agriculture

MDPI

Article

Detection of Red, Yellow, and Purple Raspberry Fruits Using YOLO Models

Kamil Buczyński *, Magdalena Kapłan and Zbigniew Jarosz

Institute of Horticulture Production, University of Life Science, 28 Głęboka Street, 20-612 Lublin, Poland; magdalena.kaplan@up.edu.pl (M.K.)
* Correspondence: kamil.buczynski@up.edu.pl

Abstract

This study presents a comprehensive evaluation of recent YOLO architectures, YOLOv8s, YOLOv9s, YOLOv10s, YOLO11s, and YOLO12s, for the detection of red, yellow, and purple raspberry fruits under field conditions. Images were collected using an smartphone camera under varying illumination, weather, and occlusion conditions. Each model was trained and evaluated using standard object detection metrics (*Precision*, *Recall*, mAP_{50}, $mAP_{50:95}$, F_1-*score*), while inference performance was benchmarked on both high-performance (NVIDIA RTX 5080) and embedded (NVIDIA Jetson Orin NX) platforms. All models achieved high and consistent detection accuracy across fruits of different colors, confirming the robustness of the YOLO algorithm design. Compact variants provided the best trade-off between accuracy and computational cost, whereas deeper architectures yielded marginal improvements at higher *Latency*. TensorRT optimization on the Jetson device further enhanced real-time inference, particularly for embedded deployment. The results indicate that modern YOLO architectures have reached a level of architectural maturity, where advances are driven by optimization and specialization rather than structural redesign. These findings underline the strong potential of YOLO-based detectors as core components of intelligent, edge-deployable systems for precision agriculture and automated fruit detection.

Keywords: convolutional neural networks; CNN; machine vision; *Rubus*; You Only Look Once

check for updates

Academic Editor: Grzegorz Lysiak

Received: 18 November 2025
Revised: 3 December 2025
Accepted: 4 December 2025
Published: 6 December 2025

Citation: Buczyński, K.; Kapłan, M.; Jarosz, Z. Detection of Red, Yellow, and Purple Raspberry Fruits Using YOLO Models. *Agriculture* **2025**, *15*, 2530. https://doi.org/10.3390/agriculture15242530

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Species of the genus *Rubus* L. are among the most widely cultivated berry crops worldwide [1] and exhibit a high level of genetic diversity [2]. They are valued not only for their desirable flavor qualities but also as an important source of nutrients and health-promoting compounds [3,4]. Raspberries are harvested both manually, as fresh dessert fruits intended for the retail market, and either mechanically or manually for processing purposes. Given the increasing labor costs and the decreasing availability of seasonal workers, which vary by cultivation region, the development of robotic systems for raspberry harvesting is highly desirable.

As various sectors of society and the economy move toward automation and autonomous solutions, robotic systems are increasingly being employed to perform complex tasks. In recent years, such systems have been widely implemented in agriculture, where they replace human labor traditionally regarded as highly time-consuming [5], and, in

particular, research on the optimization of agronomic practices in raspberry production is of great importance [6–8].

Moreover, mass harvesting methods carry the risk of crop damage; therefore, selective harvesting where a robotic system identifies and picks only ripe fruits is increasingly being adopted. However, fruit detection and automated harvesting remain challenging tasks due to occlusions, variable lighting conditions, and color similarities. Automated agriculture, particularly in the domain of autonomous harvesting, continues to represent an open field of research requiring further development and refinement [9]. With the advancement of science and technology, automated fruit harvesting integrates solutions from computer science, perception, control, and robotics, contributing to labor cost reduction and the development of modern, intelligent agriculture. Machine vision technology plays a particularly important role, as it is fundamental to the advancement of intelligent perception in agricultural systems. It is anticipated that continued technological progress, along with support from public policy, will significantly accelerate the development of automatic fruit harvesting systems [10]. Visual perception technology is of fundamental importance in fruit-harvesting robots, enabling precise recognition, localization, and grasping of the fruits [11]. In addition, key components include autonomous navigation, motion planning, and control, which constitute integral elements of the software architecture in robotic harvesting systems [12]. Fruit and vegetable harvesting robots play a key role in the modernization of agriculture through their efficiency and precision. The rapid advancement of deep learning technologies has significantly stimulated research on robotic systems for fruit and vegetable harvesting; however, their practical implementation still faces numerous technical challenges. The variability and complexity of agricultural environments demand high robustness and strong generalization capabilities from detection algorithms [13]. Deep learning, a branch of machine learning and artificial intelligence (AI), began to gain prominence in the early twenty-first century with the growing popularity of artificial neural networks, multilayer perceptrons, and support vector machines [14].

The application of deep learning to fruit recognition has become one of the main directions of technological advancement in agriculture. These methods have significantly improved the accuracy of fruit identification under complex conditions, including variable illumination and partial occlusions [15]. Among the various types of deep neural networks, convolutional neural networks (CNN) have been the most extensively studied. Although they have achieved remarkable success in experimental evaluations, many research challenges remain unresolved. The increasing complexity of contemporary CNN models entails the need for large-scale datasets and substantial computational power during training [16]. A convolutional neural network is a deep, hierarchical feed-forward architecture based on the principles of local neuron connectivity, weight sharing, and dimensionality reduction, inspired by the concept of receptive fields in biological vision. As a deep learning model, it enables end-to-end learning, wherein features of the input data are progressively transformed into increasingly complex representations used for classification or other tasks. As the complexity of computer vision problems continues to grow, there is a rising demand for CNN models with higher performance and computational efficiency [17,18]. The YOLO (You Only Look Once) algorithm, developed in 2015, rapidly gained popularity due to its high efficiency. Unlike two-stage detectors, it performs object detection within a single neural network, formulating the task as a regression problem [19]. It directly transforms the input image into bounding box coordinates and their corresponding class probabilities. The network divides the image into an $S \times S$ grid, where each cell predicts bounding boxes along with the associated class probabilities. As a result, the entire detection process is performed within a single convolutional neural network, enabling end-to-end optimization and achieving high processing speed while maintaining high accuracy [20]. The YOLO

algorithm has rapidly gained popularity in agriculture due to its high accuracy, operational speed, and compact network architecture. It has been applied to a wide range of tasks, including monitoring, detection, automation, and the robotization of agricultural processes. Although research on its applications in this field is developing dynamically, it remains fragmented and spans multiple scientific disciplines [21].

The implementation of convolutional neural networks in raspberry cultivation remains very limited, clearly indicating the need for further research aimed at optimizing this technology for use in these crops. For this reason, the aim of our study was to evaluate the effectiveness of recent YOLO architectures for detecting raspberry fruits of red, yellow, and purple color varieties under field conditions. In addition to assessing detection accuracy, the study also compares the performance of these models to clarify how architectural differences influence detection capability and computational efficiency.

2. Materials and Methods

2.1. Data Collection

2.1.1. Overview of the Research Area

The experimental raspberry plantation was located in eastern Poland at coordinates 51.194325° N, 21.816613° E. The experimental area comprised approximately 1520 m of plantation rows, containing around 3800 raspberry bushes planted at 40 cm spacing.

The red fruit images included the following floricane raspberry varieties: Glen Ample, Glen Carron, Glen Mor, and Przehyba. The dataset also comprised the pri-mocane varieties Polana, Polesie, Poemat, Polonez, Poranek, Delniwa, Enrosadira, and Husaria. All of these varieties belong to the *Rubus idaeus* L. species. The yellow raspberries, which also belong to this species, included the primocane varieties Poranna Rosa, Jantar, and Promyk. For purple raspberries, the fruit images were taken of a single primocane variety, Heban, which belongs to the species *Rubus* × *neglectus* Peck.

2.1.2. Image Acquisition Procedure

Data was collected using a Samsung Galaxy S25 smartphone (Samsung Electronics Co., Ltd., Suwon-si, Republic of Korea) equipped with RGB main camera (50 MP, 8160 × 6120 px F/1.8 aperture, sensor size 1/1.56 inches and 24 mm lens) [22], stabilized by a Hohem iSteady M7 gimbal (Shenzhen Hohem Technology Co., Ltd., Shenzhen Guangdong, China) [23]. Photographs were taken from viewpoints at distances not exceeding 50 cm from the bushes, reflecting typical conditions encountered during routine fruit harvesting and bushes inspection, and naturally included both frontal and oblique angles. Images were collected in the open-field conditions, between 20 June and 20 September 2025, covering the full fruiting period of each evaluated raspberry variety. In total, 10,115 images were captured for red raspberries, 2319 for yellow raspberries, and 1125 for purple raspberries. Images were collected across the active harvesting sections of each cultivar rather than from individually tracked plants. This strategy was intentional, as the objective of the study was to capture the full visual variability of fruit appearance under field conditions.

The difference in the number of photos resulted from the varying availability of research material; red raspberries dominate raspberry cultivation in Poland, while yellow and purple raspberries have not yet been introduced into mass production.

In order to improve the generalization performance of the trained object detection models, image data were collected under diverse agricultural, illumination, and weather conditions. The dataset includes photographs taken from different viewing angles during daylight hours, covering a wide range of times of day, cloud coverage levels, sunlight intensities, and plant surface moisture states, from dry to wet, due to rainfall or dew formation. This variability captures natural fluctuations in image contrast and specular

reflection, both of which significantly affect the robustness and operational accuracy of computer vision systems in real-world environments.

2.2. Data Preprocessing

2.2.1. Image Scaling

Original high-resolution images (8160 × 6120 px) were captured to preserve the native sensor output and avoid device-side processing typical of low-resolution modes. All images were then downscaled to 640 × 480 px while preserving the native 4:3 aspect ratio, using a reproducible Python preprocessing script. During models training the input images (640 × 480 px) were resized to 640 × 640 px via letterbox padding in the YOLO data loader, maintaining the native aspect ratio and avoiding shape distortion of the fruit targets. The 640-pixel input matches the standard YOLO training pipeline and ensures consistency across the dataset. All preprocessing, training, and evaluation scripts are publicly available in the project repository [24].

2.2.2. Images Labeling

During annotation in the LabelImg tool [25], fruits at full harvest maturity were labeled, including both fully visible and partially occluded ones. This approach aimed to better reflect real harvesting conditions, where not all fruits are entirely visible. In the case of red raspberries, a total of 83,162 objects were annotated; additionally, 18,376 objects were labeled for yellow raspberries and 8901 for purple raspberries.

2.2.3. Preparing Datasets

Each raspberry image series was stored in a separate top-level folder corresponding to a single data-collection session. The sessions were conducted on different days and under varying field conditions, including changes in natural illumination, cloud coverage, humidity, plant appearance, and camera–plant distance. Each folder therefore represents a coherent set of images acquired under a specific set of environmental conditions.

For each single-color dataset, the division into the training, validation, and test subsets was performed using a deterministic random sampling procedure with a fixed random seed of 42. Within each acquisition folder, the image list was shuffled using a seeded pseudo-random generator and assigned to the three subsets according to an approximate 80:10:10 ratio. This procedure ensures full reproducibility of the split while maintaining proportional representation of all acquisition sessions—and thus all environmental conditions—across the training, validation, and test partitions. Because each single-color dataset contains only one object class (red, yellow, or purple raspberries), stratified sampling was not required, and uniform deterministic random sampling was sufficient. Table 1 presents a detailed breakdown of the number of images and annotated objects for each subset.

Table 1. Number of images and annotated objects for each dataset.

Set		Red	Yellow	Purple
Train	Photos	8092	1855	900
	Tagged objects	66,561	14,629	7187
Valid	Photos	1011	232	113
	Tagged objects	8329	1956	888
Test	Photos	1012	232	112
	Tagged objects	8272	1791	826

In addition to the three single-color datasets, a dedicated multi-class dataset was constructed, containing images of red, yellow, and purple raspberry fruits annotated as separate classes. For this dataset, the same train/validation/test split policy was applied

as in the single-class scenarios. Equal numbers of images from each color category were selected for the training and validation subsets to ensure balanced class representation, while the full test partitions were kept unchanged. In total, the mixed-color dataset comprised 4395 images and 35,122 annotated raspberry instances. The number of images and annotations contributed by each color class to the mixed dataset is summarized in Table 2.

Table 2. Number of images and annotated objects for each class.

Set		Red	Yellow	Purple
Train	Photos	900	900	900
	Tagged objects	7313	7023	7187
Valid	Photos	113	113	113
	Tagged objects	917	905	888
Test	Photos	1012	232	112
	Tagged objects	8272	1791	826

2.3. Models Training

2.3.1. YOLO Models

The selection of YOLOv8s, YOLOv9s, YOLOv10s, YOLO11s, and YOLO12s [26] was guided by the objective of capturing the architectural evolution of the YOLO family in its most recent development stages. These models represent successive design iterations that progressively refine convolutional backbones, feature aggregation strategies, and training dynamics while maintaining a unified design philosophy focused on real-time detection.

Including the lightweight "s" variants ensured a consistent comparison under equivalent parameter scales, allowing for a direct assessment of how architectural modifications rather than model size affect accuracy, inference speed, and generalization. YOLOv8s was chosen as the reference baseline due to its maturity and extensive adoption in applied computer vision tasks. Subsequent versions, from YOLOv9s to YOLO12s, were selected to evaluate how incremental refinements in feature representation, attention mechanisms, and optimization strategies translate into real-world detection performance and computational efficiency. This sequential approach provides a structured view of model progression across generations. It enables analysis not only of performance gains but also of the diminishing returns that accompany increasingly complex architectures.

2.3.2. Training Settings

Training was executed for 200 epochs with a batch size of 32 and an image resolution of 640 × 640 pixels. The stochastic gradient descent (SGD) optimizer was employed with an initial learning rate of 0.01, a final learning-rate factor of 0.01, a momentum coefficient of 0.937, and a weight decay of 0.0005. The learning rate was governed by the default Ultralytics cosine-annealing schedule, which incorporates a brief linear warm-up phase followed by a smooth cosine decay from the initial to the final value. No custom modifications to the scheduler were introduced, ensuring full reproducibility of the training procedure. Data augmentation followed the default Ultralytics YOLO pipeline enabled during training. The augmentation procedure included horizontal flipping (p = 0.5), random scaling, translation and shear transformations (affine augmentation), HSV adjustments to hue, saturation and value, and random brightness and contrast modifications. Additionally, stochastic sharpening and blurring were applied by the dataloader. Mosaic and MixUp augmentations were not used. These techniques are intended for large, highly diverse datasets and can degrade performance on small-object detection tasks such as raspberry localization. The training therefore relied solely on the standard YOLO augmentations, which were applied on-the-fly during data loading to maintain consistent and reproducible preprocessing. Early stopping

was deactivated (patience = 0) to allow complete convergence. A fixed seed (seed = 0) ensured reproducible results. All experiments were carried out on an NVIDIA GPU with CUDA support (Figure 1).

Epochs	200
Batch	32
Imgsz	640
Optimizer	'SGD'
Momentum	0.937
Weight_decay	0.0005
lr_0	0.01
lr_f	0.01
Seed	0
Augment	True
Workers	8
Patience	0
Device	'cuda'

Figure 1. Training hyperparameter settings.

2.4. Evaluation of Trained Models

Model evaluation was performed using standard object detection metrics, namely *Precision*, *Recall*, *mean Average Precision* at an *IoU* threshold of 0.5 (mAP_{50}), *mean Average Precision* averaged over *IoU* thresholds ranging from 0.5 to 0.95 ($mAP_{50:95}$), and the F_1-*score*. The computation of these metrics was automated via lightweight Python utilities developed upon the official YOLO evaluation framework, ensuring reproducibility and consistency across models and datasets. All reported detection metrics were computed only on the independent test set, which was not used during training.

Precision and *Recall* were employed to measure the correctness and completeness of the detections, respectively, as expressed in Equations (1) and (2), where *TP*, *FP*, and *FN* denote the numbers of true positives, false positives, and false negatives.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

The *mean Average Precision* (*mAP*) metric measures the area under the *Precision–Recall curve*, expressing the relationship between detection confidence and accuracy over varying threshold levels. The mAP_{50} value corresponds to evaluation at a fixed IoU threshold of 0.5, whereas $mAP_{50:95}$ represents the average performance computed over IoU thresholds from 0.5 to 0.95 in 0.05 increments, as formulated in Equations (3) and (4).

$$mAP_{50} = \frac{1}{N}\sum_{i=1}^{N} AP_i(IoU = 0.5) \tag{3}$$

$$mAP_{50:95} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{10}\sum_{k=0}^{9} AP_i(IoU = 0.5 + 0.05k)\right) \quad (4)$$

The F_1-*score* (Equation (5)) defines the harmonic mean between *Precision* and *Recall*, providing a unified measure of detection accuracy and completeness. In conjunction, these metrics yield a comprehensive evaluation of model performance, encompassing localization precision, confidence calibration, and robustness across all image modalities.

$$F_1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (5)$$

Inference efficiency was analyzed within a controlled benchmarking setup that employed synthetic input data to decouple computational performance from dataset-induced variability. This approach ensured consistent evaluation of frame rate and *Latency* across all model architectures, independent of accuracy-related or data-specific factors.

Model inference was assessed across two computational platforms: a desktop workstation equipped with an NVIDIA RTX 5080 GPU (ASUSTek Computer Inc., Taipei, Taiwan) and an embedded device based on the Jetson Orin NX 16 GB module (Seeed Technology Co., Ltd., Shenzhen, China). Both environments utilized the same benchmarking procedure implemented within the Ultralytics YOLO framework, operating in FP16 precision and CUDA-accelerated inference mode. Evaluations were performed using the optimal model checkpoint (best.pt).

For the RTX 5080 platform, inference testing was carried out in PyTorch (CUDA) using input tensors of dimensions ($1 \times 3 \times 640 \times 640$). Each model was benchmarked in three consecutive sessions, with 50 warm-up iterations preceding and 200 timed iterations following per session. To ensure statistical reliability, five independent measurement cycles were executed for each model. Between sessions, a 90 s idle period was introduced to maintain consistent GPU temperature and clock frequency. Final throughput and *Latency* values were calculated as the mean after discarding results outside the 10th–90th percentile range, thereby minimizing transient variability and ensuring stable *FPS* and *Latency* estimates.

On the Jetson Orin NX platform, the same benchmarking methodology was employed with a reduced iteration schedule—20 warm-up iterations followed by 60 timed inferences—and an additional 10 s stabilization interval before each run to ensure GPU readiness. Models were evaluated in groups of three, with a 150 s cooldown period between sessions to mitigate potential thermal throttling effects. Two inference back-ends were examined under identical experimental conditions: PyTorch (using the best-performing checkpoint, best.pt) and TensorRT (using the optimized engine, best.engine). Each configuration was executed across five independent trials, and results were post-processed by excluding values outside the 20th–80th percentile range to improve statistical consistency and measurement stability.

For deployment on the Jetson Orin platform, all trained PyTorch models were exported to TensorRT using the Ultralytics conversion pipeline. The engines were built in FP16 precision, with a batch size of 1 and a static input resolution of 640×640 pixels. Engine construction used a 2 GB workspace, enabled graph simplification, and disabled dynamic shapes, allowing TensorRT to perform full kernel fusion and memory optimization. No INT8 calibration or post-training quantization was applied. All models were exported using the same configuration to ensure a fair and reproducible comparison of edge-device performance.

In addition to detection accuracy, evaluating the computational efficiency of the models is critical, especially for deployment on platforms with constrained hardware capabilities. Consistent with the performance assessment principles adopted in recent YOLO frameworks, three complementary indicators were employed to characterize the balance between

accuracy and real-time inference capability. All performance measurements were obtained using dedicated Python utilities developed specifically for this study.

The *Frames Per Second* (*FPS*) metric represents the rate at which images are processed during inference. It is defined as the ratio between the number of forward passes N_f and the total inference time t. Higher *FPS* values indicate faster inference speeds and, consequently, improved suitability for real-time applications (Equation (6)).

$$FPS = \frac{N_f}{t} \tag{6}$$

Latency quantifies the mean processing time required to analyze a single image, expressed in milliseconds. It is mathematically the reciprocal of the *FPS* metric and serves as a direct indicator of model responsiveness. Low *Latency* is particularly critical for onboard device inference scenarios, where timely decision-making relies on minimal computational delay (Equation (7)).

$$Latency = \frac{1}{FPS} \cdot 1000 \tag{7}$$

The *Accuracy–Speed Ratio* provides a unified scalar measure that combines detection performance with computational efficiency. It expresses the amount of detection accuracy ($mAP_{50:95}$) achieved per unit of inference speed (*FPS*). Higher values of this metric denote a more optimal trade-off between predictive accuracy and runtime performance, serving as a compact indicator of overall model efficiency (Equation (8)).

$$Accuracy\ speed\ ratio\ (ASR) = \frac{mAP_{50:95}}{FPS} \tag{8}$$

Performance indicators, including *FPS*, *Latency*, and the *Accuracy–Speed Ratio*, were calculated as the mean ± standard deviation over five independent inference trials for each model. This evaluation procedure reflects both the central tendency of computational performance and its variability across repeated measurements, ensuring a reliable characterization of model efficiency.

2.5. Hardware Configuration

Figure 2 provides an overview of the computational setup employed for model training and evaluation. All YOLO architectures were trained on a high-performance workstation equipped with an AMD Ryzen 9 9950X processor and an NVIDIA GeForce RTX 5080 graphics unit (16 GB VRAM). The system was configured to maintain stable multi-GPU operation and consistent batch throughput throughout all training sessions. The software environment comprised Microsoft Windows 11 Pro, Python 3.11, PyTorch 2.8, and CUDA 12.8, ensuring seamless integration with the Ultralytics YOLO framework. This configuration delivered reproducible training results and served as the benchmark reference for inference performance in subsequent embedded experiments.

The reComputer J4012 platform (Seeed Studio) [27] is powered by the NVIDIA Jetson Orin NX 16 GB module, which integrates an 8-core Arm® Cortex-A78AE v8.2 processor and a 1024-core NVIDIA Ampere GPU with 32 Tensor Cores. The system operated on Ubuntu 22.04.5 LTS (aarch64) with JetPack 6.1, providing CUDA 12.6, cuDNN 9.3, and TensorRT 10.3 support, as summarized in Figure 3. This configuration was dedicated to inference benchmarking, facilitating evaluation of model efficiency in embedded hardware environments.

Component	Specification
Processor	AMD Ryzen 9 9950X
Graphics Processing Unit	ASUS GeForce RTX 5080 PRIME OC, 16 GB
Motherboard	Gigabyte B850 AORUS ELITE WIFI7 AM5
Memory	Corsair Vengeance, DDR5 2 × 32GB (64 GB)
Storage	SSD Lexar NM790 2TB
Software	**Version**
Microsoft Windows	11 Pro (build 26200, 64-bit)
Python	3.11.13
PyTorch	2.8.0
CUDA	12.8
cuDNN	9.1.0.2 (NVIDIA build 91002)
Ultralytics	8.3.13

Figure 2. Setup of primary workstation.

Component	Specification
Processor	8-core Arm® Cortex®-A78AE v8.2
Graphics Processing Unit	1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores
Memory	16GB 128-bit LPDDR5
Storage	NVMe SSD Foresee XP1000F128G
Software	**Version**
Ubuntu	22.04.5 LTS (aarch64)
Jetpack/(L4T)	6.1 (R36.4.0, kernel 5.15.148-tegra)
Python	3.10.12
PyTorch	2.5.0a0+872d972e41.nv24.08
CUDA Toolkit	12.6
cuDNN	9.3.0
TensortRT	10.3.0.30-1 + CUDA 12.5
Ultralytics	8.3.221

Figure 3. Setup of the edge device.

3. Results

All YOLO architectures achieved consistently high and closely aligned results in the raspberry fruit detection task. *Precision*, *Recall*, and overall detection accuracy remained highly stable across generations, reflecting the maturity of the underlying feature extraction designs. The smallest YOLOv8s variant already provided strong predictive balance, while subsequent versions offered only marginal fluctuations in performance. Among the models, the YOLOv9s and YOLO12s generations slightly improved *Recall*, resulting in a marginal increase in the F_1-*score*, whereas other architectures maintained nearly identical outcomes. The *mAP* values at both lenient and strict thresholds confirmed that all models localized raspberry fruits with comparable reliability and robustness (Table 3). Figure 4 shows examples of red raspberry fruit detection using the trained YOLO12s model. Blue bounding boxes indicate detected fruits, and the numeric values represent confidence scores. All images were processed at 640 × 480 resolution (letterboxed to 640 × 640 during inference).

Table 3. Evaluation of red raspberry fruits detection.

Model	*Precision*	*Recall*	mAP_{50}	$mAP_{50:95}$	F_1-*Score*
YOLOv8s	0.902	0.909	0.959	0.672	0.906
YOLOv9s	0.900	0.921	0.962	0.680	0.910
YOLOv10s	0.899	0.905	0.959	0.674	0.902
YOLO11s	0.906	0.909	0.960	0.677	0.907
YOLO12s	0.906	0.919	0.962	0.679	0.912



Figure 4. Red raspberry fruit detection example by the trained YOLO12s model.

Across all evaluated models, detection performance for yellow raspberry fruits remained high and consistent, although with slightly greater variability between architectures, reflected in a wider spread of *Precision*, *Recall*, F_1-score and *mAP* values compared with the red raspberry dataset.

The YOLO11s and YOLO12s variants achieved a balanced trade-off between *Precision*, *Recall*, *mean average precision* and F_1-*Score*, yielding the most consistent overall performance. Earlier architectures preserved strong generalization with negligible degradation in detection quality (Table 4). Figure 5 shows examples of yellow raspberry fruit detection using the trained YOLO12s model. Blue bounding boxes indicate detected fruits, and the numeric values represent confidence scores. All images were processed at 640 × 480 resolution (letterboxed to 640 × 640 during inference).

Table 4. Evaluation of yellow raspberry fruit detection.

Model	*Precision*	*Recall*	mAP_{50}	$mAP_{50:95}$	F_1-*Score*
YOLOv8s	0.915	0.871	0.949	0.692	0.893
YOLOv9s	0.911	0.886	0.951	0.701	0.898
YOLOv10s	0.875	0.896	0.940	0.680	0.885
YOLO11s	0.900	0.902	0.948	0.692	0.901
YOLO12s	0.891	0.912	0.954	0.698	0.901

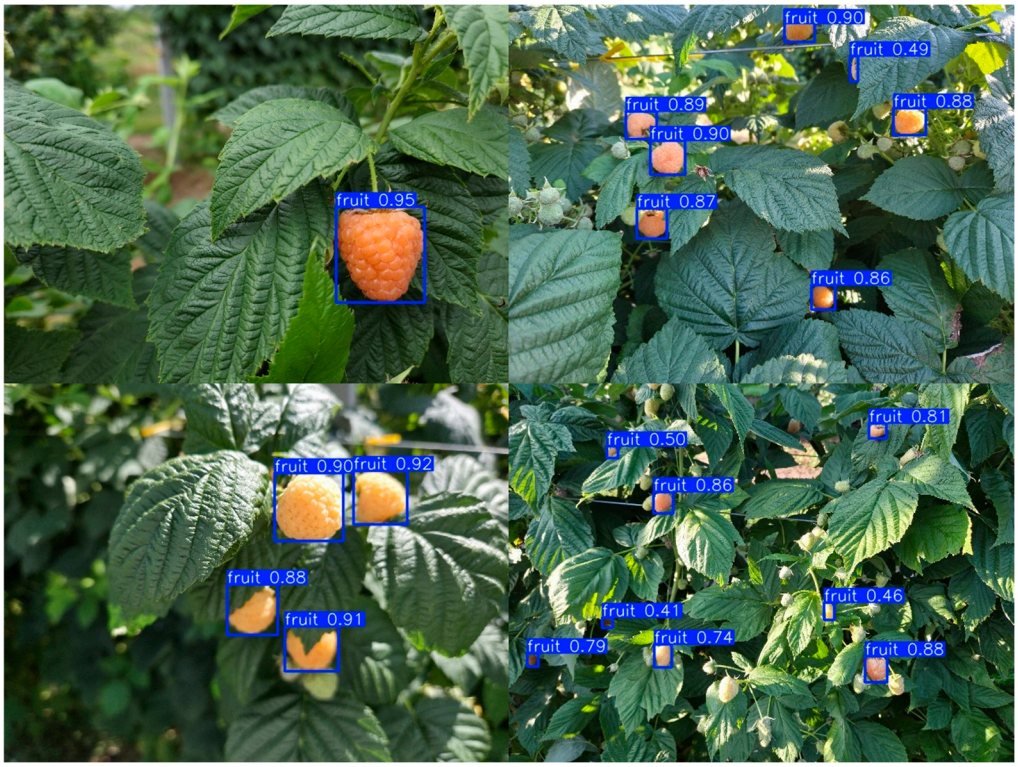


Figure 5. Yellow raspberry fruit detection example by the trained YOLO12s model.

The detection of purple raspberry fruits produced slightly lower and more variable results across models, reflected in a wider spread of *Precision*, *Recall*, F_1*-score* and *mAP* values compared with the red and yellow datasets.

All architectures maintained strong overall accuracy, yet *Precision* and *Recall* values revealed subtle trade-offs between sensitivity and selectivity. The YOLOv8s and the newest YOLO12s model achieved the most balanced predictions, while intermediate architectures exhibited minor fluctuations in performance stability. The *mAP* metrics confirmed consistent detection accuracy, with only slight differences observed among the models. The stricter *mAP* threshold values suggested a marginal decline in fine localization precision (Table 5). Figure 6 shows examples of purple raspberry fruit detection using the trained YOLO12s model. Blue bounding boxes indicate detected fruits, and the numeric values represent confidence scores. All images were processed at 640 × 480 resolution (letterboxed to 640 × 640 during inference).

Table 5. Evaluation of purple raspberry fruit detection.

Model	*Precision*	*Recall*	mAP_{50}	$mAP_{50:95}$	F_1*-Score*
YOLOv8s	0.876	0.851	0.923	0.645	0.863
YOLOv9s	0.898	0.815	0.925	0.650	0.855
YOLOv10s	0.847	0.864	0.916	0.635	0.856
YOLO11s	0.842	0.869	0.923	0.654	0.855
YOLO12s	0.909	0.855	0.924	0.655	0.881



Figure 6. Purple raspberry fruit detection example by the trained YOLO12s model.

The multi-class evaluation encompassing red, yellow, and purple raspberry fruits revealed highly consistent detection performance across all YOLO architectures. Differences between fruit classes were minor, indicating that the models maintained stable generalization across varying color distributions. Yellow and red fruits achieved closely aligned *Precision* and *Recall* values, with neither class showing a clear advantage. Purple fruits exhibited slightly lower scores overall, suggesting a modest increase in detection difficulty likely related to lower color contrast and higher spectral similarity with background foliage. Across generations, performance fluctuations were minimal and did not follow a strict upward trend, indicating that all tested architectures have reached a maturity level where further architectural modifications yield only marginal changes in detection accuracy. The observed uniformity across classes confirms that the multi-class training strategy effectively balances feature representation among fruit types, enabling consistent detection without significant class bias or degradation in localization precision. Among the evaluated architectures, the YOLO12s model achieved the highest average precision and F_1-*score*, indicating a slight overall advantage in balanced detection performance. YOLO11s followed closely, showing strong consistency across all metrics and maintaining particularly stable *Recall*. Earlier architectures such as YOLOv9s achieved competitive *Recall* and localization accuracy but exhibited marginally lower precision, while YOLOv8s and YOLOv10s maintained balanced yet slightly lower average scores. These trends suggest that the most recent YOLO generations deliver modest but measurable improvements in detection quality, primarily reflected in enhanced precision and class stability (Table 6).

Table 7 presents the performance results of the models obtained using an RTX and the PyTorch framework. Across all fruit color datasets, inference speed and *Latency* values followed consistent trends. The smallest YOLOv8s model achieved the highest frame rate and the lowest *Latency*, confirming its strong computational efficiency on high-performance GPU hardware. In contrast, the YOLOv9s variant consistently exhibited the slowest processing speeds and the highest *Latency*, which directly translated into the least favorable *Accuracy–speed ratio* among the evaluated models.

Table 6. Evaluation of mixed-color raspberry fruit detection.

Model	Fruit Color	*Precision*	*Recall*	mAP_{50}	$mAP_{50:95}$	F_1-*Score*
YOLOv8s	Red	0.891	0.856	0.936	0.643	0.873
	Yellow	0.866	0.871	0.925	0.666	0.868
	Purple	0.842	0.867	0.912	0.633	0.854
	Average	0.866	0.865	0.924	0.647	0.865
YOLOv9s	Red	0.884	0.885	0.937	0.644	0.884
	Yellow	0.910	0.869	0.941	0.677	0.889
	Purple	0.851	0.882	0.934	0.658	0.866
	Average	0.882	0.879	0.938	0.660	0.880
YOLOv10s	Red	0.848	0.887	0.931	0.636	0.867
	Yellow	0.877	0.858	0.931	0.670	0.868
	Purple	0.830	0.872	0.920	0.645	0.850
	Average	0.852	0.872	0.927	0.650	0.862
YOLO11s	Red	0.903	0.864	0.942	0.644	0.883
	Yellow	0.898	0.877	0.942	0.678	0.887
	Purple	0.866	0.878	0.925	0.646	0.872
	Average	0.889	0.873	0.936	0.656	0.881
YOLO12s	Red	0.888	0.883	0.944	0.650	0.886
	Yellow	0.926	0.868	0.948	0.687	0.896
	Purple	0.875	0.871	0.933	0.652	0.873
	Average	0.896	0.874	0.942	0.663	0.885

Table 7. YOLO models performance on RTX (PyTorch).

Fruits Color	Model	*FPS*	*Latency* (*ms*)	*Accuracy Speed Ratio*
Red	YOLOv8s	246.63 ± 0.87	4.05 ± 0.01	0.0027 ± 0.00001
	YOLOv9s	99.08 ± 0.17	10.09 ± 0.02	0.0069 ± 0.00001
	YOLOv10s	209.62 ± 0.60	4.77 ± 0.01	0.0032 ± 0.00001
	YOLO11s	196.69 ± 1.51	5.08 ± 0.04	0.0034 ± 0.00003
	YOLO12s	144.69 ± 0.99	6.91 ± 0.05	0.0047 ± 0.00003
Yellow	YOLOv8s	243.96 ± 1.32	4.10 ± 0.02	0.0028 ± 0.00002
	YOLOv9s	98.44 ± 0.18	10.16 ± 0.02	0.0071 ± 0.00001
	YOLOv10s	208.29 ± 0.77	4.80 ± 0.02	0.0033 ± 0.00001
	YOLO11s	199.01 ± 0.54	5.02 ± 0.01	0.0035 ± 0.00001
	YOLO12s	146.89 ± 0.51	6.81 ± 0.02	0.0048 ± 0.00002
Purple	YOLOv8s	247.19 ± 1.08	4.05 ± 0.02	0.0026 ± 0.00001
	YOLOv9s	98.69 ± 0.17	10.13 ± 0.02	0.0066 ± 0.00001
	YOLOv10s	207.21 ± 1.56	4.83 ± 0.04	0.0031 ± 0.00002
	YOLO11s	200.65 ± 1.12	4.98 ± 0.03	0.0033 ± 0.00002
	YOLO12s	148.64 ± 0.62	6.73 ± 0.03	0.0044 ± 0.00002
Mix Colors	YOLOv8s	245.99 ± 0.81	4.07 ± 0.01	0.0026 ± 0.00001
	YOLOv9s	98.72 ± 0.15	10.13 ± 0.01	0.0067 ± 0.00001
	YOLOv10s	208.16 ± 1.16	4.80 ± 0.03	0.0031 ± 0.00002
	YOLO11s	200.61 ± 0.31	4.98 ± 0.01	0.0033 ± 0.00001
	YOLO12s	148.38 ± 0.22	6.74 ± 0.01	0.0045 ± 0.00001

Architectures such as YOLOv10s and YOLO11s maintained stable throughput with only moderate increases in *Latency* compared to YOLOv8s, offering balanced efficiency between inference speed and predictive accuracy. The most recent YOLO12s model demonstrated a noticeable reduction in frame rate accompanied by a rise in *Latency* and accuracy–speed ratio, indicating a moderate computational overhead associated with its deeper architecture and refined feature extraction layers.

When considering dataset variations, all models demonstrated nearly identical performance across red, yellow, purple, and mixed-color subsets. The stability of *FPS* and *Latency* metrics across these subsets indicates that color variability within the fruit datasets had negligible influence on computational efficiency. Overall, YOLOv8s remained the fastest and most resource-efficient model, while YOLOv9s achieved the slowest execution, confirming a clear trade-off between architectural sophistication and inference speed.

Table 8 presents the performance results of the models obtained using an Jetson Orin and the PyTorch framework.

Table 8. YOLO models performance on Orin (PyTorch).

Fruits Color	Model	*FPS*	*Latency* (*ms*)	*Accuracy Speed Ratio*
Red	YOLOv8s	53.74 ± 0.38	18.61 ± 0.13	0.0125 ± 0.00009
	YOLOv9s	24.05 ± 0.06	41.59 ± 0.10	0.0283 ± 0.00007
	YOLOv10s	48.63 ± 0.26	20.57 ± 0.11	0.0139 ± 0.00007
	YOLO11s	46.97 ± 0.11	21.29 ± 0.05	0.0144 ± 0.00003
	YOLO12s	32.58 ± 0.08	30.70 ± 0.07	0.0208 ± 0.00005
Yellow	YOLOv8s	53.57 ± 0.37	18.67 ± 0.13	0.0129 ± 0.00009
	YOLOv9s	23.84 ± 0.04	41.95 ± 0.07	0.0294 ± 0.00005
	YOLOv10s	48.23 ± 0.19	20.73 ± 0.08	0.0141 ± 0.00005
	YOLO11s	46.62 ± 0.27	21.45 ± 0.12	0.0148 ± 0.00009
	YOLO12s	32.34 ± 0.06	30.93 ± 0.05	0.0216 ± 0.00004
Purple	YOLOv8s	53.74 ± 0.08	18.61 ± 0.03	0.0120 ± 0.00002
	YOLOv9s	23.80 ± 0.03	42.01 ± 0.06	0.0273 ± 0.00004
	YOLOv10s	48.30 ± 0.15	20.71 ± 0.06	0.0131 ± 0.00004
	YOLO11s	46.83 ± 0.12	21.36 ± 0.06	0.0140 ± 0.00004
	YOLO12s	32.20 ± 0.03	31.05 ± 0.02	0.0203 ± 0.00002
Mix Colors	YOLOv8s	53.26 ± 0.26	18.77 ± 0.09	0.0121 ± 0.00006
	YOLOv9s	23.65 ± 0.02	42.29 ± 0.04	0.0279 ± 0.00003
	YOLOv10s	48.16 ± 0.14	20.77 ± 0.06	0.0135 ± 0.00004
	YOLO11s	46.55 ± 0.13	21.48 ± 0.06	0.0141 ± 0.00004
	YOLO12s	32.20 ± 0.04	31.06 ± 0.04	0.0206 ± 0.00003

Among the evaluated architectures, YOLOv8s consistently achieved the highest frame rate and the lowest *Latency* across all fruit color subsets, confirming its superior computational efficiency and suitability for real-time edge deployment.

YOLOv9s again represented the least efficient configuration, recording the lowest throughput and the highest *Latency* across every dataset. This translated directly into the highest *Accuracy–speed ratio* values, emphasizing the significant computational cost associated with this generation. In contrast, YOLOv10s and YOLO11s maintained stable performance, with only moderate increases in *Latency* relative to YOLOv8s, offering a balanced compromise between processing speed and architectural complexity. The YOLO12s model demonstrated a noticeable slowdown accompanied by a higher *Latency* and less favorable efficiency ratios, consistent with its deeper backbone and extended feature extraction stages.

Performance trends remained stable across all color datasets, with negligible variation in inference metrics between red, yellow, purple, and mixed-color subsets. This consistency suggests that spectral diversity in the input data did not significantly influence computational demands. Overall, the hierarchy of efficiency observed on RTX was preserved on Orin, but with reduced absolute performance, underscoring the trade-off between model depth and runtime efficiency under embedded hardware constraints.

Table 9 presents the performance results of the models obtained using an Orin and the TensorRT framework. Under TensorRT, YOLOv10s delivered the highest throughput

and the lowest *Latency* across all fruit subsets, yielding the most favorable accuracy–speed trade-off. YOLOv8s and YOLO11s formed a tight cluster with slightly lower frame rates and marginally higher *Latency*, maintaining solid real-time behavior. YOLOv9s trailed these models with reduced speed and less efficient trade-offs, while YOLO12s was the slowest, reflecting the overhead of its deeper architecture on embedded hardware. Performance patterns were consistent across red, yellow, purple, and mixed-color data, indicating that input color variation did not materially affect computational efficiency.

Table 9. YOLO models performance on Orin (TensorRT).

Fruits Color	Model	*FPS*	*Latency* (*ms*)	*Accuracy Speed Ratio*
Red	YOLOv8s	65.60 ± 0.21	15.24 ± 0.05	0.0102 ± 0.00003
	YOLOv9s	55.09 ± 0.30	18.15 ± 0.10	0.0123 ± 0.00007
	YOLOv10s	75.14 ± 0.34	13.31 ± 0.06	0.0090 ± 0.00004
	YOLO11s	65.81 ± 0.23	15.20 ± 0.05	0.0103 ± 0.00003
	YOLO12s	48.44 ± 0.31	20.65 ± 0.13	0.0140 ± 0.00009
Yellow	YOLOv8s	65.76 ± 0.23	15.21 ± 0.05	0.0105 ± 0.00004
	YOLOv9s	54.74 ± 0.42	18.27 ± 0.14	0.0128 ± 0.00010
	YOLOv10s	75.31 ± 0.93	13.28 ± 0.16	0.0090 ± 0.00011
	YOLO11s	66.01 ± 0.24	15.15 ± 0.06	0.0105 ± 0.00004
	YOLO12s	48.35 ± 0.23	20.68 ± 0.10	0.0144 ± 0.00007
Purple	YOLOv8s	65.72 ± 0.22	15.22 ± 0.05	0.0098 ± 0.00003
	YOLOv9s	54.81 ± 0.41	18.25 ± 0.14	0.0119 ± 0.00009
	YOLOv10s	75.18 ± 0.39	13.30 ± 0.07	0.0084 ± 0.00004
	YOLO11s	65.82 ± 0.20	15.19 ± 0.05	0.0099 ± 0.00003
	YOLO12s	48.15 ± 0.12	20.77 ± 0.05	0.0136 ± 0.00003
Mix Colors	YOLOv8s	65.69 ± 0.24	15.22 ± 0.06	0.0098 ± 0.00004
	YOLOv9s	55.02 ± 0.45	18.18 ± 0.15	0.0120 ± 0.00010
	YOLOv10s	77.52 ± 0.46	12.90 ± 0.08	0.0084 ± 0.00005
	YOLO11s	65.67 ± 0.22	15.23 ± 0.05	0.0100 ± 0.00003
	YOLO12s	48.21 ± 0.21	20.74 ± 0.09	0.0138 ± 0.00006

Table 10 presents the improvement in model performance on the Orin using the TensorRT framework compared to the PyTorch framework. TensorRT optimization led to consistent performance gains across all evaluated YOLO models. The most pronounced acceleration was observed for the YOLOv9s architecture, which achieved an exceptional increase in frame rate and the largest proportional reduction in *Latency*. The subsequent group of models—YOLOv10s, YOLO11s, and YOLO12s also exhibited substantial improvements, though to a slightly lesser extent. Similar hierarchical patterns were observed across all evaluated metrics. The reduction in *Latency* and the decrease in the *Accuracy–speed ratio* followed the same progression as *FPS* improvements. The results remained uniform across all fruit color datasets, indicating that performance gains were driven by framework-level optimization rather than dataset characteristics.

Table 10. Comparison of models optimized with TensorRT and the PyTorch framework on the Orin.

Fruits Color	Model	+Δ*FPS*		−Δ*Latency* (*ms*)		−Δ*Accuracy Speed Ratio*	
Red	YOLOv8s	11.86	22.07%	3.37	18.11%	0.0023	18.40%
	YOLOv9s	31.04	129.06%	23.44	56.36%	0.0160	56.54%
	YOLOv10s	26.51	54.51%	7.26	35.29%	0.0049	35.25%
	YOLO11s	18.84	40.11%	6.09	28.60%	0.0041	28.47%
	YOLO12s	15.86	48.68%	10.05	32.74%	0.0068	32.69%

Table 10. *Cont.*

Fruits Color	Model	+Δ*FPS*		−Δ*Latency* (*ms*)		−Δ*Accuracy Speed Ratio*	
Yellow	YOLOv8s	12.19	22.76%	3.46	18.53%	0.0024	18.60%
	YOLOv9s	30.90	129.61%	23.68	56.45%	0.0166	56.46%
	YOLOv10s	27.08	56.15%	7.45	35.94%	0.0051	36.17%
	YOLO11s	19.39	41.59%	6.30	29.37%	0.0043	29.05%
	YOLO12s	16.01	49.51%	10.25	33.14%	0.0072	33.33%
Purple	YOLOv8s	11.98	22.29%	3.39	18.22%	0.0022	18.33%
	YOLOv9s	31.01	130.29%	23.76	56.56%	0.0154	56.41%
	YOLOv10s	26.88	55.65%	7.41	35.78%	0.0047	35.88%
	YOLO11s	18.99	40.55%	6.17	28.89%	0.0041	29.29%
	YOLO12s	15.95	49.53%	10.28	33.11%	0.0067	33.00%
MixColors	YOLOv8s	12.43	23.34%	3.55	18.91%	0.0023	19.01%
	YOLOv9s	31.37	132.64%	24.11	57.01%	0.0159	56.99%
	YOLOv10s	29.36	60.96%	7.87	37.89%	0.0051	37.78%
	YOLO11s	19.12	41.07%	6.25	29.10%	0.0041	29.08%
	YOLO12s	16.01	49.72%	10.32	33.23%	0.0068	33.01%

4. Discussion

Fruit detection represents one of the most challenging applications of computer vision in agriculture. The appearance of fruits varies substantially across species and cultivars, driven by differences in color, morphology, surface texture, and growth structure. Illumination changes, occlusions caused by leaves and shoots, and background heterogeneity further increase visual complexity. As a result, developing reliable detectors requires large and diverse datasets, as well as architectures capable of capturing fine-grained features under variable outdoor conditions. At the same time, robust fruit detection is a fundamental prerequisite for automated harvesting and yield monitoring, making the evaluation of modern detectors based on convolutional neural networks in this context both technically and practically relevant.

Raspberry fruits represent one of the more demanding cases within this domain. Their small size, soft structure, and color similarity to surrounding foliage combined with frequent occlusion by leaves, shoots, and overlapping fruits, create conditions that are particularly challenging for object detectors. These characteristics make raspberries a suitable benchmark for assessing the maturity and robustness of modern YOLO architectures.

The publicly available datasets for training models designed for raspberry fruit detection are also very limited. One of the available datasets contains images of red raspberries captured under field conditions [28], while another is intended for the automatic quality assessment of red raspberries in industrial applications [29]. These collections cover only a narrow subset of red raspberry varieties and reflect a limited range of cultivation and industrial processing systems. As a consequence, they represent only a fraction of the visual variability encountered in commercial and small-scale raspberry production. Such constraints substantially limit the ability to train models with strong generalization capability, particularly given the wide diversity of red raspberry cultivars grown worldwide. Moreover, to the best of our knowledge, no publicly accessible datasets exist for yellow, purple, or black raspberries. The absence of such resources restricts current model development to a narrow subset of phenotypes and highlights the need for broader, multi-variety datasets that capture the full spectrum of fruit colors and field conditions. Given the significant morphological variability within Rubus species, such limitations constitute a significant barrier to training models with sufficient detection ability across species of this genus.The comparative analysis of YOLO architectures demonstrated that all evaluated models achieved consistently high and stable performance in raspberry fruit detection.

The close alignment of *Precision*, *Recall*, and *mAP* metrics across generations indicates that the feature extraction mechanisms of recent YOLO designs have reached a high level of maturity. The smallest YOLOv8s variant already provided a strong balance between detection accuracy and localization precision, suggesting that compact architectures can achieve competitive performance without extensive structural complexity. Successive generations introduced only marginal variations in predictive behavior. The consistency of *mAP* values under both lenient and strict overlap thresholds further highlights the robustness of all models in localizing raspberry fruits.

When evaluated across fruit color subsets, detection remained uniformly strong. Yellow and red raspberries yielded nearly identical *Precision–Recall* characteristics, indicating reliable generalization across varying color distributions. Detection of purple fruits was slightly more challenging, showing minor decreases in *Precision* and *Recall*, likely due to reduced visual separability from background elements. Nevertheless, all models maintained high localization accuracy, with only small fluctuations in fine-grained detection performance.

The multi-class experiments confirmed the stability of YOLO detectors under shared training across fruit types. Performance differences between classes and model generations were minimal, suggesting that the current YOLO architectures effectively balance feature representation without introducing class-specific bias. The newest models, particularly YOLO12s and YOLO11s, exhibited the most uniform and balanced performance profiles, while earlier generations sustained competitive accuracy with only negligible degradation. Overall, the results indicate that the YOLO family has reached a performance plateau in this detection domain, where future improvements are likely to be incremental and driven by efficiency-oriented refinements rather than major architectural redesigns.

Species of the genus *Rubus*, including raspberries, remain poorly studied in terms of the performance of CNN-based algorithms, such as YOLO, in the context of fruit detection.

A few publications have confirmed the applicability of YOLOv5 and its improved version, HSA-YOLOv5, for detecting unripe, nearly ripe, and fully ripe raspberry fruits, demonstrating their effectiveness in this task [30]. Given the limited harvesting periods for individual fruit species during each growing season, the ability to test models under simulated conditions is also important. Under such conditions, high detection accuracy of artificial fruits using YOLO algorithms has likewise been achieved [31].

More recently, an enhanced YOLOv11n architecture has also been shown to accurately detect both unripe and ripe raspberry fruits, achieving improved overall performance compared with its baseline variant [32]. Although fruit-ripeness classification is important in automated harvesting systems, its operational relevance depends on the harvesting strategy. In many practical scenarios, only fully mature fruits are candidates for picking; therefore, detecting unripe fruits may introduce additional computational load without providing functional benefit. For this reason, the present study focuses on the detection of harvest-ready fruits, while ripeness-stage classification remains a complementary direction for future work.

In the case of blackberries, the feasibility of detecting and evaluating three maturity levels of the fruits was confirmed by testing nine YOLO models, which demonstrated their effectiveness as real-time detection tools [33]. High detection accuracy for blackberries at different ripeness stages was also achieved using YOLOv7 [34].

The collective findings from previous studies on raspberries and blackberries illustrate that computer-vision research within the genus *Rubus* is still in its early stages. Existing models demonstrate promising performance, yet their development is constrained by limited datasets and the substantial phenotypic diversity that characterizes this genus.

These factors highlight the need for broader, systematically collected image resources and further methodological work to advance robust fruit-detection systems for *Rubus* species.

The literature reports high effectiveness of YOLO-based methods in the detection of numerous fruit species belonging to genera other than *Rubus*, confirming the broad adaptability of these approaches. promising results were obtained for blueberries [35–37], citrus fruits [38], macauba [39], tomatoes [40,41], dragon fruit [42,43], pomegranates [44,45], apples [46,47], strawberries [48–51], grapes [52–54], mangoes [55–57], kiwifruit [58,59], cherries [60,61], plums [62], avocados [63], bananas [64] and pears [65]. In the case of Japanese apricots, the application of R-CNN, EfficientDet, RetinaNet, and SSD MobileNet models—based on convolutional neural networks—demonstrated their effectiveness as reliable methods for detecting fruits of this species [66].

The successful application of CNN-based algorithms to fruit species exhibiting substantial variability in morphology, color, and surface characteristics indicates their strong generalization capability and confirms their potential as a universal framework for agricultural vision systems.

The evaluation of computational performance across hardware platforms and frameworks revealed consistent trends in inference efficiency among YOLO architectures. The results demonstrated a clear hierarchy of runtime performance that correlated with architectural complexity. Compact models maintained superior frame rates and minimal *Latency*, while deeper networks introduced measurable computational overhead.

On high-performance GPU hardware, the smallest YOLOv8s variant achieved the highest throughput and lowest *Latency*, confirming its efficiency in environments without strict computational constraints. The YOLOv9s architecture consistently represented the least efficient configuration, characterized by reduced frame rates and increased *Latency* that resulted in the highest accuracy–speed ratios. Intermediate models such as YOLOv10s and YOLO11s sustained balanced efficiency, maintaining strong throughput with moderate *Latency* growth. The latest YOLO12s model produced a noticeable slowdown, reflecting the processing cost associated with its extended feature extraction design.

The comparative analysis of YOLO architectures confirmed convergent trends with those reported in the literature. Recent YOLO generations revealed consistent efficiency patterns. YOLOv9 achieved high accuracy but struggled with small objects and computational overhead. YOLOv10 traded minor accuracy for notably faster inference and better handling of overlapping targets. The YOLO11s family maintained the best accuracy–efficiency balance, while YOLO12 introduced additional complexity without meaningful performance gain, confirming predictable scaling of computational cost across architectures [67].

On the Orin platform, all models exhibited substantially reduced inference speed and increased *Latency* compared to desktop GPU execution, which is expected given the hardware constraints of embedded systems, but still based on our results, this type of device can also be deployed for real-time tasks. The overall performance decreased, but the relative hierarchy between models remained unchanged. This stability indicates that computational efficiency scales predictably across hardware classes.

TensorRT deployment further improved inference efficiency across all evaluated models; however, the magnitude of acceleration differed markedly between architectures. The largest proportional improvement was achieved by YOLOv9s, which exhibited the highest increase in frame rate and the greatest reduction in latency. This behavior reflects the architectural properties of YOLOv9, whose GELAN-based feature aggregation and deeper convolutional pathways form longer operator chains that TensorRT can fuse into highly optimized computational kernels.

Subsequent model generations—YOLOv10s, YOLO11s, and YOLO12s—also demonstrated substantial acceleration gains, though slightly smaller than those observed for

YOLOv9s. Their computational graphs still offer considerable opportunities for kernel fusion and memory-reuse optimization, yet their more streamlined operator layouts leave marginally less headroom for further acceleration.

In contrast, the YOLOv8s architecture showed the smallest relative benefit from TensorRT optimization. As one of the lightest models in the series, YOLOv8s attains high baseline efficiency in PyTorch, with limited potential for additional fusion or arithmetic-utilization improvements. A larger share of its total inference time is associated with lightweight layers and post-processing operations, such as non-maximum suppression, which TensorRT accelerates only modestly.

These findings align with the publicly documented architectural distinctions among YOLOv8s–YOLO12s in the Ultralytics model specifications, which report variations in block depth, feature-aggregation strategies, operator-path length, and overall computational complexity. The consistent decrease in *Latency* and *Accuracy–speed ratio* followed the same hierarchical order as the improvements in *FPS*, confirming that TensorRT optimization enhanced inference speed without disrupting the relative performance structure among architectures.

A recent benchmarking study of five deep-learning inference frameworks on the NVIDIA Jetson AGX Orin also provides a relevant reference point for interpreting the performance differences observed in this work. Evaluating pretrained models spanning both convolutional and transformer architectures including ResNet-152, MobileNetV2, SqueezeNet, EfficientNet-B0, VGG16, Swin Transformer, and YOLOv5s. The authors reported that TensorRT consistently achieved the lowest inference latency and highest throughput. These gains were enabled by its statically compiled execution engine, FP16 computation, and extensive operator fusion, albeit at the cost of increased power consumption. PyTorch, in contrast, offered a balanced trade-off between efficiency and usability, maintaining low latency and strong throughput through its cuDNN-accelerated dynamic computation graph [68]. The behavior observed in our experiments aligns with these findings. TensorRT provided substantial acceleration on the Orin especially for architectures with deeper operator chains while PyTorch remained competitive as a general-purpose and portable deployment optionPrevious studies have shown that inference is consistently faster with TensorRT than with the native PyTorch runtime, and this difference becomes more pronounced for complex models or constrained hardware. Embedded NVIDIA Jetson devices enable real-time detection but must be matched to the computational demands of the specific YOLO variant [69].

Overall, the findings demonstrate that YOLO architectures maintain predictable computational behavior across hardware environments. Framework-level optimization significantly enhances runtime efficiency, while architectural scaling continues to impose proportional computational costs. These results emphasize that efficiency improvements are increasingly governed by optimization techniques rather than fundamental model redesigns.

Beyond architectural comparisons, the practical value of deploying YOLO models in their out-of-the-box form also merits attention. Although recent research frequently pursues domain-specific modifications such as attention mechanisms, enhanced feature fusion, or redesigned neck and head structures standard YOLO implementations provide several advantages that are highly relevant for agricultural and robotic systems. These include rapid deployment without the need for architectural tuning, stable and predictable inference behavior across hardware platforms, broad compatibility with existing software ecosystems, and reduced engineering overhead during system integration. Our results show that even unmodified YOLO architectures deliver high detection accuracy and strong generalization across fruit colors, indicating that off-the-shelf models already meet the

operational requirements of many field-level applications. For end users and technology implementers, this reliability and ease of deployment make baseline YOLO models an effective and practical foundation for real-world agricultural vision pipelines.

The results also highlight practical implications for deployment. The RTX offered the highest inference speed, but its size, power requirements, and need for work environmental protection limit its suitability for field use. The Orin, although slower, provides a compact and energy-efficient platform designed for edge applications and can be integrated into ruggedized housings. These characteristics make Jetson-class devices a more realistic choice for in-field agricultural systems, where mobility, environmental exposure, and limited power availability are key constraints.

This study also has several limitations. Our analysis focused solely on 2D object detection; emerging directions such as 3D localization and depth-aware fruit modeling were not examined. The S25 smartphone and handheld gimbal were used exclusively for data acquisition rather than as a deployment platform, serving only to provide consistent field imagery. Accordingly, the conclusions of this work refer to the performance of the evaluated neural-network models and hardware accelerators not to the suitability of the acquisition device for real-world operation. We also did not address ripeness classification, which remains an active research topic. Finally, all experiments were conducted using a single RGB camera. Other sensing configurations, such as stereo or multispectral cameras, may further improve robustness in occluded or visually complex field environments. The balance between high detection accuracy and processing speed is particularly important in robotic implementations, where computational resources are often limited. In parallel with the progress of machine vision algorithms for fruit detection, robotic harvesting systems leveraging these advances are rapidly evolving. Recent studies have also explored their application to raspberry harvesting tasks.

At present, robotic platforms for automated fruit harvesting are under active development. Their success will also depend on robust and efficient machine vision systems, capable of detecting and localizing fruits under complex field conditions, including raspberries.

Automated agriculture, particularly autonomous harvesting using robotic systems, remains an open and evolving research domain that presents numerous challenges and opportunities for the development of novel solutions [9].

In the case of raspberries, considerable attention has been devoted to the design of end-effectors intended for their harvesting, as the delicate structure of the fruit poses a major challenge in the development of robots designed for this purpose [70,71]. This confirms the need to develop solutions that enable highly effective and accurate detection of raspberry fruits, which will help minimize damage during harvesting and reduce the number of missed fruits. Ultimately, this can also prevent delayed harvesting that might otherwise render the fruits unsuitable for consumption.

5. Conclusions

The conducted experiments demonstrated that modern YOLO architectures deliver highly reliable and consistent performance in red, yellow and purple raspberry fruits across varied visual conditions.

Across all evaluated configurations, compact YOLO models provided the best trade-off between detection accuracy and computational efficiency, whereas deeper variants delivered only marginal accuracy gains at a substantially higher processing cost. Results from both the high-performance GPU and the Orin platform confirmed that inference speed decreases systematically with increasing model complexity. Moreover, framework-level

optimization particularly with TensorRT significantly improved inference speed compared with the native PyTorch runtime on both platforms.

Due to the limited number of studies addressing the implementation of machine vision solutions based on convolutional neural networks for raspberries, further research on this topic is essential. Equally important is the development of comprehensive datasets that enable the training of models specialized for specific harvesting-related tasks. These models should be fast, accurate, and computationally efficient to meet the highest harvesting standards while remaining suitable for deployment on edge-based robotic systems with limited processing capabilities.

Author Contributions: Conceptualization, K.B. and M.K.; methodology, K.B. and M.K.; software, K.B.; validation, K.B., M.K. and Z.J.; formal analysis, K.B. and M.K.; investigation, K.B., M.K. and Z.J.; resources, K.B.; data curation, K.B.; writing—original draft preparation, K.B., M.K. and Z.J.; writing—review and editing, K.B., M.K. and Z.J.; visualization, K.B.; supervision, K.B.; project administration, K.B.; funding acquisition, K.B., M.K. and Z.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All preprocessing, training, and evaluation scripts used in this study are available in the associated repository: [https://github.com/kamilczynski/Detection-of-Red-Yellow-and-Purple-Raspberry-Fruits-Using-YOLO-Models (accesed on 10 November 2025)]. The original datasets and trained YOLO model weights (best.pt and best.engine) are not publicly available due to their large volume and storage constraints but can be obtained from the corresponding author upon reasonable request.

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

References

1. Food and Agriculture Organization of the United Nations (FAO). Available online: https://www.fao.org/faostat/en/#data/qcl (accessed on 19 October 2025).
2. Simlat, M.; Ptak, A.; Kula, A.; Orzeł, A. Assessment of Genetic Variability among Raspberry Accessions Using Molecular Markers. *Acta Sci. Pol. Hortorum Cultus* **2018**, *17*, 61–72. [CrossRef]
3. Buczyński, K.; Kapłan, M.; Jarosz, Z. Review of the Report on the Nutritional and Health-Promoting Values of Species of the Rubus L. Genus. *Agriculture* **2024**, *14*, 1324. [CrossRef]
4. Buczyński, K.; Kapłan, M.; Borkowska, A.; Kilmek, K. Wpływ Węgla Brunatnego Na Wielkość i Jakość Plonu Maliny Odmiany Polana. *Ann. Hortic.* **2024**, *32*, 5–20. [CrossRef]
5. Alaaudeen, K.M.; Selvarajan, S.; Manoharan, H.; Jhaveri, R.H. Intelligent Robotics Harvesting System Process for Fruits Grasping Prediction. *Sci. Rep.* **2024**, *14*, 2820. [CrossRef]
6. Baranowska, A.; Skowera, B.; Węgrzyn, A. Wpływ Warunków Meteorologicznych i Zabiegów Agrotechnicznych Na Wynik Produkcyjny i Ekonomiczny Uprawy Maliny Jesiennej—Studium Przypadku. *Agron. Sci.* **2025**, *79*, 169–182. [CrossRef]
7. Wróblewska, W.; Pawlak, J.; Paszko, D. The Influence of Factors on the Yields of Two Raspberry Varieties (*Rubus idaeus* L.) and the Economic Results. *Acta Sci. Pol. Hortorum Cultus* **2020**, *19*, 63–70. [CrossRef]
8. Paszko, D.; Krawiec, P.; Pawlak, J.; Wróblewska, W. Assess the Cost and Profitability of Raspberry Production under Cover in the Context of Building Competitive Advantage on Example of Selected Farm. *Ann. Pol. Assoc. Agric. Agribus. Econ.* **2017**, *XIX*, 218–223. [CrossRef]
9. Droukas, L.; Doulgeri, Z.; Tsakiridis, N.L.; Triantafyllou, D.; Kleitsiotis, I.; Mariolis, I.; Giakoumis, D.; Tzovaras, D.; Kateris, D.; Bochtis, D. A Survey of Robotic Harvesting Systems and Enabling Technologies. *J. Intell. Robot. Syst.* **2023**, *107*, 21. [CrossRef]
10. Zhang, J.; Kang, N.; Qu, Q.; Zhou, L.; Zhang, H. Automatic Fruit Picking Technology: A Comprehensive Review of Research Advances. *Artif. Intell. Rev.* **2024**, *57*, 54. [CrossRef]
11. Huang, Y.; Xu, S.; Chen, H.; Li, G.; Dong, H.; Yu, J.; Zhang, X.; Chen, R. A Review of Visual Perception Technology for Intelligent Fruit Harvesting Robots. *Front. Plant Sci.* **2025**, *16*, 1646871. [CrossRef]
12. Rajendran, V.; Debnath, B.; Mghames, S.; Mandil, W.; Parsa, S.; Parsons, S.; Ghalamzan-E., A. Towards Autonomous Selective Harvesting: A Review of Robot Perception, Robot Design, Motion Planning and Control. *J. Field Robot.* **2024**, *41*, 2247–2279. [CrossRef]

13. Tan, Y.; Liu, X.; Zhang, J.; Wang, Y.; Hu, Y. A Review of Research on Fruit and Vegetable Picking Robots Based on Deep Learning. *Sensors* **2025**, *25*, 3677. [CrossRef]
14. Sirisha, U.; Praveen, S.P.; Srinivasu, P.N.; Barsocchi, P.; Bhoi, A.K. Statistical Analysis of Design Aspects of Various YOLO-Based Deep Learning Models for Object Detection. *Int. J. Comput. Intell. Syst.* **2023**, *16*, 126. [CrossRef]
15. Zhang, Y.; Li, N.; Zhang, L.; Lin, J.; Gao, X.; Chen, G. A Review on the Recent Developments in Vision-Based Apple-Harvesting Robots for Recognizing Fruit and Picking Pose. *Comput. Electron. Agric.* **2025**, *231*, 109968. [CrossRef]
16. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent Advances in Convolutional Neural Networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
17. Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 6999–7019. [CrossRef]
18. Zhao, X.; Wang, L.; Zhang, Y.; Han, X.; Deveci, M.; Parmar, M. A Review of Convolutional Neural Networks in Computer Vision. *Artif. Intell. Rev.* **2024**, *57*, 99. [CrossRef]
19. Murat, A.A.; Kiran, M.S. A Comprehensive Review on YOLO Versions for Object Detection. *Eng. Sci. Technol. Int. J.* **2025**, *70*, 102161. [CrossRef]
20. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 779–788.
21. Badgujar, C.M.; Poulose, A.; Gan, H. Agricultural Object Detection with You Only Look Once (YOLO) Algorithm: A Bibliometric and Systematic Literature Review. *Comput. Electron. Agric.* **2024**, *223*, 109090. [CrossRef]
22. Samsung Galaxy S25. Available online: https://www.samsung.com/pl/smartphones/galaxy-s25/specs/ (accessed on 3 November 2025).
23. Hohem. iSteady M7. Available online: https://store.hohem.com/products/hohem-isteady-m7-big-phone-gimbal (accessed on 3 November 2025).
24. Kamilczynski. GitHub Repository 2025. Available online: https://github.com/Kamilczynski/Detection-of-Red-Yellow-and-Purple-Raspberry-Fruits-Using-YOLO-Models (accessed on 9 November 2025).
25. Tzutalin. LabelImg: Image Annotation Tool; Version 1.8.6, GitHub Repository. 2015. Available online: https://github.com/Tzutalin/labelImg (accessed on 1 June 2025).
26. Ultralytics. Models. Available online: https://docs.ultralytics.com/models/ (accessed on 2 November 2025).
27. Seeedstudio. reComputer J4012. Available online: https://www.seeedstudio.com/recomputer-j4012-p-5586.html (accessed on 10 October 2025).
28. Strautiņa, S.; Kalniņa, I.; Kaufmane, E.; Sudars, K.; Namatēvs, I.; Nikulins, A.; Edelmers, E. RaspberrySet: Dataset of Annotated Raspberry Images for Object Detection. *Data* **2023**, *8*, 86. [CrossRef]
29. Quintero Rincón, A.; Mora, M.; Naranjo-Torres, J.; Fredes, C.; Valenzuela, A. Raspberries-LITRP Database: RGB Images Database for the Industrial Applications of Red Raspberries' Automatic Quality Estimation. *Appl. Sci.* **2022**, *12*, 11586. [CrossRef]
30. Ling, C.; Zhang, Q.; Zhang, M.; Gao, C. Research on Adaptive Object Detection via Improved HSA-YOLOv5 for Raspberry Maturity Detection. *IET Image Process.* **2024**, *18*, 4898–4912. [CrossRef]
31. Mei, Z.; Li, Y.; Zhu, R.; Wang, S. Intelligent Fruit Localization and Grasping Method Based on YOLO VX Model and 3D Vision. *Agriculture* **2025**, *15*, 1508. [CrossRef]
32. Luo, R.; Ding, X.; Wang, J. Red Raspberry Maturity Detection Based on Multi-Module Optimized YOLOv11n and Its Application in Field and Greenhouse Environments. *Agriculture* **2025**, *15*, 881. [CrossRef]
33. Thayananthan, T.; Zhang, X.; Harjono, J.; Huang, Y.; Liu, W.; McWhirt, A.L.; Threlfall, R.T.; Chen, Y. In-Field Multi-Ripeness Blackberry Detection for Soft Robotic Harvesting. *J. ASABE*, 2025; *in press*. [CrossRef]
34. Zhang, X.; Thayananthan, T.; Usman, M.; Liu, W.; Chen, Y. Multi-Ripeness Level Blackberry Detection Using YOLOv7 for Soft Robotic Harvesting. In Proceedings of the Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping VIII, Orlando, FL, USA, 1–2 May 2023; Bauer, C., Thomasson, J.A., Eds.; SPIE: Bellingham, WA, USA, 2023; p. 15.
35. Xu, Y.; Li, H.; Zhou, Y.; Zhai, Y.; Yang, Y.; Fu, D. GLL-YOLO: A Lightweight Network for Detecting the Maturity of Blueberry Fruits. *Agriculture* **2025**, *15*, 1877. [CrossRef]
36. MacEachern, C.B.; Esau, T.J.; Schumann, A.W.; Hennessy, P.J.; Zaman, Q.U. Detection of Fruit Maturity Stage and Yield Estimation in Wild Blueberry Using Deep Learning Convolutional Neural Networks. *Smart Agric. Technol.* **2023**, *3*, 100099. [CrossRef]
37. Gai, R.; Liu, Y.; Xu, G. TL-YOLOv8: A Blueberry Fruit Detection Algorithm Based on Improved YOLOv8 and Transfer Learning. *IEEE Access* **2024**, *12*, 86378–86390. [CrossRef]
38. Al Riza, D.F.; Musahada, L.C.; Aufa, R.I.; Hermanto, M.B.; Nugroho, H.; Hendrawan, Y. Comparative Study of Citrus Fruits (Citrus Reticulata Blanco Cv. Batu 55) Detection and Counting with Single and Double Labels Based on Convolutional Neural Network Using YOLOv7. *Smart Agric. Technol.* **2025**, *10*, 100763. [CrossRef]

39. Ribeiro, D.; Tavares, D.; Tiradentes, E.; Santos, F.; Rodriguez, D. Performance Evaluation of YOLOv11 and YOLOv12 Deep Learning Architectures for Automated Detection and Classification of Immature Macauba (Acrocomia Aculeata) Fruits. *Agriculture* **2025**, *15*, 1571. [CrossRef]
40. Rong, J.; Zhou, H.; Zhang, F.; Yuan, T.; Wang, P. Tomato Cluster Detection and Counting Using Improved YOLOv5 Based on RGB-D Fusion. *Comput. Electron. Agric.* **2023**, *207*, 107741. [CrossRef]
41. Li, Y.; Li, J.; Luo, L.; Wang, L.; Zhi, Q. Tomato Ripeness and Stem Recognition Based on Improved YOLOX. *Sci. Rep.* **2025**, *15*, 1924. [CrossRef]
42. Yuan, F.; Wang, J.; Ding, W.; Mei, S.; Fang, C.; Chen, S.; Zhou, H. A Lightweight and Rapid Dragon Fruit Detection Method for Harvesting Robots. *Agriculture* **2025**, *15*, 1120. [CrossRef]
43. Zhang, B.; Wang, R.; Zhang, H.; Yin, C.; Xia, Y.; Fu, M.; Fu, W. Dragon Fruit Detection in Natural Orchard Environment by Integrating Lightweight Network and Attention Mechanism. *Front. Plant Sci.* **2022**, *13*, 1040923. [CrossRef] [PubMed]
44. Zhao, J.; Du, C.; Li, Y.; Mudhsh, M.; Guo, D.; Fan, Y.; Wu, X.; Wang, X.; Almodfer, R. YOLO-Granada: A Lightweight Attentioned Yolo for Pomegranates Fruit Detection. *Sci. Rep.* **2024**, *14*, 16848. [CrossRef]
45. Du, C.; Ma, Z.; Almodfer, R.; Wen, X.; Zhao, J.; Wang, X. A Faster and Lighter Weight Robotic Ready Model YOLO Punica for Detecting Pomegranate Fruit Development. *Sci. Rep.* **2025**, *15*, 39274. [CrossRef]
46. Gao, F.; Fang, W.; Sun, X.; Wu, Z.; Zhao, G.; Li, G.; Li, R.; Fu, L.; Zhang, Q. A Novel Apple Fruit Detection and Counting Methodology Based on Deep Learning and Trunk Tracking in Modern Orchard. *Comput. Electron. Agric.* **2022**, *197*, 107000. [CrossRef]
47. Bortolotti, G.; Piani, M.; Gullino, M.; Mengoli, D.; Franceschini, C.; Grappadelli, L.C.; Manfrini, L. A Computer Vision System for Apple Fruit Sizing by Means of Low-Cost Depth Camera and Neural Network Application. *Precis. Agric.* **2024**, *25*, 2740–2757. [CrossRef]
48. 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. [CrossRef]
49. Ma, H.; Zhao, Q.; Zhang, R.; Hao, C.; Dong, W.; Zhang, X.; Li, F.; Xue, X.; Sun, G. YOLOv11-GSF: An Optimized Deep Learning Model for Strawberry Ripeness Detection in Agriculture. *Front. Plant Sci.* **2025**, *16*, 1584669. [CrossRef] [PubMed]
50. Du, X.; Cheng, H.; Ma, Z.; Lu, W.; Wang, M.; Meng, Z.; Jiang, C.; Hong, F. DSW-YOLO: A Detection Method for Ground-Planted Strawberry Fruits under Different Occlusion Levels. *Comput. Electron. Agric.* **2023**, *214*, 108304. [CrossRef]
51. An, Q.; Wang, K.; Li, Z.; Song, C.; Tang, X.; Song, J. Real-Time Monitoring Method of Strawberry Fruit Growth State Based on YOLO Improved Model. *IEEE Access* **2022**, *10*, 124363–124372. [CrossRef]
52. Wu, X.; Tang, R.; Mu, J.; Niu, Y.; Xu, Z.; Chen, Z. A Lightweight Grape Detection Model in Natural Environments Based on an Enhanced YOLOv8 Framework. *Front. Plant Sci.* **2024**, *15*, 1407839. [CrossRef] [PubMed]
53. Li, P.; Chen, J.; Chen, Q.; Huang, L.; Jiang, Z.; Hua, W.; Li, Y. Detection and Picking Point Localization of Grape Bunches and Stems Based on Oriented Bounding Box. *Comput. Electron. Agric.* **2025**, *233*, 110168. [CrossRef]
54. Guo, C.; Zheng, S.; Cheng, G.; Zhang, Y.; Ding, J. An Improved YOLO v4 Used for Grape Detection in Unstructured Environment. *Front. Plant Sci.* **2023**, *14*, 1209910. [CrossRef]
55. Pichhika, H.C.; Subudhi, P.; Prasad Yerra, R.V. Enhanced YOLO-Based Mango Detection and Centroid Tracking for Precise Yield Estimation. *Appl. Fruit. Sci.* **2025**, *67*, 164. [CrossRef]
56. Diao, S.; Feng, J.; Zhang, B.; Xia, Y.; Gu, Y.; Li, D.; Fu, W. MangoStem-YOLOv8n: An Improved YOLOv8n for Mango and Stem Detection in Natural Orchard Environments. *Smart Agric. Technol.* **2025**, *12*, 101485. [CrossRef]
57. Pichhika, H.C.; Subudhi, P. Detection of Multi-Varieties of On-Tree Mangoes Using MangoYOLO5. In Proceedings of the 2023 11th International Symposium on Electronic Systems Devices and Computing (ESDC), Sri City, India, 4–6 May 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–6.
58. Ren, J.; Wang, W.; Tian, Y.; He, J. YOLOv10-Kiwi: A YOLOv10-Based Lightweight Kiwifruit Detection Model in Trellised Orchards. *Front. Plant Sci.* **2025**, *16*, 1616165. [CrossRef]
59. Li, L.; He, Z.; Li, K.; Ding, X.; Li, H.; Gong, W.; Cui, Y. Object Detection and Spatial Positioning of Kiwifruits in a Wide-Field Complex Environment. *Comput. Electron. Agric.* **2024**, *223*, 109102. [CrossRef]
60. Gai, R.; Chen, N.; Yuan, H. A Detection Algorithm for Cherry Fruits Based on the Improved YOLO-v4 Model. *Neural Comput. Appl.* **2023**, *35*, 13895–13906. [CrossRef]
61. Li, M.; Ding, X.; Wang, J. CMD-YOLO: A Lightweight Model for Cherry Maturity Detection Targeting Small Object. *Smart Agric. Technol.* **2025**, *12*, 101513. [CrossRef]
62. Tang, R.; Lei, Y.; Luo, B.; Zhang, J.; Mu, J. YOLOv7-Plum: Advancing Plum Fruit Detection in Natural Environments with Deep Learning. *Plants* **2023**, *12*, 2883. [CrossRef]
63. Kamat, P.; Gite, S.; Chandekar, H.; Dlima, L.; Pradhan, B. Multi-Class Fruit Ripeness Detection Using YOLO and SSD Object Detection Models. *Discov. Appl. Sci.* **2025**, *7*, 931. [CrossRef]

64. Baglat, P.; Hayat, A.; Mostafa, S.S.; Mendonça, F.; Morgado-Dias, F. Comparative Analysis and Evaluation of YOLO Generations for Banana Bunch Detection. *Smart Agric. Technol.* **2025**, *12*, 101100. [CrossRef]
65. Sun, H.; Ren, R.; Zhang, S.; Yang, S.; Cui, T.; Su, M. Detection of Young Fruit for "Yuluxiang" Pears in Natural Environments Using YOLO-CiHFC. *Signal Image Video Process.* **2025**, *19*, 382. [CrossRef]
66. Kim, E.; Hong, S.-J.; Kim, S.-Y.; Lee, C.-H.; Kim, S.; Kim, H.-J.; Kim, G. CNN-Based Object Detection and Growth Estimation of Plum Fruit (Prunus Mume) Using RGB and Depth Imaging Techniques. *Sci. Rep.* **2022**, *12*, 20796. [CrossRef] [PubMed]
67. Jegham, N.; Koh, C.Y.; Abdelatti, M.; Hendawi, A. YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions. *arXiv* **2024**, arXiv:2411.00201. [CrossRef]
68. Ratul, I.J.; Zhou, Y.; Yang, K. Accelerating Deep Learning Inference: A Comparative Analysis of Modern Acceleration Frameworks. *Electronics* **2025**, *14*, 2977. [CrossRef]
69. Lema, D.G.; Usamentiaga, R.; García, D.F. Quantitative Comparison and Performance Evaluation of Deep Learning-Based Object Detection Models on Edge Computing Devices. *Integration* **2024**, *95*, 102127. [CrossRef]
70. Chauhan, A.; Brouwer, B.; Luo, L.; Nederhoff, L.; El Harchoui, N.; Shoushtari, A.L. Measuring the Response of Soft Fruits to Robotic Handling. *Smart Agric. Technol.* **2025**, *12*, 101445. [CrossRef]
71. Junge, K.; Pires, C.; Hughes, J. Lab2Field Transfer of a Robotic Raspberry Harvester Enabled by a Soft Sensorized Physical Twin. *Commun. Eng.* **2023**, *2*, 40. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.