International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Unveiling the Potential of YOLOv9 through Comparison with YOLOv8

Hafedh Mahmoud Zayani*[1], Ikhlass Ammar[2], Refka Ghodhbani[3], Taoufik Saidani[3], Rahma Sellami[4], Mohamed Kallel[5], Amjad A. Alsuwaylimi[6], Kaznah Alshammari[6], Faheed A. F. Alrslani[6], Mohammad H. Algarni[7]

Submitted: 29/01/2024 **Revised**: 07/03/2024 **Accepted**: 15/03/2024

Abstract: This study introduces YOLOv9, a new object detection model building upon the success of YOLOv8. While YOLOv8 has delivered impressive results, there was a push to further enhance accuracy and efficiency. We delve into the architectures of both YOLO models to understand the trade-off between training speed and accuracy. YOLOv9 introduces additional complexity compared to its predecessor, which translates to superior performance. To evaluate the models' capabilities, we leveraged a tomato disease detection dataset from Roboflow. This dataset encompasses three disease classes for tomato fruits, along with a healthy class. Our experiments demonstrate that YOLOv9 achieves a significant improvement in accuracy (93.6% vs. 92%), while maintaining comparable training efficiency. However, it requires slightly longer training times compared to YOLOv8. To further substantiate these results, we present comprehensive analyses of precision, recall, F1-score, and loss functions during both training and validation stages. Additionally, in the testing phase, YOLOv9 exhibits superior precision in detecting tomato diseases. While requiring slightly more training resources, YOLOv9 offers a compelling trade-off between accuracy and efficiency. This makes it a promising choice for applications where precise object detection is paramount.

Keywords: *YOLOv9, YOLOv8, Object Detection, Training Efficiency, Accuracy.*

1. Introduction

Deep learning has revolutionized various fields, and object detection is no exception. The YOLO (You Only Look Once) family stands out for its speed and efficiency. Unlike traditional methods, YOLO predicts bounding boxes and object probabilities in a single pass. This has led to the development of several YOLO versions, each building upon the strengths of its predecessor.

Past iterations of YOLO have focused on specific goals. YOLOv3 balanced accuracy and speed, making it suitable for real-time applications [1, 2]. YOLOv4 and v6 prioritized real-time performance for scenarios like autonomous driving [3]. YOLOv5 offered a modular structure for easy customization. Recent versions, YOLOv7 and v8, have emphasized high accuracy, especially for smaller objects [4, 5].

YOLOv9, currently under development, aims to further improve accuracy and efficiency while handling complex situations more robustly (based on community discussions and resources) [6]. When choosing a YOLO version, consider your application's needs: accuracy requirements, real-time constraints, and available resources.

The versatility of YOLO lies in its wide range of applications. YOLO models are commonly used for general object detection, identifying objects like people, animals, vehicles, and everyday items [7 ,8]. They also play a role in security systems, detecting people in restricted areas or identifying authorized personnel [9, 10, 11, 12]. In agriculture, YOLO models have been used to detect plant diseases in leaves and fruits [12, 14, 15, 16, 17] and identify specific types for yield estimation or quality control [13, 18, 19].

This study delves deeper into YOLOv9, specifically comparing its architecture to YOLOv8 in the context of object detection. By analyzing their modules, we demonstrate how YOLOv9's increased complexity leads to improvements in both accuracy and efficiency. This positions YOLOv9 as a compelling choice, although it requires more memory. To verify these advantages, we trained and evaluated both models on a tomato disease detection dataset. We analyzed metrics like precision, recall,

[1] *Department of Electrical Engineering, College of Engineering, Northern Border University, Arar, Saudi Arabia.*
ORCID ID : 0000-0001-7195-9743
[2] *OASIS Laboratory, National Engineering School of Tunis, University of Tunis El Manar, Tunisia.*
Email: Ikhlass_Ammar@yahoo.fr
[3] *Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia.*
Email: refka.ghodhbani@nbu.edu.sa, taoufik.saidan@nbu.edu.sa
[4] *Department of Computer Science, Applied College, Northern Border University, Saudi Arabia.*
Email: rahma.ali@nbu.edu.sa
[5] *Department of Physics , Faculty of Sciences and arts, Northern Border University, Rafha 91911, Saudi Arabia.*
Email: mohamed.kallel@nbu.edu.sa
[6] *Department of Information Technology, Faculty of Computing and Information Technology, Northern Border University, Rafha 91911, Saudi Arabia.*
Email: amjad.alsuwaylimi@nbu.edu.sa, khaznah.alshammari2@nbu.edu.sa, f.alrslani@nbu.edu.sa
[7] *Department of Computer Science, Al-Baha University, Saudi Arabia.*
Email: malgarni@bu.edu.sa
** Corresponding Author Email: Hafedh.Zayani@nbu.edu.sa*

mean Average Precision (mAP), accuracy, and loss function during training and validation phases. Finally, we compared their detection performance on the dataset.

2. PROPOSED METHODS

In this study, we focus on two methods of YOLO: YOLOv8 and YOLOv9.

2.1. YOLOv8 architecture

YOLOv8, is emerged in 2022. It composed by the following three blocs as shown in Fig. 1:

- ***Backbone***: YOLOv8 utilizes a modified version of the CSPDarknet53 architecture as its backbone. This network is known for its efficient balance between accuracy and computational cost. It employs cross-stage partial connections to improve information flow between different layers.
- ***Neck***: Uses a simple Path Aggregation Network (PANet) for feature fusion, focusing on efficiency.
- ***Head***: Consists of multiple convolutional layers followed by fully connected layers. These layers analyze the features extracted by the backbone and make predictions for object detection.

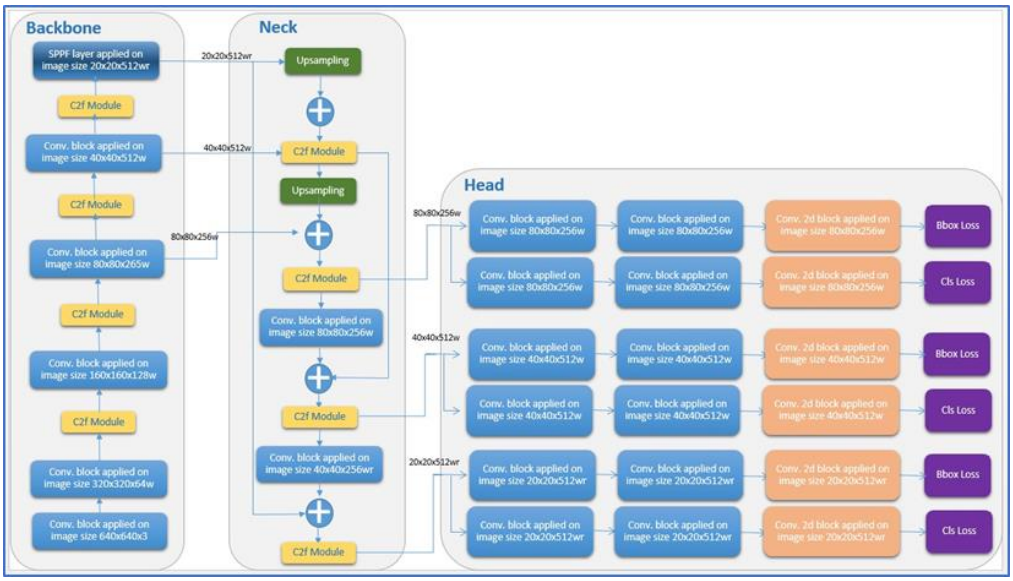

Fig 1. YOLOv8 architecture

2.2. YOLOv9 architecture

YOLOv9, the newest member of the YOLO family of object detection models, introduces key improvements focusing on accuracy, efficiency, and parameter utilization. The architecture of YOLOv9, in [6], based on two concepts:

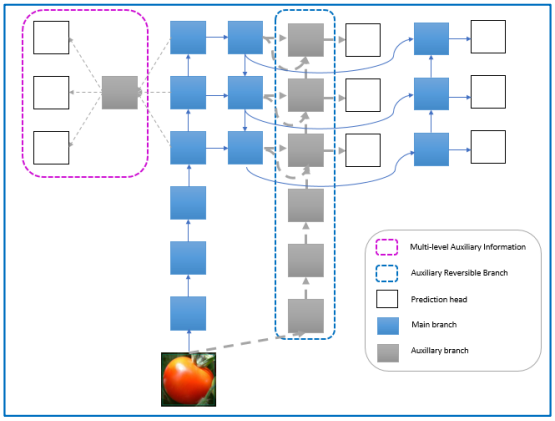

Fig 2. Programmable Gradient Information (PGI) architecture in YOLOv9

Programmable Gradient Information (PGI): This technique addresses potential information loss during training, ensuring accurate model updates. PGI, shown in Fig. 2, works by preserving information that retains all data required to calculate the model's objective function, leading to reliable gradient information for updating network weights and balancing efficiency and accuracy. PGI acts as a training tool, enhancing gradient backpropagation through the network while minimizing inference cost. It achieves this by:

- Main branch: YOLOv9 ensures that no extra inference cost is added because the other components of PGI are not necessary for the inference step.
- Removable auxiliary branch: During training, an additional branch processes information to generate reliable gradients. However, this branch is removed at inference time to maintain model compactness and speed.
- Multi-level auxiliary information: This uses an integration network to combine gradients from various network regions, providing the main branch with comprehensive information for accurate predictions.

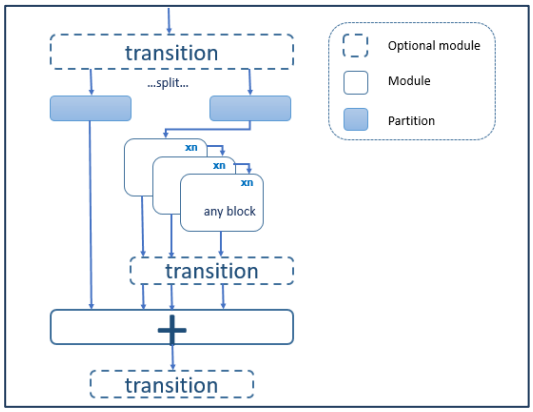

Fig 3. GELAN architecture in YOLOv9

Generalized Efficient Layer Aggregation Network (GELAN): This architecture optimizes lightweight models by maximizing parameter efficiency and combining strengths of existing approaches. GELAN, shown in Fig. 3, surpasses existing methods in its ability to leverage parameters effectively utilizing conventional convolution operations. It integrates the efficient gradient path planning of CSPNet with the fast inference capabilities of ELAN, achieving a balance between model size, speed, and accuracy.

YOLOv9 comes in four versions (v9-S, v9-M, v9-C, and v9-E) with varying parameter counts, offering flexibility based on computational resources as shown in Table 1.

Table 1. YOLOv9 versions

Module	Parameters	FLOPs(G)	Test size
YOLOv9-S	7.2	26.7	640
YOLOv9-M	20.1	76.8	640
YOLOv9-C	25.5	102.8	640
YOLOv9-E	58.1	192.5	640

2.3. Comparison between YOLOv8 and YOLOv9

Fig. 4 illustrates the differences in modules used by YOLOv8 and YOLOv9 during the training process. YOLOv9 employs a wider range of modules compared to YOLOv8. This suggests that YOLOv9 might have a more complex architecture. YOLOv9 replaces C2f module with the RepNCSELAN4 module and SPPF module with the SPPLAN module. The C2f module's architecture incorporates two parallel gradient flow branches to enhance the robustness of gradient information flow. The architecture of the RepNCSELAN4 module is an enhanced version of CSP-ELAN designed to improve the feature extraction process. The input from the initial convolutional layer is divided into two routes, each processed through a sequence of RepNCSP and convolutional layers before being combined again. The dual-path technique improves gradient flow and feature reuse, boosting the model's learning efficiency and inference speed by maintaining depth without the usual computational cost of increased complexity.

The Spatial Pyramid Pooling Fusion (SPPF) module in YOLOv8 can extract contextual information from photos at different scales, which greatly improves the model's ability to generalize. SPPLAN integrates Spatial Pyramid Pooling (SPP) into the ELAN structure to enhance layer aggregation. The process begins with a convolutional layer that modifies the channel dimensions, and then proceeds with a sequence of spatial pooling operations to gather multi-scale contextual information. The combined outputs are then processed by an additional convolutional layer to enhance the network's ability to extract detailed data from different spatial levels. YOLOv9 introduces new modules: Adown, CBLinear and CBFuse. Both YOLOv8 and YOLOv9 utilize the Upsample module.

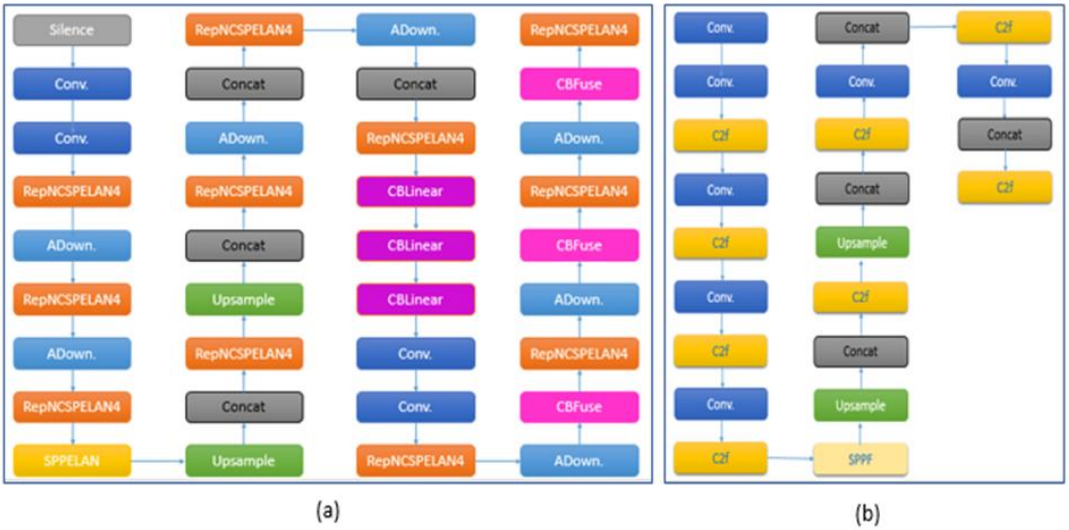

Fig. 4. Modules in: (a) YOLOv9 (b) YOLOv8

3. EXPERIMENTS

We explored the potential of YOLOv9 for tomato disease detection by comparing it to YOLOv8. The experiment utilized Google Colab, a free cloud-based platform, to optimize accuracy. While attempting to improve both algorithms, we encountered memory limitations with YOLOv9, ultimately stopping the training process. Specifically, we set the batch size to 10 and progressively increased the number of training epochs (iterations) until YOLOv9's memory requirements became excessive.

3.1. Experimental platform

For object detection of tomato diseases, we employed the YOLOv8l and YOLOv9c models. To accelerate and optimize training, we leveraged the powerful 12GB NVIDIA Tesla T4 GPU available on free Google Colab.

To validate our model's performance, we used a dataset of 159 images. Each image was resized to 640 pixels, achieving a balance between model accuracy and training efficiency.

3.2. Units

This experiment employed a dataset acquired from Roboflow, specifically focusing on three tomato disease classes in addition to a healthy class. The dataset encompasses a total of 789 images shown in Fig 5.

To facilitate training and evaluation, the dataset was strategically divided into three subsets:

- Training set: Comprises 70% (551 images) of the dataset, used to train the YOLO model.
- Validation set: Represents 20% (159 images), utilized to assess model performance during training and prevent overfitting.
- Testing set: Consists of the remaining 10% (79 images), employed for final evaluation of the trained model's generalizability.

Fig. 5. Number of images for each category

3.3. Evaluation index

YOLO models' performance is evaluated using a variety of metrics, including precision, recall, F1score, and accuracy.

Precision: is a statistic that represents the number of true positive projections inside the affirmative classification. The formula is shown in Eq. (1).

$$P = \frac{T_p}{T_p+F_p} \quad (1)$$

Where T_p represents the True Positive images that was correctly classified as positive, F_p represents the False Positive images that was incorrectly classified as positive.

Recall: is a statistic that measures the number of accurate class projections generated by the dataset's successful instances. Recall is expressed as shown in Eq. (2).

$$R = \frac{T_p}{T_p+F_N} \quad (2)$$

Where F_N represents the False Negative that was incorrectly classified as negative

F1 score: is a measure of the harmonic mean of precision and recall. The formula is given in Eq. (3).

$$F1 = \frac{P*R}{P+R} \quad (3)$$

Accuracy: refers to the proportion of occurrences anticipated accurately by the system. The accuracy formula is given in Eq. (4).

$$R = \frac{T_p+T_N}{T_p+F_N+T_N+F_p} \quad (4)$$

Where T_N represents the True Negative that was correctly classified as negative.

3.4. Experimental results analysis

This section presents the outcomes of training both YOLO algorithms on the tomato disease detection dataset. We evaluated their performance based on three key metrics: precision, recall, and mean Average Precision (mAP).

Fig. 6 visually depicts the trend of YOLOv8 through a graph, showcasing the evolution of precision, recall, and mAP as the number of epochs increases. We observed a positive correlation between the number of training epochs and the performance metrics. As the number of epochs increased, the values of precision, recall, and mAP improved, indicating that the model learned and refined its ability to accurately detect tomato diseases. Therefore, it is important to find the optimal number of epochs that balances training and generalization performance.

While YOLOv9 achieved improvements in performance compared to YOLOv8, it comes at the cost of increased complexity and computational demands. YOLOv9 utilizes a significantly deeper network with 724 layers compared to YOLOv8's 268 layers. This increased complexity allows the model to potentially learn more patterns that are intricate

and achieve higher accuracy. However, it also translates to the following:

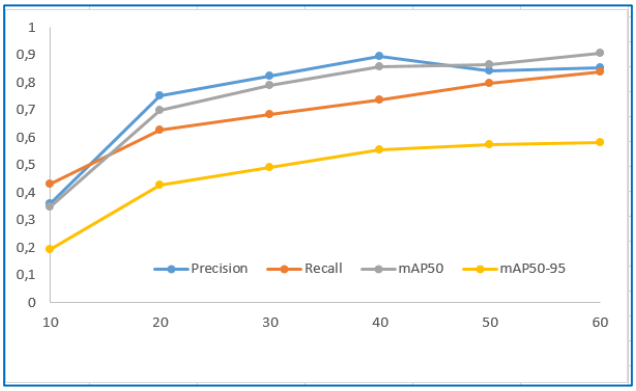

Fig. 6. Performance metrics for different epoch in YOLOv8

Higher computational cost: YOLOv9 requires 237.7 GFLOPs for calculations, whereas YOLOv8 uses 164.8 GFLOPs. This difference indicates that YOLOv9 may require more powerful hardware or longer training times.

Increased number of parameters: YOLOv9 has 50,965,560 parameters, surpassing YOLOv8's 43,609,692 parameters. More parameters can lead to better fitting of complex data but also increase the risk of overfitting and the need for larger datasets for training.

Training Interrupted at Epoch 70: Due to insufficient memory, YOLOv9 could not complete training the dataset for all 70 epochs on Google Colab. Fig. 7 shows the specific error message encountered. The proposed data produces the same error message regardless of the chosen training parameters. This occurs when using:

- A batch size of 16 and 10 epochs, and
- A dataset containing 3,000 images and 10 epochs.

```
Tried to allocate 876.00 MiB. GPU 0 has a total capacty of 14.75 GiB of which
597.06 MiB is free. Process 282540 has 14.16 GiB memory in use. Of the allocated
memory 13.91 GiB is allocated by PyTorch, and 98.58 MiB is reserved by PyTorch
but unallocated. If reserved but unallocated memory is large try setting
max_split_size_mb to avoid fragmentation. See documentation for Memory Management
and PYTORCH_CUDA_ALLOC_CONF
```

Fig. 7. Training outcome on Google Colab for 70 epochs with the YOLOv9 algorithm

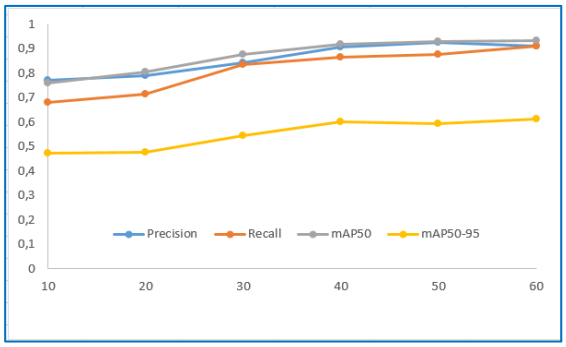

Fig. 8. Performance metrics for different epoch in YOLOv9

Performance Trend: Similar to YOLOv8, YOLOv9 exhibits a positive correlation between the number of training epochs and performance metrics (precision, recall, and mAP). Fig. 8 illustrates a graph that shows the evolution of the performance metrics. This suggests that the model continues to learn and refine its detection accuracy as the training progresses.

Confusion matrix: This matrix helps us understand how well the model distinguished between different classes (healthy tomatoes versus different disease categories) and identify potential areas for further improvement. Fig. 9 and Fig 10 display the confusion matrix for YOLOv8n and YOLOv9-c, respectively. Both models achieve reasonably good performance in classifying the five categories ("BlossomEndRot", "Healthy", "Splitting", "SunScaled", and "Background"). YOLOv9 seems to perform slightly better than YOLOv8 in terms of overall accuracy, with higher values on the diagonal (representing correct classifications) for most classes. For "BlossomEndRot" disease, both models perform well, correctly classifying over 90% of instances (YOLOv8: 93%, YOLOv9: 95%). However, YOLOv8 incorrectly classifies a slightly higher proportion of "Healthy" tomatoes category compared to YOLOv9 (YOLOv8: 3%, YOLOv9: 6%). In addition, YOLOv8 misclassifies more "Splitting" and "SunScaled" instances than YOLOv9 (YOLOv8: 28%, YOLOv9: 13%) and (YOLOv8: 14%, YOLOv9: 3%), respectively. Thus, YOLOv8 performs well for "BlossomEndRot" and "SunScaled" but struggle with "Splitting" category and classifying some "Healthy" tomatoes correctly. YOLOv9 shows improvement in "Splitting" classification compared to YOLOv8 and achieves slightly higher accuracy for "BlossomEndRot" class, but incorrectly classify a few more "Healthy" tomatoes category.

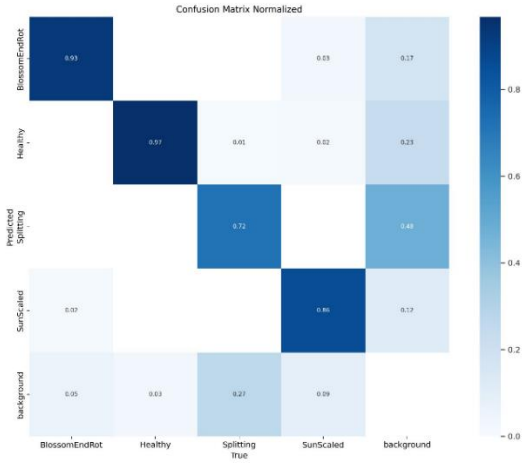

Fig. 9. Confusion matrix on YOLOv8n

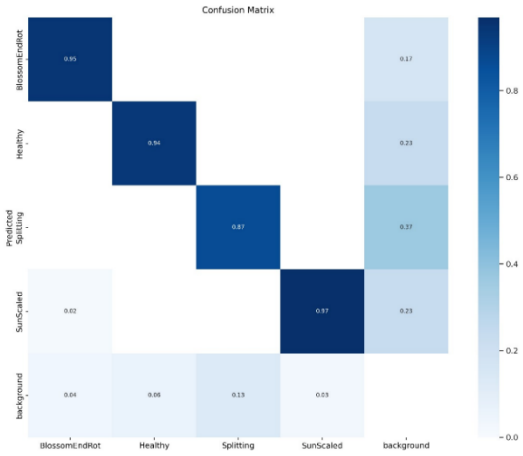

Fig. 10. Confusion matrix on YOLOv9-c

Overall, both YOLOv8 and YOLOv9 demonstrate promising results in classifying tomato disease categories. YOLOv9 appears to achieve slightly better overall performance and improvements in specific areas like "Splitting" classification and "Background" misclassification. However, it also exhibits slightly higher misclassification of "Healthy" class as "Background" class.

3.5. Comparison results

This section provides a comparison of the experimental results based on the following characteristics:

Time of training: Table 2 reveals that YOLOv8 consistently finishes training faster than YOLOv9 across all epochs shown. The table shows the difference in training time in the last column, with positive values indicating that YOLOv9 is between 0.08 and 0.44 slower than YOLOv8.

Table 2. Speed for both YOLO

Number of Epochs	YOLOv8 (hours)	YOLOv9 (hours)	*Training Time Difference*
10	0.11	0.195	*+0.085*
20	0.269	0.349	*+0.08*
30	0.334	102.8	*+0.196*
40	0.456	0.53	*+0.237*
50	0.603	0.693	*+0.285*
60	0.668	0.888	*+0.44*
70	*0.772*	*CUDA out of memory*	-

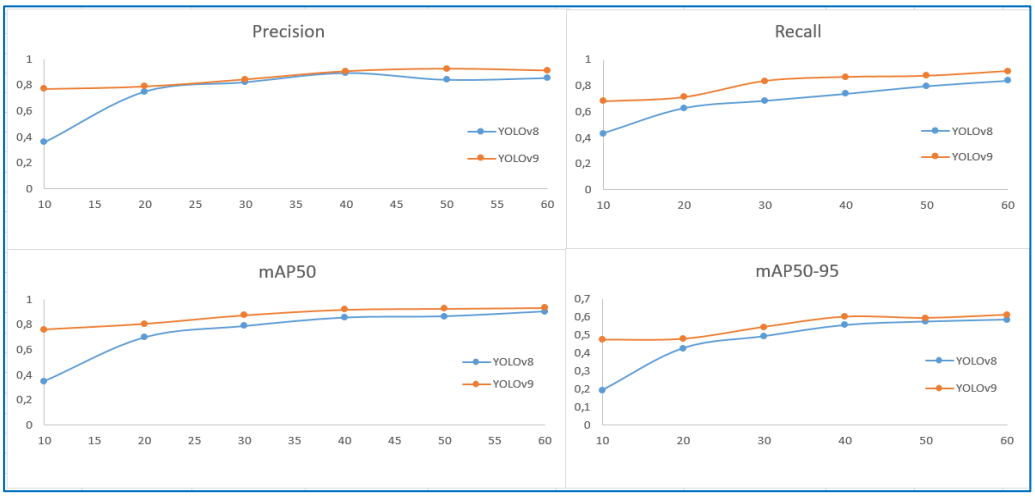

Fig. 11. Comparison of metrics for performance between YOLOv8 and YOLOv9

Performance metrics: Fig. 11 shows the precision, recall, and mAP (mean Average Precision) for YOLOv8 and YOLOv9 on the tomato disease detection task. The x-axis represents the number of epochs (training iterations) for both YOLOv8 and YOLOv9 and the y-axis represents the values of precision, recall, and mAP. These metrics usually range from 0 to 1, where: the precision measures the proportion of true positives (correctly identified diseased tomatoes) among all positive predictions (including both correct and incorrect classifications as diseased), recall measures the proportion of true positives out of all actual diseased tomatoes in the data (identifies most diseased tomatoes with minimal false negatives) and mAP provides a comprehensive overview of detection performance by considering both precision and recall across different classes (healthy and various disease types). Generally, increasing epochs lead to improvements in performance metrics. However, it's important to avoid overfitting, where the model memorizes the training data too well and performs poorly on unseen data.

Table 3. Comparison between YOLO v8 and v9

	YOLOv8				*YOLOv9*			
Classes	**P (%)**	**R (%)**	**F1 (%)**	**Acc. (%)**	**P (%)**	**R (%)**	**F1 (%)**	***Acc. (%)***
BlossomEndRot	82.3	93	43.7	94.6	84.8	95	44.8	*95.6*
Healthy	78.9	97	43.5	94.2	80.3	94	43.3	*94.2*
Splitting	60	72	32.7	84.8	70.2	87	38.8	*90*
SunScaled	*86*	*86*	*43*	*94.4*	*80.2*	*97*	*43.9*	*94.6*

Table 4. Comparison between YOLO 8 and 9

	YOLOv8	**YOLOv9**
Strengths	• YOLOv8 misclassified 3% of "Background" pixels as "Healthy". • Both models in the "BlossomEndRot" category are incorrectly labeled as SunScaled at 2%. • Both of models could not classify 17% and 23% of images on "BlossomEndRot" and "Healthy", respectively.	• YOLOv9 reduced the misclassified of "Background" class in "Splitting" category from 27% to 13%. • YOLOv9 showed improvement, only misclassifying 3% of "Background" category as "SunScaled". • YOLOv8 misclassified "BlossomEndRot" as "Background" category by 5%, higher than YOLOv9 by 1%. • YOLOv9 reduced the percentage of false negatives in "Splitting" compared to YOLOv8 by 5%.
Weaknesses	• YOLOv8 misclassified 27% of "Background" class as "splitting", higher than YOLOv9 by 14%. • YOLOv8 misclassified "Background" category as "SunScaled" (9%) and some "Healthy" class (3%).	• YOLOv9 had a slightly higher misclassification rate of 6% in "Healthy" class. • YOLOv9 has higher percentage of false negatives in "SunScaled" than YOLOv8 at 11%.

Accuracy: Based on the confusion matrix and the equations Eq. (1), Eq. (2), Eq. (3) and Eq. (4), we calculate the performance metrics illustrated in Table 3. As results, YOLOv8 demonstrates the same accuracy for "Healthy" in comparison to YOLOv9 (94.2%), however for "Splitting", YOLOv9 obtains higher accuracy (90%) than YOLOv8 (84.8%). For "BlossomEndRot" and "SunScaled" classes, YOLOv9 slightly outperforms YOLOv8 in accuracy by 1% and 0.2%, respectively. Therefore, the accuracy of YOLOv8 and YOLOv9 is equal to 92% and 93.6%, respectively. Table 4 displays the strengths and weaknesses of both models.

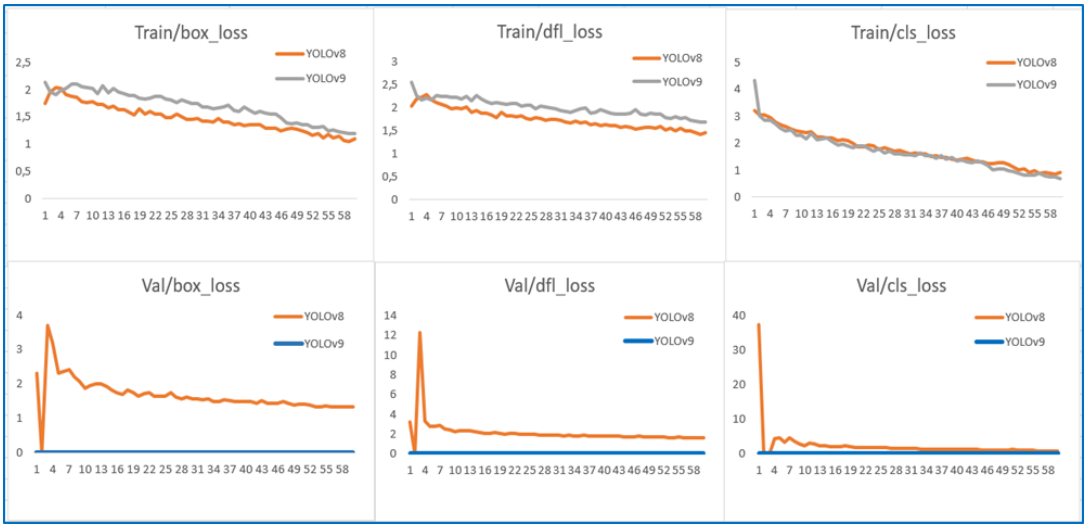

Fig. 12. Training and validation function loss comparison between YOLOv8 and YOLOv9

Loss function: Fig. 12 displays three different types of loss functions tracked during both the training and validation stages for YOLOv8 and YOLOv9 models:

- ***Boxing Loss (box_loss)***: Measures the model's ability to accurately locate and size an object using bounding boxes.
- ***Objectness Loss (dfl_loss)***: Indicates the likelihood of an object existing within a specific area of the image. Higher values suggest a greater chance of an object being present.
- ***Classification Loss (cls_loss)***: Evaluates the model's accuracy in classifying the type of object detected.

For both models, the loss values steadily decrease across epochs in both the training and validation steps. This signifies that both YOLOv8 and YOLOv9 are learning and improving their fitting to the training data. Notably, the steeper initial drops in the curves suggest faster initial learning, which gradually slows as the models fine-tune their performance.

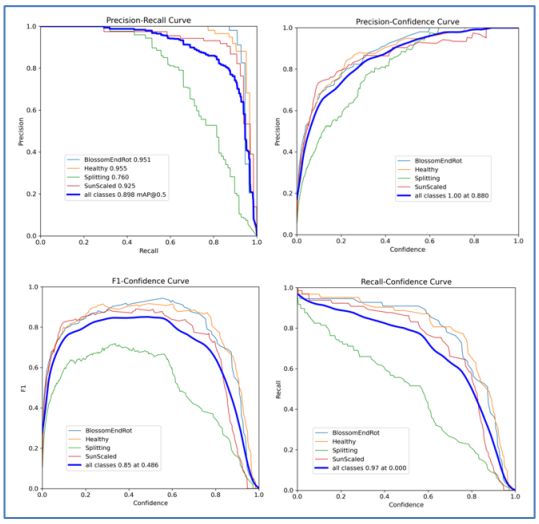

Fig. 13. Performance metrics after validation in YOLOv8

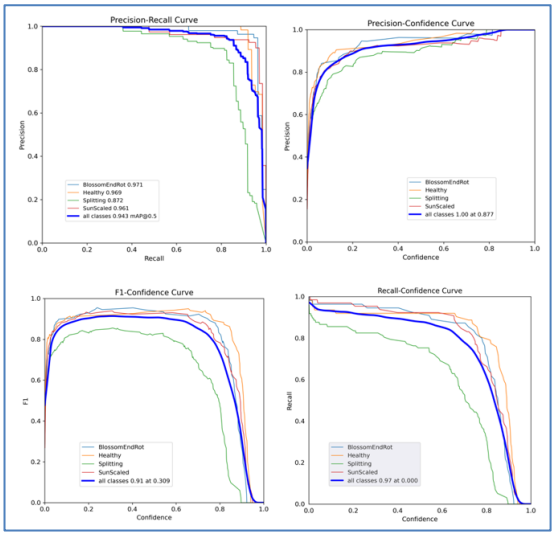

Fig. 14. Performance metrics after validation in YOLOv9

The YOLOv9 loss curves exhibit fewer fluctuations compared to YOLOv8, implying more stable training and potentially better generalization to unseen data. YOLOv9 achieves lower overall training loss compared to YOLOv8, suggesting that it has learned the training data more effectively and performs better on this specific task. This is further supported by the zero-validation loss observed in YOLOv9, which might indicate better alignment between its trained model and unseen data. Overall, YOLOv9 demonstrates improved learning, stability, and performance compared to YOLOv8 in the context of this specific tomato disease detection task, based on the analysis of loss functions.

Validation step: Having completed the initial step, we moved on to the validation phase, where we assessed the performance of the model. Fig. 13 and 14 present the precision, recall, F1-score, and precision-recall curve for both YOLOv8 and YOLOv9 models, respectively. These metrics were obtained after validation. These values are increased in their average values compared to the last training epoch (60). While both models exhibit improvements in these metrics, YOLOv9 consistently demonstrates superior performance compared to YOLOv8 across all evaluated categories. In other words, YOLOv9 achieves higher precision, recall, F1-score, and a better precision-recall curve across the entire range of recall values.

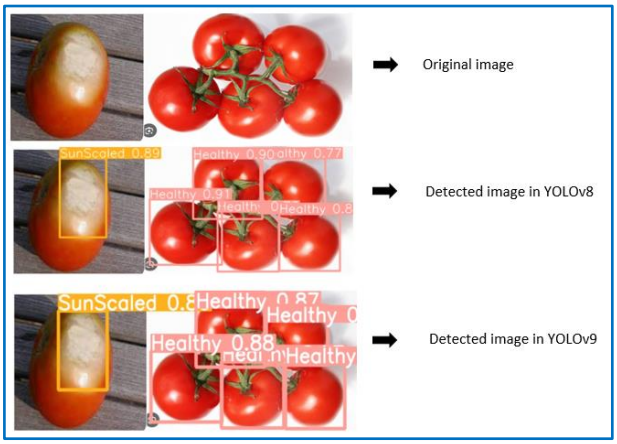

Fig. 15. Detection image in YOLOv8 and YOLOv9

Evaluation using the proposed dataset: The final step involved testing the model on the proposed dataset. We provided an original image containing various object classes, and the model's performance was evaluated based on its ability to accurately detect these classes. Fig. 15 showcases object detection results on an original image using both YOLOv8 and YOLOv9 models. While both models successfully detect SunScaled and Healthy objects with consistent precision for SunScaled at 0.89, YOLOv9 exhibits slightly more consistent precision for "healthy" objects (0.87 and 0.88) compared to YOLOv8's varying precision (0.77, 0.9, 0.8).

4. Conclusion

Our study explored YOLOv9's potential, revealing its ability to achieve higher accuracy than YOLOv8 in object detection. While YOLOv8 trains faster due to its simpler design, YOLOv9 demonstrates improved training efficiency, making it a strong contender for diverse applications that prioritize accuracy. Our experiments using a tomato disease detection dataset revealed that YOLOv8 can be trained with more epochs and larger batch sizes, but YOLOv9 achieves better precision, recall, and mAP despite requiring more memory. After 60 training epochs, YOLOv9 achieved a precision of 93.6%, exceeding YOLOv8's 92% by 1.6%. However, YOLOv8 completed training in 0.44 hours less than YOLOv9. Additionally, YOLOv9 demonstrated higher accuracy for most classes, achieving 90% for "Splitting" compared to YOLOv8's 84.8%. It has an accuracy performance similar to YOLOv8 for the "Healthy" category at 94.2%. Also, YOLOv9 maintains superior performance in validation metrics (precision, recall, mAP) compared to YOLOv8. The "Healthy" class served as an exception, where YOLOv8 achieved slightly higher recall (94.2%) and mAP (43.5%) compared to YOLOv9 (94.0% and 43.3%, respectively). YOLOv8 exhibited higher precision in detecting the "SunScaled" class compared to YOLOv9, with a difference of 5.8%. Although, both models detect objects successfully, YOLOv9 shows slightly more consistent accuracy, suggesting potential advantages in specific tasks. This analysis positions YOLOv9 as a promising contender in object detection, offering a balance between efficiency and accuracy for various applications.

In future work, we test YOLOv9 on a wider range of object detection tasks and datasets. In addition, we minimize the number of images in "Background" category to address potential class imbalance issues. Then, we optimize hyper-parameters for both YOLOv8 and YOLOv9 to potentially improve their performance and identify their relative strengths.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] J. Redmon & Ali Farhadi. "YOLOv3: An Incremental Improvement. Computer Vision and Pattern Recognition", in Computer Vision and Pattern Recognition , 2018.

[2] Lu Tan, Tianran Huangfu, Liyao Wu & Wenying Chen. "Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification", in BMC Medical Informatics and Decision Making, 2021, vol. 21, p. 324.

[3] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection", in Computer Vision and Pattern Recognition, 2020.

[4] Muhammad Abdullah. "YOLO Working principle, difference between its different Variants and Versions", in Medium, 2023, https://medium.com/ @muhabd51/ yolo-working-principle-difference-between-its-ddifferent-variants-and-versions-95b8ad7b95ab.

[5] Zhongqiang Luo, Chenghao Wang, Ziyuan Qi, Chunlan Luo, "A_YOLOv8s: A lightweight-attention YOLOv8s for oil leakage detection in power transformers", in Alexandria Engineering Journal, 2024, vol. 92, pp. 82--91.

[6] Chien-Yao Wang, I-Hau Yeh, Hong-Yuan Mark Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information", in Computer Vision and Pattern Recognition, arXiv:2402.13616, 2024.

[7] Haitong Lou, Xuehu Duan, Junmei Guo, Haiying Liu, Jason Gu, Lingyun Bi, Haonan Chen, "DC-YOLOv8: Small size Object detection algorithm based on camera sensor", 2023, doi:10.20944/preprints202304.0124.v1

[8] Moahaimen Talib, Ahmed H. Y. Al-Noori, Jameelah Suad, "YOLOv8-CAB: Improved YOLOv8 for Real-time object detection", in Karbala International Journal of Modern Science, 2024.

[9] Savan Agrawal, "Helmet Detection YOLOv3: A YOLOv3 detector, which can detect helmet", in Kaggle, 2020.

[10] Krunal Patel, Vrajesh Patel, Vikrant Prajapati, Darshak Chauhan, Adil Haji, Sheshang Degadwala, "Safety Helmet Detection Using YOLOV8" in International Conference on Pervasive Computing and Social Networking (ICPCSN), 2023, DOI: 10.1109/ICPCSN58827.2023. 00012.

[11] S. Roy, S. Mukherjee, S. K. Ghosh, "YOLO Based Real-Time Object Detection for Video Surveillance", in 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), 2019, pp. 1537-1542.

[12] G. R. Goswami, A. K. Singh, H. K. Bajaj, "Real-Time Object Detection for Security Applications Using YOLOv3", in 4th International Conference on Signal Processing, Computing and Control, 2019, pp. 147-152, https://ieeexplore.ieee. org/document/8918322.

[13] X. Zhou, Y. Yao, G. Liu, Z. Sun, Y. Hu, "Deep learning for fine-grained disease classification of tomato leaves", in IEEE Access, 2019, 107386-107400, https://ieeexplore.ieee.org/ document/

8804202.

[14] Santosh Adhikari, Bikesh Shrestha, Bibek Baiju, Er. Saban Kumar K.C, “Tomato Plant Diseases Detection System using Image Processing”, in 1st Kantipur Engineering College, Dhapakhel, Lalitpur Conference Proceedings, 2018.

[15] Md Ershadul Haque, Ashikur Rahman, Iftekhar Junaeid, Samiul Ul Hoque, Manoranjan Paul, “Rice Leaf Disease Classification and Detection using YOLOV5”, 2022, arXiv:2209.01579v.

[16] Md. Janibul Alam Soeb, Md. Fahad Jubayer, Tahmina Akanjee Tarin, Muhammad Rashed Al Mamun, Fahim Mahafuz Ruhad, Aney Parven, Nabisab Mujawar Mubarak, Soni Lanka Karri Islam Md. Meftaul, “Tealeaf disease detection and identification based on YOLOv7 (YOLO-T)”, in Scientific Reports, 2023, https://doi.org/10.1038/ s41598-023-33270-4.

[17] Mubashiru Olarewaju Lawal, “Tomato detection based on modified YOLOv3 framework”, in Scientific Reports, 2021, https://doi.org/10. 1038/s41598-021-81216-5.

[18] M. R. Shams, S. A. Sharief, M. H. Abdullah, “Performance Analysis of Deep Learning Techniques for Rice Disease Detection”, in IEEE Access, 2021, 13132-13141, https://ieeexplore. ieee.org/ document/9414761.

[19] Tran Quang Vinh, Haewon Byeon, Enhancing “Alzheimer's Disease Diagnosis: The Efficacy of the YOLO Algorithm Model”, in International Journal of Advanced Computer Science and Applications(IJACSA), 2023, vol. 14, Issue 11, DOI: 10.14569/IJACSA.2023.0141182.