scientific reports

OPEN

A comprehensive analysis of YOLO architectures for tomato leaf disease identification

Leo Thomas Ramos[1]✉ & Angel D. Sappa[1,2]✉

Tomato leaf disease detection is critical in precision agriculture for safeguarding crop health and optimizing yields. This study compares the latest YOLO architectures, including YOLOv8, YOLOv9, YOLOv10, YOLOv11, and YOLOv12, using the Tomato-Village dataset, which contains 14,368 images across six disease classes. All models are trained under identical settings to ensure a fair evaluation based on precision, recall, mean Average Precision, training time, and inference speed. Results show that YOLOv11 consistently outperforms the other architectures, achieving the highest accuracy with competitive training times and acceptable latency. YOLOv10, YOLOv8, and YOLOv12 also deliver strong results, with YOLOv12n emerging as the most effective lightweight model for resource-constrained environments. In contrast, YOLOv9 demonstrates the weakest performance, requiring more training time and exhibiting higher latency. Overall, YOLOv11 is positioned as the most effective solution for tomato leaf disease detection, providing a strong benchmark for future advancements in agricultural technology.

Keywords Leaf disease detection, Object detection, Plant disease detection, Deep learning, Precision agriculture, YOLO, Smart agriculture, Agriculture monitoring

Tomato (Solanum lycopersicum) is one of the most widely cultivated and consumed fruits globally[1]. Its regular consumption has been linked to reduced risks of various health conditions due to its nutritional benefits[2], including high levels of vitamin C, potassium, and antioxidants such as lycopene[3]. In addition, tomato plays an essential economic role by supporting millions of smallholder farmers and contributing significantly to global agricultural economies[4]. However, its cultivation is vulnerable to a wide range of plant diseases caused by bacteria, fungi, and viruses[5], which can reduce yields, deform fruit quality, and increase production costs[6–8]. These challenges can ultimately disrupt supply chains and affect consumer prices.

In tomato plants, diseases typically manifest through visible symptoms on the leaves, such as discoloration, spots, or deformities[9,10]. Traditionally, farmers have relied on manual inspection to monitor these diseases[7], a method that is time-consuming, labor-intensive, and prone to human error[7,11]. Manual inspection often fails to adequately cover large agricultural areas, leading to delayed detection and reduced treatment effectiveness[8]. To address these limitations, recent advances in Artificial Intelligence (AI) and Computer Vision (CV) have introduced automated approaches for crop monitoring[2,4]. By leveraging machine learning algorithms and image processing, CV systems enable rapid, accurate identification of disease symptoms[12], improving efficiency and enabling timely disease management at scale[5,12].

Within CV, object detection has emerged as a key technique for leaf disease detection. It enables the identification and localization of multiple objects within an image[13], allowing for the rapid processing of large datasets and providing real-time insights for disease management[14,15]. Among object detection frameworks, You Only Look Once (YOLO) stands out for its high speed and accuracy[16,17]. Unlike multi-stage approaches such as Mask R-CNN, YOLO processes the entire image in a single pass[18], making it ideal for real-time agricultural applications. Over successive versions, YOLO has consistently improved in precision, robustness, and efficiency, solidifying its position as one of the leading object detection frameworks.

Based on the previous discussion, this work conducts a comprehensive comparison of the latest YOLO architectures, YOLOv8, YOLOv9, YOLOv10, YOLOv11, and YOLOv12, for tomato leaf disease detection. These architectures represent the most recent official releases and reflect state-of-the-art advancements in object detection. Our analysis focuses on evaluating their effectiveness, efficiency, and practicality in agricultural scenarios, aiming to identify the best-performing model and provide actionable insights for real-world deployment. To the best of our knowledge, this is the first comprehensive study to benchmark these five YOLO

[1]Computer Vision Center, Universitat Autònoma de Barcelona, Barcelona 08193, Spain. [2]ESPOL Polytechnic University, 090112 Guayaquil, Ecuador. ✉email: ltramos@cvc.uab.cat; sappa@ieee.org

versions in the context of tomato leaf disease detection, offering a timely contribution for both agricultural practitioners and AI researchers. The contributions of this work are summarized as follows:

- Providing a technical overview of the latest YOLO architectures: YOLOv8, YOLOv9, YOLOv10, YOLOv11, and YOLOv12 highlighting their advancements and key features in the field of object detection.
- Conducting a comprehensive performance analysis to evaluate the effectiveness, efficiency, and accuracy of these architectures in the context of tomato leaf disease detection.
- Establishing a benchmark for future research to support the development of improved object detection models and architectures tailored for agricultural applications.
- Offering practical insights for agriculture by demonstrating the applicability of advanced computer vision techniques and their potential for integration into AI-driven crop health monitoring systems.

Related work

The task of leaf disease detection has been widely studied in recent years due to its importance in precision agriculture and crop management. YOLO-based architectures have gained significant attention due to their balance between accuracy and inference speed, making them well-suited for real-time agricultural applications. Several studies have explored the use of YOLO models for detecting leaf diseases, demonstrating their effectiveness in identifying different pathological conditions in crops. Given the impact of early disease detection on yield optimization and resource management, research in this area remains highly relevant, driving continuous efforts to refine and evaluate detection models.

To begin, the study in[19] evaluates YOLOv5, YOLOX, Scaled YOLOv4, and SSD using a dataset of 3154 images from corn, potato, and tomato plants, covering eight disease classes. Results show that YOLOv5 outperforms the others in training speed and mAP. Similarly,[20] compares YOLOv5 and YOLOv6 on the PlantDoc dataset, which contains 2569 images across 13 plant species and 17 diseases. YOLOv5 achieves a higher mAP@50, while YOLOv6 surpasses it in mAP@50:95, with YOLOv5 being 2.96 times faster in training.

The study in[21] compares YOLOv5, YOLOv7, and YOLOv8 for citrus leaf disease detection using a dataset of 2684 images across three disease classes: Anthracnose, Melanose, and Brown Spot. YOLOv8 achieves the highest performance, reaching 91.6% mAP@50:95 and significantly outperforming YOLOv5 and YOLOv7. Similarly,[22] evaluates YOLOv7, YOLOv8, and Faster R-CNN on custom datasets for mango (1522 images, three classes) and guava (647 images, four classes). YOLOv8 again surpasses the other models in F1-score and mAP, demonstrating strong adaptability to small datasets.

Expanding YOLO evaluations,[23] analyzes YOLOv5, YOLOv8, and YOLOv9 for leaf disease detection using a dataset of 8,858 images, including healthy and diseased leaves from apples, cucumbers, tomatoes, and grapes. YOLOv9 achieved the best performance, with precision, recall, and F1-scores exceeding 90%. Similarly,[24] uses the PlantDoc dataset to compare YOLOv5, YOLOv6, YOLOv8, and YOLOv9, where YOLOv9 again led in precision, recall, and mAP, while YOLOv5 attained the highest F1-score, demonstrating strong competitiveness against newer models.

Further expanding the evaluations,[25] analyzes YOLOv8, YOLOv9, and YOLOv10 for detecting Fusarium disease in banana leaves using a binary-class dataset of 450 images. YOLOv9 achieved the best mAP@50:95, highlighting its localization capability. Similarly,[26] evaluates YOLOv5, YOLOv8, YOLOv9, and YOLOv10 for melon leaf disease detection using a dataset of 600 images across five disease classes, where YOLOv9 again outperformed all models, achieving the highest mAP, precision, and recall, while YOLOv5 showed the lowest effectiveness.

As seen in the reviewed studies and summarized in Table 1, the comparative evaluation of YOLO architectures for leaf disease detection has attracted considerable attention. While multiple works have demonstrated the effectiveness of YOLO models, they differ widely in scope, datasets, and target crops. A common trend is the predominance of evaluations involving earlier YOLO versions, with limited studies addressing newer architectures such as YOLOv9, YOLOv10, and beyond. Thus, despite progress, a comprehensive assessment of the latest models remains lacking.

Another notable gap in the reviewed studies is the limited focus on tomato crops, despite their global agricultural and economic significance. Many works combine multiple plant species with small sample sizes per species, raising concerns about evaluation comprehensiveness. Additionally, the small overall dataset sizes question the robustness of performance assessments, as limited samples may not reflect real-world variability.

Models	Plant species	Number of diseases	Number of images	References
YOLOv5, YOLOvX, YOLOv4, SSD	Corn, potato, tomato	8	3154	[19]
YOLOv5, YOLOv6	13 species	17	2569	[20]
YOLOv5, YOLOv7, YOLOv8	Citrus	3	2684	[21]
YOLOv7, YOLOv8, Faster R-CNN	Guava, mango	4 (guava), 3 (mango)	1522	[22]
YOLOv5, YOLOv8, YOLOv9	Apples, cucumbers, tomatoes, and grapes	1 (infected and healthy)	8,858	[23]
YOLOv5, YOLOv6, YOLOv8, YOLOv9	13 species	17	2569	[24]
YOLOv8, YOLOv9, YOLOv10	Banana	1 (infected and healthy)	450	[25]
YOLOv5, YOLOv8, YOLOv9, YOLOv10	Melon	5	600	[26]

Table 1. Summary of related work on leaf disease detection using YOLO architectures.

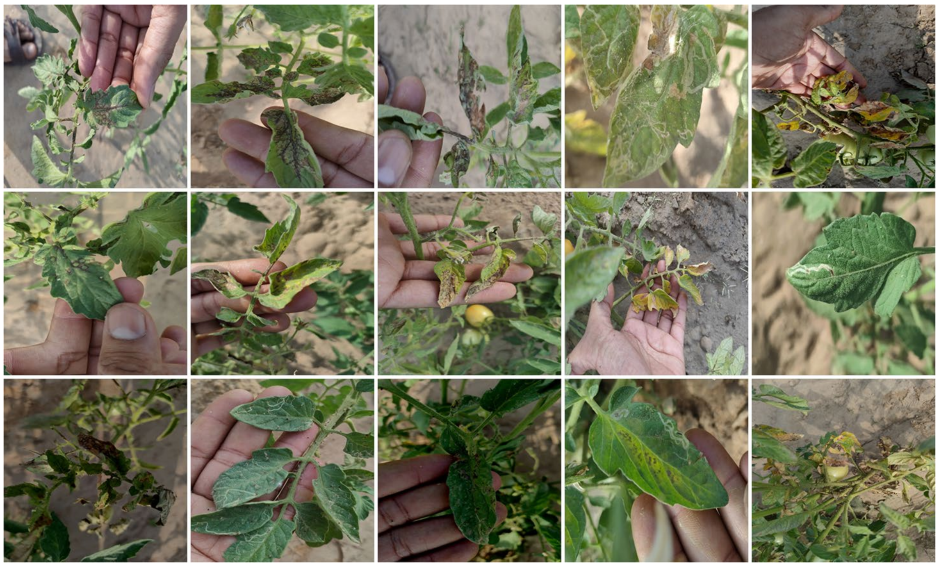


Fig. 1. Sample images from used dataset.

Disease	Description	Annotated instances
Late blight	It is caused by the fungus Phytophthora infestans. It manifests as water-soaked spots on leaves, which rapidly expand and turn dark brown or black[1].	3939
Leaf miner	It refers to the larvae of various insects that burrow into the leaf tissue, creating sinuous tunnels or spotted mines, which can produce leaf necrosis and early defoliation[27].	107,199.
Magnesium deficiency	It typically appears as interveinal chlorosis, where the tissue between the leaf veins turns yellow while the veins remain green[28]. This deficiency can lead to reduced plant vigor and yield if not corrected with appropriate fertilization.	15,870
Nitrogen deficiency	It is characterized by a general yellowing of the leaves, starting with the older leaves at the base of the plant. The affected plants exhibit stunted growth and reduced leaf size, leading to lower yields[29].	1693
Potassium deficiency	It is indicated by yellowing and browning at the leaf edges and tips, along with interveinal chlorosis[30]. This deficiency can cause poor root development, reduced disease resistance, and lower fruit quality and yield.	1126
Spotted wilt virus	It is a viral disease in which infected plants show symptoms such as bronzing and wilting of the leaves, along with small, dark spots[31]. The virus can cause severe stunting and deformation of fruits, leading to significant crop loss.	31,397

Table 2. Tomato leaf diseases and their descriptions contained in the used dataset.

These gaps highlight the need for a systematic comparison of the latest YOLO versions using a dedicated, high-quality tomato leaf disease dataset.

Materials and methods
Dataset description
The dataset used for this work is the Tomato-Village dataset[4]. This dataset comprises a collection of images of tomato leaf diseases for three tasks: multiclass classification, multilabel classification, and object detection. The images were manually captured in three districts of Rajasthan, India. For the object detection variant, the images were taken directly in the fields, as shown in Fig. 1, where a single image can contain multiple leaves with different diseases. Moreover, the images include leaves from plants of varying ages, different timings, and under diverse lighting conditions. Additionally, data augmentation techniques were applied to increase the variability of the dataset; specifically, the techniques used included: RandomRotate90, RandomBrightnessContrast, RGBShift, Rotate up to 90 degrees, RandomCrop with 30% pixel reductions, HorizontalFlip, and VerticalFlip. All images were manually annotated, resulting in a total of 14,368 images and 161,223 annotations.

The dataset, specifically the object detection variant, includes six tomato leaf diseases: late blight, leaf miner, magnesium deficiency, nitrogen deficiency, potassium deficiency, and spotted wilt virus. The 'healthy' leaf class was not included in this division as it is implied that any leaf not detected with a disease is healthy. However, it is important to mention that this class is available for the other dataset variants. Table 2 provides information about the diseases contained in this dataset, and Fig. 2 shows graphical examples of them. The dataset is available for free use on GitHub, and it is prepared in both XML Pascal VOC and YOLO formats. Additionally, the dataset is divided into training and validation subsets. For this work, we performed our own split to include a validation subset. This division was done in an 80-10-10 proportion, resulting in 11,494, 1,437, and 1,437 images for training, validation, and evaluation, respectively. Furthermore, we modified the class names to standardize their capitalization. This process ensures consistency across all annotations, facilitating more accurate and efficient data processing.

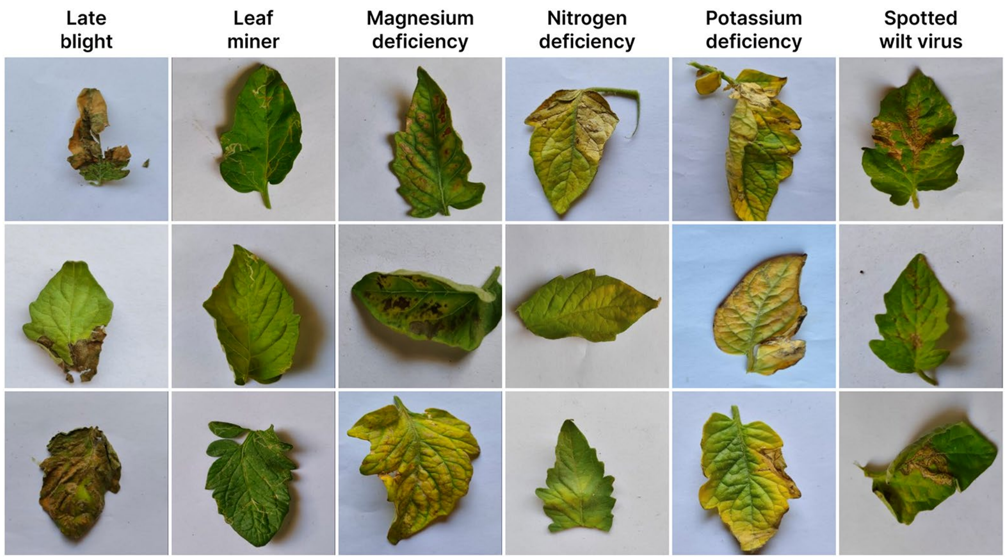


Fig. 2. Sample images of the tomato leaf diseases contained in the used dataset.

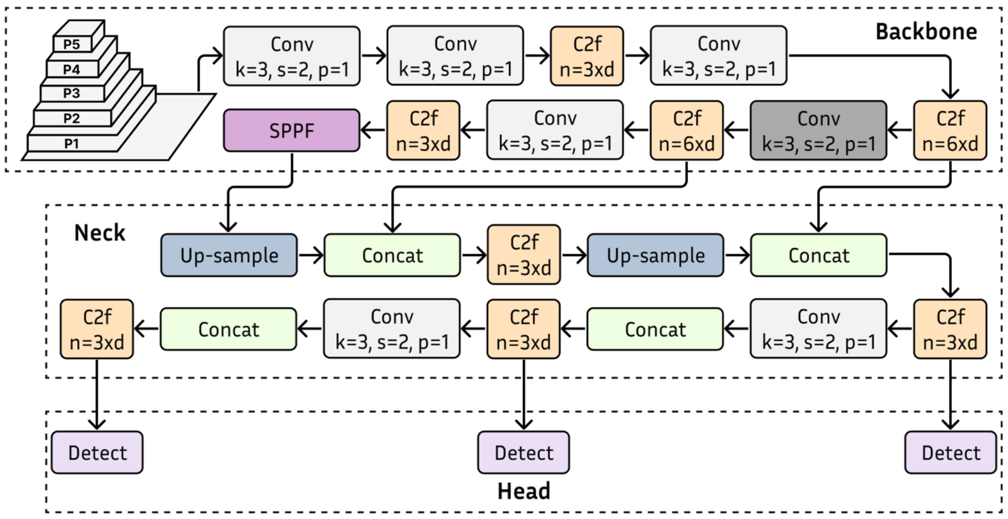


Fig. 3. YOLOv8 architecture[33].

Overview of architectures under study

YOLOv8

YOLOv8 was released by Ultralytics in 2023, the same developers as YOLOv5. This architecture follows the same structural paradigm as its predecessors, consisting of a backbone, neck, and head, as shown in Fig. 3. YOLOv8 shares some characteristics with its predecessor YOLOv5. These include the use of CSPDarknet53 for the backbone, the incorporation of Spatial Pyramid Pooling Fast (SPPF) to handle features at different scales, and the use of Path Aggregation Network (PANet)[32] to improve the flow of information through the network. However, it includes some key modifications that enhance its capabilities.

The first significant modification in YOLOv8 is the incorporation of an anchor-free paradigm, which allows the model to predict the centers of objects more directly. This reduces the number of bounding box predictions, speeding up convergence[34]. Another relevant modification is the replacement of the C3 module from YOLOv5 with a new C2F module[35]. C2F is inspired by the Efficient Layer Aggregation Network (ELAN)[36] module and allows the concatenation of outputs from all bottleneck modules to expand the receptive field; this, in turn, enables the model to learn features more effectively.

Additionally, the training of YOLOv8 incorporates techniques to improve the model's performance and generalization. This includes mosaic image augmentation, which randomly selects four images from the training dataset and combines them into a composite image[37]. This is achieved through cropping, concatenation, and scaling to fit the required input dimensions. This technique increases the dataset's variability automatically, resulting in a trained model with enhanced adaptability. The head of YOLOv8 remains consistent, integrating

Model	Size (pixels)	Params (M)	FLOPs (B)
YOLOv8n	640	3.2	8.7
YOLOv8s	640	11.2	28.6
YOLOv8m	640	25.9	78.9
YOLOv8l	640	43.7	165.2
YOLOv8x	640	68.2	257.8

Table 3. Details of the YOLOv8 variants.

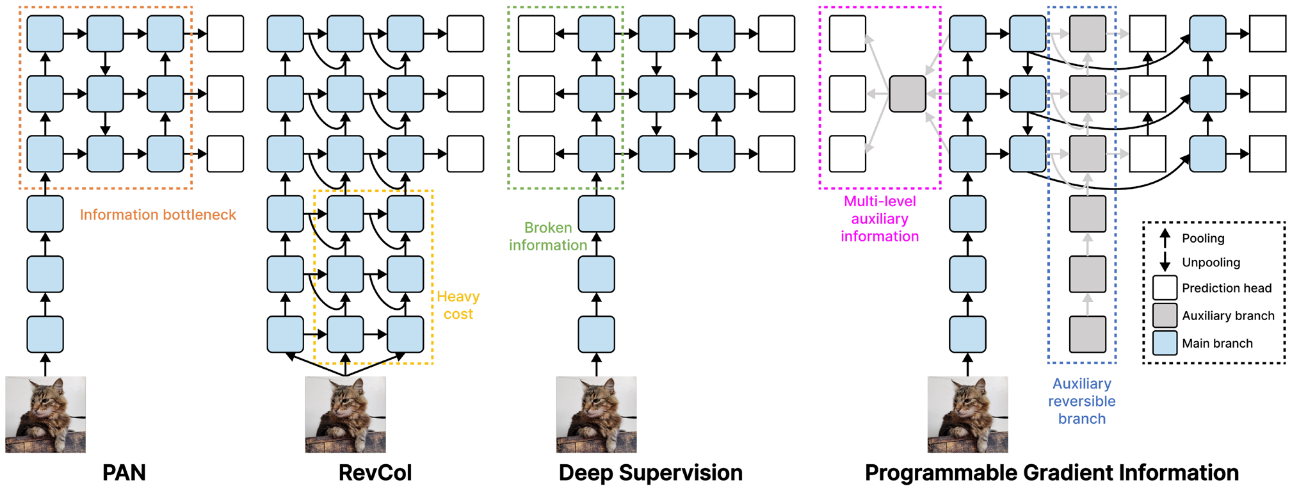


Fig. 4. Comparison of PGI, implemented in YOLOv9, and related methods.

decoupled heads to process detection and classification independently. YOLOv8 comes in five versions (n, s, m, l, and x) to meet different demands and computational resources, with details provided in Table 3.

YOLOv9

YOLOv9[38] is one of the latest versions released in the YOLO family. Although the features of YOLOv9 have not yet been explored in depth, it is known to include three key additions: reversible functions, Programmable Gradient Information (PGI), and a Generalized Efficient Layer Aggregation Network (GELAN). These modifications are designed to mitigate the issues of information loss in deep networks, where data is significantly compressed in bottlenecks, risking the loss of important information that consequently are not transmitted to subsequent layers.

To address the mentioned problem, YOLOv9 first implements reversible functions. These functions have the unique property that their inverse does not result in information loss. By integrating these functions into the architecture, it ensures the retention of the maximum amount of input information and enables its transmission to all layers. This allows the network to update gradients more accurately, thereby improving the model's capability.

To enhance the capabilities of reversible functions, PGI is incorporated to support both deep and lightweight networks. As illustrated in Fig. 4, PGI features a main branch for efficient inference with minimal computational overhead, and an auxiliary branch based on reversible concepts for precise gradient generation and parameter updates. Additionally, a multi-level auxiliary information module enables effective gradient information sharing across layers. These enhancements collectively improve the model's learning, inference, and localization capabilities.

GELAN is incorporated to complement the capabilities of the reversible functions and PGI. This component, shown in Fig. 5, is designed based on ELAN and Cross Stage Partial Network (CSPNet)[39], and it operates by combining the gradient path planning of CSPNet with the speed of ELAN during the inference process. This allows YOLOv9 to achieve fast inference without negatively impacting its accuracy. Moreover, the structure of GELAN allows for stacking multiple blocks, enabling YOLOv9 to handle a variety of scenarios and complexities effectively. Table 4 shows the five variants of YOLOv9.

YOLOv10

A few weeks after YOLOv9, YOLOv10[40] was introduced to address a major limitation in YOLO architectures: the reliance on Non-Maximum Suppression (NMS) for post-processing. NMS, used to remove duplicate detections[41], slows inference speed and hampers true end-to-end performance due to increased latency. Additionally, YOLOv10 includes design improvements and optimizations of components that had been previously overlooked.

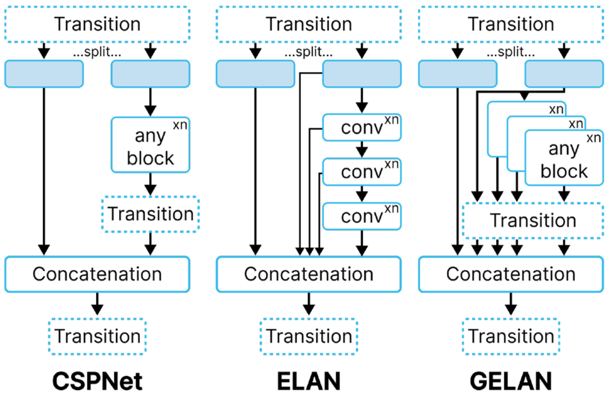


Fig. 5. GELAN module structure, used in YOLOv9.

Model	Size (pixels)	Params (M)	FLOPs (G)
YOLOv9-T	640	2.0	7.7
YOLOv9-S	640	7.1	26.4
YOLOv9-M	640	20.0	76.3
YOLOv9-C	640	25.3	102.1
YOLOv9-E	640	57.3	189.0

Table 4. Details of the YOLOv9 variants.

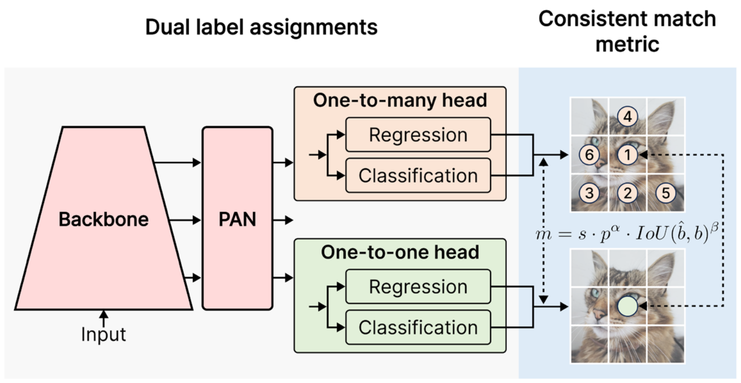
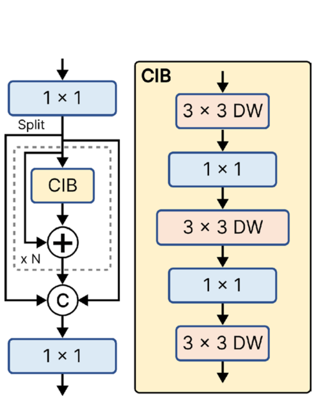


Fig. 6. Consistent dual assignment process for NMS-free training incorporated in YOLOv10.

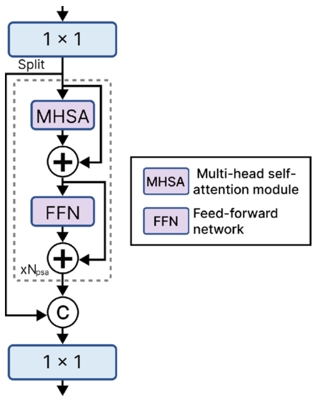
Firstly, YOLOv10 adopts an NMS-free design to reduce latency[42]. During training, YOLO traditionally assigns multiple positives per instance, enhancing performance but requiring NMS for post-processing. In contrast, one-to-one assignment removes the need for NMS but decreases accuracy and convergence speed. To leverage both approaches, YOLOv10 introduces consistent dual assignments by adding a one-to-one head alongside the standard YOLO head, as shown in Fig. 6. Both heads are optimized during training, benefiting from one-to-many learning, but during inference, only the one-to-one head is used for faster predictions. A consistent matching metric ensures alignment between both heads, selecting the best positive sample for each.

Transitioning to component modifications, YOLOv10 introduces both efficiency- and accuracy-driven changes. A lightweight classification head was developed using two depth-wise separable convolutions and a 1×1 convolution to reduce computation. For downsampling, Spatial-channel Decoupled Downsampling separates spatial and channel transformations: point-wise convolution handles channel modulation while depth-wise convolution manages spatial resolution, minimizing cost and preserving information. Additionally, intrinsic rank analysis identified redundancies in deeper stages, leading to a Compact Inverted Block (CIB) design, shown in Fig. 7a, combining depth-wise and point-wise convolutions for greater efficiency without sacrificing performance[43].

In the second set of modifications, YOLOv10 integrates large-kernel depth-wise convolutions to expand the receptive field and enhance model capability. However, to avoid performance degradation in small object detection and increased computational cost, these convolutions are applied selectively to deeper stages of smaller



(a) CIB.



(b) PSA.

Fig. 7. CIB and PSA modules introduced in YOLOv10.

Model	Size (pixels)	Params (M)	FLOPs (G)
YOLOv10-N	640	2.3	6.7
YOLOv10-S	640	7.2	21.6
YOLOv10-M	640	15.4	59.1
YOLOv10-B	640	19.1	92.0
YOLOv10-L	640	24.4	120.3
YOLOv10-X	640	29.5	160.4

Table 5. Details of the YOLOv10 variants.

model scales. Additionally, Partial Self-Attention (PSA), shown in Fig. 7b, is introduced to improve global modeling with low computational overhead. PSA divides feature channels and applies multi-head self-attention only to a subset, enhancing global feature representation without significantly increasing complexity.

All these modifications enable YOLOv10 to achieve performance on par with the best available architectures for object detection, with the added advantage of lower latency. This makes it a highly effective option for end-to-end deployment in applications where speed and accuracy are crucial, such as surveillance, autonomous driving, and real-time analysis. YOLOv10 is available in six versions (n, s, m, b, l, and x), with details provided in Table 5.

YOLOv11

YOLOv11 (YOLO11) is a recent advancement in the YOLO series, developed by Ultralytics, the creators of YOLOv5 and YOLOv8. Positioned as the direct successor to YOLOv8, it builds upon and enhances its foundational architecture. There is no official article describing in depth the characteristics of the architecture. However, some key details and innovations have already been disclosed. The design adheres to the traditional YOLO framework, which is composed of three key components: the backbone, the neck, and the head, as illustrated in Fig. 8.

The backbone is composed of alternating convolutional blocks and C3k2 blocks. Convolutional blocks combine 2D convolutions, batch normalization, and the Sigmoid Linear Unit (SiLU) activation function. The C3k2 blocks, shown in Fig. 9, evolve from the CSP bottleneck, enhancing feature extraction and information flow by splitting feature maps and applying efficient 3×3 convolutions. Inside each C3k2 block, the C3K module, similar to the C2F structure but without splitting, is designed to balance accuracy and speed during feature extraction.

In the neck, YOLOv11 retains the YOLOv8 base structure, including the SPPF, while incorporating C3k2 and C2PSA blocks. The C3k2 block was previously described. C2PSA, specifically designed for the neck, integrates Partial Spatial Attention (PSA) to enhance focus on critical image regions, aiding detection of small or occluded objects. PSA modules apply an attention layer, concatenate input and attention features, and process the result through feed-forward networks and convolutions, followed by a final concatenation, as shown in Fig. 10. Finally, the head remains consistent with those used in previous versions, featuring a multi-scale design to detect objects at three different levels of detail. YOLOv11 is available in five variants, whose characteristics are detailed in Table 6.

YOLOv12

At the time of this work, YOLOv12[44] represents the latest YOLO release, focusing on balancing inference speed and detection accuracy. Unlike previous CNN-centric versions, YOLOv12 introduces an attention-centric design, leveraging the superior modeling capabilities of attention mechanisms traditionally avoided due to

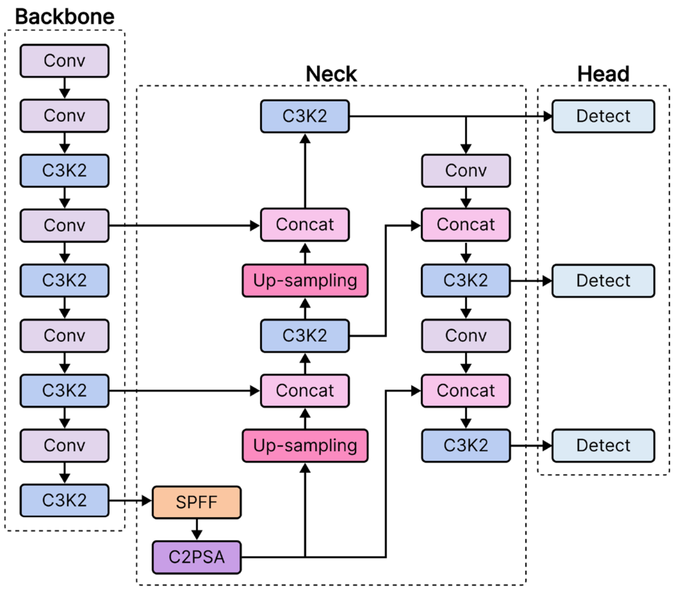


Fig. 8. High-level diagram of the YOLOv11 architecture.

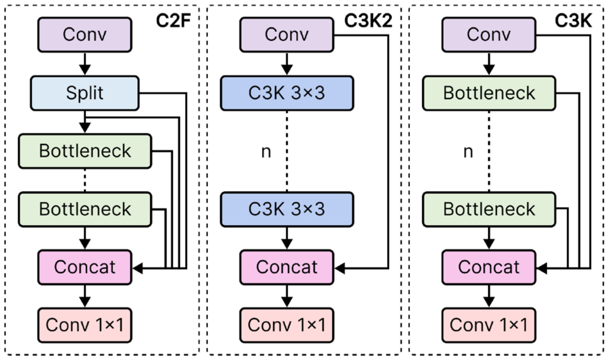


Fig. 9. Structure of a C3k2 block, and comparison with C2F.

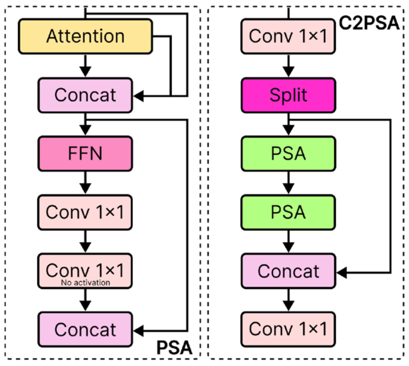


Fig. 10. Structure of C2PSA block used in YOLOv11.

latency concerns. Through architectural innovations, it demonstrates that attention-based models can achieve real-time performance comparable to CNNs. YOLOv12 maintains the classic three-part structure, backbone, neck, and head, with significant modifications to integrate attention without compromising speed.

Regarding the backbone, YOLOv12 maintains a hierarchical architecture for progressive multi-scale feature extraction, with early stages inherited from YOLOv11 and optimized for efficiency. Its main innovation is the

Model	Size (pixels)	Params (M)	FLOPs (B)
YOLOv11n	640	2.6	6.5
YOLOv11s	640	9.4	21.5
YOLOv11m	640	20.1	68.0
YOLOv11l	640	25.3	86.9
YOLOv11x	640	56.9	194.9

Table 6. Details of the YOLOv11 variants.

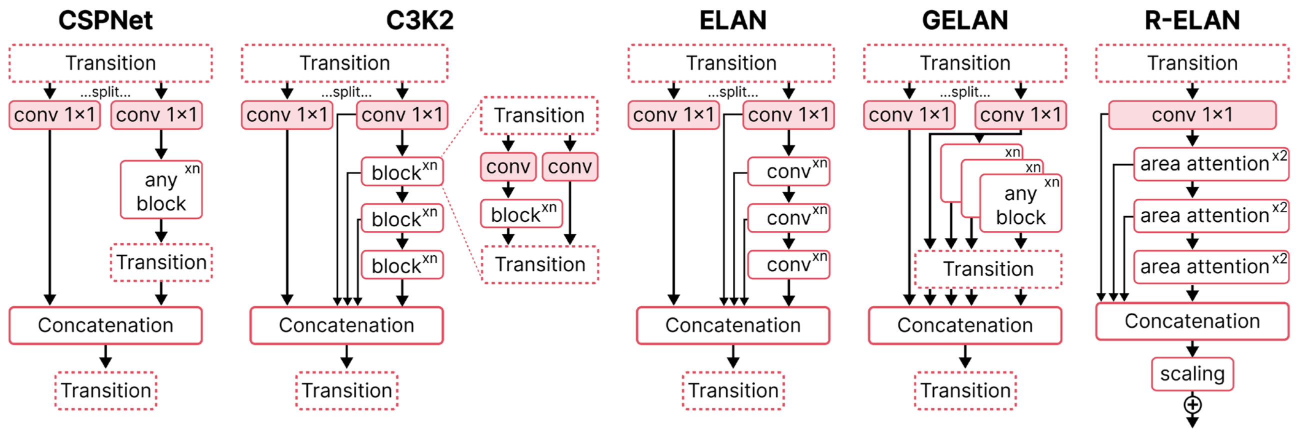


Fig. 11. Comparison of R-ELAN (introduced in YOLOv12) with prior architectural blocks including GELAN, ELAN, C3K2, and CSPNet.

Residual Efficient Layer Aggregation Network (R-ELAN), shown in Fig. 11, which replaces the traditional ELAN. R-ELAN introduces a residual shortcut from the input to the output, combined with a scaling factor, effectively mitigating gradient blockage and improving convergence, particularly in large-scale models. Additionally, the aggregation strategy has been redesigned. Instead of splitting the input and processing multiple paths in parallel as in ELAN, R-ELAN first adjusts the channel dimensions through a transition layer. After this adjustment, the features are processed sequentially and then concatenated, forming a stable and computationally efficient bottleneck structure.

To further optimize the backbone, YOLOv12 integrates 7×7 separable convolutions to maintain a wide spatial receptive field while reducing parameters and memory usage. This design eliminates the need for explicit positional encoding, enabling spatial awareness without the high computational cost of traditional large-kernel operations. Additionally, lightweight convolutional blocks based on multiple small-kernel operations are employed. By decomposing computation and increasing parallelization, the model enhances processing speed while preserving rich feature representation. Finally, unlike YOLOv8 to YOLOv11, YOLOv12 omits the triple-block stacking in the final backbone stages and instead uses a single R-ELAN block, minimizing structural redundancy and improving training stability without sacrificing representational capacity.

As for the neck, YOLOv12 continues the modular feature fusion strategy used in previous YOLO versions but adapts it to better accommodate the integration of attention mechanisms. A central innovation in this component is the use of Area Attention, a lightweight and efficient local attention mechanism tailored specifically for real-time object detection. Unlike conventional attention modules that rely on window partitioning (e.g., Swin Transformer) or complex grid patterns (e.g., axial or criss-cross attention), Area Attention segments the feature map into equal vertical or horizontal areas, as shown in Fig. 12, using a simple reshape operation, eliminating the overhead of explicit partitioning and maintaining a relatively large receptive field.

To further accelerate feature processing, YOLOv12 incorporates FlashAttention in the neck. FlashAttention addresses a key limitation of traditional attention mechanisms: inefficient memory access caused by irregular transfers between high-speed SRAM and high-bandwidth memory, leading to high latency. By restructuring attention computation into efficient memory I/O operations, FlashAttention reduces bandwidth usage and wall-clock time. Combined with Area Attention, it enables localized attention at high speed, allowing YOLOv12 to achieve real-time inference even at higher resolutions. Together, these components enhance feature discrimination in cluttered scenes, improve focus on critical regions, and retain fine-grained spatial information with minimal computational overhead.

Moreover, YOLOv12 modifies the MLP ratio in its attention blocks to improve computational efficiency. Traditionally set at 4:1 in standard vision transformer designs, this ratio determines the relative width of the intermediate feed-forward layer compared to the input dimension. YOLOv12 reduces this to 1.2:1 in smaller model scales (n,s, m) and to 2:1 in larger ones (l, x), effectively rebalancing the computational load between the attention mechanism and the subsequent feed-forward processing. This change not only reduces the overall

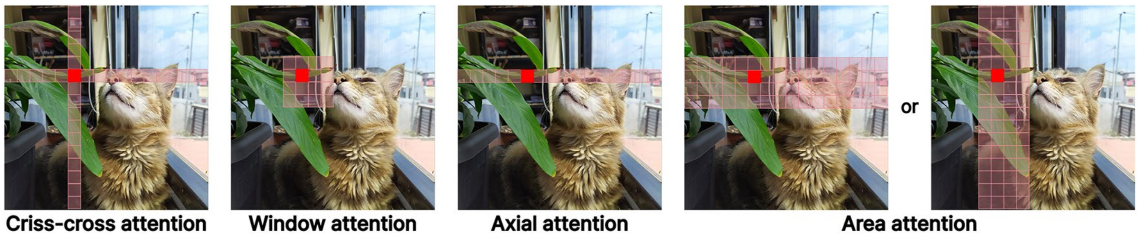


Fig. 12. Comparison between the Area Attention mechanism used in YOLOv12 and other attention mechanisms[33].

Model	Size (pixels)	Params (M)	FLOPs (G)
YOLOv12n	640	2.5	6.0
YOLOv12s	640	9.1	19.4
YOLOv12m	640	19.6	59.8
YOLOv12l	640	26.5	82.4
YOLOv12x	640	59.3	184.6

Table 7. Details of the YOLOv12 variants.

parameter count and memory usage but also accelerates inference while preserving model representational capacity.

Finally, the prediction head in YOLOv12 maintains the foundational design of earlier YOLO models, employing convolutional layers to predict class probabilities, bounding box coordinates, and objectness scores. Some refinements have been introduced to streamline the prediction pathways, enhancing efficiency and supporting consistent multi-scale detection[45]. Additionally, the head integrates Area Attention, improving spatial awareness and contributing to faster, more precise predictions. As with previous versions, YOLOv12 is available in five model variants, detailed in Table 7.

Performance metrics

Precision

Precision serves as an indicator of the reliability of positive identifications made by an object detection model. It measures the proportion of correctly identified positives against the total number of items labeled as positive[46,47]. The formula to calculate precision is given in Eq. 1:

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

where *TP* represents the count of correctly identified positive instances, and *FP* represents the instances erroneously classified as positive. A high precision score indicates that the model is effectively minimizing false positive identifications.

Recall

Recall is a metric that measures the model's ability to identify all relevant instances within a dataset[46]. It quantifies the proportion of actual positives that are correctly detected. Mathematically, recall can be defined as shown in Eq. 2:

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

where *FN* denotes the false negatives. High recall is indicative of a model's effectiveness, highlighting the model's capability to capture as many positives as possible.

Mean average precision

Mean average precision (mAP) is a metric used to evaluate the overall accuracy of object detection models across various threshold settings[46]. It aggregates the precision scores across different recall levels, providing a single figure that summarizes the model's performance. mAP is calculated by averaging the area under the precision-recall curve for each class and then computing the mean of these averages across all classes, as shown in Eq. 3:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i, \tag{3}$$

where AP_i is the average precision for each class, and N is the number of classes. Furthermore, the AP calculation, shown in Eq. 4, involves integrating precision over the range of possible recall levels, which can vary depending on the detection threshold applied[48].

$$\mathrm{AP} = \int_0^1 \mathrm{precision}(r)\mathrm{d}r. \tag{4}$$

The mAP metric can be specified with different Intersection over Union (IoU) thresholds to provide more granular insights into the model's performance. Two common variants are mAP@50 and mAP@50:95[49]. Achieving a high mAP score indicates that a detection model not only accurately identifies and localizes objects across different classes but also consistently upholds this accuracy under varying conditions of detection stringency[50].

Training and implementation details

The aim of this study is to evaluate the models within a common setting to ensure that the results are comparable and relevant. The training was conducted using default hyperparameter configurations. This design choice prioritizes fairness and comparability over individual model optimization, ensuring that all architectures are evaluated under consistent and realistic training conditions. The most relevant parameters include: a learning rate of 0.01, SGD optimizer with momentum 0.937, weight decay of 0.0005, and input image resolution of 640× 640 pixels. The only adjustment applied across all models was setting the batch size to 64 to match available GPU memory. The training process spanned 100 epochs, a standard setting applied uniformly across all architectures to maintain consistency and facilitate direct comparisons.

During the evaluation process, the best-performing model from each training session was selected based on validation performance. The comparison and analysis were carried out in two stages: first, by contrasting the results of architectures within the same version, and second, by conducting a broader comparison across different architectures. This approach allowed us to identify the strengths and weaknesses of each model under similar conditions. Regarding the implementation code, we used the official open-source notebooks provided by the authors of each YOLO version. These notebooks include complete implementations for training, evaluation, and inference, and are publicly available on their respective GitHub repositories. The training environment utilized a high-performance computing setup with the TensorFlow framework. Specifically, the training was performed using four Nvidia A100 SXM4 40GB GPUs. Additionally, the setup included 128 CPU cores and 256 GB of memory.

Results and discussion

YOLOv8

To begin with, the training times of the YOLOv8 variants, shown in Table 8, are analysed. As expected, times increase with model complexity. Notably, YOLOv8n and YOLOv8s show similar durations (0.602 and 0.641 hours, respectively), suggesting their suitability for scenarios with limited computational resources and the need for rapid, cost-effective deployment. A similar trend is observed with YOLOv8m and YOLOv8l, with training times of 1.072 and 1.169 hours, respectively. Although more complex than the nano and small variants, they exhibit comparable efficiency, making them viable options when balancing training time and model complexity for tomato leaf disease detection. Finally, YOLOv8x records the highest training time at 1.664 hours, as expected given its greater architectural complexity. However, the increase remains acceptable considering it has 65 million more parameters than the nano variant. This highlights the efficiency of YOLOv8, making even its most complex models viable for tomato leaf disease detection across varying computational environments.

Moving on to performance evaluation (Table 8), an incremental improvement is observed from the lightest to the most complex models. Notably, the mAP@50:95 increases from 0.538 (nano) to 0.776 (x-large), reflecting enhanced detection and localization accuracy as model complexity grows. A similar pattern is seen in precision, recall, and mAP@50 scores. These improvements suggest that while more complex models require additional training time, they deliver significantly better precision across multiple diseases, justifying the extra computational resources for practical deployment.

An important observation is that the most significant performance gains occur in the first three YOLOv8 variants: nano, small, and medium. Beyond the medium model, improvements with the large and x-large versions are smaller and more incremental. This suggests that, for the task and dataset in this work, heavier variants reach a point of diminishing returns. Consequently, the nano, small, and medium models offer a more efficient trade-off between performance and computational cost, making them practical options for tomato leaf disease detection without requiring the most complex variants.

When analyzing each disease separately, the heavier YOLOv8 variants generally show better performance. For instance, YOLOv8x achieves the highest precision, mAP@50, and mAP@50:95 for late blight, leaf miner, and magnesium deficiency, while YOLOv8l records the best recall for late blight. Notably, YOLOv8x consistently reports the highest mAP@50:95 across all diseases, highlighting its effectiveness in capturing fine details. However, lighter models sometimes outperform heavier ones. YOLOv8m, for example, achieves the highest precision (0.947) and mAP@50 (0.905) for nitrogen deficiency, surpassing the large and x-large variants. Additionally, YOLOv8m also reports the best recall for spotted wilt virus, demonstrating that intermediate models can offer a more balanced and effective performance in certain cases.

Regarding inference speed, the YOLOv8n and YOLOv8s variants prove to be the fastest, with times of 2.2ms and 2.4ms respectively. The YOLOv8m variant has an inference time of 3.9ms, while the large variant adds an extra millisecond, reaching 4.9ms. Finally, the most heaviest variant, YOLOv8x, reports 6.8ms, making it the slowest among all variants. However, this speed is still acceptable, and it does not rule out YOLOv8x as a viable

Model	Class	Precision	Recall	mAP@50	mAP@50:95	Speed (ms)	Training time (hours)
YOLOv8n	Late blight	0.812	0.788	0.829	0.505	2.2	0.602
	Leaf miner	0.856	0.541	0.724	0.446		
	Magnesium deficiency	0.842	0.802	0.867	0.603		
	Nitrogen deficiency	0.793	0.791	0.834	0.544		
	Potassium deficiency	0.795	0.816	0.842	0.604		
	Spotted wilt virus	0.862	0.715	0.824	0.525		
	All	0.827	0.742	0.820	0.538		
YOLOv8s	Late blight	0.874	0.846	0.896	0.623	2.4	0.641
	Leaf miner	0.894	0.720	0.838	0.575		
	Magnesium deficiency	0.912	0.873	0.926	0.719		
	Nitrogen deficiency	0.835	0.806	0.863	0.650		
	Potassium deficiency	0.883	0.851	0.918	0.724		
	Spotted wilt virus	0.910	0.824	0.898	0.660		
	All	0.885	0.820	0.890	0.658		
YOLOv8m	Late blight	0.894	0.867	0.915	0.690	3.9	1.072
	Leaf miner	0.901	0.800	0.881	0.655		
	Magnesium deficiency	0.915	0.879	0.932	0.761		
	Nitrogen deficiency	0.947	0.813	0.905	0.716		
	Potassium deficiency	0.936	0.886	0.933	0.791		
	Spotted wilt virus	0.923	0.862	0.915	0.722		
	All	0.919	0.851	0.914	0.722		
YOLOv8l	Late blight	0.902	0.883	0.927	0.738	4.9	1.169
	Leaf miner	0.923	0.811	0.896	0.695		
	Magnesium deficiency	0.933	0.863	0.927	0.787		
	Nitrogen deficiency	0.923	0.806	0.889	0.725		
	Potassium deficiency	0.937	0.917	0.950	0.823		
	Spotted wilt virus	0.923	0.846	0.914	0.752		
	All	0.923	0.854	0.917	0.753		
YOLOv8x	Late blight	0.926	0.873	0.928	0.758	6.8	1.664
	Leaf miner	0.921	0.830	0.902	0.723		
	Magnesium deficiency	0.937	0.883	0.937	0.807		
	Nitrogen deficiency	0.900	0.821	0.898	0.758		
	Potassium deficiency	0.944	0.883	0.934	0.834		
	Spotted wilt virus	0.928	0.857	0.919	0.777		
	All	0.926	0.858	0.919	0.776		

Table 8. Results of each version of YOLOv8 on evaluation set (1438 images), plus training time (100 epochs).

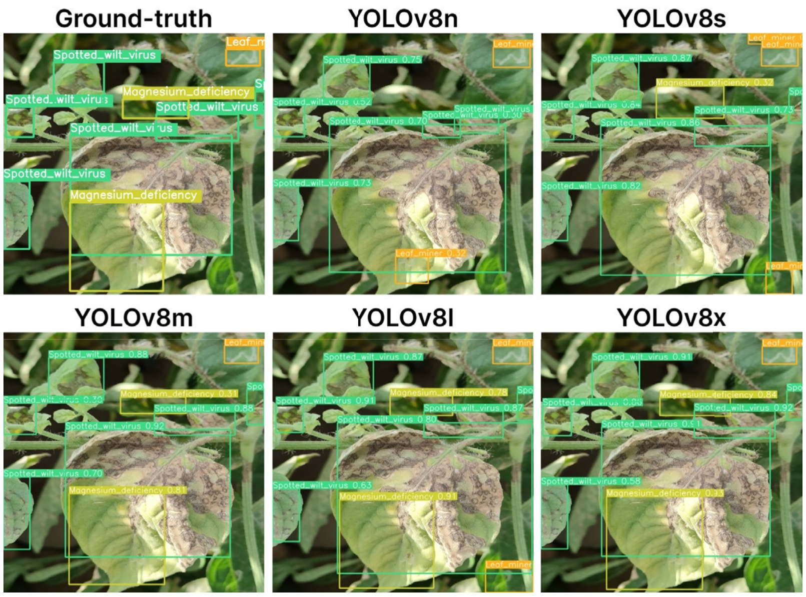
option, especially considering the task of tomato leaf disease detection, where precision and detailed detection capability are crucial.

Further insights are provided by the tests shown in Fig. 13. In the first test (Fig. 13a), lighter models like YOLOv8n and YOLOv8s fail to correctly detect magnesium deficiency and often confuse sunlight reflections with disease symptoms. In contrast, heavier models accurately identify magnesium deficiency with higher confidence scores, with YOLOv8x exceeding 90%. In the second test (Fig. 13b), YOLOv8n again performs poorly, detecting only half of the instances with low confidence. From YOLOv8m onwards, models detect most instances, with YOLOv8l and YOLOv8x achieving confidence scores above 80%. However, the medium and large variants occasionally exhibit duplicate detections, suggesting slight issues with instance overlap.

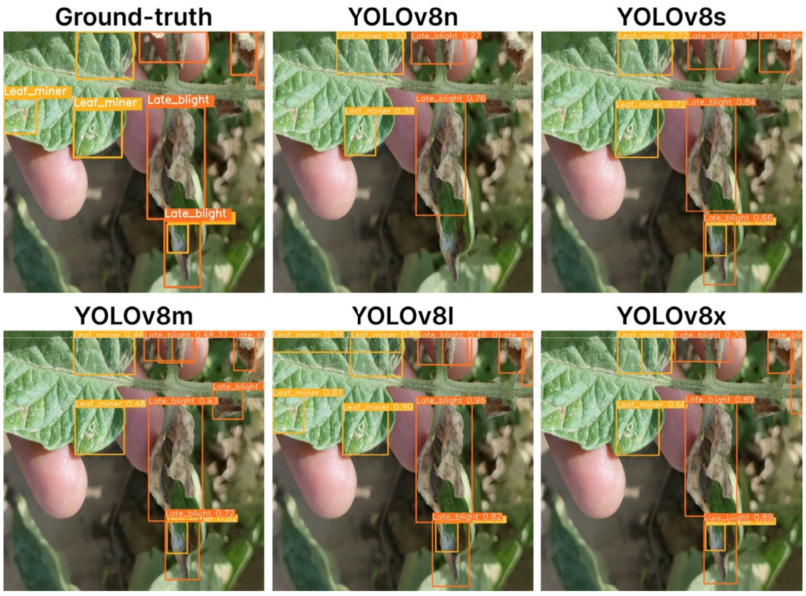
YOLOv9

The training times of YOLOv9 9 reveals that all variants require over two hours. YOLOv9-T, YOLOv9-S, and YOLOv9-M report very similar durations (2.290, 2.452, and 2.497 hours) despite differences in parameter count. YOLOv9-C shows a moderate increase (2.905 hours), while YOLOv9-E, the most complex variant, approaches four hours. Overall, YOLOv9 exhibits considerably long training times, potentially limiting its practicality under resource constraints.

Moving to performance evaluation (Table 9), heavier variants demonstrate clear superiority over lighter ones. YOLOv9-E achieves the best overall results, approaching 90% in precision and mAP@50, and notable scores of 0.795 and 0.590 in recall and mAP@50:95, respectively. YOLOv9-C follows, with precision and mAP@50 above 80%, and competitive recall and mAP@50:95 values. In contrast, YOLOv9-M and YOLOv9-S report performance below 80% in precision, recall, and mAP@50, and under 50% in mAP@50:95. YOLOv9-T performs



(a) First inference test of YOLOv8.



(b) Second inference test of YOLOv8.

Fig. 13. Inference tests of YOLOv8.

the worst across all metrics. These findings highlight that, although lighter variants train faster, their effectiveness in disease detection is significantly inferior.

Analyzing per-disease performance, a pattern similar to the overall results emerges. YOLOv9-E consistently outperforms all other variants, notably achieving 0.916 mAP@50 for magnesium deficiency, being the only case exceeding 90% across diseases and metrics. YOLOv9-C and YOLOv9-M show competitive but variable performances. YOLOv9-C excels in precision for diseases like leaf miner and magnesium deficiency, while YOLOv9-M shows higher recall in late blight and nitrogen deficiency. Both maintain mAP@50:95 scores frequently above 50%. Meanwhile, YOLOv9-S declines, surpassing 80% mAP@50 only for late blight, magnesium deficiency, and potassium deficiency. Finally, YOLOv9-T shows the weakest performance, achieving acceptable precision only for spotted wilt virus (0.745) and reporting very low mAP@50:95, below 30% for diseases like leaf miner and nitrogen deficiency.

Regarding inference speed, clear variations emerge as model complexity increases. The T variant is the fastest at 2.8 ms, while the S variant rises to 3.6 ms. YOLOv9-M shows a notable jump to over 7 ms, suggesting limitations for real-time applications. This trend continues with YOLOv9-C at 8.9 ms and YOLOv9-E reaching 11.5 ms. Although heavier variants offer better precision and detection, their higher inference times raise concerns about practical deployment in tomato leaf disease detection.

Model	Class	Precision	Recall	mAP@50	mAP@50:95	Speed (ms)	Training time (hours)
YOLOv9-T	Late blight	0.667	0.679	0.696	0.337	2.7	2.290
	Leaf miner	0.748	0.449	0.553	0.278		
	Magnesium deficiency	0.688	0.645	0.688	0.388		
	Nitrogen deficiency	0.540	0.418	0.436	0.233		
	Potassium deficiency	0.661	0.526	0.580	0.318		
	Spotted wilt virus	0.745	0.584	0.668	0.326		
	All	0.675	0.550	0.604	0.313		
YOLOv9-S	Late blight	0.742	0.771	0.804	0.440	3.6	2.452
	Leaf miner	0.789	0.540	0.652	0.345		
	Magnesium deficiency	0.768	0.765	0.818	0.502		
	Nitrogen deficiency	0.654	0.657	0.696	0.408		
	Potassium deficiency	0.795	0.715	0.801	0.513		
	Spotted wilt virus	0.789	0.698	0.771	0.421		
	All	0.756	0.691	0.757	0.438		
YOLOv9-M	Late blight	0.736	0.811	0.802	0.467	7.3	2.497
	Leaf miner	0.775	0.606	0.689	0.378		
	Magnesium deficiency	0.770	0.791	0.839	0.546		
	Nitrogen deficiency	0.690	0.716	0.737	0.456		
	Potassium deficiency	0.826	0.748	0.847	0.573		
	Spotted wilt virus	0.801	0.744	0.795	0.455		
	All	0.766	0.736	0.785	0.479		
YOLOv9-C	Late blight	0.790	0.788	0.837	0.506	8.9	2.905
	Leaf miner	0.840	0.637	0.746	0.425		
	Magnesium deficiency	0.845	0.815	0.874	0.582		
	Nitrogen deficiency	0.839	0.672	0.782	0.519		
	Potassium deficiency	0.846	0.754	0.827	0.582		
	Spotted wilt virus	0.851	0.763	0.842	0.515		
	All	0.835	0.738	0.818	0.522		
YOLOv9-E	Late blight	0.846	0.819	0.885	0.568	11.5	3.890
	Leaf miner	0.872	0.695	0.799	0.488		
	Magnesium deficiency	0.891	0.858	0.916	0.633		
	Nitrogen deficiency	0.893	0.739	0.845	0.600		
	Potassium deficiency	0.851	0.851	0.882	0.672		
	Spotted wilt virus	0.893	0.810	0.877	0.581		
	All	0.874	0.795	0.867	0.590		

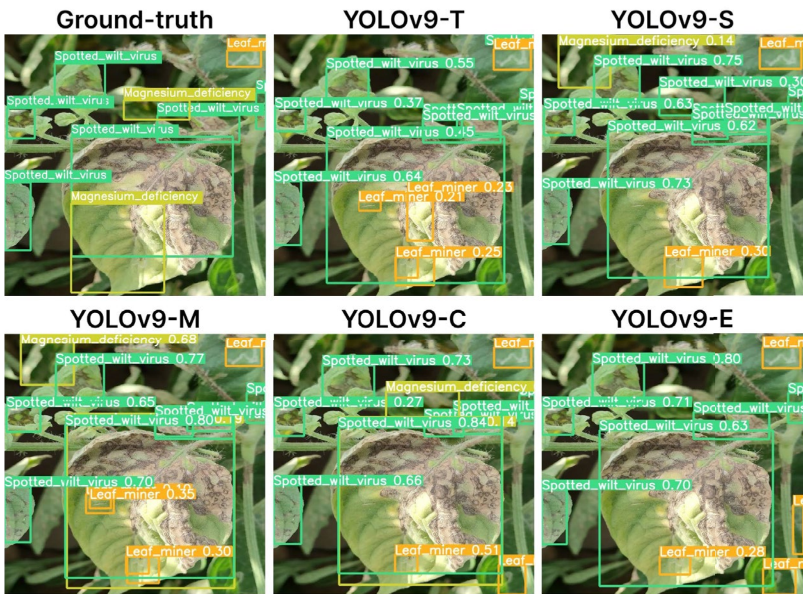
Table 9. Results of each version of YOLOv9 on evaluation set (1,438 images), plus training time (100 epochs).

Finally, Fig. 14 shows the graphical test results. In the first test (Fig. 14a), all models struggle to fully match the ground-truth detections. YOLOv9-S and YOLOv9-M incorrectly detect magnesium deficiency in unaffected areas, and all variants falsely detect leaf miner instances in the leaf center. Heavier models show improved confidence scores, though still below 90%. In the second test (Fig. 14b), overall performance improves, but issues persist, such as YOLOv9-C incorrectly detecting magnesium deficiency and some late blight instances. Again, heavier variants achieve higher confidence scores, occasionally exceeding 80%, but never surpassing 90%. These findings highlight that, despite better precision, the heavier models still face challenges in detection fine-tuning.

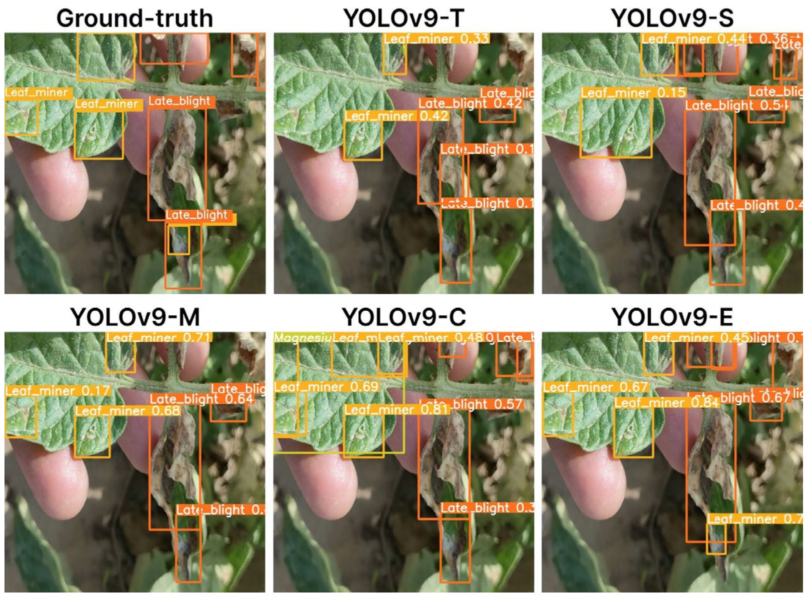
YOLOv10

Beginning with the training times of YOLOv10 models (Table 10), overall, differences between models are not excessive despite variations in parameter count. YOLOv10-N reports the shortest time at 0.830 hours, making it ideal for resource-limited scenarios. YOLOv10-S increases slightly to 0.926 hours, while YOLOv10-M and YOLOv10-B add less than half an hour more. The heavier YOLOv10-L and YOLOv10-X variants show the highest times, 1.643 and 2.008 hours, respectively. Notably, the overall difference between YOLOv10-N and YOLOv10-X is just over an hour, suggesting that even the heaviest models maintain reasonable training times, supporting their practical use for leaf disease detection.

Turning to the performance results (Table 10), overall performance improves with increasing model complexity. YOLOv10-X stands out with the highest metrics across the board. However, lighter models also achieve strong results; YOLOv10-M, for example, records a precision of 0.938, just slightly behind YOLOv10-X's 0.942. Even YOLOv10-N demonstrates notable performance, with precision and mAP@50 scores exceeding



(a) First inference test of YOLOv9.



(b) Second inference test of YOLOv9.

Fig. 14. Inference tests of YOLOv9.

80%. These results indicate that all YOLOv10 variants deliver reliable performance, offering flexibility to adapt to different computational resources and application requirements.

Analyzing the performance for each disease, the overall trend persists, with all YOLOv10 variants showing strong results. Major differences arise when comparing lighter models like YOLOv10-N and YOLOv10-S with heavier ones like YOLOv10-L and YOLOv10-X. YOLOv10-X achieves notable precision for magnesium deficiency (0.951) and spotted wilt virus (0.954), while YOLOv10-L surpasses it with 0.965 precision and 0.971 mAP@50 for potassium deficiency. Mid-complexity models like YOLOv10-B and YOLOv10-M also perform well, often exceeding 90% in key metrics; for example, YOLOv10-M achieves 0.957 precision for potassium deficiency. Although YOLOv10-N and YOLOv10-S show lower overall performance, they still present solid results, with YOLOv10-S often reaching above 90% and YOLOv10-N surpassing 80%. However, YOLOv10-N records the lowest figure (0.587 recall for leaf miner) among all architectures. Overall, the YOLOv10 family proves highly effective for tomato leaf disease detection.

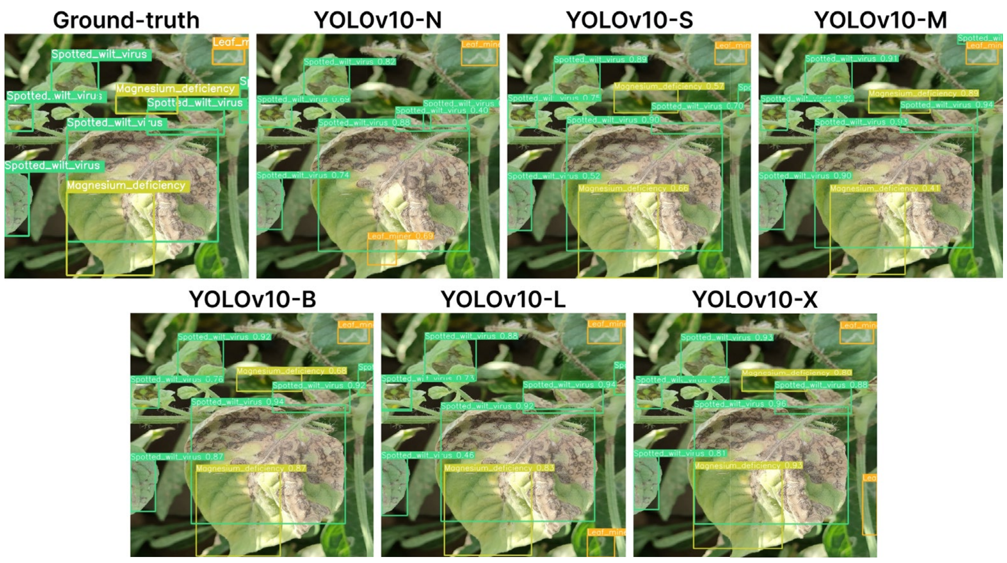
Regarding inference speeds, as model complexity increases, a corresponding rise in latency is observed. YOLOv10-N and YOLOv10-S achieve speeds of 2.1ms and 2.3ms, respectively, while YOLOv10-M reaches 3.4ms. YOLOv10-B and YOLOv10-L follow with 4.2ms and 4.9ms, remaining within an acceptable range. YOLOv10-X records the highest latency at 6.7ms, but this value is still reasonable for practical applications. Overall, despite

Model	Class	Precision	Recall	mAP@50	mAP@50:95	Speed (ms)	Training time (hours)
YOLOv10-N	Late blight	0.825	0.816	0.872	0.515	2.1	0.830
	Leaf miner	0.834	0.587	0.713	0.404		
	Magnesium deficiency	0.844	0.819	0.876	0.582		
	Nitrogen deficiency	0.780	0.746	0.792	0.520		
	Potassium deficiency	0.833	0.830	0.837	0.587		
	Spotted wilt virus	0.850	0.769	0.842	0.513		
	All	0.827	0.761	0.822	0.520		
YOLOv10-S	Late blight	0.866	0.847	0.914	0.623	2.3	0.926
	Leaf miner	0.886	0.755	0.845	0.546		
	Magnesium deficiency	0.917	0.884	0.928	0.682		
	Nitrogen deficiency	0.900	0.807	0.883	0.635		
	Potassium deficiency	0.911	0.904	0.892	0.672		
	Spotted wilt virus	0.907	0.869	0.914	0.637		
	All	0.898	0.845	0.896	0.633		
YOLOv10-M	Late blight	0.926	0.873	0.943	0.670	3.4	1.274
	Leaf miner	0.927	0.781	0.876	0.603		
	Magnesium deficiency	0.943	0.872	0.943	0.732		
	Nitrogen deficiency	0.936	0.784	0.843	0.628		
	Potassium deficiency	0.957	0.851	0.922	0.755		
	Spotted wilt virus	0.936	0.849	0.917	0.678		
	All	0.938	0.835	0.907	0.678		
YOLOv10-B	Late blight	0.933	0.879	0.951	0.693	4.2	1.416
	Leaf miner	0.919	0.812	0.886	0.630		
	Magnesium deficiency	0.941	0.886	0.943	0.744		
	Nitrogen deficiency	0.915	0.784	0.852	0.673		
	Potassium deficiency	0.929	0.895	0.929	0.794		
	Spotted wilt virus	0.935	0.862	0.925	0.704		
	All	0.929	0.853	0.915	0.706		
YOLOv10-L	Late blight	0.907	0.859	0.933	0.697	4.9	1.643
	Leaf miner	0.916	0.822	0.892	0.644		
	Magnesium deficiency	0.940	0.876	0.943	0.762		
	Nitrogen deficiency	0.920	0.768	0.839	0.684		
	Potassium deficiency	0.965	0.912	0.971	0.825		
	Spotted wilt virus	0.934	0.868	0.929	0.724		
	All	0.930	0.851	0.918	0.723		
YOLOv10-X	Late blight	0.925	0.884	0.951	0.715	6.7	2.008
	Leaf miner	0.930	0.838	0.905	0.670		
	Magnesium deficiency	0.951	0.893	0.947	0.780		
	Nitrogen deficiency	0.949	0.784	0.858	0.680		
	Potassium deficiency	0.942	0.904	0.966	0.828		
	Spotted wilt virus	0.954	0.886	0.943	0.744		
	All	0.942	0.865	0.928	0.736		

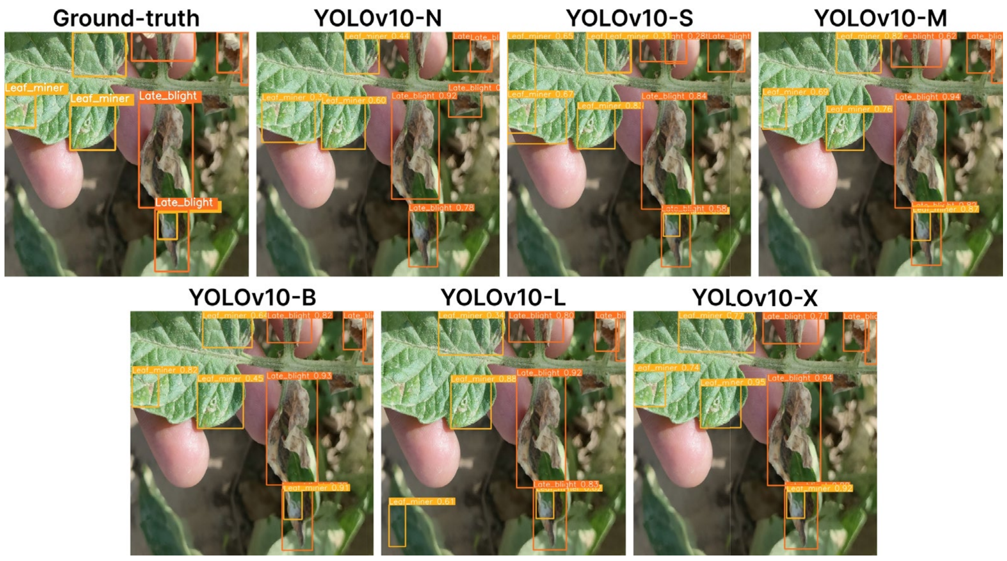
Table 10. Results of each version of YOLOv10 on evaluation set (1,438 images), plus training time (100 epochs).

higher inference times in heavier models, all variants maintain a viable balance between precision and processing speed for tomato leaf disease detection.

Turning to the graphical tests in Fig. 15, the results align with the numerical observations. In the first test (Fig. 15a), all models detect most instances, with YOLOv10-S and YOLOv10-B achieving perfect detection; YOLOv10-B also shows slightly higher confidence scores. YOLOv10-L and YOLOv10-X mistakenly detect an extra leaf miner instance, while YOLOv10-M misses one spotted wilt virus. Confidence scores generally improve with heavier models, nearing or exceeding 90%. In the second test (Fig. 15b), YOLOv10-M, YOLOv10-B, and YOLOv10-X match the ground-truth exactly, with YOLOv10-X achieving the best confidence scores, often surpassing 90%. YOLOv10-N misses one detection, while YOLOv10-S and YOLOv10-L introduce false positives. Overall, all variants perform well, with relatively few errors and high confidence levels.



(a) First inference test of YOLOv10.



(b) Second inference test of YOLOv10.

Fig. 15. Inference tests of YOLOv10.

YOLOv11

Beginning with training times (Table 11), a consistent increase is observed as model complexity grows. YOLOv11n, the lightest variant, reports the shortest time at 0.570 hours, followed by YOLOv11s at 0.679 hours, a modest increase. YOLOv11m requires 0.962 hours, approximately 42% more than YOLOv11s. The heavier variants, YOLOv11l and YOLOv11x, demand 1.227 and 1.635 hours, respectively. Notably, YOLOv11x's training time is nearly three times that of YOLOv11n, reflecting the significant computational cost associated with larger models. It is worth highlighting that even the most robust variant, YOLOv11x, requires just over 1.5 hours to train, which is a remarkably efficient time considering its complexity. Their relatively low training times, combined with their scalability across different variants, make them suitable for diverse use cases, ranging from resource-constrained environments to high-performance systems requiring robust accuracy and speed. Therefore, these models present themselves as strong candidates for practical applications and deployments.

When analyzing the performance results (Table 11), an incremental improvement in performance is observed with increasing model complexity. YOLOv11x emerges as the best-performing variant with a precision of 0.940, recall of 0.884, mAP@50 of 0.936, and mAP@50:95 of 0.790. In contrast, YOLOv11n reports the lowest values, with 0.835 precision, 0.773 recall, 0.840 mAP@50, and 0.565 mAP@50:95. YOLOv11l ranks second, achieving 0.931 precision and 0.932 mAP@50, very close to YOLOv11x. Meanwhile, YOLOv11s and YOLOv11m show intermediate but competitive results, with YOLOv11s notably achieving 0.913 precision and 0.906 mAP@50, making it a strong candidate for resource-limited applications. Overall, the YOLOv11 models offer excellent

Model	Class	Precision	Recall	mAP@50	mAP@50:95	Speed (ms)	Training time (hours)
YOLOv11n	Late blight	0.844	0.824	0.872	0.554	2.1	0.570
	Leaf miner	0.839	0.610	0.754	0.473		
	Magnesium deficiency	0.856	0.839	0.887	0.628		
	Nitrogen deficiency	0.797	0.789	0.839	0.572		
	Potassium deficiency	0.814	0.806	0.834	0.608		
	Spotted wilt virus	0.863	0.768	0.851	0.555		
	All	0.835	0.773	0.840	0.565		
YOLOv11s	Late blight	0.877	0.864	0.909	0.659	3.7	0.679
	Leaf miner	0.904	0.755	0.862	0.604		
	Magnesium deficiency	0.910	0.894	0.941	0.745		
	Nitrogen deficiency	0.931	0.804	0.889	0.688		
	Potassium deficiency	0.922	0.860	0.919	0.737		
	Spotted wilt virus	0.932	0.847	0.916	0.678		
	All	0.913	0.837	0.906	0.685		
YOLOv11m	Late blight	0.927	0.892	0.944	0.739	4.9	0.962
	Leaf miner	0.923	0.820	0.901	0.681		
	Magnesium deficiency	0.945	0.893	0.942	0.788		
	Nitrogen deficiency	0.925	0.826	0.906	0.744		
	Potassium deficiency	0.890	0.886	0.929	0.805		
	Spotted wilt virus	0.933	0.872	0.927	0.744		
	All	0.924	0.865	0.925	0.750		
YOLOv11l	Late blight	0.913	0.887	0.933	0.765	5.8	1.227
	Leaf miner	0.920	0.840	0.909	0.698		
	Magnesium deficiency	0.939	0.897	0.941	0.799		
	Nitrogen deficiency	0.912	0.821	0.903	0.748		
	Potassium deficiency	0.955	0.921	0.965	0.852		
	Spotted wilt virus	0.945	0.890	0.941	0.767		
	All	0.931	0.876	0.932	0.771		
YOLOv11x	Late blight	0.939	0.921	0.952	0.778	7.9	1.635
	Leaf miner	0.938	0.862	0.923	0.731		
	Magnesium deficiency	0.951	0.902	0.947	0.822		
	Nitrogen deficiency	0.955	0.828	0.912	0.771		
	Potassium deficiency	0.911	0.895	0.937	0.841		
	Spotted wilt virus	0.947	0.895	0.942	0.795		
	All	0.940	0.884	0.936	0.790		

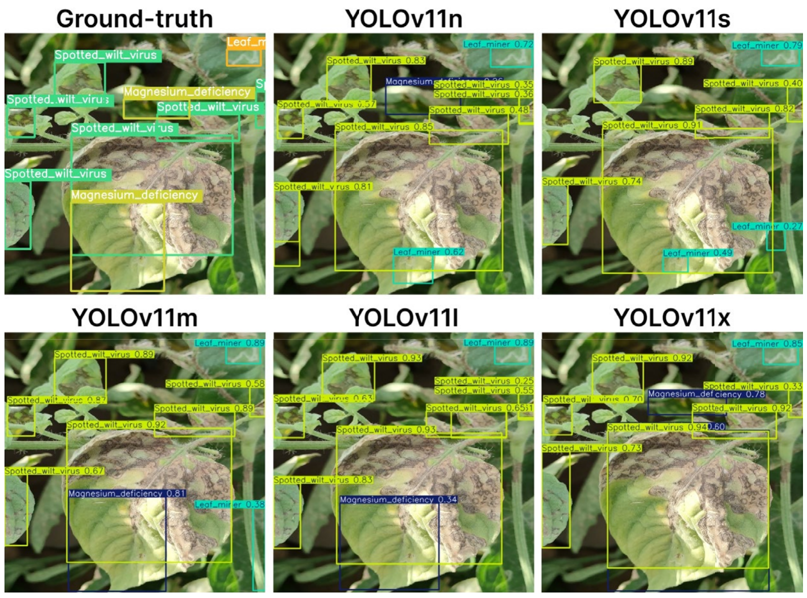
Table 11. Results of each version of YOLOv11 on evaluation set (1,438 images), plus training time (100 epochs).

performance paired with efficient training times, establishing them as robust and versatile solutions for tomato leaf disease detection.

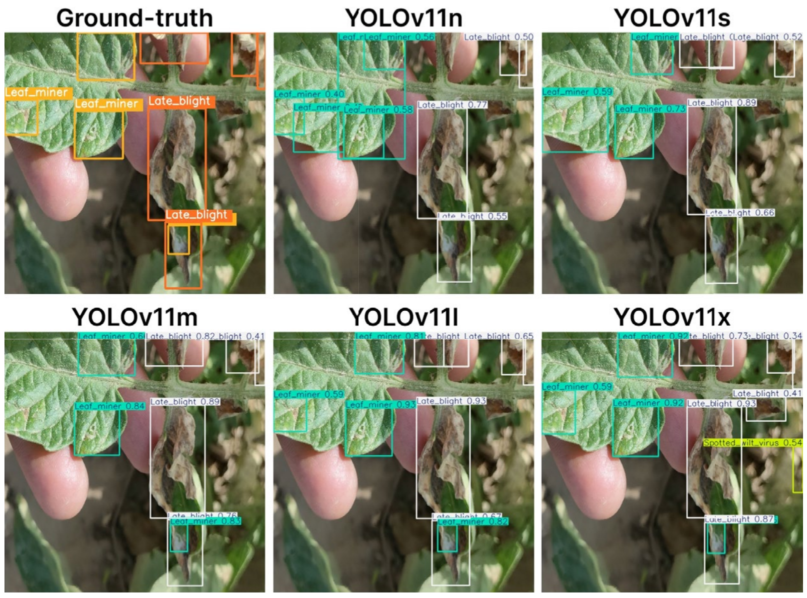
Analyzing the inference times, lighter models perform faster as expected. YOLOv11n is the quickest at 2.1 ms per image, followed by YOLOv11s at 3.7 ms, YOLOv11m at 4.9 ms, YOLOv11l at 5.8 ms, and YOLOv11x at 7.9 ms. Despite being the slowest, YOLOv11x maintains an efficient inference time suitable for most applications.

Regarding per-class performance, the more complex models, YOLOv11l and YOLOv11x, deliver the best results. YOLOv11l achieves the highest scores for potassium deficiency, while YOLOv11x excels across most classes, surpassing 95% precision in magnesium and nitrogen deficiencies and leading in late blight metrics. Lighter models also perform notably. YOLOv11s achieves precision above 90% in five of six classes, and YOLOv11m shows strong results like 0.945 precision in magnesium deficiency and 0.944 mAP@50 in late blight. Even YOLOv11n consistently exceeds 80% across categories. Overall, the YOLOv11 family balances complexity and accuracy effectively, offering robust solutions for tomato leaf disease detection across different resource constraints.

Moving forward, Fig. 15 presents the inference tests using YOLOv11 models. In the first test (Fig. 16a), all models perform satisfactorily, with well-localized bounding boxes and high confidence scores. However, none achieve perfect alignment with the ground truth: YOLOv11n, YOLOv11m, YOLOv11l, and YOLOv11x correctly predict eight out of nine instances, while YOLOv11s identifies only six. Confidence scores are generally high, often exceeding 50% and reaching above 90% in more complex models. The second test (Fig. 16b) is more challenging. YOLOv11n detects seven out of nine instances but shows overlapping and lower confidence; YOLOv11s underperforms with fewer correct detections. YOLOv11m improves with eight correct predictions



(a) First inference test of YOLOv11.



(b) Second inference test of YOLOv11.

Fig. 16. Inference tests of YOLOv11.

and stronger confidence scores. YOLOv11x detects extra instances with misplacements, while YOLOv11l achieves near-perfect detection with high confidence, reinforcing its reliability.

YOLOv12

Starting with the training times (Table 12), they increase progressively with model scale, from 0.640 hours for YOLOv12n to 2.094 hours for YOLOv12x. This growth reflects the rise in parameters and complexity across variants. Notably, the increase is not linear: while YOLOv12m requires around 1.17 hours, YOLOv12x nearly doubles that time, highlighting inefficiencies in scaling. This can be critical in environments with limited computational resources or frequent retraining needs.

Turning to the performance (Table 12), all YOLOv12 variants reflects the benefits of its attention-centric design. Precision rises from 0.860 (YOLOv12n) to 0.947 (YOLOv12x), indicating improved reliability and fewer false positives as capacity increases, largely due to Area Attention. Recall also improves, from 0.790 to 0.873, although the gain is more moderate, suggesting that while larger models detect more true instances, attention mechanisms alone may not fully compensate for limited capacity in smaller variants.

The most significant improvement appears in mAP@50:95, rising from 0.604 (YOLOv12n) to 0.783 (YOLOv12x), a relative gain of nearly 18%. This highlights the effectiveness of additions like R-ELAN and

Model	Class	Precision	Recall	mAP@50	mAP@50:95	Speed (ms)	Training time (hours)
YOLOv12n	Late blight	0.854	0.840	0.900	0.598	1.5	0.640
	Leaf miner	0.846	0.666	0.785	0.506		
	Magnesium deficiency	0.869	0.854	0.903	0.656		
	Nitrogen deficiency	0.847	0.761	0.843	0.612		
	Potassium deficiency	0.871	0.825	0.882	0.656		
	Spotted wilt virus	0.873	0.793	0.871	0.595		
	All	0.860	0.790	0.864	0.604		
YOLOv12s	Late blight	0.876	0.875	0.905	0.648	3.2	0.795
	Leaf miner	0.881	0.752	0.847	0.589		
	Magnesium deficiency	0.910	0.870	0.920	0.721		
	Nitrogen deficiency	0.890	0.799	0.875	0.675		
	Potassium deficiency	0.923	0.838	0.865	0.706		
	Spotted wilt virus	0.905	0.851	0.903	0.670		
	All	0.897	0.831	0.886	0.668		
YOLOv12m	Late blight	0.927	0.867	0.928	0.715	4.9	1.168
	Leaf miner	0.916	0.794	0.884	0.652		
	Magnesium deficiency	0.939	0.883	0.937	0.755		
	Nitrogen deficiency	0.912	0.791	0.878	0.710		
	Potassium deficiency	0.911	0.860	0.894	0.754		
	Spotted wilt virus	0.931	0.863	0.921	0.717		
	All	0.923	0.843	0.907	0.717		
YOLOv12l	Late blight	0.875	0.911	0.937	0.712	7.6	1.741
	Leaf miner	0.905	0.802	0.884	0.656		
	Magnesium deficiency	0.939	0.892	0.938	0.764		
	Nitrogen deficiency	0.900	0.813	0.884	0.704		
	Potassium deficiency	0.952	0.873	0.933	0.788		
	Spotted wilt virus	0.919	0.861	0.915	0.720		
	All	0.915	0.859	0.915	0.724		
YOLOv12x	Late blight	0.925	0.904	0.946	0.763	10.5	2.094
	Leaf miner	0.943	0.836	0.914	0.725		
	Magnesium deficiency	0.962	0.889	0.945	0.820		
	Nitrogen deficiency	0.948	0.810	0.898	0.764		
	Potassium deficiency	0.945	0.910	0.955	0.835		
	Spotted wilt virus	0.959	0.889	0.941	0.791		
	All	0.947	0.873	0.933	0.783		

Table 12. Results of each version of YOLOv12 on evaluation set (1,438 images).

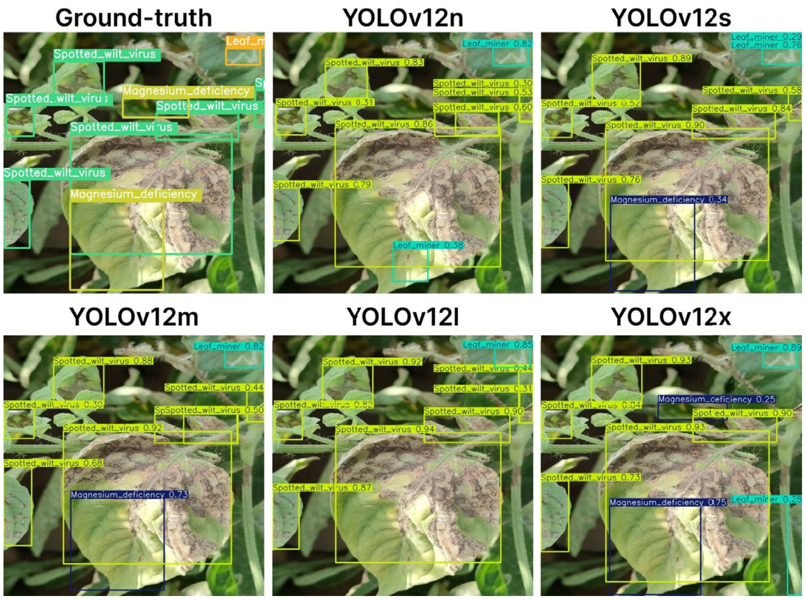
position-aware convolutions in refining localization. Notably, the jump from YOLOv12s to YOLOv12m (+0.049) shows mid-scale models already benefit considerably from attention mechanisms. Meanwhile, mAP@50 improves more gradually, from 0.864 to 0.933, indicating that all models perform well at coarse localization, with larger models primarily enhancing fine-grained accuracy. However, this performance gain comes at the cost of increased inference time, rising from 1.5 ms (YOLOv12n) to 10.5 ms (YOLOv12x). While expected, this trade-off is critical for deployment, particularly in real-time or edge-computing scenarios. Notably, YOLOv12m offers a strong balance, achieving 0.907 mAP@50 and 0.717 mAP@50:95 at just 4.9 ms, positioning it as a competitive mid-scale option for accuracy and efficiency.

When examining per-class performance across YOLOv12 variants, a clear improvement in classification and localization is observed as model complexity increases. YOLOv12n, though modest overall, achieves respectable precision for potassium deficiency (0.871) and spotted wilt virus (0.873), but struggles with leaf miner, where recall drops to 0.666. YOLOv12s shows substantial improvement, with all classes reaching at least 0.847 mAP@50, and particularly strong results in magnesium deficiency (0.920) and leaf miner (0.905), highlighting enhanced sensitivity to subtle disease patterns.

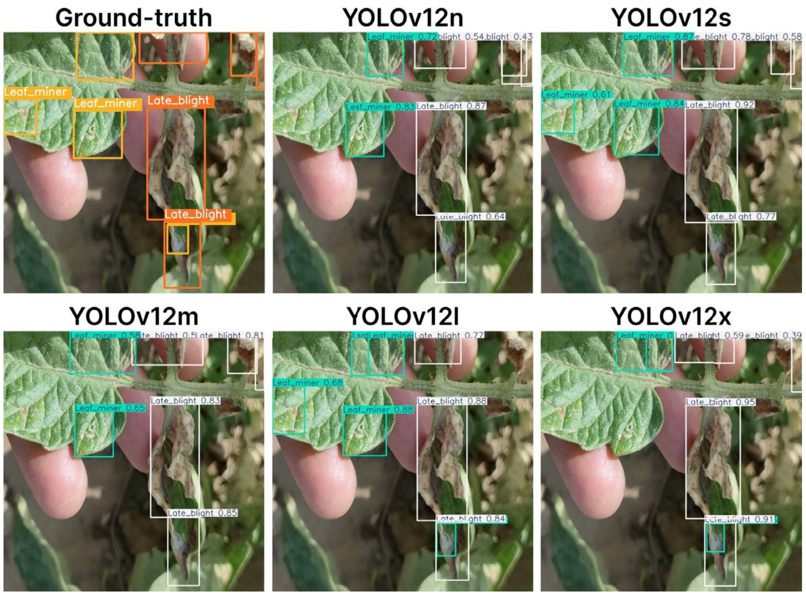
YOLOv12m further strengthens this trend, becoming the first variant where every class achieves over 0.875 mAP@50. Complex diseases like magnesium deficiency and spotted wilt virus surpass 0.920 mAP@50, reflecting the benefits of deeper attention layers and R-ELAN aggregation in improving spatial discrimination. In YOLOv12l, class-wise performance remains highly balanced. Although the overall mAP gain over YOLOv12m is modest, precision and recall stay consistently high across all classes. Magnesium deficiency reaches 0.938 mAP@50, and potassium deficiency achieves the highest mAP@50:95 at 0.788, highlighting YOLOv12l's ability to accurately detect both large, uniform symptoms and fine, scattered patterns, an outcome attributed to its deeper R-ELAN structures and integrated attention modules.

Finally, YOLOv12x maintains consistently high class-wise performance, though not uniformly. Potassium deficiency leads with 0.955 mAP@50 and 0.835 mAP@50:95, the highest across all models, while magnesium deficiency and spotted wilt virus also achieve strong metrics. In contrast, nitrogen deficiency lags slightly in mAP@50:95 (0.764) despite high precision, indicating less precise bounding box localization. Similarly, leaf miner exhibits lower recall and localization scores, likely due to its thin, linear patterns. Overall, YOLOv12x delivers top performance with high accuracy and confidence, although fine-grained symptom detection remains challenging even at maximum model capacity.

Regarding the visual tests, Fig. 17a shows the first qualitative comparison across YOLOv12 variants. Overall, models effectively control redundant detections, indicating stable confidence thresholds and well-calibrated post-processing. However, detection accuracy varies. From YOLOv12n through YOLOv12l, distinguishing spotted wilt virus from magnesium deficiency remains problematic, leading to incomplete identification of magnesium cases. Only YOLOv12x correctly detects both, likely due to deeper attention layers and enhanced spatial resolution. All models detect the single instance of leaf miner, although YOLOv12n and YOLOv12x



(a) First inference test of YOLOv12.



(b) Second inference test of YOLOv12.

Fig. 17. Inference tests of YOLOv12.

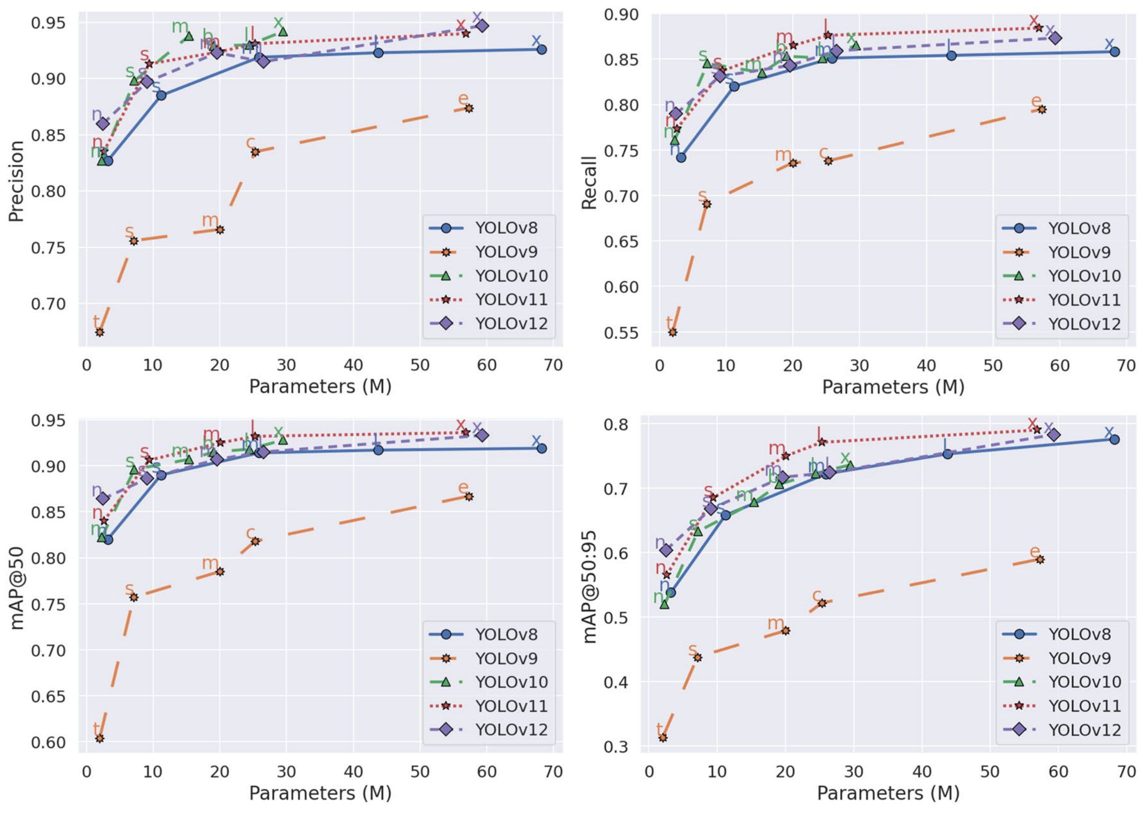


Fig. 18. Performance comparison between all architectures regarding their number of parameters.

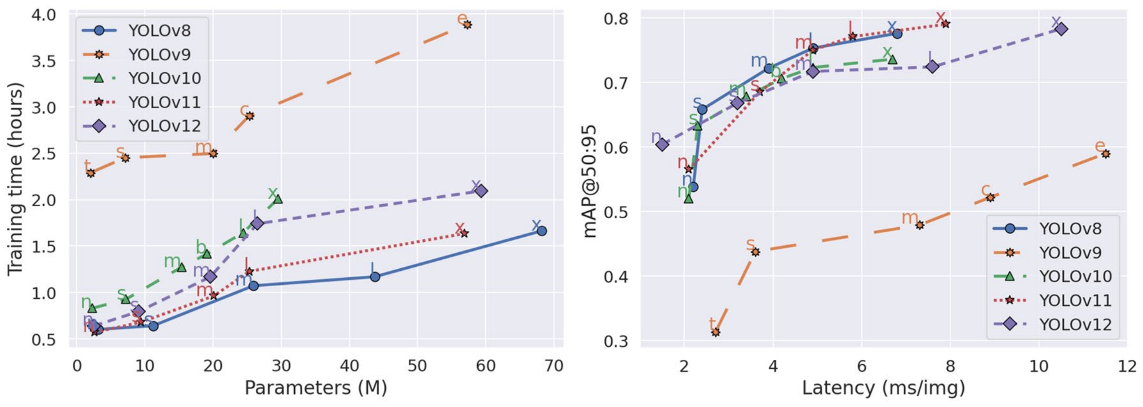


Fig. 19. Efficiency comparison between all architectures.

misclassify an additional instance, suggesting that both underparameterized and overparameterized models may confuse thin or vein-like background textures.

The second visual test, presented in Fig. 17b, again shows consistent bounding box behavior across YOLOv12 variants, with no excessive overlapping, suggesting stable confidence calibration. However, this test proves more challenging, with lighter variants (nano, small, and medium) failing to detect leaf miner instances near image borders or within occluded regions. Conversely, these models reliably detect late blight cases. YOLOv12l improves leaf miner detection but struggles with late blight at the edges, indicating sensitivity to spatial context. Surprisingly, YOLOv12x exhibits the most errors, missing leaf miner in bright areas and showing lower confidence in peripheral late blight detections. This highlights that increased model complexity does not guarantee better robustness in edge-case scenarios.

General remarks

To conclude the analysis of the architectures, Figs. 18 and 19 compare all models based on overall performance. The first key observation is that YOLOv10 models are the lightest, with even the x-large version staying below

30 million parameters. YOLOv9 increases in complexity, especially YOLOv9-E, nearing 60 million parameters. Similarly, YOLOv8's large variant reaches almost 44 million parameters, while YOLOv8x becomes the most complex model, approaching 70 million. YOLOv12 follows a comparable scaling trend, with its nano variant matching the lightest models, and its x-large model growing to just under 60 million parameters, placing it above YOLOv9, YOLOv10, and YOLOv11 x-large variants, but still below YOLOv8x.

Analyzing Fig. 18, which compares performance relative to parameter count, the YOLOv10 models stand out by delivering strong precision even in lighter variants, with the medium model already outperforming all YOLOv8 and YOLOv9 counterparts. YOLOv10-M, B, L, and X consistently surpass 90% precision, showcasing the effectiveness of their NMS-free dual-head design. YOLOv8 models follow closely, maintaining precision mostly above 85%, while YOLOv9 variants lag, with competitive precision only in C and E versions. YOLOv11 further elevates precision, with YOLOv11x nearing 95% and smaller variants matching or exceeding YOLOv8 models of similar complexity. YOLOv12 shows comparable precision to YOLOv11, especially in its nano, small, and medium versions. However, the large variant drops slightly, falling below YOLOv8-L, YOLOv10-L, and YOLOv11-L levels, nearing 90%. Despite this, YOLOv12x achieves the highest precision among all models, outperforming even YOLOv8x while using fewer parameters, confirming the strength of its attention-centric design.

Analyzing recall, the gap between YOLOv10 and YOLOv8 narrows, with only YOLOv10-X outperforming all YOLOv8 variants. YOLOv8-L and YOLOv8-X show competitive recall values, equaling or slightly surpassing most YOLOv10 and YOLOv9 models. Lighter variants from both families perform similarly, without major differences. YOLOv11 again sets a new benchmark, with its medium, large, and x-large versions all exceeding 85% recall, and even its nano model outperforming its counterparts. YOLOv12 shows intermediate recall, generally falling between YOLOv11 and YOLOv8, surpassing YOLOv8 consistently and maintaining a clear lead over YOLOv9, which remains the weakest. Among nano variants, YOLOv12n achieves the best recall, nearing 80%. However, across all models, no variant exceeds 90% recall, highlighting a common limitation.

Moving on to mAP@50, YOLOv8 and YOLOv10 show similar performance, surpassing 80% in lighter variants and 90% in heavier ones. YOLOv11 again stands out, with its medium, large, and x-large models outperforming all others, and even its nano and small variants exceeding their counterparts from YOLOv8, YOLOv9, and YOLOv10. YOLOv9 remains the weakest, with only its E variant exceeding 85%. YOLOv12 positions itself between YOLOv10 and YOLOv8, generally outperforming the latter but trailing the former. Although YOLOv12 does not surpass YOLOv11, its medium and large variants offer comparable results. Notably, YOLOv12n emerges as the best-performing nano model across all versions, confirming its strength in lightweight detection tasks.

The mAP@50:95 results confirm YOLOv11's superiority, with its variants consistently leading and several models approaching the 80% threshold, which no other versions achieve. The prior advantage of YOLOv10-X over YOLOv8 disappears here, as YOLOv8-L and YOLOv8-X outperform their YOLOv10 counterparts. YOLOv9 continues to underperform, with only its heaviest variants nearing 60%. In this stricter evaluation, YOLOv12 improves its relative position, consistently surpassing YOLOv8 and YOLOv10 and establishing itself as the second-best architecture after YOLOv11. YOLOv12n again excels among nano models, surpassing 60%, while YOLOv12x ranks just behind YOLOv11x as the second-best performer overall.

Notably, the comparatively weaker performance of YOLOv9 can be attributed to architectural and training-related factors. In particular, the integration of PGI with a reversible auxiliary branch and the GELAN backbone introduces higher training demands. While these innovations aim to maintain high accuracy and stable learning, they also introduce higher training demands and may require more extended convergence periods. Given the fixed training schedule of 100 epochs used across all models, combined with the diversity of disease classes and the density of instances in the dataset, YOLOv9 may not have had sufficient training time to fully optimize its parameters. This could explain why its performance lags behind later architectures such as YOLOv10 and YOLOv11, which adopt more efficient and lightweight components better suited for faster convergence.

Turning to Fig. 19, which compares efficiency in training time and latency, YOLOv9 models clearly require the most resources, with significantly higher training durations, making them the least practical for deployment. YOLOv10 improves efficiency, with all variants training under 2.1 hours. YOLOv8 performs even better, completing training in under 1.7 hours across all versions, positioning it among the most efficient. YOLOv11 continues this trend with slightly longer times, especially in its larger models, but still within an excellent range. YOLOv12 ranks third, with all variants training under 2.5 hours, slightly slower than YOLOv11 and YOLOv8, yet far more efficient than YOLOv9.

Regarding latency, YOLOv8 and YOLOv10 models display similar behavior, with YOLOv8 offering the best balance between speed and detection performance. YOLOv10 maintains comparable latency but slightly lags in accuracy, making its overall efficiency less favorable. YOLOv11 achieves the highest accuracy but at a modest cost in latency compared to YOLOv8, a relevant factor for strict real-time applications. YOLOv9 performs the worst, with even its better variants, YOLOv9-C and YOLOv9-E, exceeding 11 ms, limiting their practical use. YOLOv12 follows a similar latency profile to YOLOv10 and YOLOv11, with YOLOv12n standing out as the fastest nano variant. The small to large YOLOv12 models maintain comparable efficiency, while YOLOv12x exhibits a notable latency increase, ranking as the slowest model after YOLOv9-E.

Overall, for tomato leaf disease detection, YOLOv11 offers the best trade-off between model complexity, training time, and practical deployment. It consistently delivers top-tier accuracy while maintaining reasonable training and inference times. YOLOv8, YOLOv10, and YOLOv12 also provide strong alternatives, each excelling in different trade-off combinations. YOLOv10 and YOLOv12 in particular stand out for their efficient lightweight variants. Conversely, YOLOv9 underperforms despite its complexity, showing longer training times and limited gains in accuracy. These findings highlight that selecting an architecture depends not only on raw performance but also on balancing computational cost and real-world deployment needs in agricultural scenarios.

When training infrastructure is limited, opting for models such as YOLOv12n or YOLOv10-N provides significant time savings and enables faster retraining cycles, even if it comes at a small cost in detection accuracy. On the other hand, in environments where longer training times are acceptable and hardware resources are sufficient, models like YOLOv11x are preferable due to their superior generalization and precision. Ultimately, this study demonstrates that model choice should account for both resource availability and the precision demands of the target agricultural application, balancing efficiency with accuracy across varying deployment scenarios.

Limitations

Although this study makes a thorough effort to provide a comprehensive comparison, several limitations must be recognized. Firstly, the dataset used covers only six diseases, which may not fully represent the variety of diseases that could impact tomato leaves under different regional and environmental conditions. Additionally, the training and evaluation were conducted using high-performance hardware, which is not readily available in all settings, potentially affecting the replicability of the results in environments with more limited computational resources. Lastly, the focus of this work is solely on YOLO architectures, without considering other object detection models that may be of interest to certain readers.

Conclusions and future works

This work aims to compare YOLO architectures for the task of tomato leaf disease detection. Specifically, this study compares the latest versions of YOLO: YOLOv8, YOLOv9, YOLOv10, YOLOv11, and YOLOv12. For this, the Tomato-Village dataset, comprising a collection of 14,368 images of tomato leaf diseases across six types: late blight, leaf miner, magnesium deficiency, nitrogen deficiency, potassium deficiency, and spotted wilt virus, is used. Training was conducted using all available variants of the architectures, maintaining default parameters to ensure a solid comparative analysis. The results reveal significant differences in performance, highlighting the strengths and weaknesses of each architecture.

YOLOv11 models emerged as the top-performing architecture, achieving the highest precision, recall, and mAP scores while maintaining competitive training times and reasonable latency, positioning them as the most attractive option for tomato leaf disease detection. YOLOv10 and YOLOv12 follow closely, offering a strong balance between accuracy, speed, and efficiency. YOLOv8, while slightly behind YOLOv10, YOLOv11, and YOLOv12, still delivers notable performances and, in some cases, rivals heavier models. In contrast, YOLOv9 shows the weakest results, with the longest training times, poorest metrics, and highest latency. Notably, YOLOv12n stands out as the best nano variant across all architectures, offering exceptional speed and robust detection capabilities, making it ideal for extremely resource-constrained scenarios.

Future research could evaluate these models under varying environmental conditions and on different plant disease datasets to assess their robustness and generalization. It would also be valuable to test their deployment in real-time agricultural monitoring systems and compare them with other object detection techniques to identify the most accurate and efficient solutions. Finally, future work could focus on developing practical platforms based on these architectures to enable early disease detection and improve crop management strategies.

Data availability

The datasets generated during and/or analyzed during the current study are available in at: https://github.com/mamta-joshi-gehlot/Tomato-Village/tree/main

Received: 23 December 2024; Accepted: 8 July 2025
Published online: 24 July 2025

References

1. Mazumdar, P., Singh, P., Kethiravan, D., Ramathani, I. & Ramakrishnan, N. Late blight in tomato: Insights into the pathogenesis of the aggressive pathogen phytophthora infestans and future research priorities. *Planta* **253** (2021).
2. Thangaraj, R., Anandamurugan, S., Pandiyan, P. & Kaliappan, V. K. Artificial intelligence in tomato leaf disease detection: A comprehensive review and discussion. *J. Plant Dis. Prot.* https://doi.org/10.1007/s41348-021-00500-8 (2021).
3. Li, N. et al. Tomato and lycopene and multiple health outcomes: Umbrella review. *Food Chem.* **343**, 128396. https://doi.org/10.1016/j.foodchem.2020.128396 (2021).
4. Gehlot, M., Saxena, R. & Gandhi, G. C. Tomato-village: A dataset for end-to-end tomato disease detection in a real-world environment. *Multimed. Syst.* https://doi.org/10.1007/s00530-023-01158-y (2023).
5. Qasrawi, R., Amro, M., Zaghal, R., Sawafteh, M. & Polo, S. V. Machine learning techniques for tomato plant diseases clustering, prediction and classification. In *2021 International Conference on Promising Electronic Technologies (ICPET)*, pp. 40–45, https://doi.org/10.1109/ICPET53277.2021.00014 (2021).
6. Ananthi, P., Devi, K. N., D, G., Shanmugapriya, P. & S, G. Tomato leaf diseases prediction using deep learning algorithms. In *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, pp. 1–5, https://doi.org/10.1109/ADICS58448.2024.10533491 (2024).
7. Kibriya, H., Rafique, R., Ahmad, W. & Adnan, S. Tomato leaf disease detection using convolution neural network. In *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*, pp. 346–351, https://doi.org/10.1109/IBCAST51254.2021.9393311 (2021).
8. David, H. E., Ramalakshmi, K., Gunasekaran, H. & Venkatesan, R. Literature review of disease detection in tomato leaf using deep learning techniques. In *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, 274–278, https://doi.org/10.1109/ICACCS51430.2021.9441714 (2021).
9. Nanehkaran, Y. A., Zhang, D., Chen, J., Tian, Y. & Al-Nabhan, N. Recognition of plant leaf diseases based on computer vision. *J. Ambient. Intell. Humaniz. Comput.* https://doi.org/10.1007/s12652-020-02505-x (2020).
10. Sood, M. & Singh, P. K. Anomaly detection and qualitative analysis of diseases in tomato. In *Recent innovations in computing* (eds Singh, P. K. et al.) 475–487 (Springer Singapore, Singapore, 2021).

11. Chaudhary, P., Verma, A., Kukreja, V. & Sharma, R. Integrating deep learning and ensemble methods for robust tomato disease detection: A hybrid cnn-rf model analysis. In *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 1–4, https://doi.org/10.1109/ICRITO61523.2024.10522213 (2024).
12. Ghosh, H., Rahat, I. S., Pattanayak, R. M. & Mohanty, S. N. Innovative approaches in tomato leaf disease recognition using deep learning. In *2023 6th International Conference on Recent Trends in Advance Computing (ICRTAC)*, pp. 86–96, https://doi.org/10.1109/ICRTAC59277.2023.10480762 (2023).
13. Wu, X., Sahoo, D. & Hoi, S. C. Recent advances in deep learning for object detection. *Neurocomputing* **396**, 39–64. https://doi.org/10.1016/j.neucom.2020.01.085 (2020).
14. Wang, S., Xiao, T., Liu, Q. & Zheng, H. Deep learning for fast MR imaging: A review for learning reconstruction from incomplete k-space data. *Biomed. Signal Process. Control* **68**, 102579. https://doi.org/10.1016/j.bspc.2021.102579 (2021).
15. Dhanya, V. et al. Deep learning based computer vision approaches for smart agricultural applications. *Artif. Intell. Agric.* **6**, 211–229. https://doi.org/10.1016/j.aiia.2022.09.007 (2022).
16. Liu, Y., Liu, D., Wang, B. & Chen, B. Mob-yolo: A lightweight UAV object detection method. In *2022 International Conference on Sensing, Measurement & Data Analytics in the era of Artificial Intelligence (ICSMD)*, pp. 1–6 (2022).
17. Renuga Devi, R., R, K., G, N. & A, V. An automated helmet detection for bike power start control using yolov8. In *2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, pp. 1–6, https://doi.org/10.1109/SCEECS61402.2024.10481850 (2024).
18. Kaushal, M. Rapid -yolo: A novel yolo based architecture for shadow detection. *Optik* **260**, 169084. https://doi.org/10.1016/j.ijleo.2022.169084 (2022).
19. Sadi, A. A., Hossain, Z., Ahmed, A. U. & Shad, M. T. A comparative study on plant diseases using object detection models. In *Intelligent computing* (ed. Arai, K.) 419–438 (Springer Nature Switzerland, Cham, 2024).
20. Iren, E. Comparison of yolov5 and yolov6 models for plant leaf disease detection. *Eng., Technol. Appl. Sci. Res.* **14**, 13714–13719. https://doi.org/10.48084/etasr.7033 (2024).
21. Ali, U., Ismail, M. A., Ahamed, R. & Shah, A. Performance evaluation of yolo models in plant disease detection. *J. Inf. Web Eng.* **3**, 199–211. https://doi.org/10.33093/jiwe.2024.3.2.15 (2024).
22. Shetty, K. U., Javed Kutty, R., Donthi, K., Patil, A. & Subramanyam, N. Plant disease detection for guava and mango using yolo and faster r-cnn. In *2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, vol. 2, pp. 1–6, https://doi.org/10.1109/IATMSI60426.2024.10503209 (2024).
23. Boudaa, B., Abada, K., Aichouche, W. A. & Nabil Belakermi, A. Advancing plant diseases detection with pre-trained yolo models. In *2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pp. 1–6, https://doi.org/10.1109/PAIS62114.2024.10541267 (2024).
24. Natij, Y., El Karch, H., Maafiri, A. & Mezouari, A. Evaluating the performance of yolo object detectors for plant disease detection. In *2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1–6, https://doi.org/10.1109/WINCOM62286.2024.10656918 (2024).
25. Padilla, J. I. A., Pastor, H. M. R., Velasco, A. C. T. & Magsumbol, J.-A. V. Performance evaluation of yolov8, yolov9 and yolov10 models in detecting fusarium wilt disease in banana leaf plants. In *2024 IEEE International Conference on Imaging Systems and Techniques (IST)*, pp. 1–7, https://doi.org/10.1109/IST63414.2024.10759219 (2024).
26. Aisyah Mohd Robi, S. N. *et al.* Comparative analysis of yolo models for melon leaf disease classification in uav-assisted smart agriculture. In *2024 5th International Conference on Smart Sensors and Application (ICSSA)*, pp. 1–5, https://doi.org/10.1109/ICSSA62312.2024.10788572 (2024).
27. Pandey, M. et al. A review on biology and possible management strategies of tomato leaf miner, Tuta Absoluta (Meyrick), Lepidoptera: Gelechiidae in Nepal. *Heliyon* **9**, e16474. https://doi.org/10.1016/j.heliyon.2023.e16474 (2023).
28. Tian, X.-Y. et al. Physiological and molecular advances in magnesium nutrition of plants. *Plant Soil* **468**, 1–17. https://doi.org/10.1007/s11104-021-05139-w (2021).
29. Azimi, S., Kaur, T. & Gandhi, T. K. A deep learning approach to measure stress level in plants due to nitrogen deficiency. *Measurement* **173**, 108650. https://doi.org/10.1016/j.measurement.2020.108650 (2021).
30. Okazaki, K. et al. Metabolic indices related to leaf marginal necrosis associated with potassium deficiency in tomato using gc/ms metabolite profiling. *J. Biosci. Bioeng.* **130**, 520–524. https://doi.org/10.1016/j.jbiosc.2020.06.007 (2020).
31. Salonki, V., Baliyan, A., Kukreja, V. & Siddiqui, K. M. Tomato spotted wilt disease severity levels detection: A deep learning methodology. In *2021 8th International Conference on Signal Processing and Integrated Networks (SPIN)*, 361–366, https://doi.org/10.1109/SPIN52536.2021.9566053 (2021).
32. Liu, S., Qi, L., Qin, H., Shi, J. & Jia, J. Path aggregation network for instance segmentation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8759–8768, https://doi.org/10.1109/CVPR.2018.00913 (2018).
33. Ramos, L. T. & Sappa, A. D. A decade of you only look once (yolo) for object detection. *arXiv preprint* arXiv:2504.18586 (2025).
34. Vats, A. & Anastasiu, D. C. Enhancing retail checkout through video inpainting, yolov8 detection, and deepsort tracking. In *INPROCEEDINGS of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 5529–5536 (2023).
35. Kim, J.-H., Kim, N. & Won, C. S. High-speed drone detection based on yolo-v8. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–2, https://doi.org/10.1109/ICASSP49357.2023.10095516 (2023).
36. Wang, C.-Y., Liao, H.-Y. M. & Yeh, I.-H. Designing network design strategies through gradient path analysis. *arXiv preprint* arXiv:2211.04800 (2024).
37. Zhao, M. et al. Med-yolov8s: A new real-time road crack, pothole, and patch detection model. *J Real-Time Image Process* https://doi.org/10.1007/s11554-023-01405-5 (2024).
38. Wang, C.-Y. & Liao, H.-Y. M. YOLOv9: Learning what you want to learn using programmable gradient information. *arXiv preprint* arXiv:2402.13616 (2024).
39. Wang, C.-Y. *et al.* Cspnet: A new backbone that can enhance learning capability of CNN. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1571–1580, https://doi.org/10.1109/CVPRW50498.2020.00203 (2020).
40. Wang, A. *et al.* Yolov10: Real-time end-to-end object detection. *arXiv preprint* arXiv:2405.14458 (2024).
41. Guo, C. et al. ANMS: attention-based non-maximum suppression. *Multimed Tools Appl* https://doi.org/10.1007/s11042-022-12142-5 (2022).
42. Sapkota, R., Meng, Z. & Karkee, M. Synthetic meets authentic: Leveraging LLM generated datasets for yolo11 and yolov10-based apple detection through machine vision sensors. *Smart Agric Technol* **9**, 100614. https://doi.org/10.1016/j.atech.2024.100614 (2024).
43. Li, Y., Zhu, C., Zhang, Q., Zhang, J. & Wang, G. If-yolo: An efficient and accurate detection algorithm for insulator faults in transmission lines. *IEEE Access* **12**, 167388–167403. https://doi.org/10.1109/ACCESS.2024.3496514 (2024).
44. Tian, Y., Ye, Q. & Doermann, D. Yolov12: Attention-centric real-time object detectors. *arXiv preprint* arXiv:2502.12524 (2025).
45. Alif, M. A. R. & Hussain, M. Yolov12: A breakdown of the key architectural features. *arXiv preprint* arXiv:2502.14740 (2025).
46. Padilla, R., Netto, S. L. & da Silva, E. A. B. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 237–242, https://doi.org/10.1109/IWSSIP48289.2020.9145130 (2020).

47. Ramos, L. T. & Sappa, A. D. Leveraging u-net and selective feature extraction for land cover classification using remote sensing imagery. *Sci. Rep.* **15**, 784. https://doi.org/10.1038/s41598-024-84795-1 (2025).
48. Zoph, B. et al. Learning data augmentation strategies for object detection. In *Computer vision - ECCV 2020* (eds Vedaldi, A. et al.) 566–583 (Springer International Publishing, Cham, 2020).
49. Zou, Z., Chen, K., Shi, Z., Guo, Y. & Ye, J. Object detection in 20 years: A survey. *Proc. IEEE* **111**, 257–276. https://doi.org/10.1109/JPROC.2023.3238524 (2023).
50. Ramos, L. T. & Sappa, A. D. Multispectral semantic segmentation for land cover classification: An overview. *IEEE J Sel Top Appl Earth Obs Remote Sens* **17**, 14295–14336. https://doi.org/10.1109/JSTARS.2024.3438620 (2024).

Author contributions
L.T.R.: Conceptualization, Formal Analysis, Investigation, Methodology, Software, Writing – original draft, Writing – review & editing. A.D.S.: Conceptualization, Methodology, Writing – review & editing, Supervision. All authors reviewed the manuscript.

Funding
This work was supported in part by the Air Force Office of Scientific Research Under Award FA9550-24-1-0206, in part by the Agencia Estatal de Investigación, Ministerio de Ciencia e Innovación, under Grant PID2021-128945NB-I00 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe," and in part by the ESPOL project CIDIS-003-2024. The authors acknowledge the support of the Generalitat de Catalunya CERCA Program to CVC's general activities, and the Departament de Recerca i Universitats from Generalitat de Catalunya with reference 2021SGR01499.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to L.T.R. or A.D.S.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.