ORIGINAL RESEARCH **Open access**

Lightweight Query-Driven Instance Segmentation for Multi-Stage Fruit Ripeness Detection in Field Environments

Hans Müller[1*], Anna Schmid[1], Thomas Weber[2], Lukas Hoffmann[3], Petra Klein[1]

Abstract

Precise pixel-level delineation of fruit across different ripeness stages is vital for the development of autonomous harvesting platforms, particularly in unconstrained field settings. This work merges two in-field datasets covering peaches and strawberries to enable multi-stage ripeness identification and proposes a lightweight, query-driven instance segmentation architecture, termed FruitQuery. The unified dataset encompasses 3 ripeness phases for peach and 4 for strawberry, capturing diverse unstructured environments for these two common crops. The FruitQuery architecture is built from three core components: a backbone network, a pixel decoder, and Transformer decoders. Efficient multi-head self-attention mechanisms are embedded into the backbone to cut computational demands. At the same time, a pyramid pooling module is appended to the pixel decoder to strengthen multi-scale feature integration. Transformer decoders are then leveraged to extract a fixed set of queries from the feature maps and directly output instance masks, thereby circumventing the need for post-processing steps such as non-maximum suppression. FruitQuery functions in a fully end-to-end manner and merges convolutional operations with Transformer blocks to seize fine-grained details tied to distinct fruits at varying ripeness stages. Thorough evaluations on the aggregated fruit dataset reveal that the proposed FruitQuery achieves an average precision of 67.02 with only 14.08M parameters, exceeding 13 leading-edge models across 33 variants. Notably, FruitQuery outstrips three generations of YOLO (v8, v9, and v10) by a considerable gap. Ablation experiments and visual analyses further confirm its robust feature extraction with a reduced parameter count, indicating that the query-centric design is efficient at pinpointing fruit. These findings emphasize the model's favorable trade-off between segmentation accuracy and compactness, suggesting its viability for real-world field deployment.

Keywords Lightweight, Query-driven, Segmentation, Fruit, Ripeness

*Correspondence:
Hans Müller
hans.mueller@uni-bonn.de
[1] Department of Agricultural Data Science and Phenotyping, Faculty of Life Sciences, University of Bonn, Bonn, Germany
[2] Department of Machine Learning for Soil Fertility Mapping, Faculty of Agriculture, LMU Munich, Munich, Germany
[3] Department of Smart Drone Surveillance, Faculty of Agriculture, University of Stuttgart, Stuttgart, Germany

Introduction

Background

The rapid evolution of artificial intelligence has established digitization, precision agriculture, and intelligent farming as pivotal forces in advancing agricultural modernization. In automated fruit production, the swift and accurate localization of target produce is the essential foundation for constructing a fruit-harvesting robot that interprets its operational environment and steers the robotic manipulator to detach the fruit. In recent times, diverse harvesting robots have been

This is an open access article distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0), which permits non-commercial use, distribution, reproduction, and adaptation in any medium or format, provided that appropriate credit is given to the original author(s) and source, a link to the license is provided, and any changes made are indicated. Adapted material must be distributed under the same license.

© 2022 The Author(s). | Published by Agro Intelligence Research Press

engineered for various crops, such as kiwifruit [1], apple [2], litchi [3, 4], strawberry [5], green citrus [6], radiata pine [7], sweet pepper [8], and asparagus spear [9].

Multiple computer vision tasks, encompassing classification, detection, and segmentation, have found application in the agricultural domain. Segmentation is among the most promising fields for practical adoption, offering accurate pixel-wise object predictions and being widely used across numerous tasks. Segmentation approaches are divided into semantic and instance segmentation. Specifically, instance segmentation assigns separate labels to individual objects of the same class, a process that is inherently more computationally intensive and complex than semantic segmentation [10]. This technique has been deployed in agricultural applications, including mango yield forecasting [11], leaf disease diagnosis [12], plant growth tracking [13], and automated fruit plucking [14].

Concerning fruit detection and segmentation, models can be broadly categorized into those rooted in Convolutional Neural Networks (CNNs) and those based on Transformers.

CNN-based Models

CNN-driven architectures have long prevailed in visual recognition, with Mask R-CNN [15] serving as a trailblazer for instance segmentation. Santos *et al.* [16] demonstrated Mask R-CNN's proficiency in detecting, segmenting, and tracking grape bunches, confirming its robust performance across marked variations in shape, color, size, and density. A Mask R-CNN model was trained and evaluated to detect and segment individual blueberries from four cultivars at two maturity levels [17]. Likewise, Mask R-CNN and its tailored variants were leveraged to craft strawberry fruit detectors that exhibit superior generalizability and resilience compared to classic machine vision techniques [18, 19]. Jia *et al.* [20] advanced Mask R-CNN by integrating ResNet and DenseNet as feature-extraction backbones to build a vision system for picking robots, thereby drastically elevating apple recognition performance under overlapping, clustering, and occlusion scenarios.

Additionally, alternative CNN-centric instance segmentation solutions have surfaced. A rapid segmentation model for green fruits, designated FoveaMask, was devised by injecting a position-aware attention module into the embedded mask branch to pool valuable pixel-level information and enhance robustness [21]. Kang and Chen [2] proposed DaSNet-v2, which fuses an instance segmentation stream with a semantic segmentation stream within a single-stage detection framework, achieving both fruit instance segmentation and branch-level semantic segmentation. Sheng *et al.* [22] developed an edge-guided fruit segmentation framework that incorporates purpose-built modules to pinpoint prospective target zones and refine boundary precision.

It is worth noting that the YOLO (You Only Look Once) family of models has seen extensive deployment for fruit detection, categorization, and ripeness evaluation in recent years. For illustration, Yang *et al.* [23] rolled out LS-YOLOv8s by fusing YOLOv8s with an LW-Swin Transformer block to detect and classify strawberry ripeness. Xiao *et al.* [24] utilized YOLOv8 to sort apple ripeness into three grades. Moreover, He *et al.* [5] engineered a real-time, refined YOLOv5 tailored for robotic strawberry collection, Yu *et al.* [25] introduced Stolon-YOLO to detect strawberry stolons in greenhouse conditions, and Chen *et al.* [26] combined fusion clustering with YOLOv5 to handle Camellia oleifera fruit identification under severe occlusions. Additional expansions cover Zhu *et al.* [27] 's reworked lightweight YOLO for orchard-based C. oleifera fruit maturity evaluation, Ren *et al.* [28] 's YOLO-RCS for recognizing phenological phases of Yuluxiang pears, Shen *et al*. [29] 's multi-scale adaptive YOLO for grape pedicel instance segmentation, and Ma *et al.* [30] 's STRAW-YOLO for pinpointing strawberries and their critical keypoints. These investigations underscore the pervasiveness of YOLO models across a breadth of fruit-centered tasks.

Transformer-based Models

More recently, Vision Transformers (ViT, [31]) have garnered increasing research focus, operating under a sequence-to-sequence prediction paradigm. Compared with prevailing CNN-based visual perception systems, ViTs offer greater flexibility for capturing long-range dependencies in vision tasks and exhibit reduced inductive bias. Accordingly, it is

© 2022 The Author(s). | Published by Agro Intelligence Research Press

anticipated that ViTs will either supplant or amalgamate with CNNs as the foundational building block for next-generation visual perception frameworks. Building on the original ViT, subsequent models such as DETR [32], MaskFormer [33], and SegFormer [34] have been introduced.

Transformer-centric models have been integrated with CNN-based methods to tackle segmentation problems. Niu *et al.* [35] presented HSI-TransUNet, a semantic segmentation model for crop mapping that leverages plentiful spatial and spectral cues from UAV HSI data. Wang *et al.* [36] unveiled DualSeg, a parallel network topology that harnesses CNN advantages in local feature processing and Transformer strengths in global contextual interaction for grape peduncle segmentation. Guo *et al.* [37] proposed a convolutional variant of the Swin Transformer to classify disease type and severity. A novel Transformer-infused CNN model, MTYOLOX, was launched for the robust identification of whole-tree inflorescences in unpredictable and demanding orchard settings [38].

Additionally, Transformer architectures have proven effective in relevant agricultural contexts. Sun *et al.* [39] described FBoT-Net, a focal bottleneck Transformer network that merges high-level semantic insights with potent representational capacity and both global and local feature information via the focal bottleneck Transformer module for small green apple detection. Thai *et al.* [40] proposed FormerLeaf, a Transformer-powered leaf disease identification model, along with the Least Important Attention Pruning algorithm to filter the most salient attention heads within each Transformer layer. An enhanced Transformer-based method for strawberry disease classification was developed to achieve precise, rapid multi-class disease recognition [41]. He *et al.* [42] proposed a two-stream cross-attention ViT that extracts textural appearance and spatial structure to regress pig weight from paired RGB and depth imagery.

Contributions

While Transformer-centric architectures have sparked considerable enthusiasm, the bulk of related agricultural research continues to focus on CNN-based frameworks, with YOLO variants remaining the most prevalent choice. Investigations employing Transformer-based designs for fruit instance delineation and ripeness classification are notably sparse. Addressing this void is imperative, given the operational requirements for precise, resilient techniques in complex orchard landscapes. Simultaneously, achieving a compact yet efficient segmentation pipeline remains a persistent obstacle, as numerous field-deployable robotic units operate under strict computational and power budgets.

In addition, publicly shared fruit databases that feature both instance-level masks and ripeness stage labels are scarce. Although certain datasets include segmentation ground truths, they seldom incorporate annotations reflecting distinct ripeness phases, thereby slowing progress in multi-stage ripeness analysis.

To address these concerns, we make a unified fruit corpus available and propose a lightweight Transformer-based instance segmentation framework tailored for fruit ripeness recognition. The key contributions we offer are outlined as follows.

1. The StrawDI_Db1 strawberry collection received annotations for 4 ripeness levels; these ripeness designations are now released for public use.

2. The NinePeach and StrawDI_Db1 corpora were fused to facilitate multi-fruit segmentation under intricate real-world imaging conditions.

3. FruitQuery is a lightweight, query-conditioned architecture that merges convolutional and Transformer-derived representations for end-to-end ripeness segmentation.

4. FruitQuery achieves 67.02 AP with only 14.08M parameters, exceeding the performance of 13 competing models, including three iterations of YOLO (v8, v9, v10).

© 2022 The Author(s). | Published by Agro Intelligence Research Press

Datasets

Overview

In this work, two publicly available fruit datasets, NinePeach [43] and StrawDI_Db1 [19], were consolidated to establish a common evaluation standard for fruit instance segmentation. Representative image samples are shown in **Figure 1.** Each collection provides pixel-accurate, per-instance mask annotations for every fruit depicted in the imagery.



Figure 1. Sample imagery from NinePeach (left) and StraDI_Db1 (right).

Merging an arboreal fruit (peach) with a ground-dwelling berry (strawberry) exposes the assembled corpus to varied canopy configurations, overlapping patterns, and textural backdrops. Such richness yields a more demanding and inclusive testbed for segmentation algorithms, compelling them to generalize across dissimilar orchard contexts and fruit geometries.

NinePeach dataset. Building on our earlier effort [43], this compilation is the most extensive and diverse peach repository among publicly available alternatives. It holds 4599 images (1024×768) spanning nine peach varieties, all captured under ambient daylight and authentic cultivation conditions. The depicted peaches include scenarios such as fluctuating natural illumination, dense clustering of multiple fruits, and blockages caused by stems or leaves. The fruit manifests in several physical states: solitary peaches, fruits situated in tight proximity, specimens partially concealed by foliage or stalks, and fruits receiving backlighting.

The collection is split into a training split (3240 images) and a validation split (1359 images), with each peach assigned to one of three ripeness categories: unripe, semiripe, or ripe.

StrawDI_Db1 dataset. This repository contains 3100 images (1008×756) randomly selected from a large pool of strawberry photographs collected across 20 growing sites. The pictures were obtained across a range of lighting intensities, at roughly 20 cm from the planting ridge, throughout an entire harvest cycle. The dataset is organized into a training segment (2800 images), a validation segment (100 images), and a testing segment (200 images). The training and testing portions are utilized in this study.

Notably, this collection provides only class-agnostic delineations of strawberries, lacking any indication of ripeness. The following subsection details our strategy for overcoming this limitation.

StrawDI_Db1 ripeness annotation

Building on prior studies [44, 45], four ripeness phases were adopted to discriminate among strawberries in the StrawDI_Db1 corpus, using the classification guidelines summarized in **Table 1**. To carry out this sorting, we implement a straightforward yet powerful pipeline for segmenting strawberries into four groups, as shown in **Figure 2**.

Table 1. Four ripeness phases of the strawberry.

© 2022 The Author(s). | Published by Agro Intelligence Research Press

Category	Rephrased description
rs1 (Green)	Characterized by a dark green appearance with comparatively smaller sizes.
rs2 (White)	Represents the expansion stage, exhibiting a white coloration.
rs3 (Turning)	Contains less than 90% red coloration and is considered immature for harvesting.
rs4 (Red)	Exhibits more than 90% red coloration, indicating that the fruit is edible and suitable for harvest.

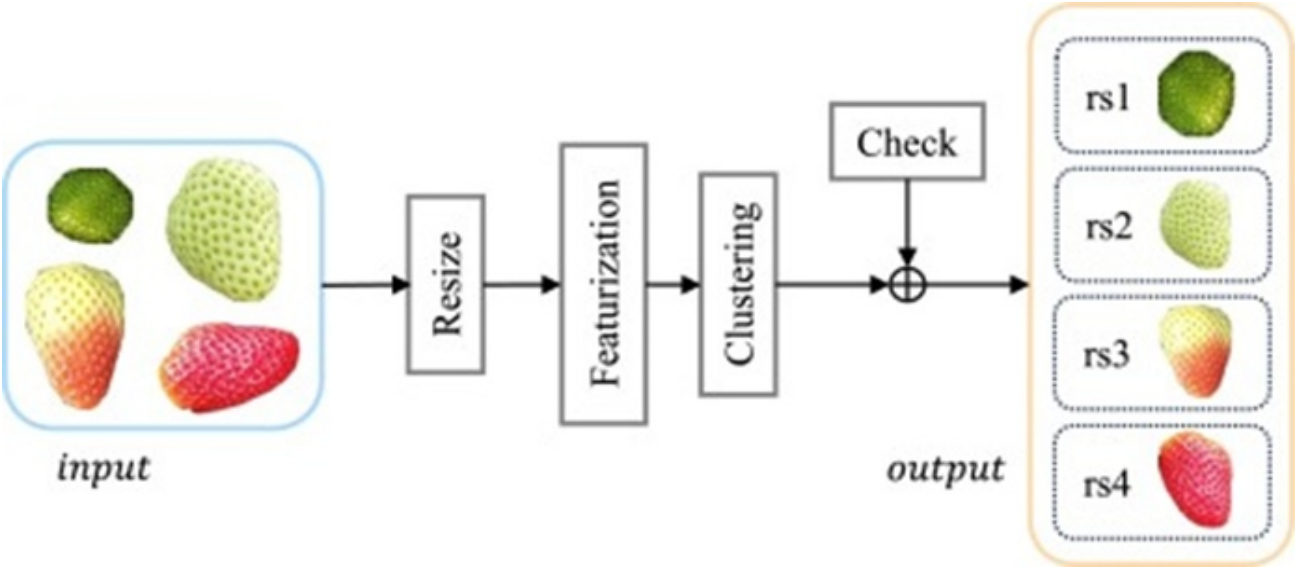

Figure 2. Workflow for strawberry mask categorization.

As a first step, individual strawberry instances are excised from the source images, and background pixels are discarded, on the premise that surrounding scene information would introduce noise rather than aid correct classification. Every extracted strawberry instance is subsequently scaled to a uniform resolution of 280×280 pixels.

The next phase involves deploying select machine learning descriptors, including Histogram of Oriented Gradients, as well as deep learning extractors such as pre-trained CNN models, to compute representations of the resized strawberry instances. Cosine similarity is then applied to gauge the distances among the derived representations, yielding similarity matrices.

In the following stage, K-means clustering is applied to decompose the similarity matrices into four distinct groupings. The clustering variant with the highest fidelity was retained to generate the preliminary assignments.

As a final step, the clustering outputs were manually scrutinized and corrected to ensure alignment with the established ripeness definitions. This polishing procedure guaranteed that the finalized cluster designations met the targeted specifications.

Dataset summary

To summarize, our investigation leverages two large fruit collections, NinePeach and StrawDI_Db1, each with per-instance mask annotations and ripeness-phase identifiers. Peaches and strawberries rank among the most broadly cultivated and ingested fruits worldwide. By unifying these two sources, we thoroughly cover heterogeneous conditions involving both tree-borne produce (peaches) and berry-type produce (strawberries).

© 2022 The Author(s). | Published by Agro Intelligence Research Press

The integrated dataset spans 7 unique categories, partitioned as 3 for peaches and 4 for strawberries. This carefully assembled structure captures the full spectrum of fruit maturation, supporting more faithful and transferable deductions in downstream analyses. Illustrations of image samples accompanied by their ground-truth labels are shown in **Figure 3**, while the tally of instances per category is compiled in **Table 2**. A noticeable trend is the gradual decline in fruit instance counts as ripening advances, mirroring an authentic dynamic that tracks the natural growth and maturing trajectory of the produce.



Figure 3. Instance annotation samples from NinePeach (left) and StraDI_Db1 (right).

Table 2. Class-wise instance distribution for the merged dataset.

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

By exposure to a fused training corpus, the model learns to navigate these intricacies across distinct object categories, thereby strengthening its resilience. Equally, the presence of dissimilar fruit types within a single integrated dataset can sharpen the model's discriminative capacity across classes, making it more flexible for practical deployments where multiple fruit categories often appear side by side.

Proposed Model

Model structure

For fruit ripeness identification, we devise FruitQuery, an instance segmentation framework inspired by the Mask2Former blueprint [46]. Its construction involves a backbone, a pixel decoder, and Transformer decoders. The schematic layout is provided in **Figure 4**.

© 2022 The Author(s). | Published by Agro Intelligence Research Press

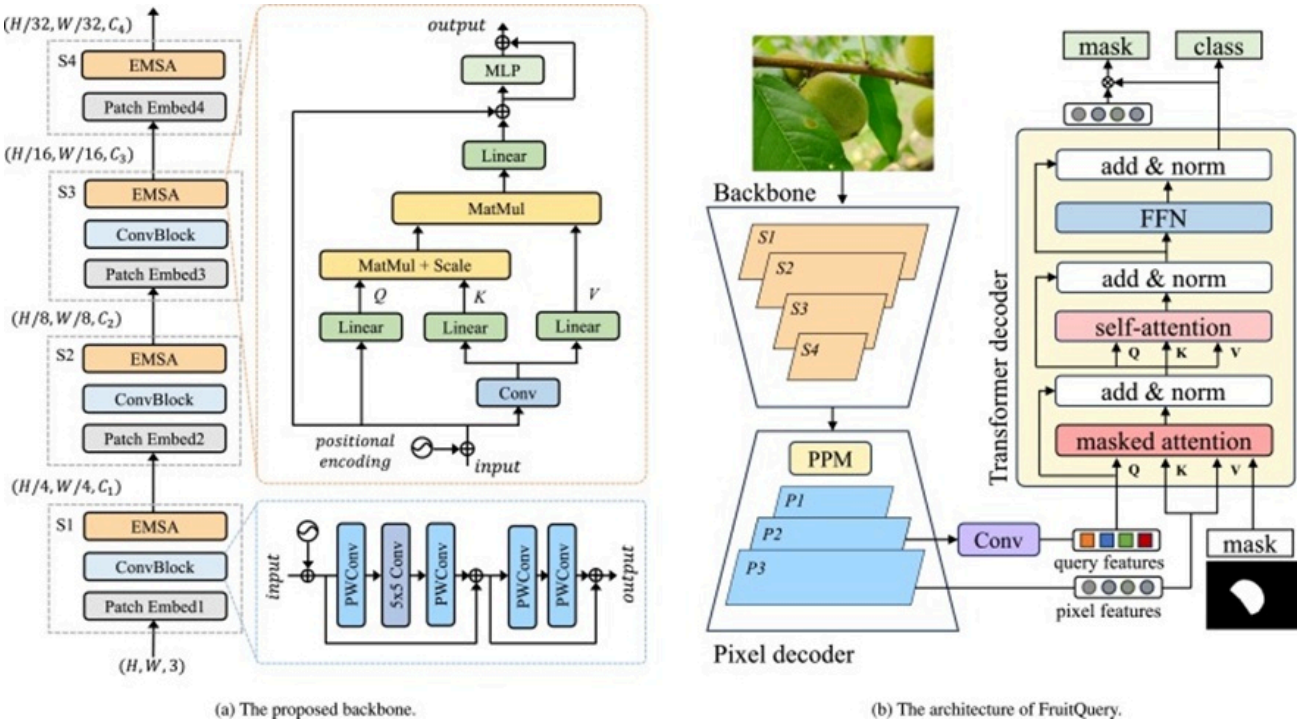

Figure 4. The proposed backbone and the overall FruitQuery architecture.

Backbone

It is commonly understood that convolutional layers exhibit inherent biases toward local patterns and translation invariance, enabling them to extract elementary, spatially confined features. Self-attention layers, by contrast, operate over a global receptive field and can assimilate long-distance contextual cues spanning the entire image. Accordingly, these two-layer families are jointly employed to assemble a backbone for hierarchical feature extraction. The layout of the proposed backbone is shown in **Figure 4a**.

By combining convolutional layers, which tend to generalize more robustly, with self-attention layers, which possess greater model capacity and learning aptitude, we postulate that the backbone can achieve enhanced generalization while retaining stronger representational power. Given an input image, the backbone yields feature maps at 4 discrete scales, providing both high-resolution coarse descriptors and low-resolution fine-grained descriptors, which typically improve fruit segmentation outcomes. In particular, the ConvBlock is removed from the final stage to keep the total parameter budget low.

Patch Embedding. The input image is carved into a grid of non-overlapping patches, each covering a square region of the image and being mapped to a fixed-length embedding vector. Depending on the chosen patch dimensions and embedding sizes, 4 distinct patch embedding blocks are prepended to their respective stages. Because patch embedding does not intrinsically conserve spatial layout within a patch, positional encodings must be supplemented for the latter two blocks.

ConvBlock. The ConvBlock is built from a sequence of convolutional layers organized around two residual pathways. In the first residual arm, two 1×1 point-wise convolutions (PWConv) flank a central 5×5 convolutional layer. The 5×5 kernel provides a wider receptive field for attending to larger local neighborhoods, enabling the detection of broader structural elements such as fruit contours and textural signatures. In the second residual arm, a pair of 1×1 point-wise convolutions mimics an MLP-like behavior: the channel dimensionality is initially quadrupled and subsequently reduced to the target output size. This arrangement is intended to amplify nonlinear representational capacity and cultivate richer feature

© 2022 The Author(s). | Published by Agro Intelligence Research Press

hierarchies, thereby bolstering model accuracy and flexibility. Owing to their per-pixel operation and modest parameter count, 1×1 point-wise convolutions are well-suited for dimensional inflation and deflation.

Efficient Multi-head Self-Attention (EMSA). Inside each head of the multi-head self-attention module, the query Q, key K, and value V are produced by subjecting the input embeddings—including positional encodings—to three separate linear mappings. Q, K, and V share a common shape N×C, where N = H×W. Attention scores are then derived via scaled dot-product attention. These scores undergo Softmax normalization to yield attention weights, which are subsequently used to compute a weighted sum over the V vectors of all tokens, as described in **Eq. 1**, where dk denotes the key dimensionality. Tokens receiving higher weights exert a proportionally larger influence on the self-attention output.

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

The chief computational bottleneck of the self-attention layer resides in its $O(N^2)$ cost, which scales quadratically with the spatial extent of the input embedding. To relieve this burden, we adopt an efficient multi-head self-attention (EMSA) based on the spatial reduction strategy introduced in PVT [47]. The central notion is to compress the sequence length by a reduction ratio R. To trim the computational load, an input sequence of shape (C, H·W) is first restructured into $\hat{K}$ of shape ($C \cdot R^2$, $H \cdot W/R^2$) in accordance with **Eq. 2.** Here, a convolutional layer with kernel_size=R and stride=R performs the restructuring operation. **Eq. 3** specifies a linear layer that ingests $\hat{K}$ and emits a new K' of shape (C, $H \cdot W/R^2$).

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

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

As a consequence, the complexity of the efficient self-attention mechanism drops from $O(N^2)$ to $O(N^2/R^2)$. It is worth remarking that a residual MLP block is appended after EMSA to fortify model capacity and counteract overfitting.

To serve a spectrum of deployment scenarios, we crafted two distinct backbone configurations (s and xs). Their detailed specifications are summarized in **Table 3**, where C denotes the embedding dimensionality, and B denotes the number of blocks.

Table 3. Specifications of our backbone configurations.

Stage	Size	Layer	s	xs
*S*1	*H*4×*W*4	Patch Embed1		Patch Size = 4, $C = C_1$
		EMSA	$C_1 = 48$	$C_1 = 36$
			$B_1 = 1$	$B_1 = 1$
			$R_1 = 4$	$R_1 = 4$

© 2022 The Author(s). | Published by Agro Intelligence Research Press

*S*2	*H*8×*W*8	Patch Embed2		Patch Size = 2, $C = C_2$
		EMSA	$C_2 = 96$	$C_2 = 72$
			$B_2 = 1$	$B_2 = 1$
			$R_2 = 2$	$R_2 = 2$
*S*3	*H*16×*W*16	Patch Embed3		Patch Size = 2, $C = C_3$
		EMSA	$C_3 = 240$	$C_3 = 144$
			$B_3 = 3$	$B_3 = 3$
			$R_3 = 2$	$R_3 = 2$
*S*4	*H*32×*W*32	Patch Embed4		Patch Size = 2, $C = C_4$
		EMSA	$C_4 = 384$	$C_4 = 288$
			$B_4 = 2$	$B_4 = 1$
			$R_4 = 1$	$R_4 = 1$

Pixel decoder

Hierarchical contextual feature maps are instrumental for image segmentation, yet deploying a complex multi-scale feature pyramid structure markedly inflates computational demands. As a case in point, the multi-scale deformable attention used in Mask2Former achieves commendable performance but also incurs a considerable parameter overhead. To construct a lightweight yet potent model, the Feature Pyramid Network (FPN, [48]) is selected as the pixel decoder, occupying less than half the footprint of multi-scale deformable attention. The FPN ingests feature maps produced by the backbone at multiple levels (S1, S2, S3, and S4) and assembles a top-down feature pyramid (P1, P2, and P3) via lateral links (S4−P1, S3−P2, S2−P3).

A pyramid pooling module (PPM, [49]) is attached to the apex layer P1 to broaden the receptive field and blend multi-scale cues; its inner workings are illustrated in **Figure 5**. The incoming feature is subdivided into several regions of varying dimensions through four adaptive average pooling operations, harvesting information at disparate receptive field scales. The pooled representations are then upsampled to match the input resolution and concatenated with the original features, producing a tensor of shape (C+4N, H, W). Finally, a plain convolutional layer restores the shape from (C+4N, H, W) back to (C, H, W) and fuses the aggregated information. Because pooling introduces no extra parameters, equipping the model with PPM improves accuracy without significantly increasing the computational burden.

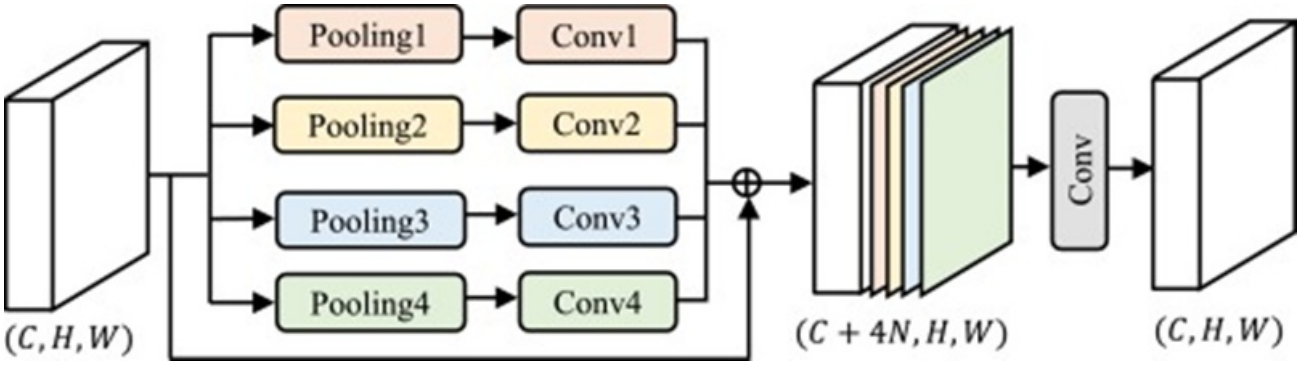

Figure 5. Depiction of the pyramid pooling module.

The final output of the pixel decoder delivers feature representations across three resolutions, unifying high-level, semantic-rich features and low-level spatially detailed features.

© 2022 The Author(s). | Published by Agro Intelligence Research Press

Transformer decoder

The Transformer decoder is a pivotal component of the model, taking the representations learned by the pixel decoder and converting them into final output predictions. As shown in **Figure 4b**, the decoder follows the canonical Transformer design, transforming N object embeddings into output embeddings. It is built as a succession of decoder layers, each housing a masked attention layer, a self-attention layer, and a feed-forward network (FFN). Every Transformer decoder layer issues mask and class predictions; however, only the prediction from the final layer is adopted as the terminal output, while earlier-layer predictions may optionally assist via auxiliary supervision. We fix the Transformer decoder layer count at 3 to secure a favorable balance between precision and model compactness, and the feature P3 from the pixel decoder serves as the pixel features.

Query Features Initialization. Query features play a decisive role in the Transformer architecture, directing the decoder to focus on the most salient segments of the input embeddings. Earlier investigations indicate that query features can either be initialized from a blank state (e.g., all zeros) [32] or progressively updated by localized features [46]. Although both strategies can successfully generate query features, they typically require a larger stack of decoder layers and longer training schedules to mature fully. Inspired by Deformable DETR, which harvests a set of query bounding boxes from pyramidal features to drive object detection, and SparseInst [50], which presents a lightweight convolutional module termed Fiam to accentuate informative zones for each foreground entity, we amalgamate the advantages of these two paradigms.

Accordingly, we incorporate a Fiam-like convolutional module to expeditiously initialize the query features in our model, directly extracting queries endowed with strong semantic content from the underlying multi-scale feature maps. This simplistic module comprises merely two convolutional layers. The first is an ordinary 3×3 convolution layer with matching input and output channels. The second is a 1×1 convolution layer that collapses the channel count to match the number of classes plus one, where the supplementary dimension stands for “no object ϕ”. Specifically, feature P2 from the pixel decoder is selected to yield N pixel embeddings with the highest foreground likelihoods, which are then designated as the query features.

Masked Attention. The cross-attention block found in the classic Transformer decoder is supplanted by masked attention. Conventional cross-attention is computed per **Eq. 4**. Here, l indexes the layer, and Xl denotes the query features of shape N×C at layer l. Ql = fq(Xl−1) is obtained by applying the linear mapping fq to the query features from the previous layer. Kl and Vl correspond to the pixel features delivered by the pixel decoder after undergoing linear transformations fk and fv.

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

Extending cross-attention, masked attention additionally incorporates an attention mask Ml−1, as prescribed in **Eq. 5**.

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

The attention mask Ml−1 evaluated at spatial coordinate (x,y) is formalized in **Eq. 6**, where ml−1(x,y) denotes the binarized mask prediction—resized accordingly—from the immediately preceding (l-1) decoder layer. m0 corresponds to the binary mask prediction derived from X0.

© 2022 The Author(s). | Published by Agro Intelligence Research Press

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

Loss function

Departing from anchor-dependent segmentation approaches that spawn an excessive volume of region proposals, our architecture harnesses the Transformer decoder to reframe fruit detection as a dictionary look-up executed in an end-to-end fashion. More precisely, the decoder progressively interprets the N trainable query embeddings layer after layer, ultimately producing a fixed-size set of N predictions. This design obviates reliance on heuristic post-processing routines such as non-maximum suppression (NMS). As a substitute, we turn to Hungarian matching—a bipartite pairing strategy—to establish the optimal correspondence between predicted outputs and ground-truth annotations for loss evaluation.

Figure 6 contrasts NMS with bipartite matching. NMS begins by generating a large number of candidate proposals and then discards redundant ones using a heuristic filter driven by overlap metrics, thereby introducing a non-differentiable step after the forward pass. Bipartite matching, on the other hand, operates on a fixed, predetermined set of proposals and relies on a cost matrix to assign each proposal to either a specific ground-truth entity or a background "no object" label. By weaving this assignment procedure directly into the training objective, our framework achieves a fully end-to-end differentiable scheme where predictions and their assignments are jointly learned.

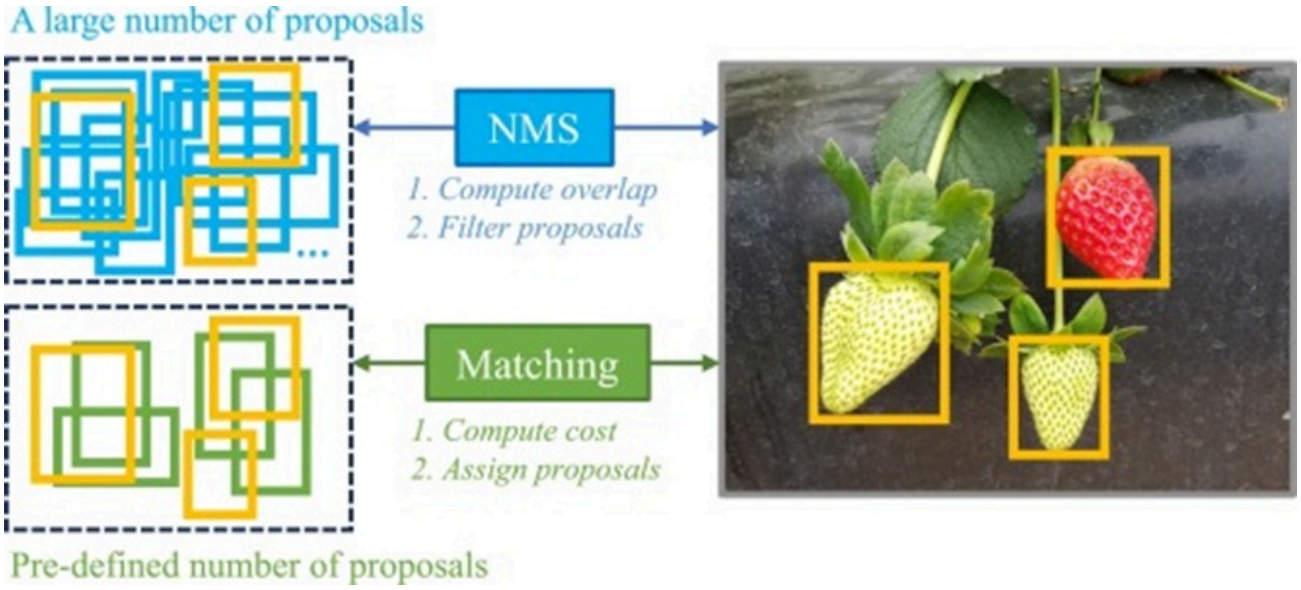

Figure 6. Comparison between non-maximum suppression and bipartite matching.

Initially, every available prediction—encompassing class and mask estimates and their corresponding class and mask targets—is used to build a cost matrix that guides prediction selection, where X denotes the total instance count in a given batch. The class-related cost and mask-related cost are quantified through cross-entropy loss and Dice loss, respectively, as captured in **Eqs. 7 and 8**:

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

© 2022 The Author(s). | Published by Agro Intelligence Research Press

Where yi represents the true class probability, and pi denotes the model's predicted probability.

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

Where mxy and txy refer to the pixel intensity at spatial location (x,y) within the predicted mask m and the ground-truth mask t, respectively.

Subsequently, the Hungarian algorithm is deployed to identify the optimal bipartite pairing by solving the constructed cost matrix, returning a matching score C(i,k) that pairs the i-th prediction with the k-th ground-truth instance. Through this process, the original N predictions are pruned down to exactly match the number of actual target objects.

The aggregate training objective for our model is expressed in **Eq. 9**:

$$\mathscr{L}_{total} = \lambda_{cl}\mathscr{L}_{class} + \lambda_{m}\mathscr{L}_{mask} + \lambda_{co}\mathscr{L}_{conv} + \lambda_{a}\mathscr{L}_{aux} \quad (9)$$

λ stands for the individual weighting factors assigned to different loss components.

Lclass denotes the cross-entropy loss evaluated between the matched class predictions and their corresponding class targets.

Lmask denotes the Dice loss evaluated between the matched mask predictions and their corresponding mask targets.

Lconv denotes the cross-entropy loss computed between the Fiam-like convolutional module's output and the ground truth.

Because each layer of the Transformer decoder emits its own class and mask estimates, we also draw upon earlier-layer predictions to formulate an auxiliary loss, Laux, specified in **Eq. 10**:

$$\mathscr{L}_{aux} = \sum_{i=0}^{D} \left(\mathscr{L}^{\mathrm{i}}{}_{class} + \mathscr{L}^{\mathrm{i}}{}_{mask} \right) \quad (10)$$

Where D indicates the total count of Transformer decoder layers. L′class and L′mask follow the identical loss definitions as Lclass and Lmask.

Taking inspiration from PointRend [51], which showed that segmentation models can be efficiently trained by restricting the computation of mask loss to a randomly sampled subset of K points rather than the entire mask, we adopt this sampling principle in our pipeline. Hence, mask loss is calculated from sampled points during both the bipartite matching phase and the final loss aggregation.

Evaluation metrics

© 2022 The Author(s). | Published by Agro Intelligence Research Press

Average Precision (AP). Precision is a conventional, widely accepted metric for assessing a network's ability to correctly pinpoint target objects, providing a summary of the network's overall capability. Precision is formally expressed by **Eq. 11**:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad (11)$$

In this expression, TP tallies the fruit targets that were correctly identified, whereas FP counts the erroneous detections where something other than a fruit is mistakenly labeled as one. Average precision is adopted as the principal criterion for benchmarking model performance. It is derived by averaging precision values across 10 Intersection over Union (IoU) cutoffs, uniformly spaced from 0.50 to 0.95, over all categories. A larger AP figure reflects more accurate model detections. For granularity, AP figures at IoU thresholds of 0.50 and 0.75 are disclosed, together with results broken down by individual category.

Learnable Parameters (Params). Learnable parameters encompass the weights and biases across a model's layers, which are adjusted during training to optimize predictive accuracy. The overall learnable parameter count is routinely interpreted as an indicator of a model's scale and architectural complexity.

Floating-point Operations (FLOPs). Floating-point operations serve as a gauge of the computational demands imposed by a deep learning model. It captures the arithmetic workload—specifically, multiplications and additions—performed during forward propagation as input data traverses the model layers to produce predictions. FLOPs are standardly reported as the combined sum of these multiplication and addition operations.

Frames per second (FPS). Frames per second (FPS) measures a deep learning model's inference throughput, indicating the number of input images the model can process within a one-second window. An elevated FPS value corresponds to faster model execution, a property of paramount importance for time-sensitive tasks such as autonomous driving or live video processing. FPS is shaped by elements such as model footprint, hardware capability, and applied optimization strategies.

Experiments and Results

Experiments

Configuration

The experimental work reported in this paper is built on the Detectron2 platform [52] and executed using Python 3.9.13 and PyTorch 1.13. The hardware environment consisted of an Intel Xeon Gold 6152 CPU clocked at 2.1 GHz, paired with 2 Nvidia Tesla V100 GPUs and 32.0 GB of system memory.

Training details

No externally pre-trained weights are used in this study; instead, the parameters of all convolutional layers are initialized from a normal distribution. The training protocol integrates multiple data augmentation techniques to fortify model robustness and generalizability. These techniques include random horizontal flipping and a multi-scale resizing scheme in which the shorter image dimension is randomly set to one of {416, 448, 480, 512, 544, 576, 608, 640} pixels, with the longer dimension strictly bounded at 768 pixels. Such scale jittering introduces considerable variation in input dimensions, compelling the model to become agnostic to object size.

The upper bound on predictions per image N is fixed at 100, predicated on the assumption that no single image will contain more fruit instances than this ceiling. The count of mask sampling points, K, is set to 12544, corresponding to a sampling grid with a resolution of 112 × 112. The weighting scheme for the loss terms is set to {λcl: 2.0, λm: 5.0, λco:

© 2022 The Author(s). | Published by Agro Intelligence Research Press

20.0, λa: 1.0}. The decoder depth D is fixed at 3, establishing the number of decoding layers engaged in prediction and achieving an equilibrium between computational load and expressive power.

Optimization is handled by the AdamW algorithm, with a step-decay schedule that commences with an initial learning rate of 0.0001 and a weight decay coefficient of 0.05. A learning rate multiplier of 0.1 is applied to the backbone subnetwork, and the learning rate is reduced by a factor of 10 at the 0.9 and 0.95 milestones of the total training iteration budget. The model undergoes 54k training iterations with a batch size of 8.

Inference details

At inference time, data augmentation is simplified exclusively to image resizing. Each incoming image is rescaled so that its shorter edge equals 640 pixels, while the longer edge is limited to 768 pixels, thereby maintaining the native aspect ratio. Auxiliary prediction branches—encompassing outputs from intermediate layers or auxiliary heads that are active during training—are deactivated during inference to streamline the forward pass and concentrate entirely on the model's final outputs. After the model produces its set of predictions, the top 100 candidates, ranked by confidence score, are retained as the final output set.

Results

We evaluated a broad suite of contemporary backbone architectures on the unified fruit corpus, adhering to the FruitQuery framework illustrated in **Figure 4b**. The complete benchmarking outcomes are consolidated in **Table 4**.

Table 4. Instance segmentation performance on the unified dataset across different backbones.

Backbone		FPS	FLOPs (G)	Params (M)	StrawDI_Db1				NinePeach	
					AP_{rs4}	AP_{rs3}	AP_{rs2}	AP_{rs1}	AP_{ripe}	$AP_{semiripe}$
CNN-based	ResNet [53]	19.50	29.56	17.54	73.39	75.80	73.64	39.52	65.72	53.50
		34.60	45.88	27.64	78.56	77.30	74.00	38.52	67.65	56.77
		31.54	50.94	30.52	77.17	75.36	75.33	41.61	**69.57**	54.78
	FasterNet [54]	17.90	91.85	35.82	77.12	78.29	74.04	44.76	63.21	50.78
		16.12	129.00	58.06	79.69	81.17	76.68	47.15	64.09	51.38
		15.60	189.00	97.70	77.86	77.72	**77.73**	**50.14**	66.47	53.61
	YOLOv8* [55]	**44.22**	46.12	11.79	75.40	65.70	61.44	32.64	68.02	49.66
		41.58	119.24	27.24	72.43	66.91	64.43	32.81	67.10	53.15
		34.36	238.48	45.94	73.78	71.46	64.14	34.11	66.79	52.63
	YOLOv9* [55]	13.4	82.26	8.64	73.54	69.95	65.99	33.85	65.10	54.33
		15.78	142.38	22.26	74.60	69.34	65.21	33.50	68.63	54.08
		15.58	171.82	27.84	75.68	69.13	66.53	33.89	68.13	55.13

© 2022 The Author(s). | Published by Agro Intelligence Research Press

	YOLOv10* [55]		44.10	9.20	74.22	66.18	68.75	33.34	63.50	50.34
		21.28	110.06	19.37	74.11	68.09	65.76	33.23	65.86	50.57
		18.94	180.68	25.52	74.92	69.35	63.70	32.71	65.39	51.82
Transformer-based	MobileViT [56]	17.94	**17.27**	**7.27**	69.19	63.06	54.66	20.12	50.81	38.27
		19.74	21.14	8.28	72.13	62.95	62.78	23.85	54.80	41.35
		19.56	26.99	11.36	70.58	70.42	62.97	27.65	56.41	44.19
		19.44								
	LightViT [57]		29.11	14.06	72.04	71.37	66.00	30.77	57.85	44.47
		14.36	28.03	23.74	74.14	70.89	68.74	31.96	59.60	46.42
		14.28	45.06	39.63	75.57	72.29	70.31	34.13	62.50	50.00
		13.49								
	NextViT [58]		105.00	37.96	76.07	77.61	73.57	48.11	62.38	49.99
		15.06	128.00	51.02	77.52	75.75	77.66	45.80	61.08	51.79
		13.10								
	GroupMixFormer [59]		83.72	17.62	79.67	78.39	74.79	42.81	63.07	52.55
		9.24	96.51	29.06	78.75	79.36	74.60	44.53	64.79	53.66
		9.02								
	MetaFormer [60]		66.92	18.40	74.69	75.95	71.14	40.86	62.98	48.77
	SegFormer [34]	18.94	57.06	10.19	77.28	75.39	67.83	40.09	58.71	47.76
		17.32								
	PoolFormer [60]		66.92	18.40	80.60	78.19	73.75	42.94	61.16	47.02
		13.50	80.94	27.88	80.24	79.89	73.76	44.69	64.78	49.63
		12.34								
	TransXNet [34]		70.65	19.33	77.44	73.96	71.59	39.81	63.03	46.70
		10.88	98.84	33.34	80.66	74.53	71.02	40.31	62.24	49.16
	CMT [61]	8.22	67.55	14.57	77.79	78.81	77.23	46.84	68.98	57.58
			78.27	20.21	80.61	78.37	76.35	47.72	68.96	57.45
	EMSA (Ours)	14.06	61.56	10.94	81.92	82.12	77.16	47.12	69.03	56.11
		13.08	69.33	14.08	**83.74**	**82.16**	76.47	47.91	68.55	**58.68**
		16.50								
		16.00								

© 2022 The Author(s). | Published by Agro Intelligence Research Press

∗ YOLO families are trained employing their segmentation head.

Overall Performance. The variant of our method equipped with EMSA-s (FruitQuery-s) claims the top positions across all principal metrics—67.02 AP, 79.17 AP50, and 70.83 AP75—placing it decisively ahead of 13 alternative models across 33 total configurations. The sibling variant with EMSA-xs (FruitQuery-xs) also posts a highly competitive AP of 66.46. These results affirm the pronounced edge our architecture has in fruit segmentation.

Turning to convolution-dominated models, the longstanding ResNet family delivers respectable results, with ResNet-50 achieving an AP of 63.92. The newer FasterNet-l also registers a commendable 65.25 AP. Examining the YOLO lineage, YOLOv9-c delivers the strongest performance among its peers at 60.41 AP, suggesting that YOLO-derived architectures face inherent limitations when applied to fruit segmentation. Every tested YOLO variant, without exception, sits below the performance envelope of our model.

On the attention-based side, the evaluated models exhibit greater diversity in their structural formulations and parameter allocations—configurations from NextViT, GroupMixFormer, and PoolFormer cluster within a narrow AP band of 62.37 to 63.50. Two members of the CMT family most closely approach our performance, registering APs of 66.00 and 66.46. These Transformer-oriented models mirror the broader field's shift toward attention-centric backbones and achieve notable improvements over many convolution-based alternatives.

Yet none surpass our FruitQuery on AP, AP50, or AP75, which suggests that the query-conditioned paradigm at the heart of our design exploits feature representations more effectively for detailed fruit instance segmentation.

Individual Performance. Within the NinePeach subset, YOLOv9-s claims the best APunripe of 56.60, while ResNet-50 attains the top APripe of 69.57. Our model, however, excels in the intermediate ripening phase, delivering the highest APsemiripe of 58.68. This result speaks to its ability to discern the subtler visual characteristics that distinguish semiripe peaches.

For the StrawDI_Db1 subset, the convolution-based FasterNet secures the leading APrs1 of 50.14 and APrs2 of 77.73. Our FruitQuery-s, in turn, outperforms all competing methods on two of the four strawberry maturity categories, posting the best APrs3 of 82.16 and APrs4 of 83.74. These gains signal that our approach handles the dramatic appearance transformations during late-stage strawberry maturation—where hue, surface texture, and overall form diverge markedly from early-stage fruit—with considerable effectiveness.

Aggregated across all seven ripeness categories present in the combined dataset, our model achieves the top AP on three of them, corroborating that the query-based formulation successfully isolates fine-grained features characteristic of different maturity levels while maintaining broadly competitive results elsewhere.

Model Complexity. A clear relationship emerges between model scale and accuracy: larger architectures typically harbor more parameters, granting them greater capacity to encode intricate patterns and dependencies.

Among convolution-centric designs, FasterNet-l is the most parameter-intensive, with 97.70M parameters and 189G FLOPs, achieving a solid AP of 65.25. The YOLO family, widely recognized for prioritizing compactness, exemplifies the opposite extreme: YOLOv9-s operates with just 8.64M parameters and 82.26G FLOPs, and YOLOv10-s trims further to 7.27M parameters and 44.10G FLOPs. Their APs, however, stand at modest 59.91 and 58.17, trailing many alternatives.

Within Transformer-oriented designs, MobileViT-xxs is the most frugal, requiring only 7.27M parameters and 17.27G FLOPs, though this minimalism comes at the cost of the lowest observed AP of 46.34. At the opposite pole, NextViT-b represents the heaviest Transformer candidate, with 51.02M parameters and 128G FLOPs, yielding an AP of 62.47.

Our model embodies a notably economical design philosophy. FruitQuery-xs expends a mere 10.94M parameters and 61.56G FLOPs to attain an AP of 66.46, while FruitQuery-s pushes the ceiling to 67.02 AP with only 14.08M parameters

© 2022 The Author(s). | Published by Agro Intelligence Research Press

and 69.33G FLOPs.

To place these figures in context, models achieving APs comparable to our FruitQuery-xs (66.46)—namely CMT-ti (66.00), CMT-xs (66.46), and FasterNet-l (65.25)—incur substantially steeper parameter and FLOP costs (14.57M/67.55G, 20.21M/78.27G, and 97.70M/189.00G, respectively) than our FruitQuery-xs (10.94M/61.56G). Viewed from the other direction, architectures that align with FruitQuery-xs in terms of parameter count 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 markedly inferior APs (57.33, 52.90, and 58.48, respectively).

Inference Speed. Convolution-based models achieve faster inference throughput than Transformer-based counterparts, consistent with the inherent efficiency advantages long attributed to convolutional computation. Among all evaluated approaches, YOLOv8-t achieves the swiftest execution at 44.22 FPS, followed closely by YOLOv8-m (41.58) and YOLOv8-l (34.36), underscoring their suitability for latency-sensitive deployments.

Our FruitQuery delivers respectable inference speeds of 16.5 and 16 FPS, demonstrating competitive real-time characteristics. Both variants outpace the entire YOLOv9 lineup, suggesting improved throughput efficiency relative to this more recent, Transformer-influenced series. Moreover, our models run faster than several efficiency-oriented Transformer backbones, including LightViT-t (14.36), CMT-ti (14.06), and NextViT-s (15.06).

While falling slightly behind MobileViT-xxs (19.74) and MobileViT-xs (19.56) in raw speed, our models substantially outrun recently proposed designs like TransXNet-s (8.22) and GroupMixFormer-s (9.02), positioning them favourably within the landscape of efficient Transformer-based architectures. Taken together, these timing results confirm that our models negotiate a propitious balance between inference latency and representational capacity.

In summary, the empirical evidence collected demonstrates that our proposed approach not only matches or exceeds alternative segmentation frameworks in predictive accuracy but also maintains a lightweight profile and high operational efficiency.

Ablation experiments

Type of pixel decoder

Table 5 contrasts two alternative pixel decoder designs—a plain FPN and an FPN augmented with PPM—in terms of resulting model accuracy. The vanilla FPN baseline delivers an AP of 64.97, AP50 of 78.41, and AP75 of 68.88. Augmenting it with PPM yields uniform improvements across the board, boosting 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 consistent gains validate that integrating PPM into the FPN structure meaningfully elevates segmentation fidelity.

Table 5. Ablation analysis on the pixel decoder.

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

Number of decoder attention heads

Table 6 explores the sensitivity of model performance to the number of attention heads employed. Restricting the design to only 2 heads yields APs of 62.36, 75.30, and 65.80, reflecting insufficient representational breadth. Expanding to 4 heads yields the optimum, with an AP of 64.46, an AP50 of 77.09, and an AP75 of 68.61. Pushing further to 8 heads achieves only a marginal gain over the 2-head baseline (AP rising to 64.46 and AP75 to 67.88) while remaining below the

© 2022 The Author(s). | Published by Agro Intelligence Research Press

4-head configuration. This pattern indicates that 4 attention heads realize an ideal compromise, providing enriched feature encoding without encountering diminishing returns.

Table 6. Ablation experiment findings based on FruitQuery-xs.

Ablation Study	Parameter	AP75	AP50	AP
Number of attention heads	2	65.80	75.30	62.30
	4	68.61	77.09	64.44
	8	67.88	76.98	64.36
Number of queries	80	68.39	77.74	64.54
	90	68.15	77.34	64.45
	100	70.15	78.88	66.55
	110	69.94	78.56	65.21
	120	69.18	78.34	65.18
Number of decoder layers	1	66.59	75.78	62.85
	2	67.72	76.93	63.98
	3	67.86	76.96	63.91
	4	70.22	78.98	66.41
	5	70.44	79.31	66.68
	6	70.85	78.95	66.80
	7	69.36	78.49	65.94
	8	69.26	78.07	65.78

Number of queries

Table 6b summarizes how varying the number of queries affects model outcomes. A setting of 100 queries enables the strongest overall metrics: AP of 66.52, AP50 of 78.84, and AP75 of 70.19. Reducing the query budget to 80 or 90 causes a uniform erosion across all evaluation criteria. Conversely, inflating the query pool to 110 or 120 offers no additional benefit. These results imply that 100 queries strike a productive balance—sufficient to capture the requisite object-level detail without incurring superfluous computation.

Number of decoder layers

Table 6c documents how decoder depth influences model behaviour. With a shallow stack of 1 to 3 layers, AP remains constrained to 62.85-63.98, suggesting inadequate representational depth. Performance improves monotonically as additional layers are introduced, peaking at 6 layers with an AP of 66.80, AP50 of 78.95, and AP75 of 70.85. Pushing beyond this depth degrades accuracy, suggesting that overly deep decoder stacks may introduce redundancy or hinder convergence. These observations identify a clear optimal configuration at 6 decoder layers.

Combined versus separated training

We examine how FruitQuery-xs performs when trained on the unified dataset versus when trained independently on each constituent dataset, and present the comparative results in **Table 7**. For the NinePeach subset, the unified training

© 2022 The Author(s). | Published by Agro Intelligence Research Press

strategy yields substantial accuracy gains across all ripeness levels: APunripe climbs by 7.55 points (from 43.72 to 51.27), APsemiripe by 4.86 points (from 49.07 to 53.93), and APripe by 4.89 points (from 62.05 to 66.94).

Table 7. Training on separate datasets versus the combined dataset.

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

On the StrawDI_Db1 side, outcomes are more uneven. APrs2 sees a pronounced 4.18-point improvement (from 72.55 to 76.73), and APrs4 similarly gains 1.65 points (from 77.99 to 79.64). The remaining two categories, however, experience modest regression: APrs1 drops from 45.57 to 42.29, and APrs3 retreats from 79.87 to 78.40.

Stepping back to assess the overall picture, joint training on the merged corpus lifts the model's global AP from 61.08 to 64.63. This result suggests that exposure to a richer, more heterogeneous fruit distribution generally strengthens generalization across the majority of ripeness stages, even if isolated category-specific compromises occur.

Model parameter distribution

We detail the parameter breakdown for YOLO and our FruitQuery in **Table 8**. Drawing on the earlier evidence in **Table 4**, YOLOv9 is the top-performing iteration among the three YOLO families. It is therefore selected for direct head-to-head comparison with our approach, as well as for subsequent analytical discussions.

Table 8. Parameter breakdown for YOLO versus our FruitQuery.

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

YOLOv9-s maintains the leanest overall footprint at 8.64M parameters, with the head component consuming 2.92M of that budget, yet it musters only 59.91 AP. Progressing from YOLOv9-s to YOLOv9-m swells the total parameter count to 22.26M through expansions in both backbone and head, but the return is a negligible AP increment from 59.91 to 60.04.

© 2022 The Author(s). | Published by Agro Intelligence Research Press

YOLOv9-c improves upon YOLOv9-m slightly, achieving 60.41 AP, though this requires a backbone of 19.95M and a head of 7.89M.

Our model, in stark contrast, outperforms YOLOv9 while using fewer parameters. Both FruitQuery-xs and FruitQuery-s share a compact 4.18M head, which is smaller than the heads of either YOLOv9-m or YOLOv9-c. The key differentiator between the two FruitQuery variants lies in the backbone: FruitQuery-s deploys a more expressive backbone and, as a result, attains the markedly superior AP of 67.02.

These comparisons not only confirm that FruitQuery achieves a substantially more favourable balance between segmentation quality and architectural footprint than the YOLO family, but also underscore its lightweight design, which further strengthens the case for practical deployment in real-world agricultural settings.

Visualization

Figure 7 presents illustrative segmentation outputs generated by our FruitQuery model. As an initial observation, FruitQuery supports concurrent delineation of peaches and strawberries within a single framework, eliminating the need to train separate models for each fruit category. A second noteworthy trait is the model's resilience to scale differences between fruit types, a property rooted in its adept multi-scale feature blending. The typical peach occupies a considerably larger image footprint than the average strawberry, yet FruitQuery renders accurate boundaries for both large and small specimens alike. A third strength lies in the model's dependability amidst difficult orchard conditions: scenarios involving trunk or leaf occlusions, as well as variable lighting, are navigated without sacrificing boundary fidelity. Collectively, these properties demonstrate that FruitQuery provides reliable fruit localization for subsequent processing pipelines.



Figure 7. Segmentation outputs of our FruitQuery for NinePeach (left) and StrawDI_Db1 (right).

We further offer a side-by-side comparison of FruitQuery against YOLO in **Figure 8a**. Looking at example (1), even though YOLOv9-c is not fundamentally an anchor-based detector, its output on the strawberry resembles a coarse, anchor-style approximation. In contrast, our method recovers the strawberry's actual outline far more faithfully. In example (2), YOLOv9-c's segmentation contour largely mirrors the rectangular confines of its bounding proposal, while our model adheres tightly to the peach's genuine perimeter. Beyond this, YOLOv9-c completely misses a smaller peach positioned in the background, whereas ours correctly captures it. Moving to example (3), YOLOv9-c fails to detect and segment a conspicuously visible strawberry, whereas our approach does so without issue. In example (4), a peach partly obscured by background elements goes unrecognized by YOLOv9-c, but our model successfully discriminates it despite the minimal exposed surface.

© 2022 The Author(s). | Published by Agro Intelligence Research Press

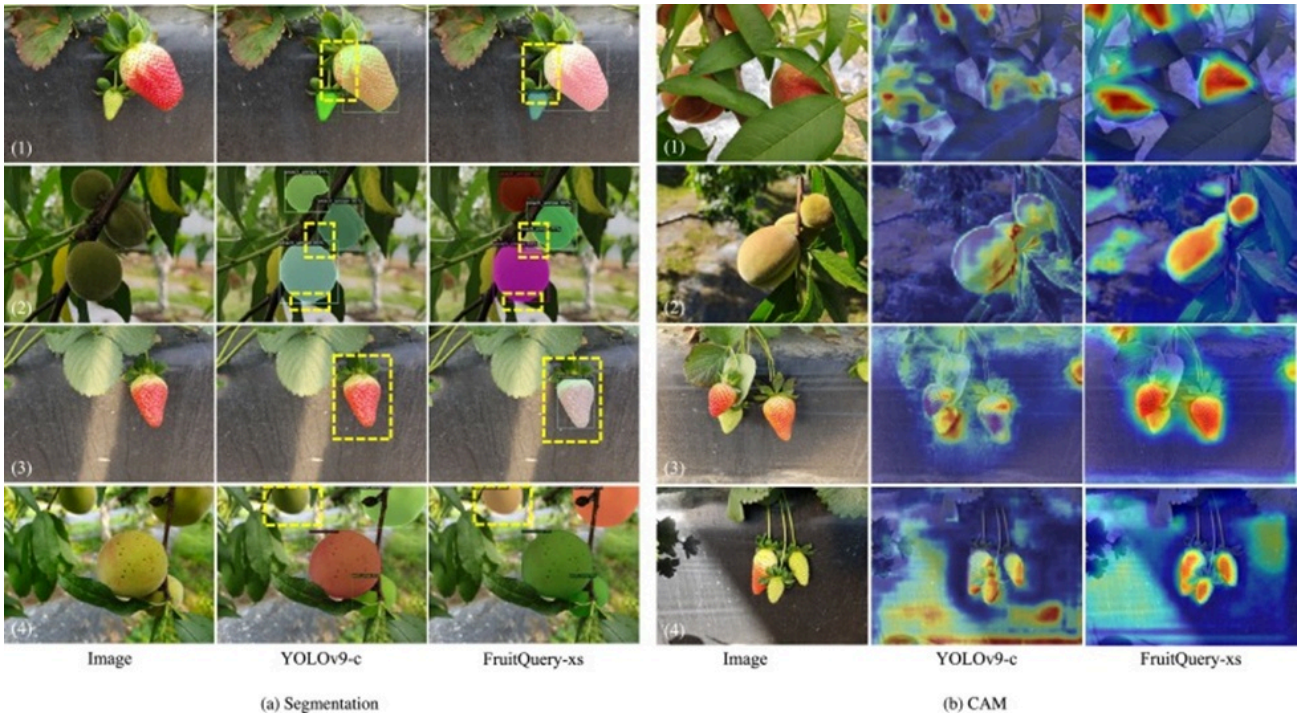

Figure 8. Visual comparison between YOLO and our FruitQuery.

Class activation map analysis

Class Activation Maps (CAM, [62]) serve as a well-established explanatory tool that illuminates which spatial regions of an input image most heavily shape a network's predictions. By mapping learned feature importance back onto the original image plane, CAM reveals where the model focuses, thereby opening a window into its internal reasoning. We present a CAM-based comparison of YOLO and our FruitQuery in **Figure 8b**.

Inspecting the CAM visualizations, YOLOv9-c tends to produce attention distributions that are comparatively dispersed and, at times, mislocalized, focusing on expansive or less diagnostic portions of the scene. For example, in scenarios (1) and (2), YOLOv9-c's activation pattern flickers uncertainly around the fruit and gets drawn toward the surrounding foliage. FruitQuery, on the other hand, maintains activation that is markedly more compact and precisely centered on the fruit targets. The contrast becomes most apparent in scenarios (3) and (4): YOLOv9-c spreads its attention across a broad, blurred halo encircling the fruit and inadvertently assigns weight to inconsequential background clutter, whereas FruitQuery sharply demarcates fruit from non-fruit context, picking up on subtle textural details specific to peaches and strawberries and confining high-activation zones tightly to the fruit bodies.

Thus, the activation map evidence underscores FruitQuery's strengthened capacity to learn distinguishing fruit attributes —contour shape, colour transitions, and margin boundaries—which, in turn, translate into more interpretable behaviour and superior segmentation outcomes.

Deployment

We benchmarked the inference characteristics of the proposed FruitQuery across a spectrum of hardware configurations: a datacenter-grade GPU (NVIDIA Tesla V100), a low-power edge device (NVIDIA Jetson Orin Nano), and a consumer-grade CPU (Apple M1).

Table 9 reports per-image inference latency for four distinct model serialization formats: native PyTorch checkpoint (.pth), Open Neural Network Exchange (.onnx), TensorRT engine (.trt), and Core ML (.mlmodel). Two levels of arithmetic

© 2022 The Author(s). | Published by Agro Intelligence Research Press

precision are examined: FP32 (full single-precision) and FP16 (half-precision).

Table 9. Single-image inference latency of FruitQuery across hardware platforms.

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

The recorded latencies show that TensorRT consistently yields the lowest runtime across all hardware targets under both precision settings. When executed on the Tesla V100, FruitQuery-xs completes inference in a mere 0.0196 seconds per image with FP16 and TensorRT—a figure that betters the FP32 PyTorch baseline by a factor exceeding 2.5. The Jetson Orin Nano likewise derives substantial benefit from TensorRT compilation, dipping beneath the 0.1-second threshold for FruitQuery-xs in FP16 mode. These observations highlight TensorRT's value for latency-sensitive edge scenarios.

Compared with the V100 and Jetson ecosystems, the Apple M1 exhibits longer inference times. Deployment through ONNX yields per-image runtimes above 1 second for both model scales. What is more, Core ML execution in the .mlmodel format incurs additional slowdown, with FruitQuery taking around 2.5 seconds per frame under FP16. That said, Core ML compatibility opens the door to integration within Apple-native application stacks, and future hardware revisions may narrow this performance gap.

In aggregate, the proposed FruitQuery variants demonstrate capable inference performance spanning heterogeneous deployment settings. The measurements attest to the models' adaptability and their portability across multiple runtime backends and precision configurations. The pairing of lightweight architectural design with TensorRT optimization and half-precision arithmetic enables responsive inference on both cloud-class GPUs and resource-constrained edge nodes.

Discussion

Limitations

One constraint is that FruitQuery depends on a fairly extensive, meticulously annotated training corpus, rendering out-of-the-box transfer to unfamiliar fruit species or previously unseen orchard environments difficult without additional labeling effort.

A second limitation concerns computational load: while FruitQuery is comparatively parsimonious, the present architecture still imposes moderate resource requirements that could impede real-time performance on heavily constrained embedded hardware. The inference rate of our model also presents opportunities for further acceleration. These acknowledged drawbacks delineate concrete pathways for future improvement, including the exploration of more aggressive model compression approaches.

© 2022 The Author(s). | Published by Agro Intelligence Research Press

Future work

Building on the findings reported here, we aim to compress and deploy our FruitQuery on embedded computing modules for practical in-field fruit segmentation. By shrinking the model footprint and harnessing computational optimization methods such as quantization, we hope to enhance orchard efficiency through near-instantaneous ripeness feedback, thereby enabling autonomous harvesting platforms to target exclusively ripe produce selectively.

In parallel, we envision augmenting the unified dataset with a broader range of fruit cultivars, thereby broadening its relevance across diverse horticultural contexts. The assembly of a large-scale, multi-fruit instance segmentation resource, complete with ripeness stage labels, would simultaneously reduce duplicated annotation effort and accelerate innovation in autonomous fruit-picking technology.

Conclusion

In the present work, we merged two field-collected fruit datasets—covering peach and strawberry—each encompassing 3 ripeness grades for peach and 4 for strawberry. Building upon this merged resource, we introduced FruitQuery, a lightweight query-conditioned instance segmentation framework purpose-built for multi-stage ripeness identification.

The consolidated dataset enables a single model to simultaneously perform ripeness discrimination across both fruit types, obviating the need to replicate the training workflow for each crop. FruitQuery is structured around three principal modules: a feature-extraction backbone, a pixel decoder, and a stack of Transformer decoders. Efficient multi-head self-attention (EMSA) blocks are woven into the backbone to reduce computational overhead. At the same time, a pyramid pooling module (PPM) is incorporated into the pixel decoder to enhance multi-scale feature integration. Transformer decoders are subsequently leveraged to derive a fixed set of queries that directly yield instance masks, thereby sidestepping the need for post-processing steps such as non-maximum suppression.

Through the synergistic integration of convolutional and Transformer-based processing, FruitQuery executes in a fully end-to-end fashion and allocates attention precisely to fruit regions, resolving nuanced differences in morphology and maturity. The architectural choices behind FruitQuery culminate in state-of-the-art performance: a leading AP of 67.02, achieved with merely 14.08M parameters, surpassing 13 competing models from both CNN-centric and Transformer-centric paradigms. In particular, it outperforms three successive YOLO generations under taxing conditions, including partial occlusion and lighting variation. That said, FruitQuery's reliance on annotated supervision poses obstacles for immediate adaptation to novel fruit types, and latency considerations may present challenges when porting the model to embedded processors.

Moving forward, we intend to further tailor FruitQuery for operational in-field deployment, investigating techniques such as quantization to better suit edge computing environments. Plans are also in place to broaden the unified dataset with additional fruit varieties, progressing toward a comprehensive, large-scale instance segmentation benchmark annotated with ripeness stages. Through these ongoing refinements, we aspire to deepen FruitQuery's impact in orchard automation, enabling more accurate and efficient fruit maturity assessment and advancing the broader advancement of precision agriculture.

Acknowledgements

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

Conflict of interest

None

© 2022 The Author(s). | Published by Agro Intelligence Research Press

Financial support

None

Ethics statement

None

Received: 27 Jul 2021 Revised: 30 Sep 2021 Accepted: 15 Oct 2021
Published online: 25 May 2022

Rights and permissions

Open Access The author(s) retain copyright. This article is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. It may be shared and adapted for non-commercial purposes with appropriate attribution, an indication of changes, and distribution of adaptations under the same license. Third-party material may be subject to separate terms identified in its credit line. View the license at https://creativecommons.org/licenses/by-nc-sa/4.0/.

References

1. Mu L, Cui G, Liu Y, Cui Y, Fu L, Gejima Y. Design and simulation of an integrated end-effector for picking kiwifruit by robot. Inf Process Agric. 2020;7:58-71. https://doi.org/10.1016/j.inpa.2019.05.004..

2. Kang H, Chen C. Fruit detection, segmentation and 3D visualisation of environments in apple orchards. Comput Electron Agric. 2020;171:105302.

3. Liang C, Xiong J, Zheng Z, Zhong Z, Li Z, Chen S, et al. A visual detection method for nighttime litchi fruits and fruiting stems. Comput Electron Agric. 2020;169:105192. https://doi.org/10.1016/j.compag.2019.105192..

4. Jiao Z, Huang K, Jia G, Lei H, Cai Y, Zhong Z. An effective litchi detection method based on edge devices in a complex scene. Biosyst Eng. 2022;222:15-28. https://doi.org/10.1016/j.biosystemseng.2022.07.009..

5. He Z, Khanal SR, Zhang X, Karkee M, Zhang Q. Real-time Strawberry Detection Based on Improved YOLOv5s Architecture for Robotic Harvesting in Open-field Environment. arXiv. 2023;arXiv:2308.03998. https://doi.org/10.48550/arXiv.2308.03998..

6. Lu J, Chen P, Yu C, Lan Y, Yu L, Yang R, et al. Lightweight green citrus fruit detection method for practical environmental applications. Comput Electron Agric. 2023;215:108205. https://doi.org/10.1016/j.compag.2023.108205..

7. McGuinness BJ, Duke MD, Au KC, Lim HS. Field factory for the automated harvesting of forestry tree stock. Biosyst Eng. 2023;227:52-67. https://doi.org/10.1016/j.biosystemseng.2023.01.003..

8. Ning Z, Luo L, Ding X, Dong Z, Yang B, Cai J, et al. Recognition of sweet peppers and planning the robotic picking sequence in high-density orchards. Comput Electron Agric. 2022;196:106878.

https://doi.org/10.1016/j.compag.2022.106878..

9. Peebles M, Lim SH, Duke M, McGuinness B. Investigation of optimal network architecture for asparagus spear detection in robotic harvesting. IFAC-PapersOnLine. 2019;52:283-7. https://doi.org/10.1016/j.ifacol.2019.12.535..

10. Hafiz AM, Bhat GM. A survey on instance segmentation: state of the art. Int J Multimed Inf Retr. 2020;9:171-89. https://doi.org/10.1007/s13735-020-00195-x..

11. Kestur R, Meduri A, Narasipura O. MangoNet: A deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard. Eng Appl Artif Intell. 2019;77:59-69. https://doi.org/10.1016/j.engappai.2018.09.011..

12. Gonçalves JP, Pinto FAC, Queiroz DM, Villar FMM, Barbedo JGA, Del Ponte EM. Deep learning architectures for semantic segmentation and automatic estimation of severity of foliar symptoms caused by diseases or pests. Biosyst Eng. 2021;210:129-42. https://doi.org/10.1016/j.biosystemseng.2021.08.011..

13. Trivedi M, Gupta A. Automatic monitoring of the growth of plants using deep learning-based leaf segmentation. Int J Appl Sci Eng. 2021;18:1-9.

14. Li Q, Jia W, Sun M, Hou S, Zheng Y. A novel green apple segmentation algorithm based on ensemble U-Net under complex orchard environment. Comput Electron Agric. 2021;180:105900.

15. He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN, 2017 IEEE International Conference on Computer Vision (ICCV). 2017:2980-8. https://doi.org/10.1109/ICCV.2017.322..

16. Santos TT, de Souza LL, dos Santos AA, Avila S. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. Comput Electron Agric. 2020;170:105247. https://doi.org/10.1016/j.compag.2020.105247..

17. Ni X, Li C, Jiang H, Takeda F. Deep learning image segmentation and extraction of blueberry fruit traits associated with harvestability and yield. Hortic Res. 2020;7:110.

18. Yu Y, Zhang K, Yang L, Zhang D. Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. Comput Electron Agric. 2019;163:104846. https://doi.org/10.1016/j.compag.2019.06.001..

19. Pérez-Borrero I, Marín-Santos D, Gegúndez-Arias ME, Cortés-Ancos E. A fast and accurate deep learning method for strawberry instance segmentation. Comput Electron Agric. 2020;178:105736. https://doi.org/10.1016/j.compag.2020.105736..

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

21. Jia W, Zhang Z, Shao W, Hou S, Ji Z, Liu G, et al. FoveaMask: a fast and accurate deep learning model for green fruit instance segmentation. Comput Electron Agric. 2021;191:106488. https://doi.org/10.1016/j.compag.2021.106488..

22. Sheng X, Kang C, Zheng J, Lyu C. An edge-guided method to fruit segmentation in complex environments. Comput Electron Agric. 2023;208:107788. https://doi.org/10.1016/j.compag.2023.107788..

© 2022 The Author(s). | Published by Agro Intelligence Research Press

23. Yang S, Wang W, Gao S, Deng Z. Strawberry ripeness detection based on YOLOv8 algorithm fused with LW-Swin Transformer. Comput Electron Agric. 2023;215:108360.

24. Xiao B, Nguyen M, Yan WQ. Fruit ripeness identification using YOLOv8 model. Multimed Tools Appl. 2023. https://doi.org/10.1007/s11042-023-16570-9..

25. Yu J, Bai Y, Yang S, Ning J. Stolon-YOLO: a detecting method for stolon of strawberry seedling in glass greenhouse. Comput Electron Agric. 2023;215:108447. https://doi.org/10.1016/j.compag.2023.108447..

26. Chen S, Zou X, Zhou X, Xiang Y, Wu M. Study on fusion clustering and improved YOLOv5 algorithm based on multiple occlusion of Camellia oleifera fruit. Comput Electron Agric. 2023;206:107706. https://doi.org/10.1016/j.compag.2023.107706..

27. Zhu X, Chen F, Zheng Y, Chen C, Peng X. Detection of Camellia oleifera fruit maturity in orchards based on modified lightweight YOLO. Comput Electron Agric. 2024;226:109471. https://doi.org/10.1016/j.compag.2024.109471..

28. Ren R, Zhang S, Sun H, Wang N, Yang S, Zhao H, et al. YOLO-RCS: a method for detecting phenological period of 'Yuluxiang' pear in unstructured environment. Comput Electron Agric. 2025;229:109819. https://doi.org/10.1016/j.compag.2024.109819..

29. Shen Q, Zhang X, Shen M, Xu D. Multi-scale adaptive YOLO for instance segmentation of grape pedicels. Comput Electron Agric. 2025;229:109712. https://doi.org/10.1016/j.compag.2024.109712..

30. Ma Z, Dong N, Gu J, Cheng H, Meng Z, Du X. STRAW-YOLO: a detection method for strawberry fruits targets and key points. Comput Electron Agric. 2025;230:109853.

31. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, et al. An image is worth 16x16 words: transformers for image recognition at scale. International Conference on Learning Representations (ICLR). 2020. https://openreview.net/forum?id=YicbFdNTTy.

32. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S. End-to-end object detection with transformers. Computer Vision – ECCV 2020: 16th European Conference, Proceedings, Part I, Glasgow, UK; 2020, Springer-Verlag, Berlin, Heidelberg. 2020:213-29. https://doi.org/10.1007/978-3-030-58452-8_13..

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

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

35. Niu B, Feng Q, Chen B, Ou C, Liu Y, Yang J. HSI-TransUNet: A transformer based semantic segmentation model for crop mapping from UAV hyperspectral imagery. Comput Electron Agric. 2022;201:107297. https://doi.org/10.1016/j.compag.2022.107297..

36. Wang J, Zhang Z, Luo L, Wei H, Wang W, Chen M, et al. DualSeg: fusing transformer and CNN structure for image segmentation in complex vineyard environment. Comput Electron Agric. 2023;206:107682. https://doi.org/10.1016/j.compag.2023.107682..

37. Guo Y, Lan Y, Chen X. CST: Convolutional Swin Transformer for detecting the degree and types of plant diseases. Comput Electron Agric. 2022;202:107407.

© 2022 The Author(s). | Published by Agro Intelligence Research Press

https://doi.org/10.1016/j.compag.2022.107407..

38. Xia X, Chai X, Li Z, Zhang N, Sun T. MTYOLOX: multi-transformers-enabled YOLO for tree-level apple inflorescences detection and density mapping. Comput Electron Agric. 2023;209:107803. https://doi.org/10.1016/j.compag.2023.107803..

39. Sun M, Zhao R, Yin X, Xu L, Ruan C, Jia W. FBoT-Net: focal bottle-neck transformer network for small green apple detection. Comput Electron Agric. 2023;205:107609. https://doi.org/10.1016/j.compag.2022.107609..

40. Thai HT, Le KH, Nguyen NLT. FormerLeaf: an efficient vision transformer for Cassava Leaf Disease detection. Comput Electron Agric. 2023;204:107518. https://doi.org/10.1016/j.compag.2022.107518..

41. Li G, Jiao L, Chen P, Liu K, Wang R, Dong S, et al. Spatial convolutional self-attention-based transformer module for strawberry disease identification under complex background. Comput Electron Agric. 2023;212:108121. https://doi.org/10.1016/j.compag.2023.108121.

42. He W, Mi Y, Ding X, Liu G, Li T. Two-stream cross-attention vision transformer based on RGB-D images for pig weight estimation. Comput Electron Agric. 2023;212:107986. https://doi.org/10.1016/j.compag.2023.107986.

43. Zhao Z, Hicks Y, Sun X, Luo C. Peach ripeness classification based on a new one-stage instance segmentation model. Comput Electron Agric. 2023;214:108369. https://doi.org/10.1016/j.compag.2023.108369.

44. Basson CE, Groenewald JH, Kossmann J, Cronjé C, Bauer R. Sugar and acid-related quality attributes and enzyme activities in strawberry fruits: invertase is the main sucrose hydrolysing enzyme. Food Chem. 2010;121:1156-62. https://doi.org/10.1016/j.foodchem.2010.01.064..

45. Sánchez-Sevilla JF, Vallarino JG, Osorio S, Bombarely A, Posé D, Merchante C, et al. Gene expression atlas of fruit ripening and transcriptome assembly from RNA-seq data in octoploid strawberry (Fragaria × ananassa). Sci Rep. 2017;7:13737. https://doi.org/10.1038/s41598-017-14239-6.

46. Cheng B, Misra I, Schwing AG, Kirillov A, Girdhar R. Masked-attention mask transformer for universal image segmentation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022:1290-9. https://openaccess.thecvf.com/content/CVPR2022/html/Cheng_Masked-Attention_Mask_Transformer_for_Universal_Image_Segmentation_CVPR_2022_paper.html

47. Wang W, Xie E, Li X, Fan DP, Song K, Liang D, et al. Pyramid vision transformer: a versatile backbone for dense prediction without convolutions. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, Montreal, QC, Canada. 2021:548-58. https://doi.org/10.1109/ICCV48922.2021.00061.

48. Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S. Feature pyramid networks for object detection. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017:936-44. https://doi.org/10.1109/CVPR.2017.106.

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

50. Cheng T, Wang X, Chen S, Zhang W, Zhang Q, Huang C, et al. Sparse instance activation for real-time instance segmentation. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, New Orleans, LA, USA. 2022:4423-32. https://doi.org/10.1109/CVPR52688.2022.00439.

© 2022 The Author(s). | Published by Agro Intelligence Research Press

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

52. Wu Y, Kirillov A, Massa F, Lo WY, Girshick R. Detectron2. https://github.com/facebookresearch/detectron2

53. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016:770-8.

54. Chen J, Kao SH, He H, Zhuo W, Wen S, Lee CH, et al. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. IEEE Computer Society. 2023:12021-31. https://doi.org/10.1109/CVPR52729.2023.01157.

55. Jocher G, Qiu J, Chaurasia A. Ultralytics YOLO. https://github.com/ultralytics/ultralytics

56. Mehta S, Rastegari M. MobileViT: light-weight, general-purpose, and mobile-friendly vision transformer. http://arxiv.org/abs/2110.02178

57. Huang T, Huang L, You S, Wang F, Qian C, Xu C. LightViT: towards light-weight convolution-free vision transformers. http://arxiv.org/abs/2207.05557.

58. Li J, Xia X, Li W, Li H, Wang X, Xiao X, et al. Next-ViT: next generation vision transformer for efficient deployment in realistic industrial scenarios. arXiv:2207.05501 [cs]. 2022. https://doi.org/10.48550/arXiv.2207.05501..

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

60. Yu W, Luo M, Zhou P, Si C, Zhou Y, Wang X, et al. Meta-Former is actually what you need for vision. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2022:10809-19. https://doi.org/10.1109/CVPR52688.2022.01055.

61. Guo J, Han K, Wu H, Tang Y, Chen X, Wang Y, et al. CMT: convolutional neural networks meet vision transformers. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, New Orleans, LA, USA. 2022:12165-75. https://doi.org/10.1109/CVPR52688.2022.01186.

62. Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A. Learning deep features for discriminative localization. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA. 2016:2921-9. https://doi.org/10.1109/CVPR.2016.319..

© 2022 The Author(s). | Published by Agro Intelligence Research Press