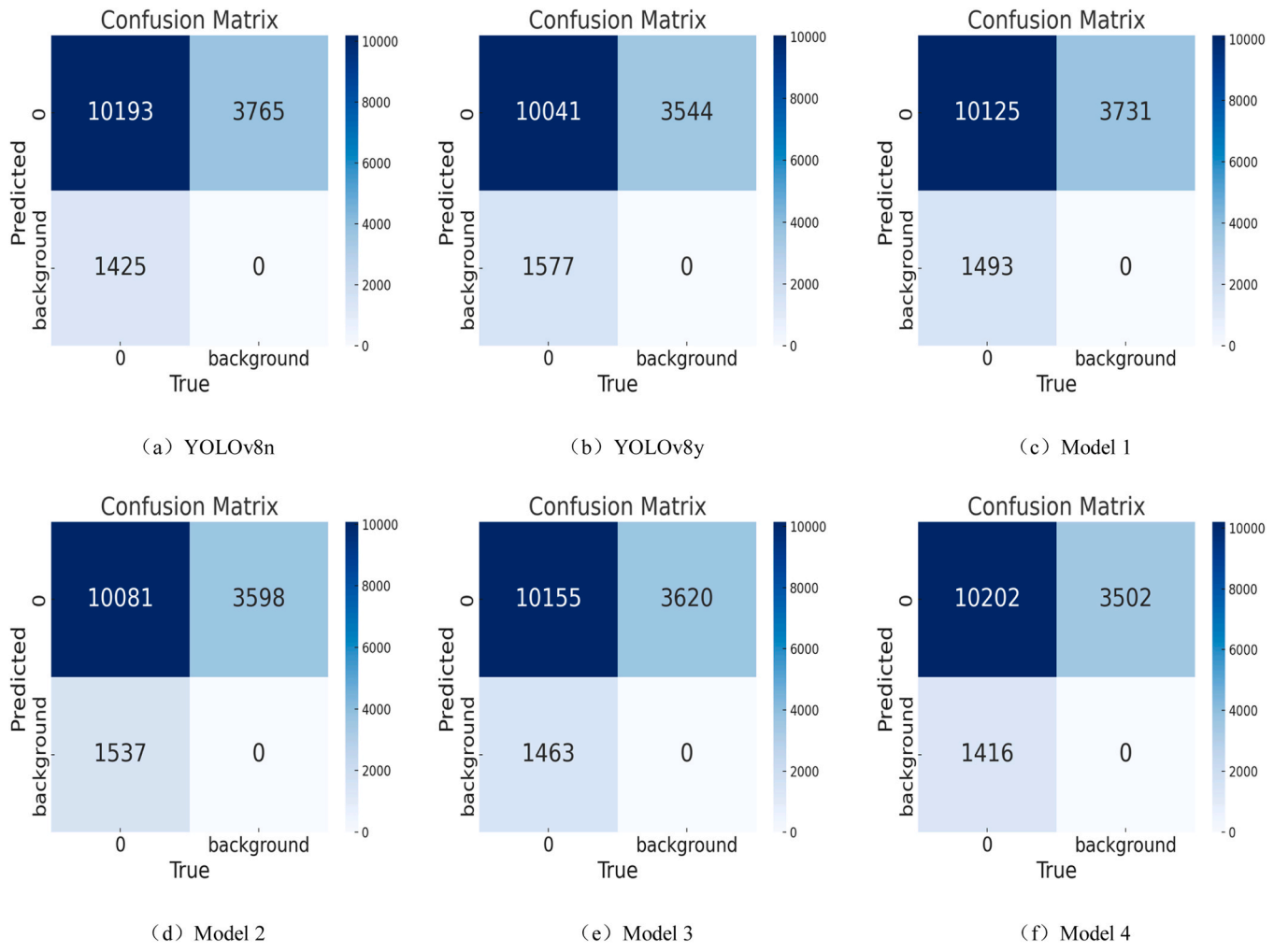


Fig. 7. Confusion matrix performance of ablation experiments on the test set.

Table 5
Experimental results of different IoU losses were compared.

LOSS	mAP50 (%)	mAP50-95 (%)	Parameters (M)	FLOPs (G)	Weight (M)
DIOU	88.23	64.98	0.98	4.4	2.1
EOIU	88.04	65.03	0.98	4.4	2.1
GIOU	88.21	65.09	0.98	4.4	2.1
SIOU	88.34	65.11	0.98	4.4	2.1
CIOU	88.22	65.16	0.98	4.4	2.1
SHAPEIOU	88.31	65.01	0.98	4.4	2.1
PIOUV2	88.67	65.59	0.98	4.4	2.1

Negative count (TN = 10,202) and the lowest False Positive (FP = 3502) and False Negative (FN = 1416) rates, indicating its superior ability in both background detection and accurate target recognition. Model 3 also delivered strong performance with well-balanced metrics across all categories. In contrast, while the YOLOv8n and YOLOv8y models excelled in background detection with high true-negative rates, they showed relatively higher false-positive and false-negative rates, suggesting room for improvement, particularly in minimizing false alarms and reducing missed detections.

3.3. Loss function IOU comparison

In this study, we evaluated the performance of various IoU loss functions applied to Model 2 before pruning to validate the model's robustness. The comparative experiments, involving different feature extraction modules and IoU loss functions, revealed that all IoU variants achieved similar reductions in loss. However, PIoUv2 distinguished itself by attaining the highest mAP50 and mAP50-95 scores of 88.67 % and 65.59 %, respectively, indicating superior accuracy. As shown in Tables 5 and 6, all IoU loss variants exhibited a similar trend in loss reduction during training, with PIoUv2 demonstrating the fastest and most consistent improvement. Fig. 8 illustrates the bounding box regression loss curves for the various loss functions. While the loss

Table 6
Performance comparison of Model 3 under different pruning and optimization strategies.

Model Index	Model 3	Model 3-Pruned	VBP-YOLO-prune (No-global)	VBP-YOLO-prune (Global)
Global Pruning	–	–	×	√
Fine-tuning after Pruning	–	×	√	√
Inference Speed-up	–	1.3	1.3	1.3
Parameters(M)	0.98	0.611	0.76	0.608
FLOPs(G)	4.4	3.3	3.4	3.2
Weight (M)	2.1	1.5	1.8	1.5
mAP50 (%)	88.68	75.33	88.61	89.0
mAP50–95 (%)	66.01	49.81	66.0	66.26

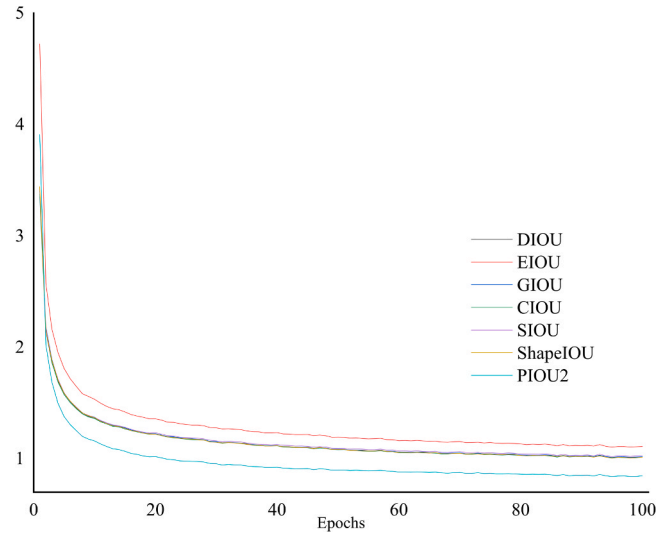


Fig. 8. Bounding box regression loss curves for different loss functions in the validation set.

curves of DIOU[50], EIOU[46], GIOU[29], CIOU(Du et al., 2021), SIOU [7], and ShapeIoU[47] are close, PIoUv2 maintains a lower loss level throughout the training, indicating its superior optimization performance and faster convergence.

As shown in Fig. 8, PIoUv2 consistently outperforms other IoU-based loss functions, maintaining lower loss values and achieving faster convergence throughout the training process.

3.4. Model pruning and Comparative Analysis

To enhance the efficiency of the YOLOv8y-based apple detection model, we applied structured channel pruning and assessed its impact on both model complexity and detection performance. As shown in Fig. 9, the number of channels in each convolutional layer was significantly reduced after pruning. The yellow bars represent the original channel count, while the red bars indicate the reduced channels.

To compress Model 3 and improve inference efficiency, we adopted a pruning strategy based entirely on LAMP (Layer-Adaptive Magnitude-based Pruning). In the initial pruning stage, the number of parameters was reduced from 0.98 M to 0.611 M, FLOPs decreased from 4.4 G to 3.3 G, and the model's weight size shrank from 2.1MB to 1.5MB, resulting in an approximate 1.3 × speed-up in inference. However, this pruning led to a significant performance drop, with mAP50 dropping from 88.68 % to 75.33 %, and mAP50-95 falling from 66.01 % to 49.81 %. To recover the lost accuracy, we applied fine-tuning to the pruned model and explored two pruning approaches within the LAMP framework. First, VBP-YOLO-prune (No-global) applied LAMP in a layer-wise (non-global) manner. Fine-tuning effectively restored performance, achieving 88.61 % mAP50 and 66.0 % mAP50-95, with a moderate increase in parameters (0.76 M) and FLOPs (3.4 G). The second variant, VBP-YOLO-prune (Global), utilized the same LAMP method but evaluated channel importance across the entire network, enabling globally structured pruning. This approach led to further improvements: with fewer parameters (0.608 M) and lower FLOPs (3.2 G), it achieved 89.0 % mAP50 and 66.26 % mAP50-95. These results demonstrate that, under the same pruning framework, the global strategy provides superior accuracy and efficiency compared to the non-global approach.

The training dynamics under both pruning strategies are illustrated in Fig. 10, showing the mAP50 (yellow and blue) and mAP50-95 (red and green) curves over 100 training epochs. Global pruning results in a more stable training process, particularly during the early stages, and consistently achieves higher accuracy across both metrics. In contrast, non-global pruning exhibits greater fluctuations and converges to slightly lower final performance. These training curves further validate the effectiveness of our LAMP-based pruning framework and highlight the advantages of adopting a global pruning strategy.

3.5. Effect of different lightweight networks on model detection

This study primarily aims to enable lightweight model deployment. To this end, the YOLOv8n backbone was reconstructed using various lightweight architectures, with performance results summarized in Table 7. The tested backbones including Shufflenetv2[23], Vanillant[6], GhostNet [10], EfficientNetv2[37], and MobileNetv3, each aiming to reduce computational complexity and parameter count. Among all the tested models, VBP-YOLO-prune demonstrates the most competitive performance. It achieves mAP50 and mAP50–95 scores of 89.0 % and

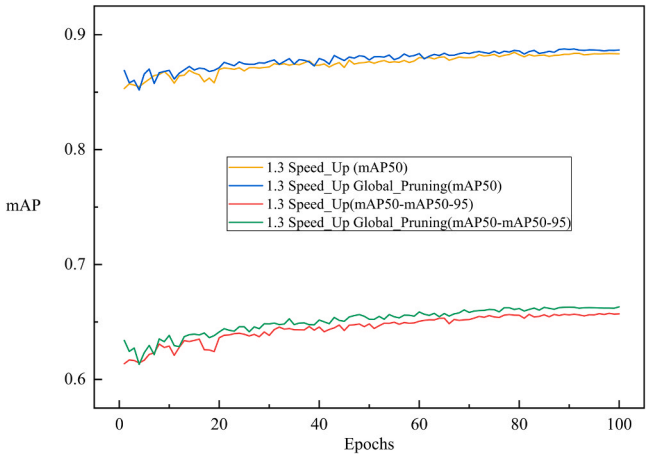


Fig. 10. mAP trained by finetune pruning.

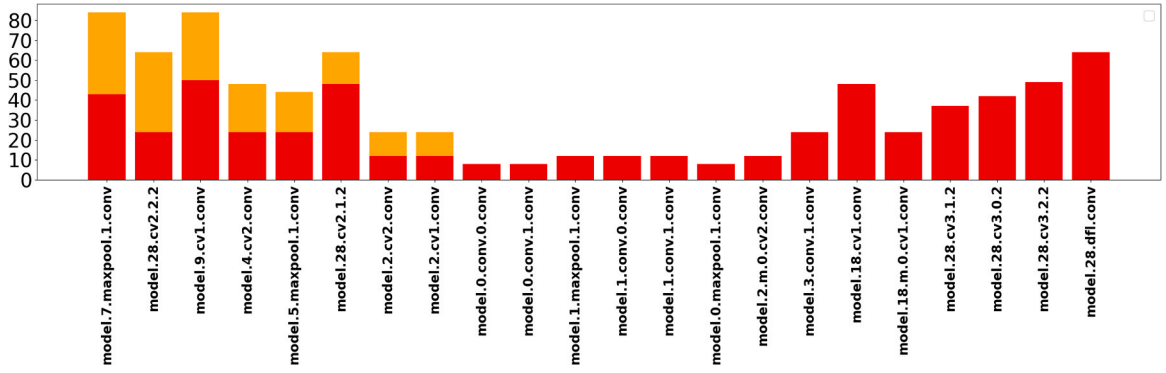


Fig. 9. Changes in channels before and after channel pruning.

Table 7
Comparative experiments of different lightweight backbones.

Model	P (%)	R (%)	mAP50 (%)	mAP50-95 (%)	Parameters(M)	FLOPs(G)	Weight (M)	FPS (S)
YOLOv8n-Shufflenetv2	82.91	79.11	86.99	61.71	1.83	5.1	3.9	85.1
YOLOv8n-Vanillant	82.66	78.97	86.71	61.01	1.73	5.1	3.7	87.3
YOLOv8n-GostNet	83.19	79.33	87.15	63.42	1.72	5.1	3.8	87.0
YOLOv8n-EfficientNetv2	82.73	78.78	85.87	60.76	2.12	2.6	4.5	88.9
YOLOv8n-MobileNetv3	83.15	79.45	87.11	61.95	2.92	7.1	6.0	77.8
VBP-YOLO-prune	84.01	80.52	89.0	66.26	0.61	3.2	1.5	102.6

Table 8
The 10-Fold cross-validation experiment of the VBP-YOLO-prune.

group	Train set	Val set	Test set	P (%)	R (%)	mAP50 (%)	mAP50-95 (%)
1	①②③④⑤⑥⑦⑧	⑨	⑩	84.01	80.52	**89.0**	**66.26**
2	②③④⑤⑥⑦⑧⑩	⑨	①	83.72	80.18	88.83	66.11
3	①③④⑤⑥⑦⑧⑩	⑨	②	83.78	80.24	88.85	66.16
4	①②④⑤⑥⑦⑧⑩	⑨	③	83.55	80.00	88.71	66.06
5	①②③⑤⑥⑦⑧⑩	⑨	④	83.89	80.31	88.90	66.21
6	①②③④⑥⑦⑧⑩	⑨	⑤	83.68	80.15	88.84	66.07
7	①②③④⑤⑦⑧⑩	⑨	⑥	83.74	80.22	88.87	66.10
8	①②③④⑤⑥⑧⑩	⑨	⑦	83.81	80.28	88.91	66.20
9	①②③④⑤⑥⑦⑩	⑨	⑧	83.61	80.14	88.77	66.07
10	①②③④⑤⑥⑦⑧	⑩	⑨	83.64	80.17	88.79	66.10
Ave.	/	/	/	83.75	80.22	88.85	66.14

Note: The bold font represents the optimal value of the indicator in the experiments.

66.26 %, respectively, significantly outperforming its counterparts. With only 0.61 M parameters, it substantially reduces both storage requirements and computational complexity. Although its FLOPs (3.2 G) are slightly higher, the low parameter count effectively controls the computational overhead. Moreover, its model size of just 1.5MB makes it particularly well-suited for deployment on resource-constrained platforms, such as edge devices and mobile systems. Compared to FPS, the proposed model is the fastest in this experiment. Additionally, VBP-YOLO-prune achieves a precision (P) of 84.01 % and recall (R) of 80.52 %, showcasing its superior accuracy in this lightweight setup.

3.6. The k-fold across-validation experiment of VBP-YOLO-prune

To rigorously assess the generalization capability of the proposed VBP-YOLO-prune model, a 10-fold cross-validation was performed on the entire dataset. The data were evenly split into ten subsets labeled ① through ⑩. In each fold, eight subsets were used for training, one for validation, and one for testing, ensuring that every sample was tested exactly once. This procedure provides a statistically robust evaluation of model performance.Table 8 summarizes four key evaluation metrics: Precision (P), Recall (R), mean Average Precision at an IoU threshold of 0.5 (mAP50), and mean Average Precision averaged over IoU thresholds from 0.5 to 0.95 (mAP50–95). The mAP50 scores ranged from 88.71 % (Fold 4) to 89.00 % (Fold 1), averaging 88.85 %. Correspondingly, mAP50–95 varied between 66.06 % and 66.26 %, with a mean of 66.14 %. Precision fluctuated narrowly from 83.55 % to 84.01 % (ave 83.75 %), while Recall ranged from 80.00 % to 80.52 % (ave 80.22 %). The small deviations between the maximum and average values—0.15 % for mAP50, 0.20 % for mAP50–95, 0.46 % for Precision, and 0.52 % for Recall—demonstrate the model's consistent detection performance across different data splits. These findings confirm that VBP-YOLO-prune maintains excellent stability and robustness, supporting its practical applicability in diverse real-world environments.

3.7. Comparison Experiments

To evaluate the performance of VBP-YOLO-prune with nine widely used object detection models: EfficientDet, YOLOv3-tiny, YOLOv5n, YOLOv7-tiny, YOLOv8n, YOLOv10n, YOLOv11n, and YOLOv12n. Table 9 shows key metrics: Precision (P), Recall (R), mAP50, mAP50–95, model parameters, FLOPs, model size, and inference speed (FPS).

VBP-YOLO-prune outperforms all models with the highest mAP50 (89.0 %) and mAP50–95 (66.26 %), while using only 0.61 M parameters and 3.2 GFLOPs. It also achieves the fastest FPS (102.6) and the smallest model size (1.5 MB). Its recission and recall (84.01 % and 80.52 %) surpass other models, including YOLOv11n and YOLOv12n, despite their similar accuracy. In contrast, YOLOv11n and YOLOv12n require significantly more parameters and FLOPs, yielding lower inference speeds. Other lightweight models like YOLOv3-tiny and YOLOv7-tiny have lower accuracy, while EfficientDet underperforms with the

Table 9
Comparative experiments of different models.

Model	P (%)	R (%)	mAP50 (%)	mAP50-95 (%)	Parameters(M)	FLOPs (G)	Weigh (M)	FPS (S)
EfficientDet	37.92	35.87	43.33	29.76	15.0	5.23	3.87	81.3
YOLOv3-tiny	79.22	78.38	84.49	54.81	8.67	13.0	17.5	66.7
YOLOv5n	82.76	79.53	88.08	62.1	1.76	4.2	3.9	90.8
YOLOv7-tiny	79.93	78.77	87.68	60.33	6.01	13.2	12.3	67.8
YOLOv8n	84.01	79.8	88.68	66.01	3.01	8.2	6.3	79.4
YOLOv10n	82.36	79.11	87.96	65.11	2.7	8.4	5.3	80.6
YOLOv11n	83.83	79.14	88.57	65.06	2.6	6.2	5.5	82.7
YOLOv12n	83.01	78.82	88.11	64.94	2.52	6.0	5.4	83.9
VBP-YOLO-prune	84.01	80.52	89.0	66.26	0.61	3.2	1.5	102.6

Table 10
Comparison of counting results for different models.

	YOLOv5n	YOLOv7-tiny	YOLOv8n	YOLOv10n	YOLOv11n	YOLOv12n	VBP-YOLO-prune
R^2	0.86	0.84	0.90	0.87	0.89	0.88	0.91
RMSE	4.04	4.11	3.86	3.97	3.89	3.95	3.75

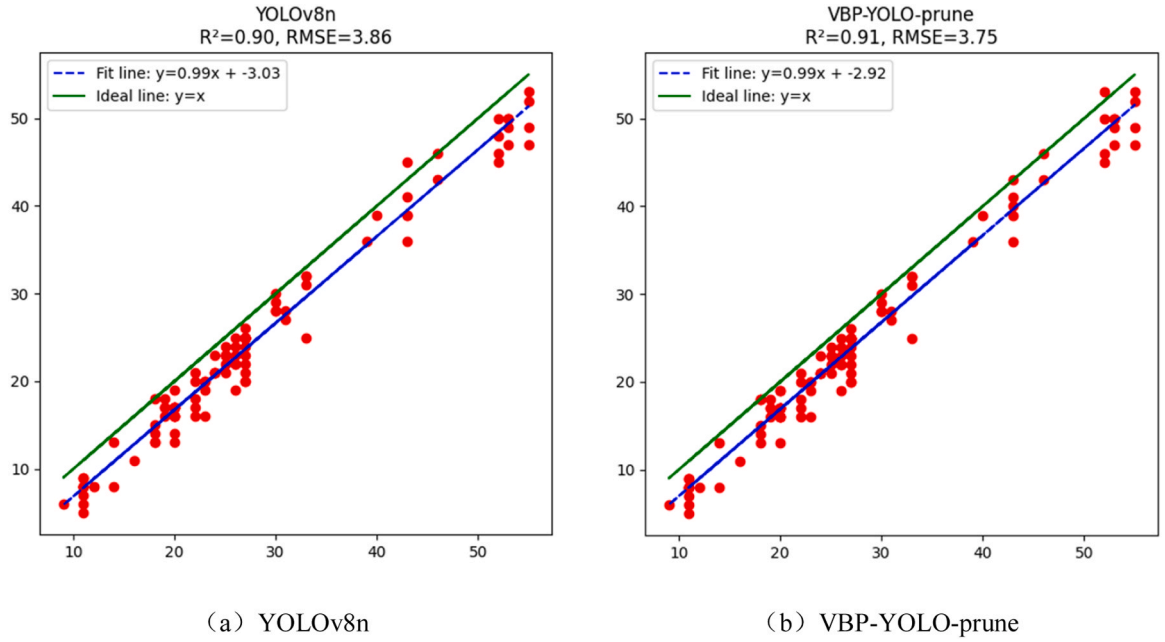


Fig. 11. Count regression curves of YOLOv8n and VBP-YOLO-prune models.

lowest mAP50 (43.33 %) and high complexity. These results show that VBP-YOLO-prune strikes a superior balance between accuracy, efficiency, and model size, making it ideal for real-time edge applications.

3.8. Model counts

To systematically evaluate apple counting accuracy, we conducted regression analysis on 100 images. Initial tests revealed that EfficientDet and YOLOv3-tiny performed poorly under complex conditions, leading to their exclusion from further evaluation. Table 10 summarizes key metrics for the selected models, highlighting the superior performance of VBP-YOLO-prune and YOLOv8n. Specifically, VBP-YOLO-prune achieved the highest accuracy with an R^2 of 0.91 and an RMSE of 3.75, slightly outperforming YOLOv8n (R^2 = 0.90, RMSE = 3.86). Models such as YOLOv5n and YOLOv7-tiny exhibited weaker performance, with R^2 values below 0.86 and RMSE above 4.0.

To clearly demonstrate the effectiveness of our optimization strategy, we compared YOLOv8n directly with its optimized variant, VBP-YOLO-prune, which was refined through pruning and parameter optimization. This direct comparison isolates the improvements specifically attributable to our strategy by eliminating confounding factors from differences in the underlying model architecture. Fig. 11 compares the regression results of the two models. YOLOv8n exhibited a slight underestimation (y = 0.99x - 3.03, R^2 = 0.90, RMSE = 3.86), whereas VBP-YOLO-prune showed better alignment with ideal conditions (y = 0.99x - 2.92, R^2 = 0.91, RMSE = 3.75). Both models used Non-Maximum Suppression (NMS) to effectively manage apple overlap and occlusion. Specifically, NMS eliminated redundant detections using an Intersection over Union (IoU) threshold of 0.5 and confidence scores above 0.25. Additionally, we introduced further post-processing rules to enhance counting accuracy, including a clustering-based verification step. In this step, closely overlapping detections were grouped and evaluated collectively, enabling the system to distinguish overlapping apples more accurately. Detection boxes with significant overlapping but distinct centroids were considered separate apples, while those with high overlapping and closely matched centroids were merged into a single count. These measures significantly improved the robustness of apple counting under challenging conditions. While the accuracy levels of both models are similar, VBP-YOLO-prune outperforms in terms of model efficiency, drastically reducing parameters, computational load, and overall model size. This highlights its practicality and effectiveness for deployment in orchard environments.

4. Discussion

4.1. Applicability analysis of the model

Based on the previous discussion, it is evident that the detection model used in this experiment demonstrates high accuracy. To further evaluate the model's robustness in detecting apples within complex environments, we conducted a comparative visual analysis. This analysis differentiates between correctly detected boxes and misses detection cases. It is important to note that this experiment simulates real-world scenarios of apple counting during picking, where challenges such as falling apples and capturing distant apples are inevitable. Therefore, these scenarios were included in our evaluation to comprehensively assess the model's effectiveness in handling diverse fruit detection conditions.

4.1.1. Apple detection in normal weather

As shown in Fig. 12, eight object detection models—EfficientDet, YOLOv3-Tiny, YOLOv5n, YOLOv7-Tiny, YOLOv8n, YOLOv10n, YOLOv11n, YOLOv12n, and the proposed VBP-YOLO-prune—were evaluated for apple detection under normal weather conditions, with an

Fig. 12. Apple detection in normal weather.

unannotated image for reference. EfficientDet detected only 7 apples and produced redundant boxes, indicating weak performance. YOLOv3-Tiny detected 14 apples but generated numerous overlapping predictions, resulting in high false positives. YOLOv5n through YOLOv10n also detected 14 apples each but missed those blending with foliage due to color similarity. YOLOv11n, YOLOv12n, and VBP-YOLO-prune detected 15 apples each. However, YOLOv12n had repeated detections, reducing localization precision. In contrast, VBP-YOLO-prune produced more accurate and compact bounding boxes, demonstrating superior localization and robustness in visually complex environments.

4.1.2. *Evening low-light conditions*

As shown in Fig. 13, the models were evaluated under dim or backlit conditions. EfficientDet detected only 8 apples, performing poorly due to the low contrast between the apples and background, leading to frequent missed detections. YOLOv3-Tiny detected 12 apples but

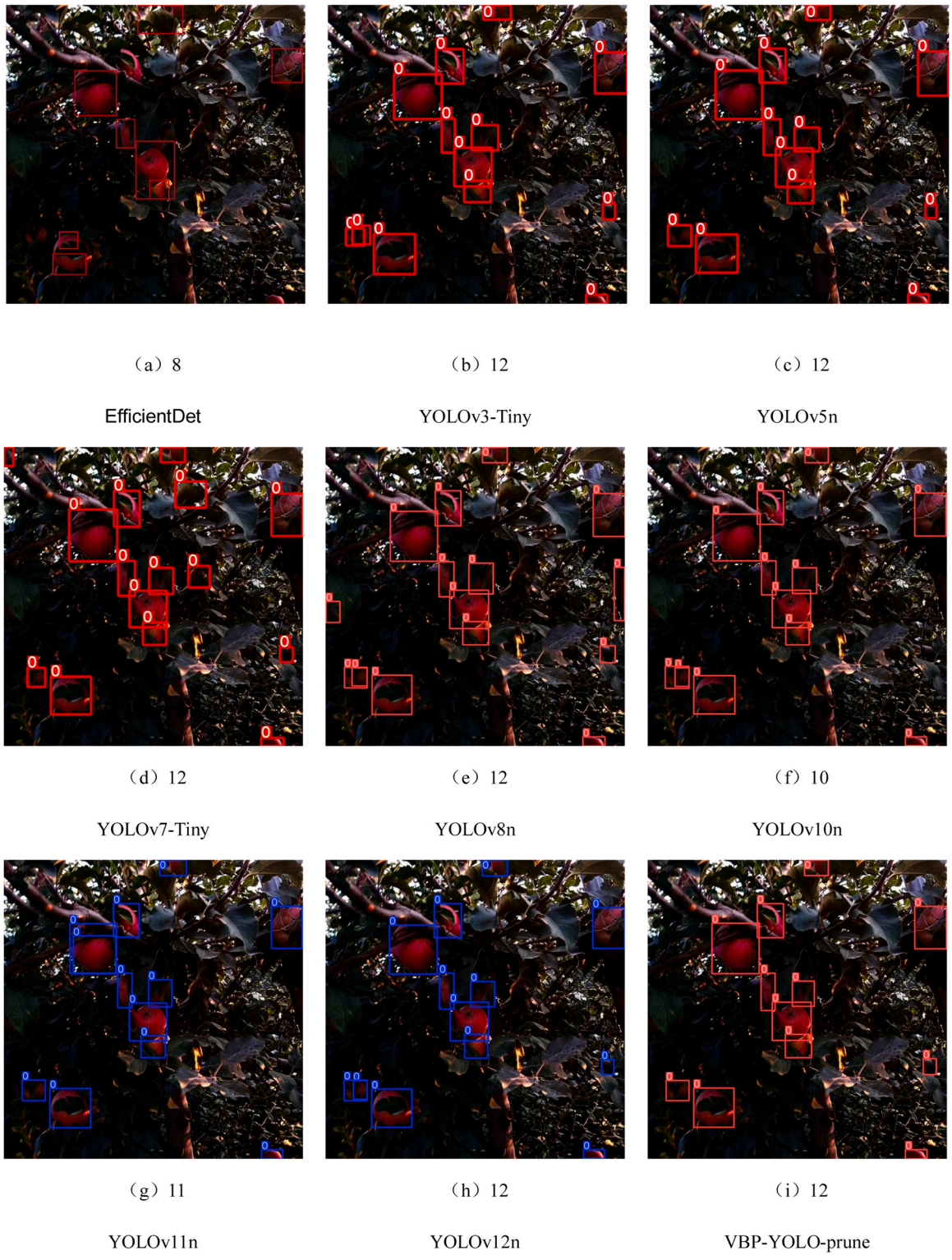


Fig. 13. Weak light in the evening.

generated overlapping boxes, increasing false positives. YOLOv5n, YOLOv7-Tiny, and YOLOv8n each detected 12 apples with improved performance, though errors persisted. YOLOv7-Tiny misclassified tree trunks as apples, while YOLOv8n had one duplicate and two missed detections. YOLOv10n detected 10 apples with minor inaccuracies, and YOLOv11n detected 11 apples, missing one due to poor lighting. YOLOv12n detected 12 apples but had one duplicate detection, reducing the localization precision. VBP-YOLO-prune achieved the highest detection count with more precise, compact bounding boxes, offering similar accuracy to YOLOv5n with a significantly lighter architecture, making it ideal for resource-limited environments.

4.1.3. Simulation of early morning bright light

As shown in Fig. 14, strong illumination and color similarity present challenges for all models. EfficientDet detects only 4 apples, indicating poor sensitivity. YOLOv3-Tiny detects 22 apples but suffers from

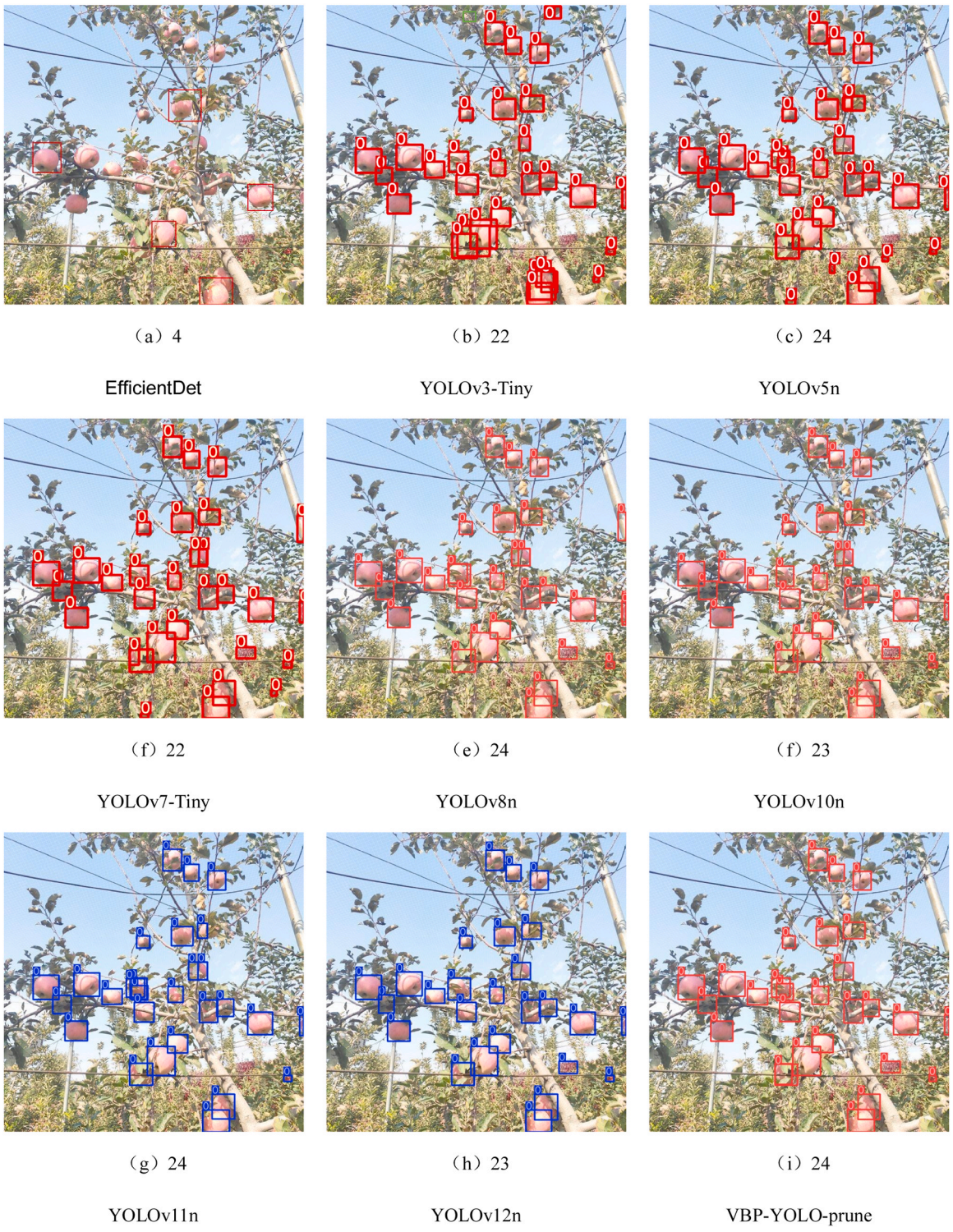


Fig. 14. simulates bright light in the early morning.

redundant boxes and poor discrimination in dense areas. YOLOv5n, YOLOv7-Tiny, and YOLOv8n detect 24 apples each, though the latter two misidentify metallic objects and duplicate occluded targets. YOLOv10n detects 23 apples with minor localization shifts. YOLOv11n matches the highest detection count but produces redundant and incomplete boxes. YOLOv12n detects 23 apples but misses one, with less compact bounding boxes compared to VBP-YOLO-prune. VBP-YOLO-prune delivers accurate, non-redundant detections with compact localization, demonstrating its robustness under high-contrast conditions.

4.1.4. Complex weather conditions, light fog and rain conditions

As shown in Fig. 15, all models were tested under light fog conditions. EfficientDet performed the worst, detecting only 6 apples due to low contrast and background blur. YOLOv3-Tiny and YOLOv5n each detected 17 apples, with one missing and one redundant detection. YOLOv7-Tiny and YOLOv10n also showed minor counting errors. YOLOv8n detected 17 apples, with partial redundancy and a missed target. YOLOv11n exhibited one missing and one redundant detection. YOLOv12n detected all 18 apples, but in one case two apples were

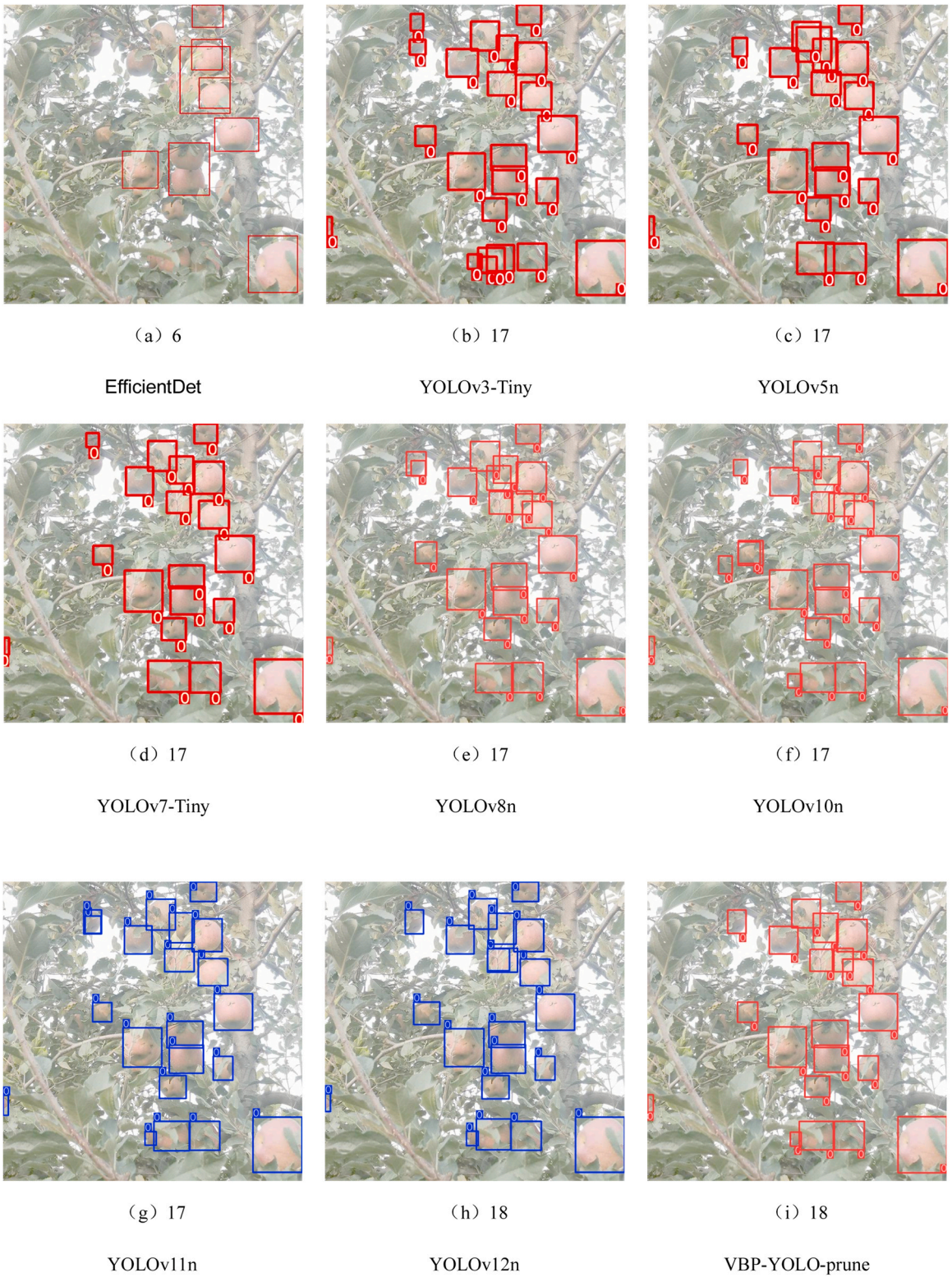


Fig. 15. Light fog in complex weather conditions.

enclosed by a single bounding box, slightly affecting localization precision. In contrast, VBP-YOLO-prune achieved the best result, detecting all 18 apples with precise and compact bounding boxes, highlighting its robustness in low-visibility environments.

As shown in Fig. 16, this experiment simulates light rain, reflecting realistic orchard conditions where harvesting is not possible during heavy rainfall. EfficientDet detects only 4 apples, struggling with low contrast and weather interference. YOLOv3-Tiny detects 41 apples but suffers from significant redundancy, indicating poor robustness in complex scenes. YOLOv5n, YOLOv7-Tiny, and YOLOv8n show moderate performance, with some missed or false detections. YOLOv10n performs well in occluding regions and is the only model to detect one extremely small apple, though it misses two targets overall, limiting its reliability. YOLOv11n detects 39 apples with some bounding box inconsistencies.

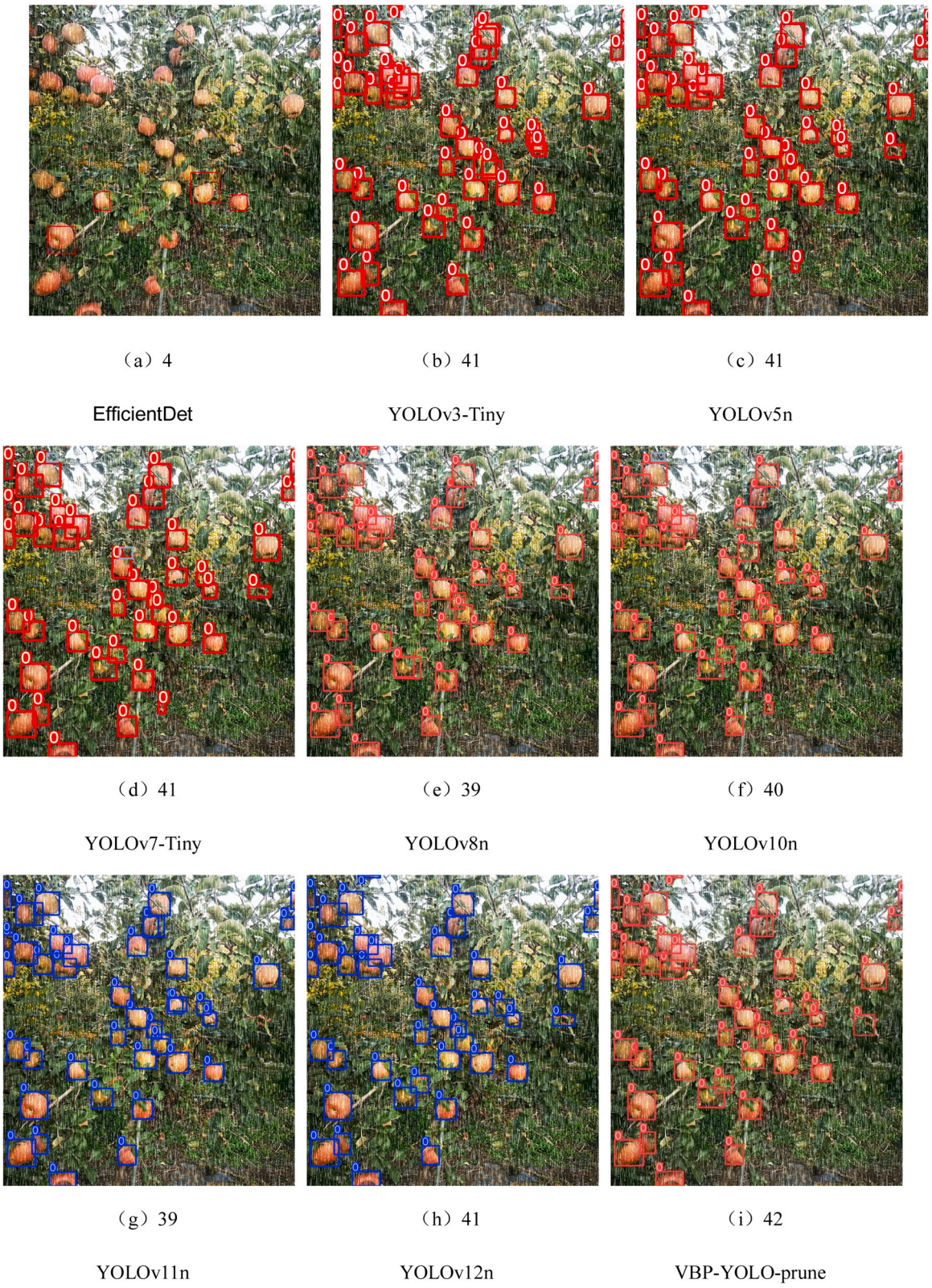


Fig. 16. Detection of rainy days in complex weather.

YOLOv12n detects 41 apples but misses one target, which is correctly identified by VBP-YOLO-prune. VBP-YOLO-prune achieves the best result, detecting 42 apples with only one minor miss. Its accurate and compact localization under rain-affected conditions demonstrates strong resilience and superior feature extraction capabilities.

4.2. Deployment of the improved YOLOv8 model

4.2.1. Introduction of NVIDIA Jetson Orin Nano

The NVIDIA Jetson Orin Nano is a high-performance AI embedded platform, featuring an Ampere-architecture GPU and a multi-core Arm

Table 11
Basic information and deployment environment of the NVIDIA Jetson Orin Nano.

Hardware/Software Environment	version
Development board	NVIDIA Jetson Orin Nano
CPU	6-core Cortex-A78AE, up to 1.5 GHz
GPU	Ampere architecture GA10B, 1024 CUDA cores, 32 Tensor Cores, up to 625 MHz
Memory	8 GB LPDDR4x
Interfaces	Camera interfaces, Type-C, DisplayPort, etc.
Power Consumption	15 W
Operating system	Ubuntu 20.04
python	3.8.18
torch	1.11.0
torchvision	0.12.0
ultralytics	8.2.83
timm	0.9.8
mmcv	2.0.1

processor. It includes dedicated accelerators for deep learning and computer vision, supports high-speed interfaces, and accommodates multiple sensor inputs with high memory bandwidth. Despite its compact size and low power consumption, it delivers up to 20 TOPS of AI inference performance. Due to its impressive performance and energy efficiency, the Jetson Orin Nano was chosen as the edge deployment platform for the enhanced YOLOv8 model in this study.

4.2.2. Deployment of improved YOLOv8 model

To evaluate the practical application value and operational performance of the optimized lightweight YOLOv8 model on resource-constrained devices, it was deployed on an NVIDIA Jetson Orin Nano. Leveraging the model's efficient design and the computing power of edge devices, this deployment achieves low-latency, real-time apple detection while maintaining accuracy. Table 11 details the deployment environment and configurations. The outcomes contribute to enhancing intelligent orchard management and providing reliable technical support for automated fruit detection and harvesting.

Experiments were conducted using an improved YOLOv8 model on the NVIDIA Jetson Orin Nano platform. Apple images were captured using a connected USB color camera in both indoor laboratory conditions and simulated orchard environments. Real-time apple detection results were displayed on the screen (as shown in Fig. 17), highlighting the model's bounding box accuracy, detection stability, and low inference latency. To evaluate deployment feasibility in real-world agricultural scenarios, the system was tested under various lighting conditions and occlusion situations. Thanks to its lightweight architecture and efficient feature fusion modules, the optimized YOLOv8 model demonstrated robust real-time performance on the edge device. This setup confirms the practical potential of the proposed detection system for resource-constrained edge computing platforms in precision agriculture applications, such as automated yield estimation and robotic fruit harvesting.

4.3. Verifying the generalizability of the model

To evaluate the generalization performance of the proposed model, we conducted experiments involving both intra-class variation and cross-category fruit types. For intra-class evaluation, the model was tested on a publicly available apple dataset [24], which is based on the University of Minnesota's apple dataset and includes augmented samples of the 'Honeycrisp' variety under various simulated weather conditions. As shown in Fig. 18(a–d), the model maintained stable detection performance under illumination variation and intra-class differences. While the overall localization accuracy remained satisfactory, instances of missed or redundant detections were observed, particularly in occluded or low-contrast regions.

For cross-category evaluation, the model was applied—without any retraining or fine-tuning—to fruit datasets containing citrus, guava [41, 48], and mangoes[9]. As shown in Fig. 18(e–h), the model was able to detect and localize fruits from these unseen categories, demonstrating a certain degree of generalization ability. However, the predicted bounding boxes exhibited reduced precision in cases involving small or partially occluded fruits. Future work will involve training with more diverse crops and environmental data and refining anchor settings and feature extraction to enhance detection accuracy and robustness under real-world conditions.

4.4. Discussion of Detection

Compared to recent lightweight YOLO variants, our VBP-YOLO-prune introduces structured channel pruning, which effectively reduces redundant parameters while maximizing the reduction of model size and computational cost without compromising detection accuracy. Specifically, PG-YOLO, YOLOv4-weeds[39], ASE-YOLOv8n[16] and Star-YOLO[22] achieve lightweight design by modifying the backbone network and incorporating attention mechanisms to reduce computational overhead on embedded platforms, whereas Faster-YOLO-AP adopts depth-wise separable convolutions and adjusts the width multiplier to improve efficiency. However, as these methods rely mainly on architectural modifications or attention-based enhancements, they often leave substantial redundancy in the model. Other approaches, such as YOLOv7-SAP[32], YOLO-TBD[20], and YOLOv8n-DWF[52], mainly improve detection accuracy under occlusion and complex backgrounds. However, these enhancements often come at the cost of increased model complexity and computational overhead, making them less suitable for

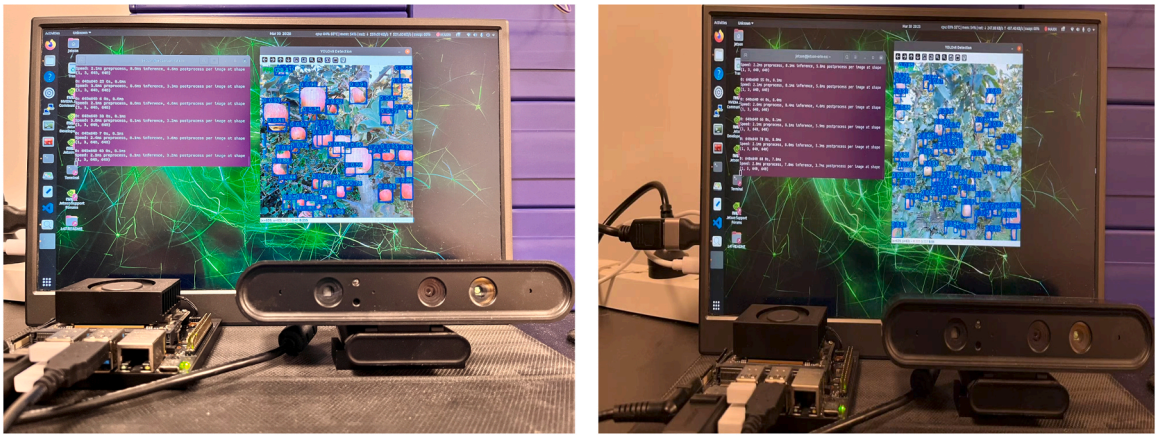


Fig. 17. Deployment diagram of the improved YOLOv8 model on the Jetson Orin Nano development board.

Fig. 18. Experiments for validating the generalization ability of the proposed model. (a–c) on varying weather conditions, (d) on a different apple variety (green apple), and (e–j) show results on different fruit types.

real-time inference on resource-constrained edge devices. To overcome these challenges, the proposed VBP-YOLO-prune integrates scaling factors, V7 downsampling, and BiFPN to enhance multi-scale extraction while maintaining structural efficiency. By incorporating the PIOUv2 loss function, the model also improves localization precision. Furthermore, structured channel pruning eliminates unnecessary parameters and computations, improving both inference speed and resource usage. Importantly, our training pipeline includes targeted data augmentation simulating fog, rain, and low-light conditions, which was not considered in the above-mentioned studies, including Faster-YOLO-AP and YOLOv7-SAP, both of which also focus on apple detection. This allows our model to achieve superior robustness in real-world orchard environments under variable weather conditions. Finally, VBP-YOLO-prune is deployed on the NVIDIA Jetson Orin Nano, demonstrating real-time processing capability and validating its effectiveness for edge applications.

While the model performs well in experimental environments, challenges remain in practical deployment. Hardware limitations, such as processing speed when handling high-resolution images or large datasets, may cause delays. Additionally, environmental factors like lighting, weather, and object occlusion can affect accuracy and stability. Future work will focus on broader testing in real-world environments, particularly integrating the model into intelligent picking robots for large-scale automated fruit harvesting. Despite these challenges, VBP-YOLO-prune has achieved significant results in apple detection, but further optimization is needed, especially in detecting small targets at long distances and distinguishing objects with similar colors. Future research will aim to enhance small target detection and improve accuracy in varying distances, resolutions, and complex backgrounds.

5. Conclusion

This study presents a lightweight and high-performance apple detection model, VBP-YOLO-prune, designed for complex orchard environments. A diverse dataset was constructed using data augmentation to simulate various weather conditions, enhancing the model's robustness. Based on YOLOv8n, the model integrates an optimized YOLOv8y backbone, V7 downsampling, BiFPN, and the PIOUv2 loss function, combined with structured pruning to reduce complexity. Experimental results demonstrated that VBP-YOLO-prune achieved 84.01 % Precision and 80.52 % Recall, along with 89.0 % mAP50 and 66.26 % mAP50–95, with only 0.61 M parameters, 3.2 GFLOPs, and a 1.5 MB model size—representing reductions of 79.7 % in parameters and 61 % in computation compared to YOLOv8n.

Experimental results demonstrated that VBP-YOLO-prune achieved 89.0 % mAP50 and 66.26 % mAP50–95, with only 0.61 M parameters, 3.2 GFLOPs, and a 1.5 MB model size—representing reductions of 79.7 % in parameters and 61 % in computation compared to YOLOv8n. A 10-fold cross-validation confirmed the model's stability. Further comparisons showed that VBP-YOLO-prune outperforms recent models such as YOLOv11n and YOLOv12n, offering superior accuracy and lower resource demands. The model was successfully deployed on the NVIDIA

Jetson Orin Nano, achieving real-time inference under edge computing constraints. It also exhibited strong generalization ability when tested on unseen apple varieties and other fruits (e.g., guavas, oranges, mangoes) without retraining. While the proposed model demonstrates excellent performance in controlled experiments, further research is required to extend its applicability under real-world agricultural conditions.

Future work will focus on enhancing domain robustness by leveraging unsupervised domain generalization and continual learning techniques, enabling the model to adapt across different orchard types, crop varieties, and imaging devices without extensive retraining. In addition, we aim to investigate multi-task perception frameworks that unify object detection with tasks such as disease identification, fruit ripeness estimation, and yield prediction, promoting a more comprehensive intelligent orchard system. To facilitate deployment across a broader range of embedded hardware, including low-power platforms such as Raspberry Pi and RK3588, we will explore lightweight architecture search, quantization-aware training, and knowledge distillation strategies. Furthermore, integration into autonomous orchard robots will be considered, enabling end-to-end systems capable of detection, decision-making, and path planning under unstructured environmental conditions.

Ethics and Consent Statement

This study did not involve human participants, animal experiments, and did not collect any data that required ethical approval, so ethical review and participant consent are not applicable.

Consent Statement

This study does not address any material or content that requires specific consent for publication and therefore this statement does not apply.

CRediT authorship contribution statement

Hao Wang: Investigation, Funding acquisition. **Zhanchen Wei:** Formal analysis. **Haohai You:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Xuefang Li:** Supervision, Software. **Yingying Yin:** Supervision, Software, Project administration, Conceptualization. **Chunguang Bi:** Funding acquisition, Formal analysis, Conceptualization. **Lijuan Zhang:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could affect the work reported herein.

The authors declare the following financial interests/personal relationships that may be considered as potential competing interests

Data availability

The dataset and source code used in this study are publicly available to support reproducibility and facilitate further research. The apple detection dataset includes images collected under various environmental conditions, including rain, low light, and strong sunlight, and is enhanced to reflect the real-world variability of orchard ecosystems. The source code corresponds to the proposed VBP-YOLO-prune model, which is optimized for lightweight deployments on edge devices. All resources can be accessed via the following links:

(https://pan.baidu.com/s/1-kQMdLMa8pyLmBhNQO-Eqg?pwd=c5tf)

References

[1] Y. Bai, J. Yu, S. Yang, J. Ning, An improved YOLO algorithm for detecting flowers and fruits on strawberry seedlings, Biosyst. Eng. 237 (2024) 1–12, https://doi.org/10.1016/j.biosystemseng.2023.11.008.

[2] H. Bilal, Y. Tian, I. Ullah, S. Garg, B.J. Choi, M.M. Hassan, M-CNN-RF: a hybrid deep learning model for accurate pediatric skeletal age estimation using hand bone radiographs, Alex. Eng. J. 129 (2025) 289–301, https://doi.org/10.1016/j.aej.2025.05.090.

[3] X. Chen, T. Liu, K. Han, X. Jin, J. Wang, X. Kong, J. Yu, TSP-yolo-based deep learning method for monitoring cabbage seedling emergence, Eur. J. Agron. 157 (2024) 127191, https://doi.org/10.1016/j.eja.2024.127191.

[4] W. Chen, M. Liu, C. Zhao, X. Li, Y. Wang, MTD-YOLO: Multi-task deep convolutional neural network for cherry tomato fruit bunch maturity detection, Comput. Electron. Agric. 216 (2024) 108533.

[5] J. Chen, H. Mai, L. Luo, X. Chen, K. Wu, Effective feature fusion network in BIFPN for small object detection, in: IEEE international conference on image processing (ICIP), 2021, IEEE, 2021, pp. 699–703.

[6] H. Chen, Y. Wang, J. Guo, D. Tao, Vanillanet: the power of minimalism in deep learning, Adv. Neural Inf. Process. Syst. 36 (2024).

[7] Gevorgyan, Z., 2022. SIoU loss: More powerful learning for bounding box regression. arXiv preprint arXiv:2205.12740.

[8] Y. Gong, X. Yu, Y. Ding, X. Peng, J. Zhao, Z. Han, Effective fusion factor in FPN for tiny object detection, Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (2021) 1160–1168.

[9] Z. Gu, D. He, J. Huang, J. Chen, X. Wu, B. Huang, T. Dong, Q. Yang, H. Li, Simultaneous detection of fruits and fruiting stems in mango using improved YOLOv8 model deployed by edge device, Comput. Electron. Agric. 227 (2024) 109512, https://doi.org/10.1016/j.compag.2024.109512.

[10] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, C. Xu, Ghostnet: more features from cheap operations, proceedings of, IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (2020) 1580–1589.

[11] W. Jia, Y. Tian, R. Luo, Z. Zhang, J. Lian, Y. Zheng, Detection and segmentation of overlapped fruits based on optimized mask R-CNN application in apple harvesting robot, Comput. Electron. Agric. 172 (2020) 105380, https://doi.org/10.1016/j.compag.2020.105380.

[12] Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., Kwon, Y., Michael, K., Fang, J., Yifu, Z., Wong, C., Montes, D., 2022. ultralytics/yolov5: v7. 0-yolov5 sota realtime instance segmentation. Zenodo.

[13] A. Kasthuri, A. Suruliandi, E. Poongothai, S.P. Raja, Deep Learning-based texture feature extraction technique for face annotation, Int. J. Pattern Recognit. Artif. Intell. 39 (03n04) (2025) 2532001, https://doi.org/10.1142/S0218001425320015.

[14] L. Kuang, Z. Wang, Y. Cheng, H. Li, J. Li, Y. Shen, J. Zhang, G. Xu, Cultivar and origin authentication of 'Fuji' and 'gala' apples from two dominant origins of China based on quality attributes, Food Chem. X 23 (2024) 101643, https://doi.org/10.1016/j.fochx.2024.101643.

[15] H. Li, Z. He, Y. Wang, X. Ding, Y. Cui, Research on the mechanized harvesting strategy for clustered kiwi fruits based on deep reinforcement learning, Comput. Electron. Agric. 237 (2025) 110686, https://doi.org/10.1016/j.compag.2025.110686.

[16] X. Liang, H. Jia, H. Wang, L. Zhang, D. Li, Z. Wei, H. You, X. Wan, R. Li, W. Li, M. Yang, ASE-YOLOv8n: a method for cherry tomato ripening detection, Agronomy 15 (5) (2025) 1088.

[17] Z. Liu, R.M. Rasika D. Abeyrathna, R. Mulya Sampurno, V. Massaki Nakaguchi, T. Ahamed, Faster-YOLO-AP: a lightweight apple detection algorithm based on improved YOLOv8 with a new efficient PDWConv in orchard, Comput. Electron. Agric. 223 (2024) 109118, https://doi.org/10.1016/j.compag.2024.109118.

[18] C. Liu, K. Wang, Q. Li, F. Zhao, K. Zhao, H. Ma, Powerful-IoU: more straightforward and faster bounding box regression loss with a nonmonotonic focusing mechanism, Neural Netw. 170 (2024) 276–284, https://doi.org/10.1016/j.neunet.2023.11.041.

[19] J. Liu, Y. Zhou, Smart fitness with YOLO-Fit IoT: Real-time pose analysis and personalized training via IoT and RL, Alex. Eng. J. 129 (2025) 216–225, https://doi.org/10.1016/j.aej.2025.05.068.

[20] Z. Liu, L. Zhuo, C. Dong, J. Li, YOLO-TBD: tea bud detection with Triple-Branch attention mechanism and Self-Correction group convolution, Ind. Crops Prod. 226 (2025) 120607, https://doi.org/10.1016/j.indcrop.2025.120607.

[21] F. Long, L. Li, Y. Liu, H. Chen, Y. Zhang, H. Wang, A lightweight detection algorithm for camellia oleifera buds, stamens, and flowers using you only look once with large selective kernel network, Eng. Appl. Artif. Intell. 158 (2025) 111275, https://doi.org/10.1016/j.engappai.2025.111275.

[22] Z. Lu, Z. Chengao, L. Lu, Y. Yan, W. Jun, X. Wei, X. Ke, T. Jun, Star-YOLO: a lightweight and efficient model for weed detection in cotton fields using advanced YOLOv8 improvements, Comput. Electron. Agric. 235 (2025) 110306, https://doi.org/10.1016/j.compag.2025.110306.

[23] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, Shufflenet v2: practical guidelines for efficient cnn architecture design, Proc. Eur. Conf. Comput. Vis. (ECCV (2018) 116–131.

[24] L. Ma, L. Zhao, Z. Wang, J. Zhang, G. Chen, Detection and counting of small target apples under complicated environments by using improved YOLOv7-tiny, Agronomy 13 (5) (2023) 1419.

[25] S. Maruthai, R. Selvanarayanan, T. Thanarajan, S. Rajendran, Hybrid vision GNNs based early detection and protection against pest diseases in coffee plants, Sci. Rep. 15 (1) (2025) 11778, https://doi.org/10.1038/s41598-025-96523-4.
[26] E.L. Pereira, W.C. Celeste, L.J. Silvestre, J.F. Fardin, H.R.O. Rocha, Deep learning for multi-class electrical load detection in smart X-energy systems on edge computing devices, Expert Syst. Appl. 292 (2025) 128461, https://doi.org/10.1016/j.eswa.2025.128461.
[27] H. Ramamoorthy, M. Ramasundaram, S.P. Raja, K. Randive, An efficient classification of multiclass brain tumor image using hybrid artificial intelligence with honey bee optimization and probabilistic U-RSNet, Int. J. Image Graph. 25 (01) (2023) 2450059, https://doi.org/10.1142/S0219467824500591.
[28] M.L. Rao, V.S. Rajapu, M.V.S. Ramprasad, D. Manasa, M. Swetha, Development of image-based automatic disease symptom detection model for onion downy mildew using multi-dilated efficient attention network, Expert Syst. Appl. (2025) 128720, https://doi.org/10.1016/j.eswa.2025.128720.
[29] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, S. Savarese, Generalized intersection over union: a metric and a loss for bounding box regression, Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (2019) 658–666.
[30] R. Selvanarayanan, S. Rajendran, Y. Alotaibi, Early detection of colletotrichum kahawae disease in coffee cherry based on computer vision techniques, Comput. Model. Eng. \ Sci. 139 (1) (2024) 759–782.
[31] R. Selvanarayanan, R. S, T. G, L. K, Hybrid vision transformer and CNN for detection of overripe coffee berry disease (OCBD) in coffee plantation, 2024 International Conference Emerging Research Computational Science (ICERCS) (2024) 1–7.
[32] B. Shi, C. Hou, X. Xia, Y. Hu, H. Yang, Improved young fruiting apples target recognition method based on YOLOv7 model, Neurocomputing 623 (2025) 129186, https://doi.org/10.1016/j.neucom.2024.129186.
[33] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning, J. big data 6 (1) (2019) 1–48.
[34] T. Sidhaarth, P.A. Obla, N.N. Patil, Z. Stamenkovic, S.P. Raja, Enhanced miss forest and multivariate time series prediction of wind speed using deep learning, J. Circuits Syst. Comput. (2025) 2530006, https://doi.org/10.1142/S0218126625300065.
[35] M. Sohan, T. Sai Ram, R. Reddy, C. Venkata, A review on yolov8 and its advancements. International Conference on Data Intelligence and Cognitive Informatics, Springer, 2024, pp. 529–545.
[36] P. Sree, R. S, R. Selvanarayanan, Leveraging CNN and AlexNet algorithms for improved coffee leaf disease identification and detection, 2024 First Int. Conf. Innov. Commun. Electr. Comput. Eng. (ICICEC) (2024) 1–6.
[37] M. Tan, Q. Le, Efficientnetv2: smaller models and faster training, Int. Conf. Mach. Learn. PMLR (2021) 10096–10106.
[38] Y. Tianjing, M. Mhamed, Developments in automated harvesting equipment for the apple in the orchard: review, Smart Agric. Technol. 9 (2024) 100491, https://doi.org/10.1016/j.atech.2024.100491.
[39] H. Wan, S. Wang, Weed detection in cornfields based on improved lightweight neural network model, Alex. Eng. J. 122 (2025) 334–343, https://doi.org/10.1016/j.aej.2025.03.016.
[40] D. Wang, D. He, Channel pruned YOLO V5s-based deep learning approach for rapid and accurate apple fruitlet detection before fruit thinning, Biosyst. Eng. 210 (2021) 271–281.
[41] J. Wang, M. Liu, Y. Du, M. Zhao, H. Jia, Z. Guo, Y. Su, D. Lu, Y. Liu, PG-YOLO: an efficient detection algorithm for pomegranate before fruit thinning, Eng. Appl. Artif. Intell. 134 (2024) 108700.
[42] J. Wu, T. Zhong, G. Li, Y. Lan, W. Xiong, Tree species identification based on convolutional neural network and feature fusion, Comput. Electron. Agric. 237 (2025) 110691, https://doi.org/10.1016/j.compag.2025.110691.
[43] Y. Xing, X. Zhang, Z. Feng, W. Ni, H. Xie, Y. Guan, Z. Zhu, S. Ge, Y. Jiang, Optimizing 'red Fuji' apple quality: Auxin-mediated calcium distribution via fruit-stalk in bagging practices, Food Chem. (2024) 141126, https://doi.org/10.1016/j.foodchem.2024.141126.
[44] B. Yan, P. Fan, X. Lei, Z. Liu, F. Yang, A Real-Time apple targets detection method for picking robot based on improved YOLOv5, Remote Sens. 13 (9) (2021) 1619.
[45] J. Yu, X. Liu, W. Wang, L. Zhang, C. Wang, Q. Zhang, J. Wang, M. Du, L. Sheng, D. Hu, MdCIbHLH1 modulates sugar metabolism and accumulation in apple fruits by coordinating carbohydrate synthesis and allocation, Hortic. Plant J. (2024), https://doi.org/10.1016/j.hpj.2024.01.007.
[46] Y.-F. Zhang, W. Ren, Z. Zhang, Z. Jia, L. Wang, T. Tan, Focal and efficient IOU loss for accurate bounding box regression, Neurocomputing 506 (2022) 146–157.
[47] Zhang, H., Zhang, S., 2023. Shape-iou: More accurate metric considering bounding box shape and scale. arXiv preprint arXiv:2312.17663.
[48] J. Zhao, R. Almodfer, X. Wu, X. Wang, A dataset of pomegranate growth stages for machine learning-based monitoring and analysis, Data Brief. 50 (2023) 109468, https://doi.org/10.1016/j.dib.2023.109468.
[49] Y. Zheng, Y. Cui, X. Gao, An infrared Dim-small target detection method based on improved YOLOv7, Proc. 2023 Asia Conf. Comput. Vis. Image Process. Pattern Recognit. (2023) 1–5.
[50] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, D. Ren, Distance-IoU loss: faster and better learning for bounding box regression, Proc. AAAI Conf. Artif. Intell. (2020) 12993–13000.
[51] X. Zhu, F. Chen, Y. Zheng, X. Peng, C. Chen, An efficient method for detecting camellia oleifera fruit under complex orchard environment, Sci. Hortic. 330 (2024) 113091, https://doi.org/10.1016/j.scienta.2024.113091.
[52] Y. Zhu, S. Sui, W. Du, X. Li, P. Liu, Picking point localization method of table grape picking robot based on you only look once version 8 nano, Eng. Appl. Artif. Intell. 146 (2025) 110266, https://doi.org/10.1016/j.engappai.2025.110266.