Smart Agricultural Technology 12 (2025) 101068

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.journals.elsevier.com/smart-agricultural-technology

FruitQuery: A lightweight query-based instance segmentation model for in-field fruit ripeness determination

Ziang Zhao [a,*], Yulia Hicks [a], Xianfang Sun [b], Chaoxi Luo [c]

[a] School of Engineering, Cardiff University, Cardiff, CF243AA, Wales, United Kingdom
[b] School of Computer Science and Informatics, Cardiff University, Cardiff, CF244AG, Wales, United Kingdom
[c] College of Plant Science and Technology, Huazhong Agricultural University, Wuhan, 430070, China

ARTICLE INFO

Dataset link: ripeness_info

Keywords:
Query-based
Lightweight
Segmentation
Fruit
Ripeness

ABSTRACT

Accurate fruit instance segmentation at different ripeness stages is critical for developing autonomous harvesting robots, particularly given the unstructured in-field conditions. In this paper, we combine two in-field fruit datasets of peaches and strawberries for multiple ripeness stages determination, and propose a lightweight query-based instance segmentation model named FruitQuery.

The combined dataset contains 3 peach ripeness stages and 4 strawberry ripeness stages, covering various unstructured conditions of two popular fruits. The model FruitQuery consists of three parts: a backbone, a pixel decoder and Transformer decoders. Efficient multi-head self-attention modules are introduced to the backbone to reduce computational overhead, and a pyramid pooling module is added to the pixel decoder to enhance multi-scale feature fusion. Transformer decoders are then applied to learn a fixed number of queries from features and generate instance masks, avoiding postprocessing like non-maximum suppression. FruitQuery runs in an end-to-end way and incorporates the convolution and Transformer to capture fine-grained features related to different fruits at different ripeness stages.

Extensive experiments on the combined fruit dataset demonstrate that our FruitQuery achieves the highest average precision of 67.02 with only 14.08M parameters, outperforming 13 state-of-the-art models with 33 variants. It is noted that FruitQuery surpasses three series of YOLO (v8, v9 and v10) by a large margin. Ablation studies and visualizations also show its robust feature extraction with fewer parameter usage, indicating that the query-based design is effective in localizing fruit. These results highlight FruitQuery's compelling balance between segmentation performance and model size, offering the potential for in-field application.

1. Introduction

1.1. Background

Rapid developments in artificial intelligence have made digitization, precision, and smart farming key elements in driving the modernization of agriculture. In the context of fruit automation production, the efficient and precise location of the target fruit serves as the foundational prerequisite to building a fruit-picking robot, which senses the working surroundings and guides the robotic arm to detach the fruit. In recent years, various harvesting robots have been developed for fruits and crops, such as kiwifruit [35], apple [23], litchi [29,21], strawberry [17], green citrus [31], radiata pine [33], sweet pepper [37] and asparagus spear [39].

Various computer vision tasks, such as classification, detection and segmentation, have been applied in the agricultural sector. Segmentation is one of the most promising domains for practical implementation, and it has been widely used in various applications because it provides accurate pixel-level prediction for objects. Segmentation methods include semantic segmentation and instance segmentation. Specifically, instance segmentation assigns different labels for individual instances belonging to the same category, which is relatively more resource-consuming and complicated than semantic segmentation [13]. Segmentation has been applied to various agricultural applications like mango yield estimation [24], leaf disease analysis [10], plant growth monitoring [48] and automatic fruit picking [28].

In terms of fruit detection and segmentation, the models can be mainly divided into Convolutional Neural Network (CNN) based models and Transformer-based models.

* Corresponding author.
E-mail address: zhaoz60@cardiff.ac.uk (Z. Zhao).

https://doi.org/10.1016/j.atech.2025.101068
Received 17 February 2025; Received in revised form 18 May 2025; Accepted 1 June 2025
Available online 5 June 2025

1.2. CNN-based models

CNN-based models have dominated vision architectures for a long time, of which Mask R-CNN [14] is the pioneer of instance segmentation. Santos et al. [42] showcased the effectiveness of Mask R-CNN in detecting, segmenting, and tracking grape clusters, demonstrating its robust performance across significant variations in shape, colour, size, and compactness. A Mask R-CNN model was trained and tested to detect and segment individual blueberries of four cultivars to two stages of maturity [36]. Similarly, Mask R-CNN and its modified version are adopted to build strawberry fruit detectors with better universality and robustness than traditional machine vision algorithms [58,40]. Jia et al. [19] improved Mask R-CNN by adopting ResNet and DenseNet as the feature extraction backbone to construct a picking robot vision detector, which significantly improves the recognition accuracy of apples in the environment of overlaps, agglomerations and occlusions. Besides, some other CNN-based instance segmentation also emerged. A fast segmentation model for green fruits called FoveaMask was constructed by introducing a position attention module to the embedding mask branch to aggregate pixels containing valuable information and enhance robustness capability [20]. Kang and Chen [23] proposed DaSNet-v2 which combines an instance and a semantic segmentation branch into the one-stage detection network, which can perform fruit instance segmentation and branches semantic segmentation. Sheng et al. [44] designed an edge-guided fruit segmentation model, which included modules specially designed to locate potential target areas and sharpen the edges.

It is noted that You Only Look Once (YOLO) series models have been widely adopted in fruit detection, classification, and ripeness determination over the past few years. For instance, Yang et al. [55] introduced LS-YOLOv8s by integrating YOLOv8s with an LW-Swin Transformer module to detect and grade strawberry ripeness. Xiao et al. [53] applied YOLOv8 to classify apple ripeness into three stages. In addition, He et al. [17] proposed a real-time improved YOLOv5s for robotic strawberry harvesting, Yu et al. [56] introduced Stolon-YOLO to detect strawberry stolons in greenhouse environments, and Chen et al. [4] combined fusion clustering with YOLOv5 to tackle *Camellia oleifera* fruit detection under multiple occlusions. Further extensions include Zhu et al. [62]'s modified lightweight YOLO for *C. oleifera* fruit maturity assessment in orchards, Ren et al. [41]'s YOLO-RCS for detecting the phenological period of Yuluxiang pears, [43]'s multi-scale adaptive YOLO for grape pedicel instance segmentation, and Ma et al. [32]'s STRAW-YOLO for identifying strawberries and their key points. These studies reflect the popularity of YOLO models across diverse fruit applications.

1.3. Transformer-based models

Recently, Vision Transformer (ViT, [9]) have received growing research attention, which is based on sequence-to-sequence prediction. Compared to the widely-used CNNs in visual perception, ViTs enjoy great flexibility in modelling long-range dependencies in vision tasks and introduce less inductive bias. Thus, it is expected that ViTs are likely to replace or combine with CNNs and serve as the basic component in the next-generation visual perception system. Based on the vanilla ViT, several successive models like DETR [2], MaskFormer [6] and SegFormer [54] etc have arisen.

Transformer models have been combined with CNN-based models to conduct segmentation. Niu et al. [38] proposed a semantic segmentation model called HSI-TransUNet for crop mapping, which could make full use of the abundant spatial and spectral information of UAV HSI data simultaneously. Wang et al. [49] proposed a parallel network structure DualSeg, which leverages the advantages of CNN at local processing and Transformer at global interaction for grape peduncle segmentation. Guo et al. [12] presented a Convolutional version of Swin Transformer to recognize the degree and kind of disease using a convolutional design. A novel Transformer-based CNN model MTYOLOX, is introduced for robustly detecting full tree inflorescences in the uncontrolled and challenging orchard environment [52].

Besides, the Transformer models also have demonstrated good performance in relevant agricultural applications. Sun et al. [45] presented a focal bottleneck Transformer network FBoT-Net to incorporate high-level semantic information with strong representation ability and global and local feature information through the focal bottleneck Transformer module for small green apple detection. Thai et al. [47] introduced a Transformer-based leaf disease detection model, namely FormerLeaf along with the Least Important Attention Pruning algorithm to select the most important attention heads of each layer in the Transformer model. An improved Transformer-based strawberry disease identification method was proposed to achieve precise and fast recognition of multiple classes of strawberry diseases [26]. He et al. [16] proposed a two-stream cross-attention ViT to extract texture appearance and spatial structure for regressing pig weight based on both RGB and depth images.

1.4. Contributions

Despite significant interest in Transformer-based models, most current related research in agriculture focused on CNN-based models, of which YOLO is the most frequently used model. There is a distinct lack of exploration into Transformer-based models for fruit instance segmentation and ripeness determination. This gap is critical given the practical need for accurate and robust methods in complex orchard environments. Meanwhile, ensuring lightweight and efficient segmentation remains a major challenge, as many in-field robotic platforms have strict limitations on computational resources and power.

Furthermore, publicly available fruit datasets that include both instance-level masks and ripeness annotations are difficult to obtain. Although some datasets provide segmentation labels, they rarely include labels on ripeness stages, hindering progress in the multi-stage evaluation of ripeness.

To address these issues, we release a combined fruit dataset and propose a lightweight Transformer-based instance segmentation model for fruit ripeness determination. In the following, we highlight our main contributions.

1. The strawberry dataset StrawDI_Db1 was annotated with 4 ripeness stages; the ripeness labels are now publicly available.
2. The NinePeach and StrawDI_Db1 datasets were merged to support multi-fruit segmentation in complex real-world scenarios.
3. FruitQuery is a lightweight query-based model combining CNN and Transformer features for end-to-end ripeness segmentation.
4. FruitQuery achieves 67.02 AP with only 14.08M parameters, surpassing 13 other models, including three YOLOs (v8, v9, v10).

2. Datasets

2.1. Overview

In this paper, we combined two public fruit datasets, NinePeach dataset [60] and StrawDI_Db1 dataset [40] to form a unified benchmark for fruit instance segmentation. Sample images are shown in Fig. 1. Both datasets provide pixel-wise individual annotation masks for every single fruit shown in the image.

By merging a tree-fruit (peach) and a berry-fruit (strawberry), the dataset spans diverse canopy structures, occlusion patterns, and background textures. This variety offers a more challenging and comprehensive setting for segmentation models, as they must adapt to different orchard conditions and fruit morphologies.

NinePeach dataset. This dataset is from our previous work [60], which is the largest and most varied peach dataset among publicly available peach datasets. It comprises 4599 images (1024×768) of nine peach cultivars, which were taken under natural illumination and in real-world

Fig. 1. Images of NinePeach (left) and StraDI_Db1 (right).

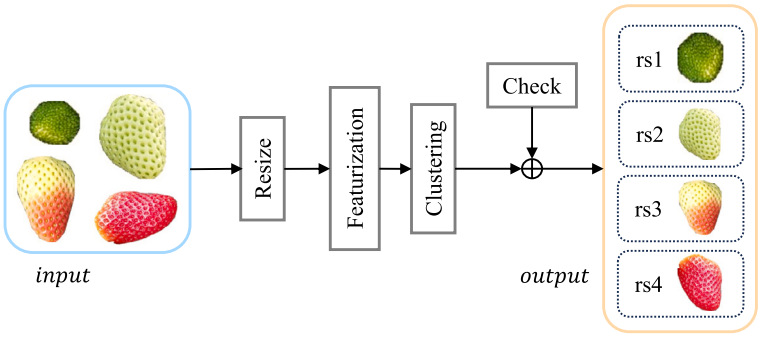


Fig. 2. The process of strawberry mask classification.

production settings, including peaches with factors like different intensities of natural light, multi-fruit adhesion, and occlusion caused by stems and leaves. The peaches demonstrated different physical scenarios such as isolated peaches, peaches that are in close proximity to one another, peaches that are partially obscured by leaves or stalks, and peaches that are illuminated from the opposite side.

This dataset is divided into training (3240 images) and validation (1359 images) subsets, and each peach is categorized into three ripeness stages: unripe, semiripe, and ripe.

StrawDI_Db1 dataset. This dataset contains a total of 3100 images (1008×756) that were randomly selected from a large number of strawberries images, taken from 20 plantations, where images were taken under different conditions of brightness, at a distance of approximately 20 cm from the ridge during a full picking campaign. This dataset is divided into training (2800 images), validation (100 images) and testing (200 images) subsets. The training and testing sets are used in this paper.

Unfortunately, this dataset only offers class-agnostic annotations for strawberries, with no information provided on ripeness. Therefore, we present our solution to this problem in the following section.

2.2. StrawDI_Db1 ripeness annotation

Based on the previous work [1,46], four ripeness stages are selected to distinguish the strawberries from StrawDI_Db1 dataset, with the criterion described in Table 1. To achieve this classification, we adopt a simple but effective method for dividing strawberries into four stages, as illustrated in Fig. 2.

First, the strawberry instances are cropped from the original images and background pixels are filtered, as the contextual information from the background was assumed to introduce noise rather than contribute to the classification accuracy. All strawberry instances are resized to 280×280 pixels.

Second, some machine learning methods like Histogram of Oriented Gradients, and deep learning methods like pre-trained CNN models are employed to extract features of the resized strawberry instances. Then, the cosine-similarity is adopted to calculate the distance between features, resulting in similarity matrices.

Table 1
Four ripeness stages of strawberry.

Category	Description
rs1 (Green)	Dark green, the sizes are relatively small.
rs2 (White)	Expanding, the colour is white.
rs3 (Turning)	Below 90% red and not ready to be harvested.
rs4 (Red)	Over 90% red, edible and ready to be harvested.

Third, we applied K-means clustering to solve the similarity matrices, partitioning them into four clusters. The clustering method with the best performance was chosen to give the predictions.

Lastly, the clustering results were manually reviewed and corrected to ensure alignment with the predefined ripeness criteria. This refinement ensured that the final clustering outcomes adhered to the anticipated standards.

2.3. Dataset summary

In summary, our study leverages two large fruit datasets NinePeach and StrawDI_Db1, and both of them have individual mask annotations and ripeness stage labels. Peaches and strawberries are two popular fruits that are widely grown and consumed across the world. By integrating these two datasets, we can effectively cover different scenarios involving both tree-fruit (peaches) and berry-fruit (strawberries).

The combined dataset contains 7 different classes, with 3 classes corresponding to peaches and 4 classes to strawberries. This detailed dataset structure ensures a comprehensive representation of fruit development stages, facilitating more accurate and generalizable insights in subsequent analyses. Examples of images and their associated annotations are presented in Fig. 3, and the distribution of instance categories is summarized in Table 2. It is noted that the quantity of fruit instances decreases progressively over time as ripeness advances, revealing a real pattern that aligns with the natural growth and ripening process of fruit.

Fig. 3. Examples of fruit instance annotation in NinePeach (left) and StraDI_Db1 (right).

Table 2
The category distribution of the combined dataset.

NinePeach			StrawDI_Db1		
Category	Train	Val	Category	Train	Val
unripe	3669	1717	rs1	6693	453
semiripe	3312	1307	rs2	4014	319
ripe	1698	737	rs3	3010	212
/	/	/	rs4	2517	148
Instance	8679	3761	Instances	16234	1132
Image	3240	1359	Images	2800	100

By training on a combined dataset, the model learns to handle these complexities across different object types, which enhances its robustness. Additionally, the inclusion of varied fruit types in a unified dataset can improve the model's ability to distinguish between different objects, making it more adaptable to real-world applications where multiple fruit categories are often present simultaneously.

3. Proposed model

3.1. Model structure

For fruit ripeness determination, we propose an instance segmentation model FruitQuery following the design of Mask2Former [5], which consists of a backbone, a pixel decoder and Transformer decoders. The architecture is illustrated in Fig. 4.

3.1.1. Backbone

It is well-known that the convolutional layer has inductive biases of locality and spatial invariance, which is capable of extracting low-level small local features. The self-attention layer has a global receptive field and allows capturing global context information within an image. Therefore, these two types of layers are considered to build the backbone for multi-level feature extraction. The proposed backbone is illustrated in Fig. 4a.

By combining convolutional layers with stronger generalization performance and self-attention layers with higher model capacity and stronger learning ability, we assume that the backbone can achieve better generalization performance and learning ability. Given an input image, the backbone can generate 4 levels of features, which provide high-resolution coarse features and low-resolution fine-grained features that usually boost the performance of fruit segmentation. It is noted that ConvBlock is removed in the last block in order to reduce the model parameters.

Patch Embedding. The input image is divided into a grid of non-overlapping patches, and each patch normally covers a square region of the image and is transformed into a fixed-dimensional embedding vector. According to different patch sizes and embedding dimensions, 4 different patch embedding blocks are attached in front of each block. As patch embedding does not inherently preserve positional information within each patch, it is required to add positional encoding to the subsequent two blocks.

ConvBlock. The ConvBlock is made of several convolutional layers, with two residual connections. In the first residual connection, two 1×1 point-wise convolutional layers (PWConv) are respectively placed before and after a 5×5 convolutional layer. The 5×5 convolutional layer has a larger receptive field to consider larger local regions and is expected to capture large-scale features like fruit edges, and textures in images. In the second residual connection, two 1×1 point-wise convolutional layers are used to perform MLP-like behaviour: increase the dimension to 4 times and then decrease it to the desired output dimension. This operation is designed to increase nonlinear representation capacity and learn richer feature representations, thereby enhancing the model performance and generalization ability. The 1×1 point-wise convolutional layers only involve a single pixel and have fewer parameters to learn, therefore it is suitable for dimension expansion and compression.

Efficient Multi-head Self-Attention (EMSA). For each head of the multi-head self-attention, the query Q, key K and value V are obtained by applying three linear projections to the input embedding, including positional encoding. Q, K and V have the same dimensions $N \times C$, where $N = H \times W$. Then, attention scores are calculated by the scaled dot-product attention. The scores are normalized using the Softmax function to obtain attention weights, which is used to compute a weighted sum of the V vectors of all tokens, as shown in Equation (1), where d_k refers to the dimensionality of the key. Tokens with higher scores contribute more to the output of the self-attention mechanism.

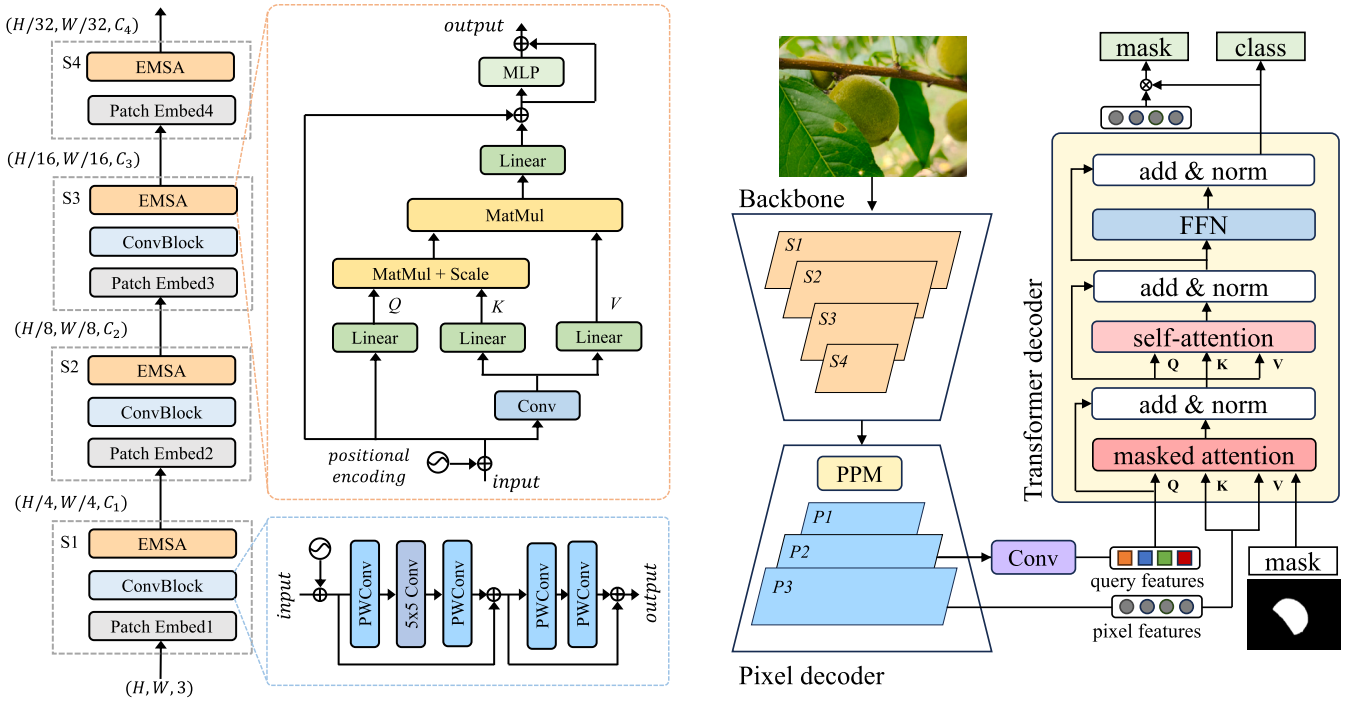
$$\text{Attention} = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

The main bottleneck of the self-attention layer lies in its computation cost of $O(N^2)$, which scales quadratically with spatial dimension based on the input embedding. To alleviate this problem, we introduce an efficient multi-head self-attention (EMSA) based on the spatial reduction method proposed in PVT [50]. The main idea of it is to reduce the length of the sequence with a reduction ratio R. For reducing computations, an input sequence with shape $(C, H \cdot W)$ is reshaped to the $\hat{K}$ with shape $(C \cdot R^2, H \cdot W/R^2)$ based on Equation (2). Here we use a convolutional layer with $kernel\_size = R$ and $stride = R$ to perform the reshape operation. Equation (3) refers to a linear layer taking $\hat{K}$ as input and generating a new K' with shape $(C, H \cdot W/R^2)$ as output.

$$\hat{K} = \text{Reshape}(K, R) \tag{2}$$

$$K' = \text{Linear}(C \cdot R^2, C)(\hat{K}) \tag{3}$$

As a result, the complexity of the efficient self-attention mechanism is reduced from $O(N^2)$ to $O(N^2/R^2)$. It is noted that a residual MLP layer is appended at the end of EMSA to increase the model capacity and avoid overfitting.



(a) The proposed backbone. (b) The architecture of FruitQuery.

Fig. 4. The proposed backbone and overall architecture of FruitQuery.

Table 3
Specification of our backbones.

Stage	Size	Layer	xs	s
$S1$	$\frac{H}{4} \times \frac{W}{4}$	Patch Embed1	Patch Size = 4, $C = C_1$	
		EMSA	$C_1 = 36$ $B_1 = 1$ $R_1 = 4$	$C_1 = 48$ $B_1 = 1$ $R_1 = 4$
$S2$	$\frac{H}{8} \times \frac{W}{8}$	Patch Embed2	Patch Size = 2, $C = C_2$	
		EMSA	$C_2 = 72$ $B_2 = 1$ $R_2 = 2$	$C_2 = 96$ $B_2 = 1$ $R_2 = 2$
$S3$	$\frac{H}{16} \times \frac{W}{16}$	Patch Embed3	Patch Size = 2, $C = C_3$	
		EMSA	$C_3 = 144$ $B_3 = 3$ $R_3 = 2$	$C_3 = 240$ $B_3 = 3$ $R_3 = 2$
$S4$	$\frac{H}{32} \times \frac{W}{32}$	Patch Embed4	Patch Size = 2, $C = C_4$	
		EMSA	$C_4 = 288$ $B_4 = 1$ $R_4 = 1$	$C_4 = 384$ $B_4 = 2$ $R_4 = 1$

To cater to diverse scenarios, we proposed two different settings for the backbone (s and xs). The specifications are presented in Table 3, where C represents the number of embedded dimensions, and B denotes the number of blocks.

3.1.2. Pixel decoder

Multi-level contextual feature maps play a crucial role in image segmentation, but employing a complex multi-scale feature pyramid network escalates the computational workload. For instance, multi-scale deformable attention used in Mask2Former demonstrates good performance, but it also brings a large number of parameters. To build a lightweight but effective model, the Feature Pyramid Network (FPN, [30]) is selected as the pixel decoder, which occupies less than half the size of the multi-scale deformable attention. FPN works by taking the feature maps produced by the backbone at different levels ($S1$, $S2$, $S3$ and $S4$), and building a feature pyramid from top to down ($P1$, $P2$ and $P3$) through lateral connections ($S4 - P1$, $S3 - P2$, $S2 - P3$).

A pyramid pooling module (PPM, [59]) is added to the top layer $P1$ to enlarge the receptive field and fuses the multi-scale features, of which the detail is shown in Fig. 5. The input feature is divided into multiple regions of different sizes, using four different adaptive average pooling to capture information at different receptive field sizes. Then the pooled features are resized to the same size as the input, and concatenated with the input feature, resulting in a feature of shape $(C + 4N, H, W)$. Finally, a simple convolutional layer is used to transform the shape of $(C + 4N, H, W)$ back to (C, H, W) and fuse all information. Since the pooling operation does not introduce any new parameters, the introduction of PPM enhances the model's performance without significantly increasing its computational complexity.

The final output of the pixel decoder comprises features at three resolutions, incorporating both high-level features rich in semantics and low-level features rich in spatial information.

The final output of the pixel decoder integrates multi-scale features from three different resolutions, combining both high-level semantic information and low-level spatial details.

3.1.3. Transformer decoder

The Transformer decoder plays a crucial component in the model, which takes the learned features from the pixel decoder and processes them to produce the final output predictions. As shown in Fig. 4b, the decoder follows the paradigm of the standard architecture of the original Transformer, transforming N embeddings of objects into output embeddings. It is a stack of decoder layers, each of which consists of a masked attention layer, a self-attention layer and a feed-forward network (FFN). Each Transformer decoder layer generates predictions for mask and class, but only the prediction of the last layer is used as the final prediction, prior layer predictions can be used for auxiliary predictions optionally. We set the number of the Transformer decoder layers as 3 to achieve a better trade-off between accuracy and model size, and the feature $P3$ from the pixel decoder is used as pixel features.

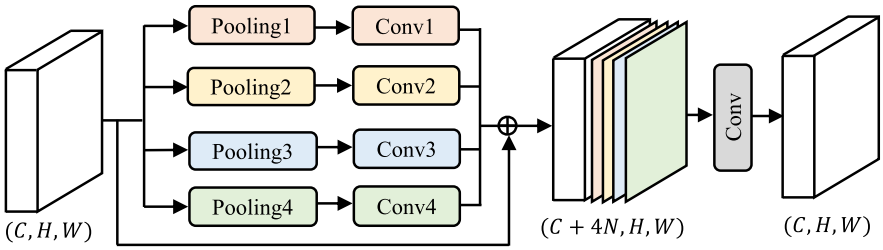


Fig. 5. The pyramid pooling module.

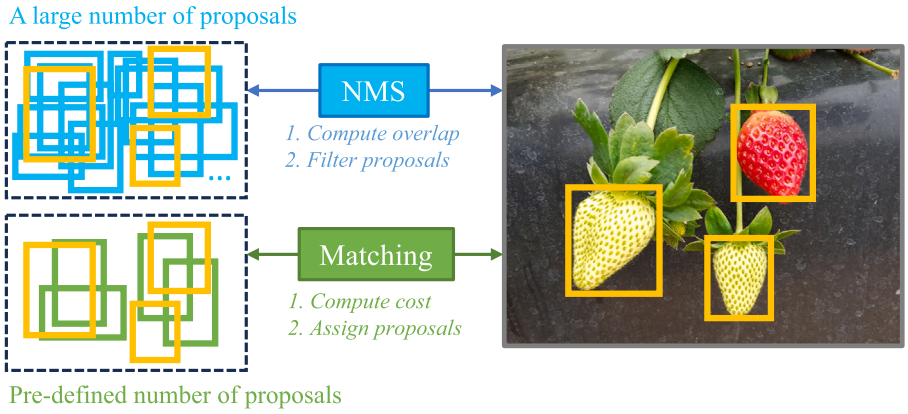


Fig. 6. Non-maximum suppression and bipartite matching.

Query Features Initialization. The query features are important in the Transformer model, as they guide the decoder to attend to the most significant parts of the input embedding. Previous research indicates that query features can be initialized from zero [2], or can be updated by local features [5]. Although these two strategies are effective in generating query features, they require more decoders and longer training iterations to refine. Inspired by Deformable DETR, which selects a set of query bounding boxes from pyramidal features to perform object detection, and SparseInst [7], which introduces a simple convolutional module F_{iam} to highlight informative regions for each foreground object.

Therefore, we combine these two advantages to our model. A F_{iam}-like convolutional module is added to efficiently initialize the query features in our model, which directly picks the queries with high semantics from underlying multi-scale feature maps. The simple module only consists of two convolutional layers. The first convolutional layer is a typical 3×3 convolution layer with the same input and output dimensions. The second convolutional layer is a 1×1 convolution layer to reduce the number of dimensions to the number of classes $+1$, where the extra one means "no object ϕ". Specifically, feature $P2$ from the pixel decoder is selected to generate N pixel embeddings with the highest foreground probabilities as the query features.

Masked Attention. The cross-attention in the original Transformer decoder is replaced with masked attention. The standard cross-attention is computed by Equation (4). l is the layer index, X_l indicates the query features with the shape $N \times C$ at the l-th layer. $Q_l = f_q(X_{l-1})$ is calculated by applying a linear transformation f_q on the query features of previous layer. K_l and V_l are the pixel features from pixel decoder after linear transformations f_k and f_v.

$$X_l = \text{Softmax}(Q_l K_l^t) V_l + X_{l-1} \tag{4}$$

Based on cross-attention, masked attention adds an attention mask $\mathcal{M}_{l-1}$, as calculated in Equation (5).

$$X_l = \text{Softmax}(\mathcal{M}_{l-1} + Q_l K_l^t) V_l + X_{l-1} \tag{5}$$

The attention mask $\mathcal{M}_{l-1}$ at feature location (x, y) is calculated in Equation (6), where $m_{l-1}(x, y)$ is the binary output of the resized mask prediction of the previous (l-1) decoder layer. m_0 is the binary mask prediction obtained from X_0.

$$\mathcal{M}_{l-1} = \begin{cases} 0 & \text{if } m_{l-1}(x, y) = 1 \\ -\infty & \text{otherwise} \end{cases} \tag{6}$$

3.2. Loss function

Different from anchor-based segmentation models that generate a large number of anchor proposals, our model employs the Transformer decoder to treat fruit detection as an end-to-end dictionary lookup task. Specifically, the decoder generates a fixed number of N predictions by decoding the N learnable query embeddings layer by layer. Therefore, the necessity for manual processes like non-maximum suppression (NMS) is eliminated. Instead, we adopt Hungarian matching, which is a kind of bipartite matching method, to find the best matching between predictions and ground truths for loss computation.

The difference between NMS and bipartite matching is illustrated in Fig. 6. NMS generates a large number of proposals and applies heuristic filtering based on overlap scores to remove redundant detections. This introduces a non-differentiable post-processing step. In contrast, bipartite matching employs a pre-defined fixed number of proposals and assigns each proposal to a specific ground truth or a "no object" class based on a cost matrix. By integrating this matching process directly into the optimization framework, our model enables a fully end-to-end differentiable pipeline where predictions and assignments are jointly optimized.

First, all of the predictions including class predictions, mask predictions and class targets, mask targets are used to calculate a cost matrix for prediction selection, where X indicates the number of instances in a batch. The class cost and mask cost are calculated by cross entropy loss and Dice loss respectively, as shown in Equation (7) and Equation (8),

$$CE(y, p) = -\Sigma_i y_i \cdot \log(p_i) \tag{7}$$

where y_i represents the ground truth probability and p_i represents the predicted probability.

$$\text{Dice}(m,t) = \frac{2 \cdot \Sigma_{x,y} m_{xy} \cdot t_{xy}}{\Sigma_{x,y} m_{xy}^2 + \Sigma_{x,y} t_{xy}^2} \tag{8}$$

where m_{xy} and t_{xy} refer to the value of pixel located at (x, y) in predicted mask m and ground truth t respectively.

Second, the Hungarian algorithm is used to search for the best bipartite matching by solving the cost matrix, resulting in a matching score $C(i,k)$ for i-th prediction and k-th ground truth object. Therefore, the number of predictions is decreased from N to match that of the targets.

The total training loss for our proposed model is defined in Equation (9):

$$\mathcal{L}_{total} = \lambda_{cl}\mathcal{L}_{class} + \lambda_m\mathcal{L}_{mask} + \lambda_{co}\mathcal{L}_{conv} + \lambda_a\mathcal{L}_{aux} \tag{9}$$

λ indicates the different loss weights.

$\mathcal{L}_{class}$ is the cross entropy loss between the selected class predictions and class targets.

$\mathcal{L}_{mask}$ is the dice loss between the selected mask predictions and mask targets.

$\mathcal{L}_{conv}$ is the cross entropy loss between the output of F_{iam}-like convolutional module and ground truth.

As each Transformer decoder layer generates class prediction and mask prediction, we use the prior predictions to calculate auxiliary loss $\mathcal{L}_{aux}$, as shown in Equation (10),

$$\mathcal{L}_{aux} = \sum_{i=0}^{D} (\mathcal{L}^{\geq}{}_{class} + \mathcal{L}^{\geq}{}_{mask}) \tag{10}$$

where D indicates the number of Transformer decoders. $\mathcal{L}'{}_{class}$ and $\mathcal{L}'{}_{mask}$ use the same loss functions as $\mathcal{L}_{class}$ and $\mathcal{L}_{mask}$.

Based on PointRend [25], which demonstrated that a segmentation model can be effectively trained by calculating its mask loss on a subset of randomly K sampled points instead of the entire mask, we incorporate this strategy into our model. Consequently, we compute the mask loss using sampled points both in the matching process and the final loss calculation.

3.3. Evaluation metrics

Average Precision (AP) Precision serves as a standard and widely-used metric for evaluating the network's ability to accurately identify target objects, reflecting the comprehensive performance of the network. The definition of precision is shown by Equation (11):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{11}$$

Here, TP represents the count of correctly detected fruit targets, and FP represents the count of cases where the target is wrongly detected as a fruit when it is not. The average precision is selected as the primary metric to assess model performance. It is computed by averaging the precision scores at 10 Intersection over Union (IoU) thresholds, ranging from 0.50 to 0.95, across all categories. A higher AP value indicates better detection accuracy of the model. Specifically, the AP values for IoU thresholds of 0.50 and 0.75 are reported, as well as for each individual category.

Learnable Parameters (Params) The learnable parameters refer to the weights and biases within the model's layers, which are adjusted during the training process to optimize performance and make accurate predictions. The total number of learnable parameters in a model is often considered an indicator of its capacity and complexity.

Floating-point Operations (FLOPs) Floating-point operations is a measure of the computational complexity of a deep learning model. It represents the number of arithmetic operations performed by the model during the process of forward propagation, where input data passes through the layers of the model to produce output predictions. FLOPs is typically quantified in terms of the number of multiplications and additions performed by the model.

Frame per second (FPS) Frames per second (FPS) is a measure of the inference speed of a deep learning model, indicating how many input images the model can process per second. A higher FPS reflects faster model execution, which is critical for real-time applications such as autonomous driving or video analysis. FPS is influenced by factors such as model size, hardware performance, and optimization techniques.

4. Experiments and results

4.1. Experiments

4.1.1. Configuration

In this paper, experiments are conducted based on Detectron2 [51] and have been carried out using Python 3.9.13 and PyTorch 1.13 on a computer with an Intel Xeon Gold 6152 @2.1 GHz CPU, 2 Nvidia Tesla V100 GPUs and 32.0 GB memory.

4.1.2. Training details

No pre-trained weights are utilized in this work, and the parameters of all convolution layers are initialized by a normal distribution. The training process incorporates diverse data augmentation strategies to improve the model's robustness and generalization. These strategies contain random horizontal flips, resizing the input images such that the shortest side is one of 416, 448, 480, 512, 544, 576, 608 or 640 pixels while the longest is at most 768. This multi-scale resizing introduces variability in the input sizes, encouraging the model to adapt to objects of different scales.

The maximum prediction per image N is set to 100, based on the assumption that this value is sufficient to include all fruit present in a single image. The number of mask sampling points K is set to 12544, corresponding to a grid resolution of 112×112. The loss weights are set to $\{\lambda_{cl}$:2.0, λ_m:5.0, λ_{co}:20.0, λ_a:1.0$\}$. The depth of decoder layers) D is set to 3, indicating the number of decoding layers used for prediction tasks, providing a balance between computational efficiency and representational capacity.

We use the AdamW optimizer with a step learning rate schedule, of which the initial rate is 0.0001, and the weight decay is 0.05. A learning rate multiplier of 0.1 is applied to the backbone, and decay the learning rate by 10 at fractions 0.9 and 0.95 of the total number of training iterations. We train our model for 54k iterations with a batch size of 8.

4.1.3. Inference details

The data augmentation strategy used in inference is only resizing the input images such that the shortest side is 640 pixels while the longest is at most 768 pixels. Auxiliary predictions are not used during inference. The top 100 candidates with the highest confidence are selected as final predictions.

During inference, the data augmentation strategy is simplified to resize the input images. Specifically, each image is resized such that the shortest side is scaled to 640 pixels while ensuring the longest side does not exceed 768 pixels, preserving the aspect ratio. Auxiliary predictions like outputs from intermediate layers or heads used during training, are not used during inference to streamline the process and focus solely on the final model predictions. After the model generates predictions, the top 100 candidate predictions with the highest confidence scores are selected as the final predictions.

4.2. Results

We conducted a comprehensive segmentation comparison of different state-of-the-art backbones on the combined fruit dataset, using FruitQuery's architecture shown in Fig. 4b, and the results are summarized in Table 4.

Table 4
Instance segmentation results on the combined dataset with different backbones.

Backbone		Type	AP	AP50	AP75	NinePeach			StrawDI_Db1				Params (M)	FLOPs (G)	FPS
						AP_{unripe}	$AP_{semiripe}$	AP_{ripe}	AP_{rs1}	AP_{rs2}	AP_{rs3}	AP_{rs4}			
CNN-based	ResNet [15]	18	61.74	75.68	65.37	50.61	53.50	65.72	39.52	73.64	75.80	73.39	17.54	29.56	19.50
		34	63.17	77.06	66.98	49.42	56.77	67.65	38.52	74.00	77.30	78.56	27.64	45.88	34.60
		50	63.92	77.29	67.70	53.62	54.78	**69.57**	41.61	75.33	75.36	77.17	30.52	50.94	31.54
	FasterNet [3]	s	62.56	75.04	66.20	49.72	50.78	63.21	44.76	74.04	78.29	77.12	35.82	91.85	17.90
		m	64.52	76.13	67.68	51.46	51.38	64.09	47.15	76.68	81.17	79.69	58.06	129.00	16.12
		l	65.25	77.01	69.15	53.24	53.61	66.47	**50.14**	**77.73**	77.72	77.86	97.70	189.00	15.60
	YOLOv8* [22]	s	57.33	72.44	63.67	48.49	49.66	68.02	32.64	61.44	65.70	75.40	11.79	46.12	**44.22**
		m	58.70	74.36	64.85	54.06	53.15	67.10	32.81	64.43	66.91	72.43	27.24	119.24	41.58
		l	59.74	75.50	66.17	55.32	52.63	66.79	34.11	64.14	71.46	73.78	45.94	238.48	34.36
	YOLOv9* [22]	s	59.91	75.23	66.12	**56.60**	54.33	65.10	33.85	65.99	69.95	73.54	8.64	82.26	13.4
		m	60.04	75.72	66.57	54.90	54.08	68.63	33.50	65.21	69.34	74.60	22.26	142.38	15.78
		c	60.41	76.07	67.13	54.37	55.13	68.13	33.89	66.53	69.13	75.68	27.84	171.82	15.58
	YOLOv10* [22]	s	58.17	73.91	64.53	50.86	50.34	63.50	33.34	68.75	66.18	74.22	9.20	44.10	21.28
		m	58.60	74.23	65.18	52.56	50.57	65.86	33.23	65.76	68.09	74.11	19.37	110.06	18.94
		b	58.69	73.60	64.87	52.95	51.82	65.39	32.71	63.70	69.35	74.92	25.52	180.68	17.94
Transformer-based	MobileViT [34]	xxs	46.34	63.25	50.76	28.26	38.27	50.81	20.12	54.66	63.06	69.19	**7.27**	**17.27**	19.74
		xs	50.29	67.14	54.81	34.17	41.35	54.80	23.85	62.78	62.95	72.13	8.28	21.14	19.56
		s	52.90	68.62	56.82	38.08	44.19	56.41	27.65	62.97	70.42	70.58	11.36	26.99	19.44
	LightViT [18]	t	54.65	69.90	58.59	40.05	44.47	57.85	30.77	66.00	71.37	72.04	14.06	29.11	14.36
		s	56.31	71.25	60.30	42.43	46.42	59.60	31.96	68.74	70.89	74.14	23.74	28.03	14.28
		b	58.57	73.17	62.39	45.18	50.00	62.50	34.13	70.31	72.29	75.57	39.63	45.06	13.49
	NextViT [27]	s	62.36	74.88	66.02	48.79	49.99	62.38	48.11	73.57	77.61	76.07	37.96	105.00	15.06
		b	62.47	75.03	65.81	47.71	51.79	61.08	45.80	77.66	75.75	77.52	51.02	128.00	13.10
	GroupMixFormer [8]	t	62.67	75.45	66.21	47.42	52.55	63.07	42.81	74.79	78.39	79.67	17.62	83.72	9.24
		s	63.50	76.31	67.79	48.81	53.66	64.79	44.53	74.60	79.36	78.75	29.06	96.51	9.02
	MetaFormer [57]	id	59.65	73.16	63.43	43.19	48.77	62.98	40.86	71.14	75.95	74.69	18.40	66.92	18.94
	SegFormer [54]	b0	58.48	71.53	62.36	42.33	47.76	58.71	40.09	67.83	75.39	77.28	10.19	57.06	17.32
	PoolFormer [57]	s12	61.12	74.19	64.48	44.19	47.02	61.16	42.94	73.75	78.19	80.60	18.40	66.92	13.50
		s24	62.61	75.17	66.28	45.27	49.63	64.78	44.69	73.76	79.89	80.24	27.88	80.94	12.34
	TransXNet [54]	t	59.26	72.48	62.66	42.31	46.70	63.03	39.81	71.59	73.96	77.44	19.33	70.65	10.88
		s	60.43	73.64	63.87	45.07	49.16	62.24	40.31	71.02	74.53	80.66	33.34	98.84	8.22
	CMT [11]	ti	66.00	78.44	69.43	54.80	57.58	68.98	46.84	77.23	78.81	77.79	14.57	67.55	14.06
		xs	66.46	78.60	70.10	55.79	57.45	68.96	47.72	76.35	78.37	80.61	20.21	78.27	13.08
	EMSA (Ours)	xs	66.46	78.49	70.27	51.77	56.11	69.03	47.12	77.16	82.12	81.92	10.94	61.56	16.50
		s	**67.02**	**79.17**	**70.83**	52.05	**58.68**	68.55	47.91	76.47	**82.16**	**83.74**	14.08	69.33	16.00

∗ YOLO series are trained using their segment head.

Overall Performance Our proposed model with EMSA-s (FruitQuery-s) achieves the highest overall AP of 67.02, AP50 of 79.17, and AP75 of 70.83, significantly outperforming 13 other models with a total of 33 variants. Our model with EMSA-xs (FruitQuery-xs) also delivers a competitive AP of 66.46. This illustrates the superior performance of our model in fruit segmentation.

Among CNN-based models, the widely-used ResNet series shows solid results, with ResNet-50 reaching an AP of 63.92. The recent FasterNet-l also achieves a competitive AP of 65.25. Turning to the YOLO series, YOLOv9-c attains the highest AP of 60.41 among its variants, indicating that the YOLO series has limited performance on fruit segmentation. In comparison, all YOLO variants fall short of our proposed model.

On the Transformer-based side, models demonstrate more different designs and parameter counts. The variants of NextViT, GroupMixFormer, and PoolFormer generate similar results of AP ranging from 62.37 to 63.50. Two CMT variants reach APs of 66.00 and 66.46, coming closest to our performance. These Transformer-based models reflect the trend toward attention-driven backbones, with noticeable performance gains over many CNN counterparts.

However, they still fall short of our FruitQuery in AP, AP50 and AP75, suggesting that our proposed query-based design leverages features more effectively for precise fruit instance segmentation.

Individual Performance For NinePeach dataset, YOLOv9-s achieves the highest AP_{unripe} of 56.60, while ResNet-50 delivers the highest AP_{ripe} of 69.57. However, our model attains the best performance on semiripe peaches with an $AP_{semiripe}$ of 58.68, underscoring its ability to capture the more subtle visual cues present in intermediate ripeness for peaches.

For StrawDI_Db1 dataset, CNN-based FasterNet obtains the highest AP_{rs1} of 50.14 and AP_{rs2} of 77.73, while ours-s outperforms all counterparts in half of the strawberry ripeness stages, with the highest AP_{rs3} of 82.16 and AP_{rs4} of 83.74. These gains indicate that our model can effectively handle the appearance variations in later strawberry growth, where colour, texture, and shape have significant changes compared to earlier stages.

Overall, within seven ripeness stages of the combined dataset, our model delivers the best AP for three of them, indicating that our model with the query-based design is capable of capturing fine-grained features within different fruit ripeness levels, and generating comparable results.

Model Complexity The broad range of model sizes is generally related to performance: larger models typically have more parameters, which allows them to capture more complex patterns and relationships.

On the CNN-based models, FasterNet-l is the largest CNN-based model with parameters of 97.70M and FLOPs of 189G, and it achieves a competitive AP of 65.25. Notably, the YOLO series are well-known for their lightweight design, with YOLOv9-s having 8.64M parameters and 82.26G FLOPs, and YOLOv10-s having 7.27M parameters and 44.10G FLOPs, but their AP of 59.91 and 58.17 are lower than many other models.

On the Transformer-based models, MobileViT-xxs exhibits the smallest parameter count of 7.27M and FLOPs of 17.27G, while it comes with the lowest AP of 46.34. NextViT-b is the most complex Transformer-

Table 5
Ablation on the pixel decoder.

Module	AP	AP50	AP75
FPN	64.97	78.41	68.88
PPM-FPN	**66.57**	**78.98**	**70.22**

based model with 51.02M parameters and 128G FLOPs, delivering an AP of 62.47.

Our model shows a highly cost-efficient design. Specifically, the ours-xs only utilizes 10.94M parameters and 61.56G FLOPs to achieve an AP of 66.46, and ours-s attains the highest AP of 67.02 with 14.08M parameters and 69.33G FLOPs.

In contrast, models with similar APs to ours-xs (66.46), such as CMT-ti (66.00), CMT-xs (66.46) and FasterNet-l (65.25), require larger parameters and FLOPs (14.57M/67.55G, 20.21M/78.27G and 97.70M/189.00G) than ours-xs (10.94M/61.56G). On the other hand, models that match ours-xs in parameters and FLOPS (10.94M/61.56G), such as YOLOv8-s(11.79M/46.12G), MobileViT-s (11.36M/26.99G) and SegFormer-s24 (10.19M/57.06G), deliver poorer APs (57.33, 52.90 and 58.48).

Inference Speed CNN-based models exhibit higher inference speeds compared to Transformer-based models, consistent with the established efficiency advantages of convolutional architectures. Among all evaluated models, YOLOv8-t achieves the highest FPS at 44.22, followed by YOLOv8-m (41.58) and YOLOv8-l (34.36), highlighting the real-time capabilities.

Our proposed FruitQuery achieves relatively high inference speeds (16.5 and 16 FPS), demonstrating competitive inference performance. They outperform all YOLOv9 variants, suggesting improved speed efficiency relative to this recent Transformer-based series. In addition, our models surpass a number of widely used Transformer-based backbones such as LightViT-t (14.36), CMT-ti (14.06), and NextViT-s (15.06), which are specifically designed for efficiency.

While slightly slower than MobileViT-xxs (19.74) and MobileViT-xs (19.56), our models are notably faster than recent models like TransXNet-s (8.22) and GroupMixFormer-s (9.02), positioning them among the faster Transformer-based designs. These results indicate that our models strike a favourable balance between inference speed and model complexity.

In summary, the results demonstrate that our proposed model not only exhibits comparable or even superior results to other segmentation models but also maintains lightweight model size and higher efficiency.

4.3. Ablation experiments

4.3.1. Type of pixel decoder

Table 5 compares two different pixel decoders FPN and PPM-FPN in terms of model performance. The baseline FPN achieves an AP of 64.97, AP50 of 78.41, and AP75 of 68.88. In contrast, the PPM-FPN variant leads to a consistent performance boost across all metrics, improving AP by 1.6 points from 64.97 to 66.57, AP50 by 0.57 points from 78.41 to 78.98, and AP75 by 1.34 points from 68.88 to 70.22. These results indicate that incorporating PPM into the FPN enhances the overall segmentation performance.

4.3.2. Number of decoder attention head

Table 6a compares the effect of different numbers of attention heads on model performance. With just 2 heads, the model attains an AP of 62.36, AP50 of 75.30, and AP75 of 65.80, indicating limited representational capacity. Increasing to 4 heads yields the highest AP of 64.46, AP50 to 77.09 and AP75 of 68.61. Although further increasing the number of heads to 8 slightly boosts AP to 64.46 and AP75 to 67.88 compared to 2 heads, it still lags behind the 4-head configuration. These results suggest that 4 attention heads provide an optimal balance, offering richer feature representations without incurring diminishing returns.

4.3.3. Number of query

Table 6b shows the effect of different numbers of queries on model performance. When the number of query is set to 100, the model achieves its highest overall AP of 66.52, AP50 of 78.84 and AP75 of 70.19. Decreasing the number of queries to 80 or 90 leads to a consistent drop in performance across all metrics. On the other hand, increasing the number of queries to 110 or 120 offers no further improvement. These results suggest that using 100 queries strikes an effective balance between capturing sufficient object-level information and maintaining computational efficiency.

4.3.4. Number of decoder layers

Table 6c illustrates the effect of different numbers of decoder layers on model performance. With only 1 to 3 layers, AP stays between 62.85 and 63.98, indicating limited representational depth. As more layers are added, accuracy steadily improves, peaking at 6 layers with an AP of 66.80, AP50 of 78.95, and AP75 of 70.85. Beyond 6 layers, model performance begins to decline, suggesting that excessive stacking of decoder blocks may introduce redundancy or complicate training. These findings highlight an optimal spot at 6 decoder layers.

4.4. Combined or separated training

We compare the performance difference of FruitQuery-xs trained on combined (t/o combined) and separate (t/o separate) datasets, and the results are shown in Table 7. For NinePeach, the combined training strategy produces notable improvements across all ripeness levels, with AP_{unripe} increases of 7.55 points from 43.72 to 51.27, 4.86 points for $AP_{semiripe}$ from 49.07 to 53.93, and 4.89 points for AP_{ripe} from 62.05 to 66.94.

In contrast, results on StrawDI_Db1 are mixed: AP_{rs2} has a significant gain of 4.18 points from 72.55 to 76.73, and AP_{rs4} also increases 1.65 points from 77.99 to 79.64. However, the other two categories AP_{rs1} drops from 45.57 to 42.29 and AP_{rs3} drops from 79.87 to 78.40.

Overall, training on the combined dataset boosts the model's overall AP from 61.08 to 64.63, indicating that learning from a broader, integrated fruit distribution can enhance generalization for the majority of fruit ripeness stages despite limited category-specific trade-offs.

4.5. Model parameter distribution

We summarize the parameter distribution of YOLO and our FruitQuery in Table 8. Based on previous results in Table 4, YOLOv9 is the best-performing version of three YOLO series, therefore it is selected to compare with our model and also in later comparisons.

YOLOv9-s has the least number of parameters of 8.64M, with a head of 2.92M, but produces the lowest AP of 59.91. When changing the model from YOLOv9-s to YOLOv9-m, the total parameters increase to 22.26M, with a bigger backbone and head, but bring a tiny AP gain from 59.91 to 60.04. YOLOv9-c performs better than YOLOv9-m with the AP of 60.41, but occupies a backbone of 19.95M and a head of 7.89M.

On the other hand, our model demonstrates its ability to outperform YOLOv9 with fewer parameter counts. Specifically, FruitQuery-xs and FruitQuery-s have an identical head of 4.18M, which is smaller than YOLOv9-m and YOLOv9-c. The main difference between the two variants of FruitQuery lies in the backbone, FruitQuery-s has a more complex backbone and delivers a better AP of 67.02.

These results not only demonstrate that our FruitQuery achieves a significantly better balance between the segmentation performance and model size compared to YOLO, but also highlight its lightweight design, which enhances the potential for in-field applications.

4.6. Visualization

We visualize the segmentation performance of our FruitQuery in Fig. 7. First, FruitQuery is capable of simultaneously segmenting

Table 6
Results of ablation experiments based on FruitQuery-xs.

(a) Ablation on the number of attention head.

Head	AP	AP50	AP75
2	62.36	75.30	65.80
4	**64.46**	**77.09**	**68.61**
8	64.36	76.98	67.88

(b) Ablation on the number of query.

Query	AP	AP50	AP75
80	64.54	77.74	68.39
90	64.45	77.38	68.15
100	**66.52**	**78.84**	**70.19**
110	65.91	78.56	69.48
120	65.48	78.34	69.11

(c) Ablation on the number of decoder layers.

Layer	AP	AP50	AP75
1	62.85	75.78	66.59
2	63.98	76.93	67.72
3	63.91	76.69	67.86
4	66.41	78.98	70.20
5	66.68	79.31	70.40
6	**66.80**	**78.95**	**70.85**
7	65.94	78.49	69.36
8	65.78	78.07	69.26

Fig. 7. Segmentation visualization of our FruitQuery on NinePeach (left) and StrawDI_Db1 (right).

Table 7
Comparison of training on separate and combined dataset.

Dataset	Category	t/o separate	t/o combined
NinePeach	unripe	43.72	51.27
	semiripe	49.07	53.93
	ripe	62.05 66.94	
StrawDI_Db1	rs1	45.57	42.29
	rs2	72.55	76.73
	rs3	79.87	78.40
	rs4	77.99	79.64
Overall		61.08	**64.63**

Table 8
Parameters comparison of YOLO and our FruitQuery.

	YOLOv9			FruitQuery	
Type	s	m	c	xs	s
Backbone	5.72	15.52	19.95	4.07	7.15
Neck	/	/	/	2.70	2.76
Head	**2.92**	6.74	7.89	**4.18**	**4.18**
Total (M)	**8.64**	22.26	27.84	**10.94**	14.08
AP	59.91	60.04	60.41	66.46	**67.02**

peaches and strawberries without requiring separate training for each fruit type. Second, FruitQuery demonstrates strong generalization ability on fruit size due to effective multi-scale feature fusion. Specifically, the size of peaches is relatively large compared to that of strawberries, and FruitQuery can accurately segment both large and small fruit. Third, FruitQuery maintains high robustness in complex in-field conditions, such as occlusions from tree trunks and leaves, delivering precise fruit segmentation. These indicate that our FruitQuery can accurately predict fruit locations for downstream applications.

We also compare the visualization of our FruitQuery and YOLO in Fig. 8a. In case (1), although YOLOv9-c is not an anchor-based model, it still gives an inaccurate anchor-like prediction on the strawberry, while ours provides a more precise delineation of the strawberry's shape. In case (2), YOLO-v9c's segmentation boundary tends to follow the rectangular outline of the bounding box, while ours closely tracks the actual peach boundary. Additionally, YOLO-v9c ignores the small peach behind, while ours correctly detects it. In case (3), YOLO-v9c fails to detect an evidently visible strawberry, while our model successfully identifies and segments it. In case (4), YOLO-v9c is unable to recognize a peach partially hidden in the background, whereas ours correctly distinguishes the peach despite the limited visible part.

4.7. Class activation map analysis

Class Activation Maps (CAM, [61]) is a popular visualization technique that highlights the regions in an image most influential to a model's prediction. By projecting learned feature weights back onto the original input, CAM reveals where the model allocates its attention and provides an interpretable window into the decision-making process. We illustrated the CAM comparison of YOLO and our FruitQuery in Fig. 8b.

In the CAM visualizations, YOLOv9-c exhibits relatively diffuse and occasionally misaligned attention, focusing on broader or less discriminative regions. For example, in cases (1) and (2), YOLOv9-c has uncertain attention on the fruit and is affected by the surrounding leaves. By contrast, our FruitQuery maintains a more localized and precise concentration of high-intensity activation around the fruit. This difference is particularly evident in cases (3) and (4), YOLOv9-c looks at a

Fig. 8. Comparison of YOLO and our FruitQuery.

Table 9
Inference time of FruitQuery on a single image across different devices.

Device	Format	FruitQuery-xs		FruitQuery-s	
		FP32	FP16	FP32	FP16
NVIDIA Tesla V100	.pth	0.0605	0.0544	0.0625	0.0595
	.onnx	0.2661	0.2545	0.2674	0.2563
	.trt	**0.0253**	**0.0196**	**0.0254**	**0.0198**
NVIDIA Jetson Orin Nano	.pth	0.3307	0.2246	0.3427	0.2253
	.onnx	0.3494	0.3150	0.3570	0.3242
	.trt	0.1168	0.0792	0.1216	0.0856
Apple M1	.onnx	1.1554	1.4055	1.7193	1.6731
	.mlmodel	2.5133	2.3406	2.7164	2.5292

large blur region around fruit and gives attention to the irrelevant background, while our FruitQuery accurately distinguishes between fruit and background context, capturing finer textural cues on peaches and strawberries and generating tightly focused activation zones.

Consequently, the visualizations demonstrate the enhanced ability of our FruitQuery to learn discriminative features of peaches and strawberries, such as shape, colour transition, and edge boundaries, eventually resulting in interpretable and improved segmentation performance.

4.8. Deployment

We compared inference performance of the proposed FruitQuery on different hardware platforms, including a high-performance GPU (NVIDIA Tesla V100), an edge computing device (NVIDIA Jetson Orin Nano), and a general-purpose CPU (Apple M1).

Table 9 summarizes the inference time per image using four model formats: PyTorch checkpoint (.pth), Open Neural Network Exchange (.onnx), TensorRT engine (.trt) and Core ML (.mlmodel). Two numerical precision modes are compared: FP32 (single-precision floating point) and FP16 (half-precision floating point).

Inference results demonstrate that TensorRT achieves the fastest inference across all tested hardware for both precision modes. On the Tesla V100, FruitQuery-xs runs in as little as 0.0196 seconds per image using FP16 and TensorRT, which is more than 2.5 times faster than the PyTorch baseline under FP32. Jetson Orin Nano also benefits significantly from TensorRT acceleration, achieving inference times below 0.1 seconds for FruitQuery-xs in FP16. This highlights the suitability of TensorRT for edge deployment when latency is critical.

Compared to the V100 and Jetson platforms, the Apple M1 incurs higher inference latency. The ONNX-based deployment shows inference times exceeding 1 second per image for both model variants. Moreover, Core ML deployment with .mlmodel results in even longer running time, with FruitQuery-s requiring approximately 2.5 seconds per image under FP16. Despite this, the Core ML format enables compatibility with Apple-native applications, which may be improved in future hardware iterations.

Overall, the proposed FruitQuery models demonstrate efficient inference across diverse deployment scenarios. The results demonstrate the flexibility of the models and their compatibility with multiple deployment backends and precision modes. Particularly, the combination of lightweight architectures, TensorRT optimisation, and half-precision inference enables fast inference on both cloud GPUs and edge platforms.

5. Discussion

5.1. Limitations

First, our FruitQuery depends on a relatively large and precisely annotated dataset, making it challenging to generalize seamlessly to new fruit varieties or orchard conditions without further labelling efforts.

Second, while our FruitQuery is comparably efficient, the current architecture still demands moderate computational resources, which may limit real-time applications on highly constrained edge devices. The inference speed of our FruitQuery also can be optimized. These limitations provide clear directions for improvement, such as exploring further model compression techniques.

5.2. Future work

Building on current results, we plan to compress and deploy our FruitQuery to the embedded devices for in-field fruit segmentation. By reducing model size and optimizing computational methods such as quantization, we aim to streamline orchard operations by providing immediate feedback on fruit ripeness, thereby guiding in-field harvesting robots to selectively pick the ripe fruit only.

Furthermore, we propose to expand the combined dataset by incorporating a broader range of fruit varieties, to enhance its applicability across different fruit types. Building a large-scale fruit instance segmentation dataset with ripeness labels will not only reduce redundant annotation efforts, but also accelerate the development of autonomous fruit-picking robots.

6. Conclusion

In this work, we combined two in-field fruit datasets of peaches and strawberries, which contain 3 ripeness stages for peaches and 4 ripeness stages for strawberries. Then we introduced FruitQuery, a lightweight query-based instance segmentation model for fruit ripeness determination.

The combined dataset enables training the model to handle the ripeness determination of two fruits at the same time, reducing the effort to replicate the training. FruitQuery is composed of three main components: a backbone, a pixel decoder, and Transformer decoders. We integrated EMSA modules into the backbone to reduce computational overhead, and introduced a PPM in the pixel decoder to improve multi-scale feature fusion. Transformer decoders were employed to learn a fixed number of queries for instance masks, eliminating the need for postprocessing like NMS.

By combining the advantages of convolution and Transformer, FruitQuery runs in an end-to-end way and precisely attends to fruit regions, capturing subtle distinctions in shape and ripeness. The design of FruitQuery leads to state-of-the-art performance, achieving the highest AP of 67.02 with 14.08M parameters and surpassing 13 other CNN-based and Transformed-based models. Notably, it outperforms three series of YOLO, under challenging conditions such as occlusion and varying illumination. However, FruitQuery's dependence on labelled data makes it challenging for immediate adaptation to new fruit varieties. Additionally, latency issues may be a problem for our model when applied on embedded platforms.

Moving forward, we will further optimize FruitQuery for in-field applications, exploring strategies like quantization for edge deployment. The combined dataset is planned to be expanded with more fruit varieties, ultimately building a large-scale fruit instance segmentation dataset with ripeness labels. Through these enhancements, we aim to increase FruitQuery's utility in orchard automation, enabling more accurate and efficient fruit ripeness determination and helping the development of precise agriculture.

CRediT authorship contribution statement

Ziang Zhao: Writing – original draft, Visualization, Software, Methodology, Data curation, Conceptualization. **Yulia Hicks:** Writing – review & editing, Supervision, Methodology, Formal analysis, Conceptualization. **Xianfang Sun:** Writing – review & editing, Supervision, Methodology, Formal analysis, Conceptualization. **Chaoxi Luo:** Writing – review & editing, Resources, Data curation, Conceptualization.

Declaration of competing interest

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

Acknowledgement

We appreciate the computational resources provided by Advanced Research Computing at Cardiff (ARCCA).

Data availability

The ripeness information for StrawDI_Db1 can be accessed by following this link ripeness_info.

References

[1] C.E. Basson, J.H. Groenewald, J. Kossmann, C. Cronjé, R. Bauer, Sugar and acid-related quality attributes and enzyme activities in strawberry fruits: invertase is the main sucrose hydrolysing enzyme, Food Chem. 121 (2010) 1156–1162, https://doi.org/10.1016/j.foodchem.2010.01.064, https://www.sciencedirect.com/science/article/pii/S0308814610001445.

[2] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: Computer Vision – ECCV 2020: 16th European Conference, Proceedings, Part I, Glasgow, UK, August 23–28, 2020, Springer-Verlag, Berlin, Heidelberg, 2020, pp. 213–229.

[3] J. Chen, S.h. Kao, H. He, W. Zhuo, S. Wen, C.H. Lee, S.H.G. Chan, Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks, IEEE Computer Society, 2023, pp. 12021–12031, https://www.computer.org/csdl/proceedings-article/cvpr/2023/012900m2021/1POTrhb8Tzq.

[4] S. Chen, X. Zou, X. Zhou, Y. Xiang, M. Wu, Study on fusion clustering and improved YOLOv5 algorithm based on multiple occlusion of Camellia oleifera fruit, Comput. Electron. Agric. 206 (2023) 107706, https://doi.org/10.1016/j.compag.2023.107706, https://linkinghub.elsevier.com/retrieve/pii/S0168169923000947.

[5] B. Cheng, I. Misra, A.G. Schwing, A. Kirillov, R. Girdhar, Masked-attention mask transformer for universal image segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 1290–1299, https://openaccess.thecvf.com/content/CVPR2022/html/Cheng_Masked-Attention_Mask_Transformer_for_Universal_Image_Segmentation_CVPR_2022_paper.html.

[6] B. Cheng, A.G. Schwing, A. Kirillov, Per-pixel classification is not all you need for semantic segmentation, http://arxiv.org/abs/2107.06278, https://doi.org/10.48550/arXiv.2107.06278, arXiv:2107.06278 [cs], 2021.

[7] T. Cheng, X. Wang, S. Chen, W. Zhang, Q. Zhang, C. Huang, Z. Zhang, W. Liu, Sparse instance activation for real-time instance segmentation, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, New Orleans, LA, USA, 2022, pp. 4423–4432, https://ieeexplore.ieee.org/document/9880463/.

[8] Y. Cui, C. Jiang, G. Wu, L. Wang, MixFormer: end-to-end tracking with iterative mixed attention, http://arxiv.org/abs/2302.02814, 2023, arXiv:2302.02814 [cs].

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: transformers for image recognition at scale, in: International Conference on Learning Representations (ICLR), 2020, https://openreview.net/forum?id=YicbFdNTTy.

[10] J.P. Gonçalves, F.A.C. Pinto, D.M. Queiroz, F.M.M. Villar, J.G.A. Barbedo, E.M. Del Ponte, Deep learning architectures for semantic segmentation and automatic estimation of severity of foliar symptoms caused by diseases or pests, Biosyst. Eng. 210 (2021) 129–142, https://doi.org/10.1016/j.biosystemseng.2021.08.011, https://www.sciencedirect.com/science/article/pii/S1537511021001951.

[11] J. Guo, K. Han, H. Wu, Y. Tang, X. Chen, Y. Wang, C. Xu, CMT: convolutional neural networks meet vision transformers, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, New Orleans, LA, USA, 2022, pp. 12165–12175, https://ieeexplore.ieee.org/document/9879701/.

[12] Y. Guo, Y. Lan, X. Chen, CST: Convolutional Swin Transformer for detecting the degree and types of plant diseases, Comput. Electron. Agric. 202 (2022) 107407, https://doi.org/10.1016/j.compag.2022.107407, https://www.sciencedirect.com/science/article/pii/S0168169922007153.

[13] A.M. Hafiz, G.M. Bhat, A survey on instance segmentation: state of the art, Int. J. Multimed. Inf. Retr. 9 (2020) 171–189, https://doi.org/10.1007/s13735-020-00195-x.

[14] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980–2988, ISSN: 2380-7504.

[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, ISSN: 1063-6919.

[16] W. He, Y. Mi, X. Ding, G. Liu, T. Li, Two-stream cross-attention vision transformer based on RGB-D images for pig weight estimation, Comput. Electron. Agric. 212 (2023) 107986, https://doi.org/10.1016/j.compag.2023.107986, https://www.sciencedirect.com/science/article/pii/S0168169923003745.

[17] Z. He, S.R. Khanal, X. Zhang, M. Karkee, Q. Zhang, Real-time strawberry detection based on improved YOLOv5s architecture for robotic harvesting in open-field environment, http://arxiv.org/abs/2308.03998, https://doi.org/10.48550/arXiv.2308.03998, arXiv:2308.03998 [cs], 2023.

[18] T. Huang, L. Huang, S. You, F. Wang, C. Qian, C. Xu, LightViT: towards lightweight convolution-free vision transformers, http://arxiv.org/abs/2207.05557, 2022, arXiv:2207.05557 [cs].

[19] 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, https://www.sciencedirect.com/science/article/pii/S0168169919326274.

[20] W. Jia, Z. Zhang, W. Shao, S. Hou, Z. Ji, G. Liu, X. Yin, FoveaMask: a fast and accurate deep learning model for green fruit instance segmentation, Comput. Electron. Agric. 191 (2021) 106488, https://doi.org/10.1016/j.compag.2021.106488, https://www.sciencedirect.com/science/article/pii/S0168169921005056.

[21] Z. Jiao, K. Huang, G. Jia, H. Lei, Y. Cai, Z. Zhong, An effective litchi detection method based on edge devices in a complex scene, Biosyst. Eng. 222 (2022) 15–28, https://doi.org/10.1016/j.biosystemseng.2022.07.009, https://www.sciencedirect.com/science/article/pii/S1537511022001714.

[22] Jocher Glenn, Jing Qiu, Ayush Chaurasia, Ultralytics YOLO, https://github.com/ultralytics/ultralytics, 2023.

[23] H. Kang, C. Chen, Fruit detection, segmentation and 3D visualisation of environments in apple orchards, Comput. Electron. Agric. 171 (2020) 105302, https://doi.org/10.1016/j.compag.2020.105302, https://www.sciencedirect.com/science/article/pii/S0168169919323889.

[24] R. Kestur, A. Meduri, O. Narasipura, MangoNet: a deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard, Eng. Appl. Artif. Intell. 77 (2019) 59–69, https://doi.org/10.1016/j.engappai.2018.09.011, https://www.sciencedirect.com/science/article/pii/S0952197618301970.

[25] A. Kirillov, Y. Wu, K. He, R. Girshick, PointRend: image segmentation as rendering, http://arxiv.org/abs/1912.08193, https://doi.org/10.48550/arXiv.1912.08193, arXiv:1912.08193 [cs], 2020.

[26] G. Li, L. Jiao, P. Chen, K. Liu, R. Wang, S. Dong, C. Kang, Spatial convolutional self-attention-based transformer module for strawberry disease identification under complex background, Comput. Electron. Agric. 212 (2023) 108121, https://doi.org/10.1016/j.compag.2023.108121, https://www.sciencedirect.com/science/article/pii/S0168169923005094.

[27] J. Li, X. Xia, W. Li, H. Li, X. Wang, X. Xiao, R. Wang, M. Zheng, X. Pan, Next-ViT: next generation vision transformer for efficient deployment in realistic industrial scenarios, http://arxiv.org/abs/2207.05501, https://doi.org/10.48550/arXiv.2207.05501, arXiv:2207.05501 [cs], 2022.

[28] Q. Li, W. Jia, M. Sun, S. Hou, Y. Zheng, A novel green apple segmentation algorithm based on ensemble U-Net under complex orchard environment, Comput. Electron. Agric. 180 (2021) 105900, https://doi.org/10.1016/j.compag.2020.105900, https://www.sciencedirect.com/science/article/pii/S0168169920331057.

[29] C. Liang, J. Xiong, Z. Zheng, Z. Zhong, Z. Li, S. Chen, Z. Yang, A visual detection method for nighttime litchi fruits and fruiting stems, Comput. Electron. Agric. 169 (2020) 105192, https://doi.org/10.1016/j.compag.2019.105192, https://www.sciencedirect.com/science/article/pii/S0168169919313274.

[30] T.Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 936–944, https://ieeexplore.ieee.org/document/8099589, ISSN: 1063-6919.

[31] J. Lu, P. Chen, C. Yu, Y. Lan, L. Yu, R. Yang, H. Niu, H. Chang, J. Yuan, L. Wang, Lightweight green citrus fruit detection method for practical environmental applications, Comput. Electron. Agric. 215 (2023) 108205, https://doi.org/10.1016/j.compag.2023.108205, https://www.sciencedirect.com/science/article/pii/S0168169923005938.

[32] Z. Ma, N. Dong, J. Gu, H. Cheng, Z. Meng, X. Du, STRAW-YOLO: a detection method for strawberry fruits targets and key points, Comput. Electron. Agric. 230 (2025) 109853, https://doi.org/10.1016/j.compag.2024.109853, https://linkinghub.elsevier.com/retrieve/pii/S0168169924012444.

[33] B.J. McGuinness, M.D. Duke, K.C. Au, H.S. Lim, Field factory for the automated harvesting of forestry tree stock, Biosyst. Eng. 227 (2023) 52–67, https://doi.org/10.1016/j.biosystemseng.2023.01.003, https://www.sciencedirect.com/science/article/pii/S1537511023000089.

[34] S. Mehta, M. Rastegari, MobileViT: light-weight, general-purpose, and mobile-friendly vision transformer, http://arxiv.org/abs/2110.02178, 2022, https://doi.org/10.48550/arXiv.2110.02178, arXiv:2110.02178 [cs].

[35] L. Mu, G. Cui, Y. Liu, Y. Cui, L. Fu, Y. Gejima, Design and simulation of an integrated end-effector for picking kiwifruit by robot, Inf. Process. Agric. 7 (2020) 58–71, https://doi.org/10.1016/j.inpa.2019.05.004, https://www.sciencedirect.com/science/article/pii/S2214317318304372.

[36] X. Ni, C. Li, H. Jiang, F. Takeda, Deep learning image segmentation and extraction of blueberry fruit traits associated with harvestability and yield, Hortic. Res. 7 (2020) 110, https://doi.org/10.1038/s41438-020-0323-3.

[37] Z. Ning, L. Luo, X. Ding, Z. Dong, B. Yang, J. Cai, W. Chen, Q. Lu, Recognition of sweet peppers and planning the robotic picking sequence in high-density orchards, Comput. Electron. Agric. 196 (2022) 106878, https://doi.org/10.1016/j.compag.2022.106878, https://www.sciencedirect.com/science/article/pii/S0168169922001958.

[38] B. Niu, Q. Feng, B. Chen, C. Ou, Y. Liu, J. Yang, HSI-TransUNet: a transformer based semantic segmentation model for crop mapping from UAV hyperspectral imagery, Comput. Electron. Agric. 201 (2022) 107297, https://doi.org/10.1016/j.compag.2022.107297, https://www.sciencedirect.com/science/article/pii/S0168169922006093.

[39] M. Peebles, S.H. Lim, M. Duke, B. McGuinness, Investigation of optimal network architecture for asparagus spear detection in robotic harvesting, IFAC-PapersOnLine 52 (2019) 283–287, https://doi.org/10.1016/j.ifacol.2019.12.535, https://www.sciencedirect.com/science/article/pii/S2405896319324541.

[40] I. Pérez-Borrero, D. Marín-Santos, M.E. Gegúndez-Arias, E. Cortés-Ancos, A fast and accurate deep learning method for strawberry instance segmentation, Comput. Electron. Agric. 178 (2020) 105736, https://doi.org/10.1016/j.compag.2020.105736, https://www.sciencedirect.com/science/article/pii/S0168169920300624.

[41] R. Ren, S. Zhang, H. Sun, N. Wang, S. Yang, H. Zhao, M. Xin, YOLO-RCS: a method for detecting phenological period of 'Yuluxiang' pear in unstructured environment, Comput. Electron. Agric. 229 (2025) 109819, https://doi.org/10.1016/j.compag.2024.109819, https://linkinghub.elsevier.com/retrieve/pii/S0168169924012109.

[42] T.T. Santos, L.L. de Souza, A.A. dos Santos, S. Avila, Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association, Comput. Electron. Agric. 170 (2020) 105247, https://doi.org/10.1016/j.compag.2020.105247, https://www.sciencedirect.com/science/article/pii/S0168169919315765.

[43] Q. Shen, X. Zhang, M. Shen, D. Xu, Multi-scale adaptive YOLO for instance segmentation of grape pedicels, Comput. Electron. Agric. 229 (2025) 109712, https://doi.org/10.1016/j.compag.2024.109712, https://linkinghub.elsevier.com/retrieve/pii/S0168169924011037.

[44] X. Sheng, C. Kang, J. Zheng, C. Lyu, An edge-guided method to fruit segmentation in complex environments, Comput. Electron. Agric. 208 (2023) 107788, https://doi.org/10.1016/j.compag.2023.107788, https://www.sciencedirect.com/science/article/pii/S016816992300176X.

[45] M. Sun, R. Zhao, X. Yin, L. Xu, C. Ruan, W. Jia, FBoT-Net: focal bottleneck transformer network for small green apple detection, Comput. Electron. Agric. 205 (2023) 107609, https://doi.org/10.1016/j.compag.2022.107609, https://www.sciencedirect.com/science/article/pii/S0168169922009176.

[46] J.F. Sánchez-Sevilla, J.G. Vallarino, S. Osorio, A. Bombarely, D. Posé, C. Merchante, M.A. Botella, I. Amaya, V. Valpuesta, Gene expression atlas of fruit ripening and transcriptome assembly from RNA-seq data in octoploid strawberry (Fragaria × ananassa), Sci. Rep. 7 (2017) 13737, https://doi.org/10.1038/s41598-017-14239-6, https://www.nature.com/articles/s41598-017-14239-6, publisher: Nature Publishing Group.

[47] H.T. Thai, K.H. Le, N.L.T. Nguyen, FormerLeaf: an efficient vision transformer for Cassava Leaf Disease detection, Comput. Electron. Agric. 204 (2023) 107518, https://doi.org/10.1016/j.compag.2022.107518, https://www.sciencedirect.com/science/article/pii/S0168169922008262.

[48] M. Trivedi, A. Gupta, Automatic monitoring of the growth of plants using deep learning-based leaf segmentation, Int. J. Appl. Sci. Eng. 18 (2021) 1–9, https://doi.org/10.6703/IJASE.202106_18(2).003, https://gigvvy.com/journals/ijase/articles/ijase-202106-18-2-003, publisher: Chaoyang University of Technology.

[49] J. Wang, Z. Zhang, L. Luo, H. Wei, W. Wang, M. Chen, S. Luo, DualSeg: fusing transformer and CNN structure for image segmentation in complex vineyard environment, Comput. Electron. Agric. 206 (2023) 107682, https://doi.org/10.1016/j.compag.2023.107682, https://www.sciencedirect.com/science/article/pii/S0168169923000704.

[50] W. Wang, E. Xie, X. Li, D.P. Fan, K. Song, D. Liang, T. Lu, P. Luo, L. Shao, Pyramid vision transformer: a versatile backbone for dense prediction without convolutions, in: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, Montreal, QC, Canada, 2021, pp. 548–558, https://ieeexplore.ieee.org/document/9711179/.

[51] Y. Wu, A. Kirillov, F. Massa, W.Y. Lo, Ross Girshick, Detectron2, https://github.com/facebookresearch/detectron2, 2019.

[52] X. Xia, X. Chai, Z. Li, N. Zhang, T. Sun, MTYOLOX: multi-transformers-enabled YOLO for tree-level apple inflorescences detection and density mapping, Comput. Electron. Agric. 209 (2023) 107803, https://doi.org/10.1016/j.compag.2023.107803, https://www.sciencedirect.com/science/article/pii/S0168169923001916.

[53] B. Xiao, M. Nguyen, W.Q. Yan, Fruit ripeness identification using YOLOv8 model, Multimed. Tools Appl. (2023), https://doi.org/10.1007/s11042-023-16570-9.

[54] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J.M. Alvarez, P. Luo, SegFormer: simple and efficient design for semantic segmentation with transformers, http://arxiv.org/abs/2105.15203, arXiv:2105.15203 [cs], 2021.

[55] S. Yang, W. Wang, S. Gao, Z. Deng, Strawberry ripeness detection based on YOLOv8 algorithm fused with LW-Swin Transformer, Comput. Electron. Agric. 215 (2023) 108360, https://doi.org/10.1016/j.compag.2023.108360, https://www.sciencedirect.com/science/article/pii/S0168169923007482.

[56] J. Yu, Y. Bai, S. Yang, J. Ning, Stolon-YOLO: a detecting method for stolon of strawberry seedling in glass greenhouse, Comput. Electron. Agric. 215 (2023) 108447, https://doi.org/10.1016/j.compag.2023.108447, https://linkinghub.elsevier.com/retrieve/pii/S0168169923008359.

[57] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, S. Yan, MetaFormer is actually what you need for vision, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 10809–10819, https://ieeexplore.ieee.org/document/9879612, ISSN: 2575-7075.

[58] Y. Yu, K. Zhang, L. Yang, D. Zhang, Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN, Comput. Electron. Agric. 163 (2019) 104846, https://doi.org/10.1016/j.compag.2019.06.001, https://www.sciencedirect.com/science/article/pii/S0168169919301103.

[59] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, http://arxiv.org/abs/1612.01105, arXiv:1612.01105 [cs], 2017.

[60] Z. Zhao, Y. Hicks, X. Sun, C. Luo, Peach ripeness classification based on a new one-stage instance segmentation model, Comput. Electron. Agric. 214 (2023) 108369, https://doi.org/10.1016/j.compag.2023.108369, https://www.sciencedirect.com/science/article/pii/S0168169923007573.

[61] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA, 2016, pp. 2921–2929, http://ieeexplore.ieee.org/document/7780688/.

[62] X. Zhu, F. Chen, Y. Zheng, C. Chen, X. Peng, Detection of Camellia oleifera fruit maturity in orchards based on modified lightweight YOLO, Comput. Electron. Agric. 226 (2024) 109471, https://doi.org/10.1016/j.compag.2024.109471, https://linkinghub.elsevier.com/retrieve/pii/S0168169924008627.