International Journal of Innovative Research and Scientific Studies, 9(6) 2026, pages: 119-142

IJIRSS
International Journal of Innovative Research & Scientific Studies

ISSN: 2617-6548

URL: www.ijirss.com

An efficient YOLO-based framework for multi-class plant disease detection

Khalid Alghamdi

Department of Electrical and Computer Engineering, King Abdulaziz University, Jeddah, Makkah Region, Saudi Arabia.

(*Email: kalghamdi0418@stu.kau.edu.sa*)

Abstract

Plant health plays a critical role in agriculture, climate balance, and economic stability. However, plant diseases caused by bacteria, fungi, and viruses can significantly reduce crop productivity if not detected early. Traditional manual inspection methods are time-consuming, labor-intensive, and prone to human error, especially in large-scale farming. To address these challenges, this study proposes an automated and accurate plant disease detection system using deep learning-based object detection models for early disease diagnosis in agriculture. A publicly available dataset containing 38 different plant leaf diseases annotated in You Only Look Once (YOLO) format is used, along with a standardized preprocessing pipeline to ensure data quality and consistency. Three modern architectures: YOLOv8, YOLOv11, and YOLOv26 were trained and evaluated under identical conditions using the Ultralytics framework on Google Colab. Experimental results show that YOLOv11 achieves the highest accuracy in terms of precision, recall, and mean Average Precision (mAP), while YOLOv8 provides the fastest inference speed with lower computational complexity. Based on the results, the study concludes that YOLO-based models show great potential for plant disease detection, with YOLOv11 offering superior detection accuracy among the evaluated models. The practical implications of these findings lie in the potential for precision agriculture to monitor diseases in real-time, minimize crop losses, and aid in timely decision-making for farmers and agricultural stakeholders.

Keywords: Computer Vision, Convolution neural networks, Deep Learning, Object Detection, Plant disease detection, Precision Agriculture, YOLO.

DOI: 10.53894/ijirss.v9i6.11771
Funding: This study received no specific financial support.
History: Received: 14 April 2026 / **Revised:** 1 June 2026 / **Accepted:** 8 June 2026 / **Published:** 23 June 2026
Copyright: © 2026 by the author. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).
Competing Interests: The author declares that there are no conflicts of interests regarding the publication of this paper.
Transparency: The author confirms that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.
Publisher: Innovative Research Publishing

1. Introduction

Agriculture plays a vital role in maintaining global food security and supporting economic growth. Healthy crops are necessary for the production of enough food to meet the needs of the growing population. However, plant diseases have always remained a major challenge for farmers because they significantly reduce both crop yield and quality. If diseases are not identified and managed in the early stages, they can spread quickly and cause serious damage to crops [1].

Traditionally, plant diseases have been identified through manual inspection by farmers or agricultural experts. However, manual disease detection has several limitations. This process is time-consuming and may lead to inaccurate results due to human error or misinterpretation of symptoms. In addition, the inspection of large agricultural fields manually is difficult and inefficient, especially when quick disease detection is required to prevent crop loss. With recent advancements in artificial intelligence and computer vision, automated plant disease detection systems have gained significant attention. Machine learning and deep learning techniques are increasingly used to analyze plant leaf images and identify diseases automatically [2, 3].

Convolutional Neural Network (CNN) based classification models have achieved high accuracy in identifying plant diseases from leaf images. However, these models usually classify the entire image and cannot determine the exact location of disease symptoms. In many cases, plant diseases appear as small spots or lesions on leaves, which require accurate localization. For this reason, object detection techniques are more suitable for plant disease detection. Several object detection algorithms, including Faster Region-based Convolutional Neural Network (R-CNN), Single Shot Detector (SSD), and You Only Look Once (YOLO), are widely used in computer vision applications. Among them, the YOLO architecture is particularly popular because of its high detection speed and strong accuracy. YOLO performs detection in a single stage, which enables real-time image processing [4].

This study ensures a comparative analysis of multiple YOLO-based architectures for plant disease detection. The research aims to identify plant diseases by analyzing leaf images using advanced object detection models. The study focuses on the detection of 38 different plant leaf diseases using modern YOLO models, including YOLOv8, YOLOv11, and YOLOv26. These models are selected because of their strong performance in object detection tasks and their ability to identify small disease symptoms in images. To determine the most effective model, the performance of each architecture is evaluated using standard evaluation metrics such as precision, recall, F1-score, and mean Average Precision (mAP). This study makes several important contributions to the field of plant disease detection.

- First, it provides a comparative evaluation of three recent YOLO-based models, helping to understand their relative strengths and limitations.
- Second, it develops a multi-class detection framework capable of identifying diseases across different crop types, making it more suitable for real-world agricultural applications.
- Third, it highlights the advantage of object detection over traditional classification by enabling precise localization of infected regions in leaf images.
- Fourth, the study ensures a comprehensive evaluation by using multiple performance metrics, including accuracy, precision, recall, and mAP.
- Finally, it contributes to the advancement of smart agriculture by integrating deep learning techniques for efficient and automated plant disease diagnosis.

The rest of this article is structured as follows: Section 2 presents the literature review, Section 3 describes the methodology, Section 4 provides the results and their analysis, Section 5 discusses the findings, and the final section concludes the study along with directions for future work.

2. Literature Review

Plant diseases greatly affect crop production and global food security. Detecting diseases at an early stage is important to reduce crop damage and improve agricultural sustainability. Plant diseases were identified through manual inspection by agricultural experts. However, this method is slow, dependent on human experience, and its application is difficult in large farming areas.

2.1. Traditional Approaches

Recent progress in computer technology and artificial intelligence has made it possible to detect plant diseases automatically with the help of machine learning and deep learning methods. In early stages of research, plant disease detection mainly depended on traditional image processing and machine learning algorithms such as Support Vector Machines (SVM), K-Nearest Neighbor (KNN), Decision Trees, and Random Forest [5-9]. These methods usually require researchers to manually extract features from images, including color, texture, and shape characteristics. Although these techniques produced reasonably good results in some cases, they had several drawbacks. Their performance often depended heavily on lighting conditions and image quality, and background noise could easily manipulate the results. In addition, the need for manual feature extraction made the process time-consuming and dependent on expert knowledge. Because of these limitations, researchers later started exploring more advanced deep learning approaches that can automatically learn features from images.

2.2. Deep Learning Approaches

Deep learning methods have significantly improved the performance of plant disease detection systems. Among these methods, CNNs are widely used because they can automatically learn important features from images without manual feature extraction. One of the earliest studies in this area was conducted by Mohanty, et al. [3] who applied CNN architectures such as AlexNet and GoogleNet to the PlantVillage dataset containing more than 54,000 leaf images from 14 crop species and 26 diseases. The study reported classification accuracy above 99% under controlled conditions, highlighting the strong potential of deep learning for automated plant disease identification. Sladojevic, et al. [10] proposed a deep neural network model for automatic plant disease recognition using leaf images. Their system was able to

identify 13 different plant diseases and achieved high accuracy on controlled datasets. However, the approach required preprocessed images and did not identify the specific infected regions on the leaf surface. Fuentes, et al. [11] developed a deep learning–based system for detecting tomato diseases and pests using object detection models such as Faster R-CNN and SSD. Unlike earlier research that relied on laboratory images, this study used images collected from real agricultural environments. The proposed model successfully detected multiple diseases simultaneously with high accuracy. However, the computational complexity of Faster R-CNN limited its ability to operate efficiently in real-time applications. Ferentinos [12] applied deep convolutional neural networks to detect plant diseases across several crop species. Different CNN architectures were trained using the PlantVillage dataset, and the proposed system achieved an overall accuracy of 99.53%. Although the results were promising, the research mainly focused on classification and did not provide localization of disease symptoms on plant leaves. Too, et al. [13] conducted a comparative study of several deep learning architectures, including VGG16, ResNet50, DenseNet121, and Inception models. Their results showed that DenseNet-based models achieved better performance due to improved feature propagation and reduced overfitting. The study also confirmed that deep CNN models generally outperform traditional machine learning methods in plant disease detection tasks.

2.3. Object Detection Approaches

In recent years, object detection models such as the YOLO family have become popular for plant disease detection. Unlike traditional CNN classification models, YOLO models can detect and localize multiple disease regions in real time. This makes them suitable for practical agricultural applications where both speed and accuracy are important Aldakheel, et al. [14]. Ali, et al. [15] introduced YOLOv4, which improved detection accuracy while maintaining high inference speed. Although initially designed for general object detection, YOLOv4 has been widely applied in agriculture, including plant disease detection, due to its improved mean average precision (mAP) and efficiency compared to previous YOLO versions.

Shill and Rahman [16] applied YOLOv3 to detect multiple disease symptoms in tomato leaves. Their system outperformed traditional CNN classifiers and demonstrated the advantages of object detection for localizing disease regions. Ahmed and Abd-Elkawy [17] developed a YOLOv4-based framework for tomato disease detection, achieving an accuracy of approximately 96%, further confirming the suitability of YOLO models for real-time plant disease detection. YOLOv5, developed by Jocher, et al. [18] introduced improvements in both training efficiency and inference speed, making it widely used in agricultural research. Following this, Zayani, et al. [19] applied YOLO-based models to detect paddy leaf diseases, demonstrating effective identification of multiple disease spots in a single image. More recent studies have explored YOLOv5, YOLOv7, and YOLOv8 for various crops. Wang and Liu [20] developed a YOLOv5-based system for real-time field monitoring of plant diseases, achieving both high accuracy and fast inference. YOLOv8 was introduced later, offering higher accuracy and better computational efficiency, and has been applied to plant disease detection and crop monitoring. YOLOv8 further improved disease detection performance Önler and Köycü [21]. Abid, et al. [22] applied YOLOv8 to detect wheat leaf diseases, achieving faster inference and higher accuracy than earlier YOLO models. Ghafar, et al. [23] adapted YOLOv8 for corn disease detection, adding feature enhancement modules to improve detection of small disease spots. Yang, et al. [24] evaluated multiple YOLO models for citrus leaf diseases and found that YOLOv8 achieved the highest detection accuracy. Na, et al. [25] introduced an enhanced YOLOv8 model with attention mechanisms to detect small lesions more effectively. Lightweight and specialized YOLO models have also been developed for practical deployment. Abulizi, et al. [26] created YOLOv8-tiny for rice disease detection, optimized for edge devices like smartphones and drones. Chen, et al. [27] proposed a multi-crop disease detection system that emphasizes real-world applicability across several crop species. Most recently, Wang, et al. [28] and Zhu, et al. [29] developed YOLOv8-CMS and multi-scale feature extraction models for citrus and general crops, achieving higher accuracy and robust detection under varying environmental conditions. Table 1 represents previous approaches used for plant disease detection.

Table 1.
Previous Studies on Plant Disease Detection.

Author & References	Year	Model	Dataset	Diseases	Accuracy
Mohanty, et al. [3]	2016	CNN	PlantVillage	26	99.35%
Singh, et al. [6]	2022	Hybrid CNN + Bayesian SVM + Random Forest	Plant leaf images	Multiple	High accuracy
Sahu and Pandey [7]	2023	Hybrid multiclass SVM + Fuzzy C-means	Plant leaf images	Multiple	High accuracy
Alhwaiti, et al. [8]	2023	Histogram of Oriented Gradients (HOG) + SVM	Tomato leaves	Late blight	High accuracy
Sai, et al. [9]	2019	SVM	Leaf images	Unhealthy leaf	High accuracy
Sladojevic, et al. [10]	2016	CNN	Leaf images	13	96%
Fuentes, et al. [11]	2017	Faster R-CNN	Tomato dataset	9	87%
Ferentinos [12]	2018	Deep CNN	Multi-crop	58 classes	99.53%
Too, et al. [13]	2019	DenseNet	PlantVillage	38	99%
Aldakheel, et al. [14]	2024	YOLOv4	Plant leaves	Multiple	High accuracy

Ali, et al. [15]	2024	YOLOv8	Citrus dataset	3	96.1% mAP
Shill and Rahman [16]	2021	YOLOv3 / YOLOv4	Plant leaf images	Multiple	High accuracy
Ahmed and Abd-Elkawy [17]	2024	YOLOv5 / YOLOv8	Tomato leaves	Multiple	High accuracy
Zayani, et al. [19]	2024	YOLOv8	Tomato leaves	Multiple	High accuracy
Wang and Liu [20]	2024	YOLOv8	Greenhouse vegetables	Multiple	High accuracy
Önler and Köycü [21]	2024	YOLOv8	Wheat leaves	Powdery mildew	High accuracy
Abid, et al. [22]	2024	YOLOv8	Paddy leaves	Multiple	High accuracy
Ghafar, et al. [23]	2024	YOLOv8	Crop leaves	Multiple	High accuracy
Yang, et al. [24]	2024	YOLOv8	Corn leaves	Spot disease	High accuracy
Na, et al. [25]	2023	YOLOv8	Wheat leaves	Multiple	High accuracy
Abulizi, et al. [26]	2025	YOLOv9	Tomato leaves	Multiple	High accuracy
Chen, et al. [27]	2024	YOLOv8-ACCW	Grape leaves	Multiple	High accuracy
Wang, et al. [28]	2025	YOLOv8	Rice leaves	Multi-scale diseases	High accuracy
Zhu, et al. [29]	2025	YOLOv8-CMS	Citrus leaves	4	High accuracy

Although deep learning has enhanced plant disease detection, some challenges remain. Many studies use datasets with only a few diseases and often test only one model instead of comparing different YOLO versions. Detection of small disease spots on leaves is difficult because they occupy very small areas in images. Therefore, this study compares multiple YOLO models to detect 38 different leaf diseases and improve detection performance in disease diagnosis.

3. Methodology

In this section, the methodology of the development of an automated plant disease detection system based on deep learning is outlined and detailed. It explains the study's dataset, the general research process, and the preprocessing steps used to prepare data to be used in training. Additionally, a detailed discussion is given on the architecture and selection of YOLO-based object detection models such as YOLOv8, YOLOv11, and YOLOv26. The overall proposed methodology is represented in Figure 1.

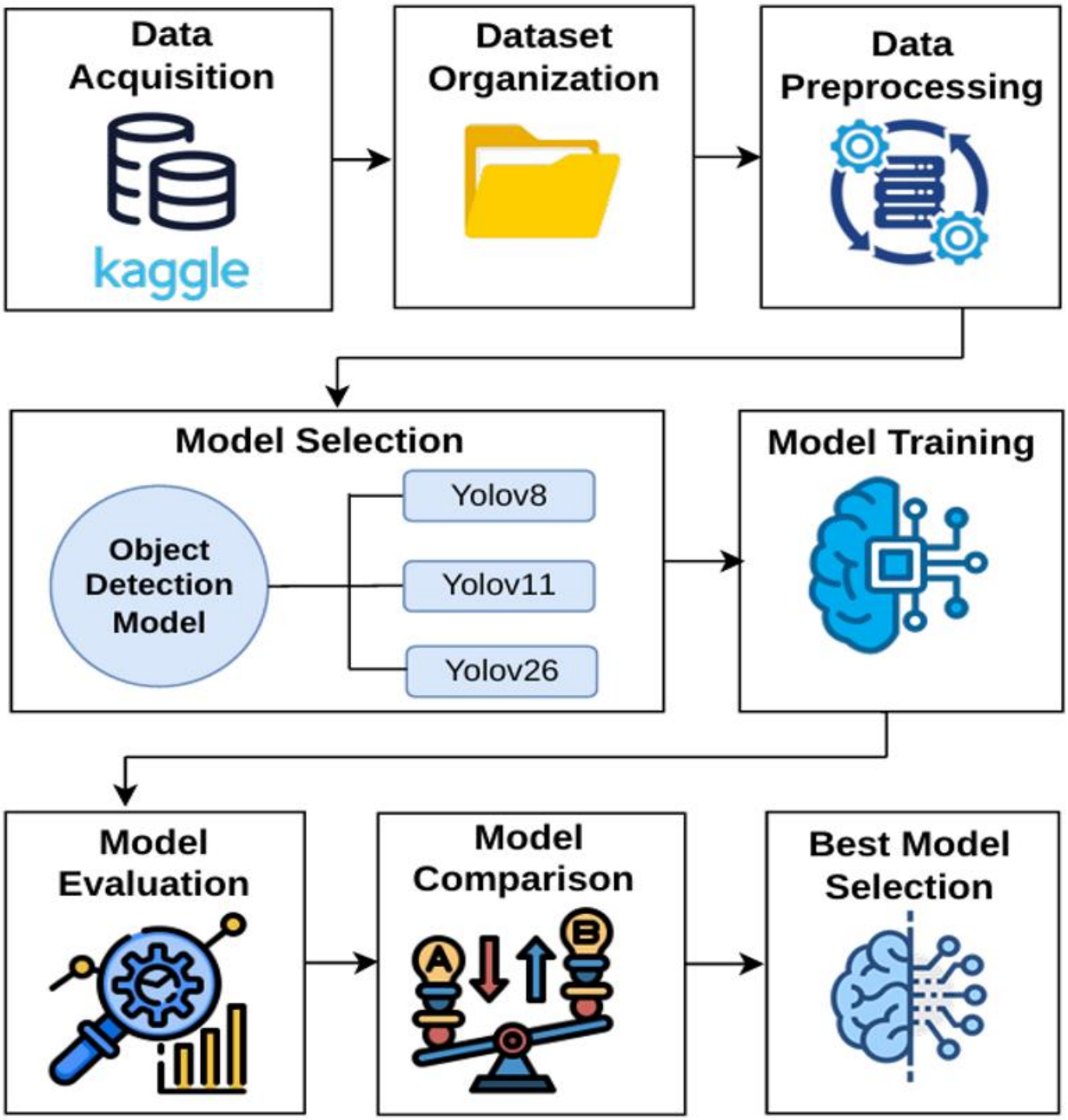


Figure 1.
Proposed Research Methodology.

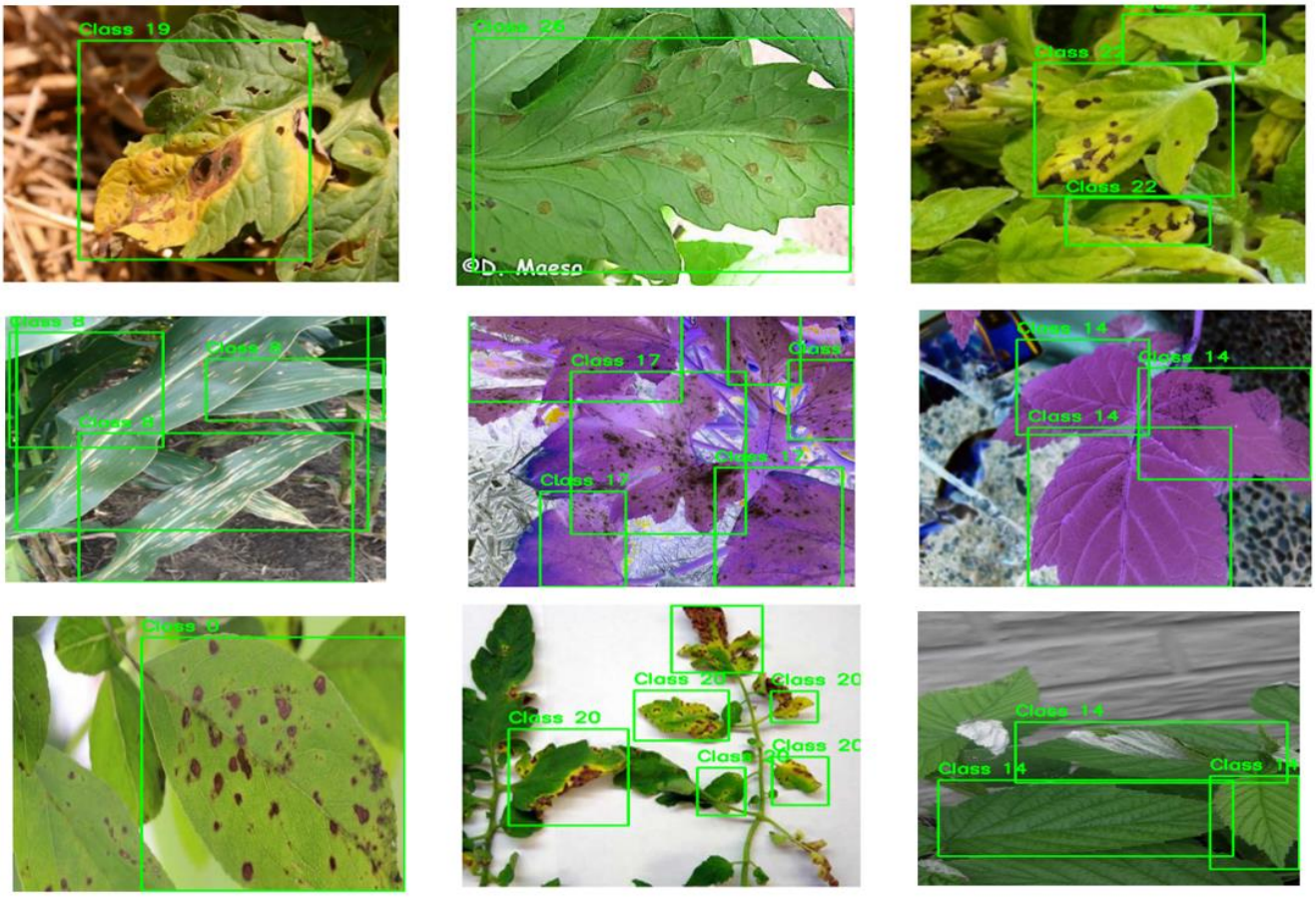


Figure 2.
Sample images from the dataset.

3.1. Dataset Description

For this study, we obtained a publicly accessible dataset (plant disease detection dataset) from Kaggle [30]. The dataset comprises 2569 images of 13 different plant species, as shown in Figure 2. Each image is labeled as either healthy or diseased and is assigned to one of thirty-eight distinct classes. The dataset includes multiple crop types with both healthy and diseased leaf categories. It includes apple (scab and rust), bell pepper (leaf spot), corn (gray leaf spot, leaf blight, and rust), potato (early and late blight), and grape (black rot). Additionally, several healthy leaf classes, such as blueberry, cherry, peach, raspberry, soybean, squash, and strawberry, are included. The tomato category is the most diverse, covering multiple diseases, including early blight, septoria leaf spot, bacterial spot, late blight, mosaic virus, yellow leaf virus, mold, and two-spotted spider mites. Table 2 represents overall description of dataset.

Table 2.
Crop Dataset Description.

Crop	Leaf Condition / Disease Type
Apple	Healthy, Scab, Rust
Bell Pepper	Healthy, Leaf Spot
Blueberry	Healthy
Cherry	Healthy
Corn	Gray Leaf Spot, Leaf Blight, Rust
Peach	Healthy
Potato	Healthy, Early Blight, Late Blight
Raspberry	Healthy
Soybean	Healthy
Squash	Powdery Mildew
Strawberry	Healthy
Tomato	Healthy, Early Blight, Septoria Leaf Spot, Bacterial Spot, Late Blight, Mosaic Virus, Yellow Leaf Virus, Mold, Two-Spotted Spider Mites
Grape	Healthy, Black Rot

The dataset is divided into a training set and a validation set. Out of 2569 images, the training data consisted of 2328 images, while 239 images were used for validation. This division ensures that the model is evaluated on unseen data,

improving generalization capability. This dataset is a useful tool for object recognition and image classification tasks because it contains 8,851 labelled objects in total. Each image is associated with a corresponding annotation file in YOLO format, which contains class labels and bounding box coordinates.

3.2. Data Preprocessing and Verification

Data preprocessing was performed to ensure dataset consistency and improve model performance. All images were resized to 416 × 416 pixels to match YOLO input requirements. Data cleaning was conducted to remove duplicate images, incorrect annotations, and missing labels. Bounding box coordinates were normalized to a range of 0 to 1, as required by the YOLO format. Finally, data verification was carried out by visually inspecting sample images with bounding boxes to ensure annotation accuracy. These steps resulted in a clean and reliable dataset for training.

3.3. Data Annotation Format

The dataset is in the YOLO annotation format where the objects are described by a single line in a text file. The format is defined as;

<class_id> <x_center> <y_center> <width> <height>

Where:

- *class_id* represents the category of the object.
- *x_center* and *y_center* are the center of the bounding box.
- *width* and *height* are the size of the bounding box.

Every value is normalized in relation to the dimensions of the image. This format makes it possible to process and train object detection models effectively.

3.4. Object Detection Models

In this research, the three object detection models were selected based on the YOLO method, which are YOLOv8, YOLOv11, and YOLOv26, to detect plant diseases. These models are selected based on their architectural design, detection, and precision for real-time image analysis processes.

YOLO was introduced by Redmon, et al. [4]. It was the first prototype to introduce a real-time end-to-end object detection method. The name YOLO means You Only Look Once because it could handle the detection task in a single pass of the network compared to earlier methods which would either use sliding windows and then a classifier that had to be applied to hundreds or thousands of times to each image or where the task was divided into two steps, with the first step detecting possible regions with objects and the second step to run a classifier on the proposals [31]. The *backbone*, *neck*, and *detection head* are the three primary components of the Yolo overall architecture. The *backbone* is responsible for finding featured maps of the input image at different scales. It has convolutional layers and special blocks that identify significant visual information like edges, textures and shapes. The *neck* then aggregates the feature maps obtained from the different stages of the backbone. It boosts feature representation by grouping the information of various scales with operations like up sampling and concatenation. This enables the model to effectively identify objects of various sizes. The *detecting head* generates the final predictions which include confidence scores, class names and bounding box coordinates. Large, medium and small objects are normally detected with multiple heads [32].

1. YOLOv8 [33] was published by Ultralytics, the firm behind YOLOv5, in January 2023. Five scaled variants were offered by YOLOv8: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large). Numerous vision tasks, including object identification, segmentation, pose estimation, tracking, and classification, are supported by YOLOv8.

YOLOv8 consists of three detection head components, which are used to detect objects of various sizes: Small, medium, and large objects as shown in Figure 3. The multi-scale detection feature makes YOLOv8 useful in detection of plant diseases of different sizes [32].

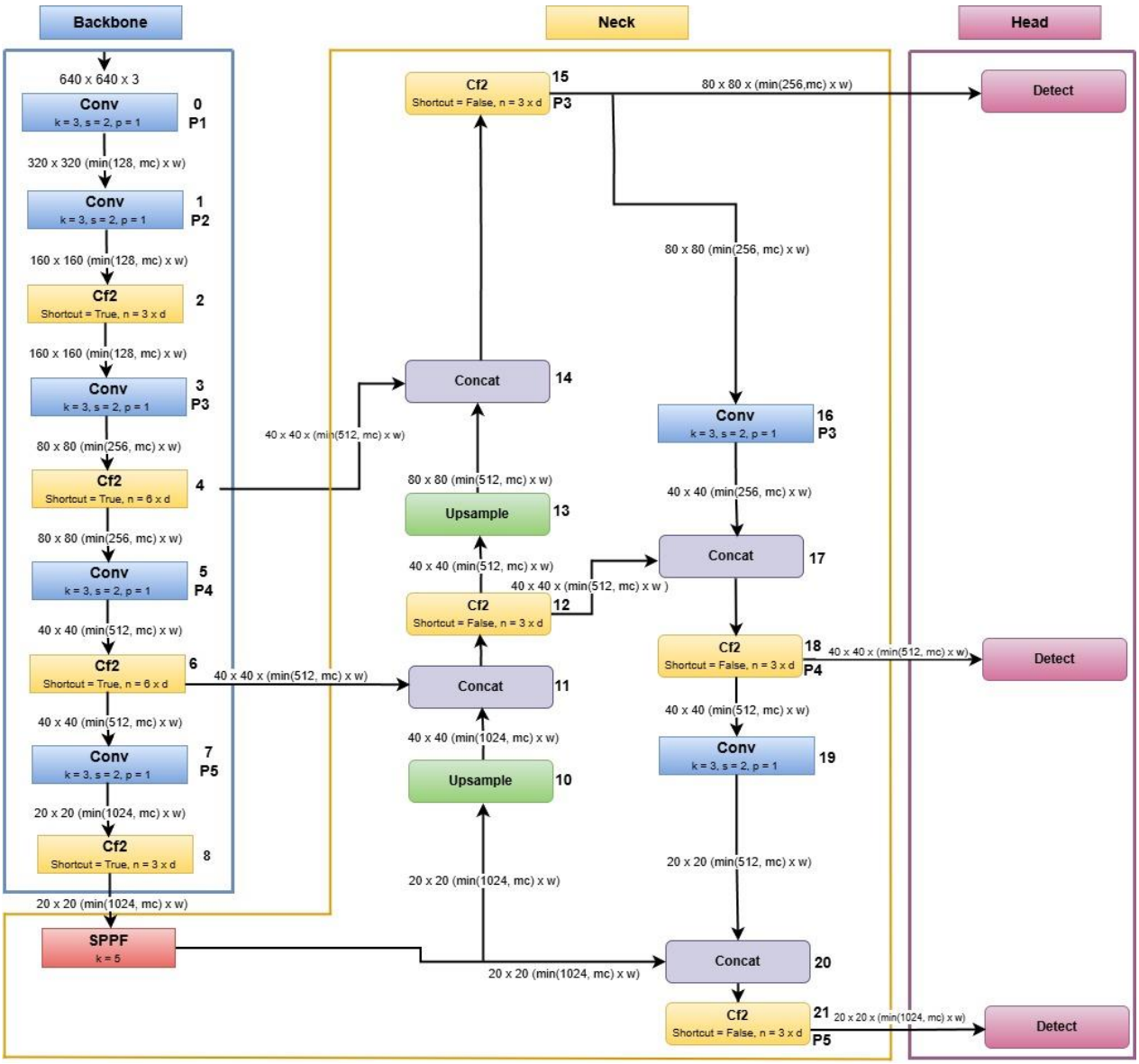


Figure 3.
Yolov8 Architecture.
Source: Hidayatullah, et al. [32].

2. The YOLOv11 [34] network is based on the previous versions of YOLO but adds new developments in terms of feature representation and extraction. It uses a superior block called C3k2, an improvement of the C2f block used in YOLOv8 as represented in Figure 4. The block will enhance efficiency, as it will not use a lot of parameters, while maintaining strong feature learning capability [35].

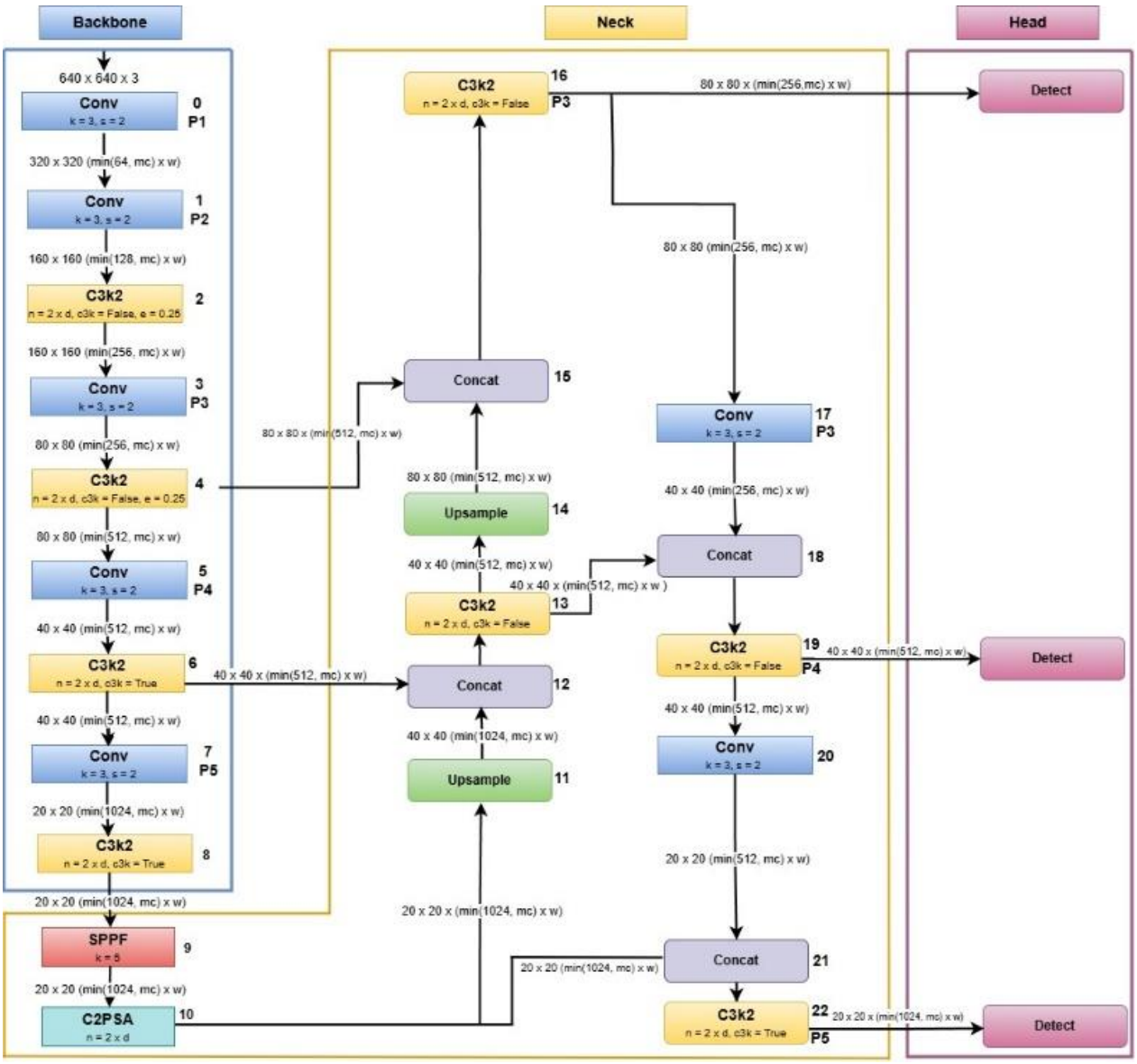


Figure 4.
Yolov11 Architecture
Source: Hidayatullah, et al. [32].

3. YOLOv26 is a more complex and deeper object detector model that has better features representation and detection accuracy. The architecture has more layers and parameters as shown in Figure 5 making the model able to compensate the small-scale visual details. This is especially useful in identifying plant diseases whose differences in texture and color are subtle [36].

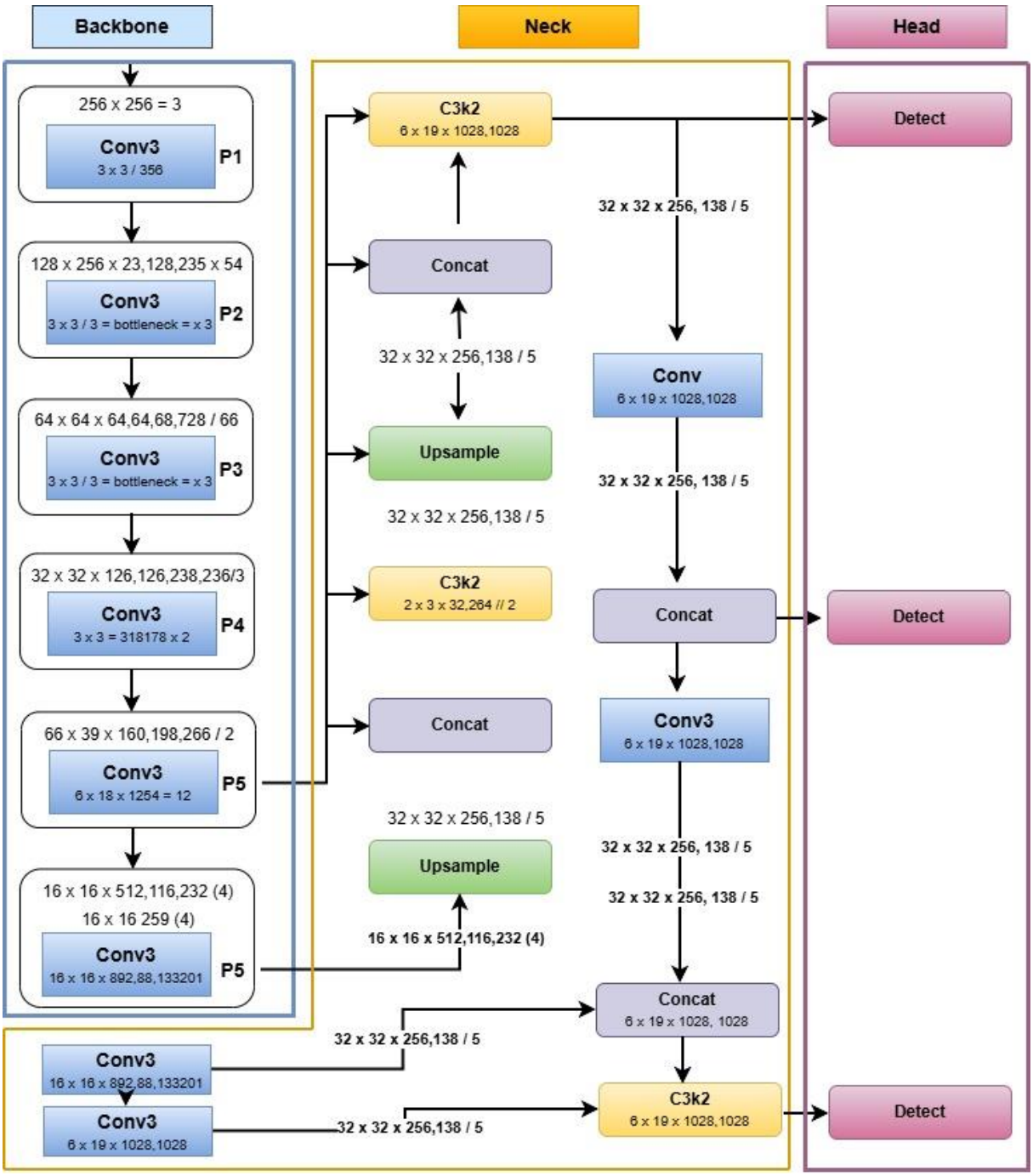


Figure 5.
Yolov26 Architecture.
Source: Sapkota and Karkee [37].

4. Result Analysis

The experimental assessment of the proposed plant disease detection models is presented in this section. Standard object detection criteria, training behavior, and computational efficiency are used to evaluate the performance of various YOLO variations.

4.1. Experimental Setup

The plant disease object detection experiments were performed using Google Colab [38]. The training and evaluation of the model were conducted in an environment with an AMD Ryzen 5 processor with Radeon Graphics card, 8GB RAM and 64 bits Windows 10 operating system. Model development and evaluation were performed using the Ultralytics [37] deep learning framework in Python.

4.2. Evaluation Parameters

All models were trained and tested against the same dataset with the same parameters to ease a fair comparison. A commonly used object detection metric was used to assess the proposed models' performance such as Precision, Recall,

Mean Average Precision and Mean Average Precision at different Intersection over Union (IoU) thresholds, Inference Time, and Model Size.

4.2.1. Precision (P)

This metric calculate *"out of all the objects the model claimed to detect, what fraction were actually correct?"* According to the equation below high precision indicates that the model produces very few false positives.

$$P = \frac{TP}{TP + FP}$$

4.2.2. Recall (R)

"Out of all the objects that actually exist in the image, what fraction did the model find?" can be answered by using this metric. High recall of model indicates that it rarely misses an object and avoids false negatives. The following equation can be used to compute it.

$$R = \frac{TP}{TP + FN}$$

4.2.3. Mean Average Precision (mAP)

To measure the performance of computer vision models, especially for a task like object detection and instance segmentation mAP is the most widely used metric. mAP assesses how well a model finds objects *and* how accurately it positions the bounding box around them as compare to the simplest accuracy, which only checks if an image is classified correctly or not. Because of this, it serves as the main standard for contrasting cutting-edge systems like YOLO26 with earlier iterations.

The calculation of mAP is based on Intersection over Union (IoU) which measures the spatial overlap between the predicted box and the ground truth annotation have been measured by IoU.

$$IoU = \frac{Area(Predicted\ Box\ \cap\ Ground\ Truth)}{Area(Predicted\ Box\ \cup\ Ground\ Truth)}$$

Where the numerator represents the overlap area between the predicted and ground truth bounding boxes, while the denominator represents the total area covered by both boxes .IoU values vary from 0 to 1, with higher values denoting more accurate localization. A prediction is often considered a *"True Positive"* only if the IoU exceeds a specific threshold, such as 0.5 or 0.75.

The Average Precision (AP) is computed from the area under the Precision-Recall curve for each class. The Mean Average Precision (mAP) is then calculated as the mean of AP values across all classes and can be expressed as:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP\ i$$

Where *N* denotes the total number of classes and AP_i represents the Average Precision of the *i-th* class.

The two mAPs that we employ in our study are mAP@0.5 and mAP@0.5:0.95.

- mAP@50: The detection is considered correct if the IoU value is at least 0.50.
- mAP@50-95: This is the average of mAP computed in steps of 0.05 at IoU thresholds ranging from 0.50 to 0.95. This rigorous metric rewards models that achieve high localization accuracy [39].

4.2.4. Inference Time

The amount of computational needed by a trained model to process an input image and generate detection results is referred to as inference time. It is critical for real-time applications like field-level disease monitoring and precision agriculture, and is affected by model complexity and hardware [40]. We use this metric to compare the inference time of the models.

4.2.5. Model Size

Model size [41] refers to the memory usage and complexity of the trained network and is usually measured in megabytes (MB). Smaller models are better suited for mobile platforms and edge devices.

4.3. Loss Functions

Loss functions play a crucial role in training deep learning models by measuring the difference between predicted outputs and ground-truth annotations. During training, the model minimizes these losses to improve both classification and localization performance. YOLO employs three main loss functions: Box Loss, Classification Loss, and Distribution Focal Loss (DFL). These losses collectively optimize object localization, class prediction, and bounding box regression accuracy.

4.3.1. Box Loss

Box Loss evaluates the localization accuracy of predicted bounding boxes with respect to the ground-truth boxes. YOLOv8 utilizes Complete Intersection over Union (CIoU) loss, which considers overlap area, center point distance, and aspect ratio consistency between the predicted and ground-truth boxes.

$$L_{CIoU} = 1 - IoU + \left(\frac{p^2(b, b_{gt})}{c^2}\right) + \propto v$$

where *IoU* is the Intersection over Union, $\rho(b, b_{gt})$ is the distance between center points of predicted and ground truth box, c represents the diagonal length of smallest enclosing box, α denotes the positive trade-off parameter and v is the aspect ratio consistency term [42].

4.3.2. Classification Loss

Classification Loss measures the discrepancy between the predicted class probabilities and the ground-truth class labels. YOLOv8 employs Binary Cross-Entropy (BCE) loss for multi-class classification tasks.

$$L_{cls} = -[\, y \log(p) + (1 - y) \log(1 - p)]$$

where y is the ground-truth label and p is the predicted probability [43].

4.3.3. Distribution Focal Loss (DFL)

Distribution Focal Loss (DFL) enhances bounding box regression by modeling object boundary locations as probability distributions rather than direct coordinate values. This improves localization precision and detection performance.

$$L_{dfl} = -(y_l \log(p_l) + y_r \log(p_r))$$

where p_l and p_r represent the predicted probabilities of neighboring discrete locations, while y_l and y_r denote their corresponding target weights [44].

4.4. Training Hyperparameters

Table 3 represents the hyperparameters utilized during the training of the Yolo models for plant disease object detection. All models were trained and evaluated on the same plant disease dataset using identical hyperparameters to ensure consistency and fairness in comparative analysis.

Table 3.
Hyperparameters for proposed models training.

S. No	Parameter	Value
1	Image size	416 x 416
2	Batch size	16
3	Number of Epochs	50
4	Training Sample	80%
5	Validation Sample	20%
6	Test Images	10
7	Learning rate	0.001
8	Optimizer	adamW

4.5. Yolov8 Results

YOLOv8 was evaluated as the baseline architecture for the plant disease detection models in this study. As presented in the Figure 6 the trained YOLOv8 model consists of 129 layers with approximately 11.15 million parameters and requires 28.7 Giga Floating Point Operations (GFLOPs), making it a lightweight yet efficient detector. After an optimization process and weight fusion, the resulting model has a size of about 22.5MB, which makes it suitable to be deployed in resource-constrained environments, such as mobile and edge devices used in precision agriculture.

```
Model summary: 129 layers, 11,147,210 parameters, 11,147,194 gradients, 28.7 GFLOPs
```

Figure 6.
YOLOv8 model architecture's specifications.

The learning efficiency of YOLOv8 model is depicted in Figure 7. From the box loss, classification loss, and Distribution Focal Losses (DFLs), we can see that they are all monotonically decreasing. The classification loss reduced considerably at the beginning of the training, suggesting that the model learned the features from pretrained weights efficiently and converged quickly.

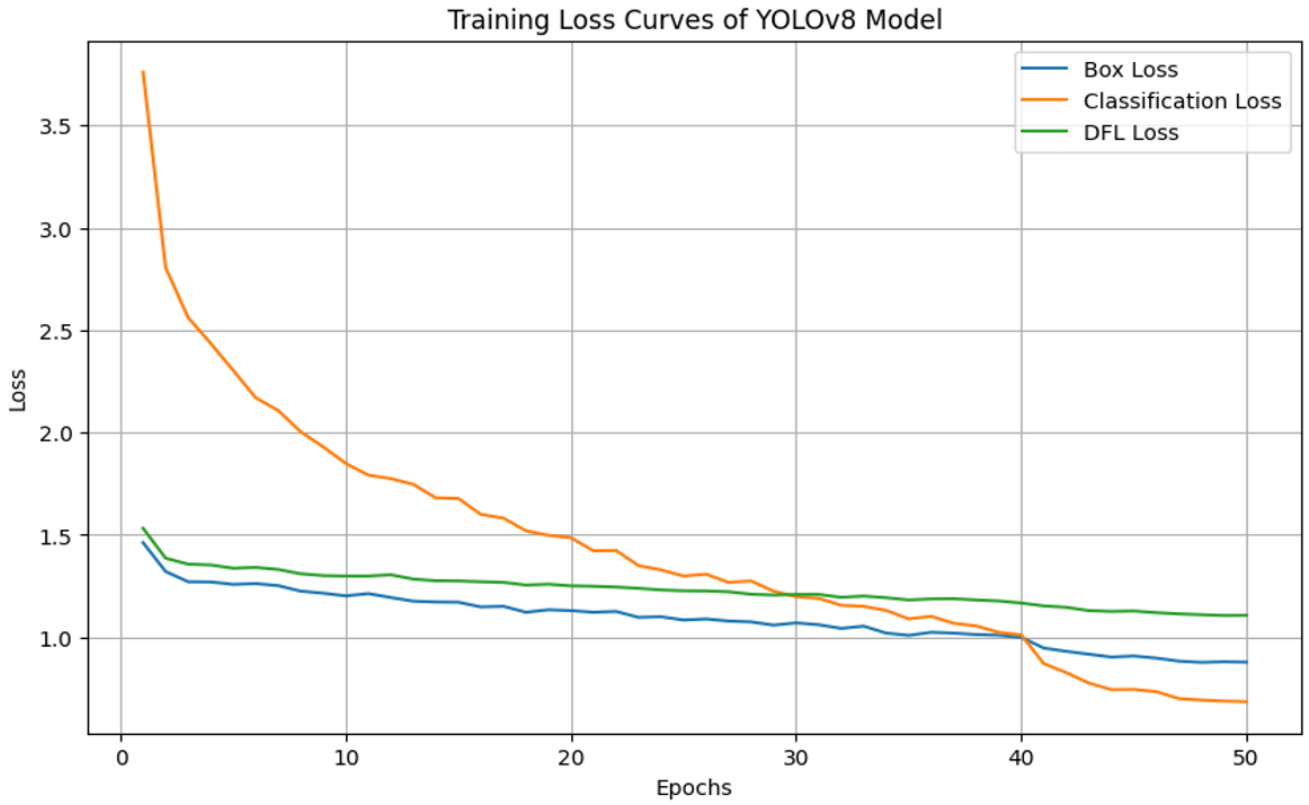
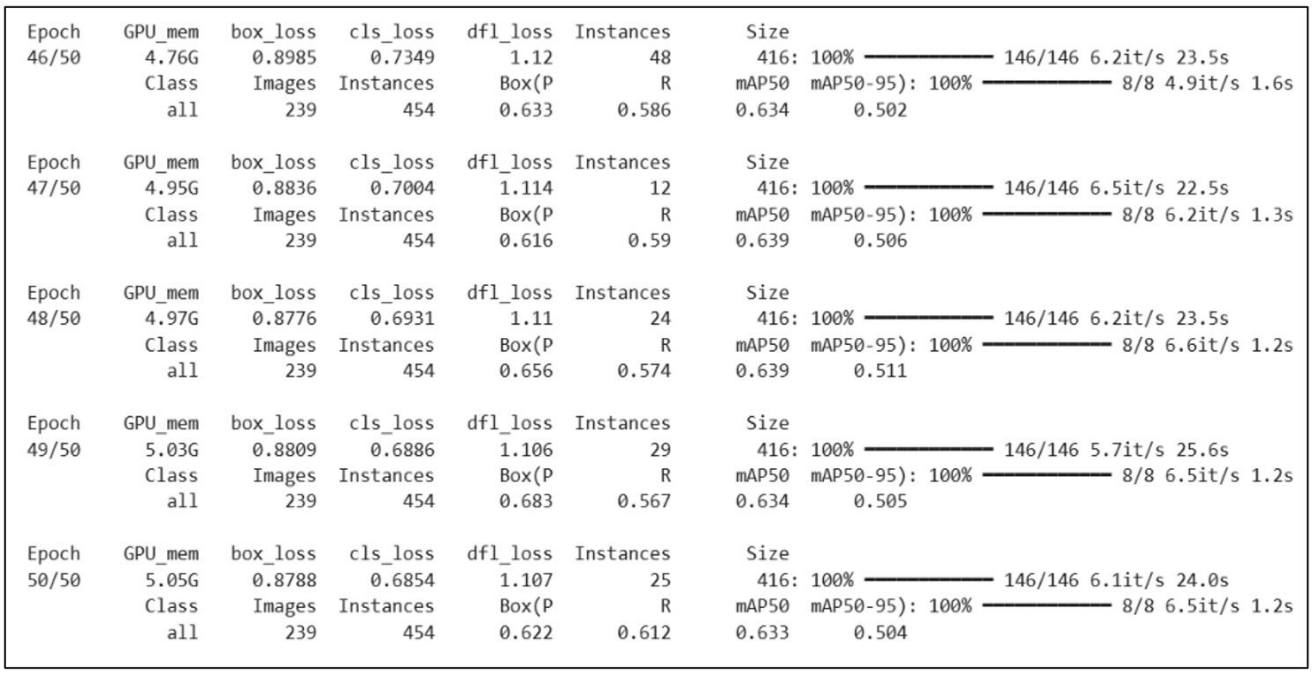


Figure 7.
Training loss curves of YOLOv8 model.

The training and validation performance across the first and last five epochs as presented in the Figures 8 and 9 indicate that the model converged without overfitting.

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
1/50	1.67G	1.462	3.761	1.532	59	416: 100% ━━━━━━━━━━━━ 146/146 5.3it/s 27.5s
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.5it/s 2.3s
	all	239	454	0.392	0.238	0.205 0.163
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
2/50	2.05G	1.321	2.806	1.386	64	416: 100% ━━━━━━━━━━━━ 146/146 5.8it/s 25.2s
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% ━━━━━━━━━━━━ 8/8 4.4it/s 1.8s
	all	239	454	0.245	0.419	0.301 0.228
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
3/50	2.09G	1.271	2.562	1.357	60	416: 100% ━━━━━━━━━━━━ 146/146 6.0it/s 24.5s
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% ━━━━━━━━━━━━ 8/8 6.3it/s 1.3s
	all	239	454	0.433	0.378	0.373 0.288
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
4/50	2.12G	1.27	2.436	1.353	58	416: 100% ━━━━━━━━━━━━ 146/146 6.0it/s 24.3s
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% ━━━━━━━━━━━━ 8/8 6.4it/s 1.3s
	all	239	454	0.543	0.343	0.377 0.286
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
5/50	2.16G	1.258	2.304	1.337	96	416: 100% ━━━━━━━━━━━━ 146/146 6.1it/s 24.1s
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% ━━━━━━━━━━━━ 8/8 4.8it/s 1.7s
	all	239	454	0.39	0.452	0.441 0.342

Figure 8.
Top five epochs performance during yolov8 model training.



```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
46/50      4.76G     0.8985     0.7349       1.12         48        416: 100% ━━━━━━━━━━━━ 146/146 6.2it/s 23.5s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 4.9it/s 1.6s
             all        239        454      0.633      0.586      0.634      0.502

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
47/50      4.95G     0.8836     0.7004      1.114         12        416: 100% ━━━━━━━━━━━━ 146/146 6.5it/s 22.5s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 6.2it/s 1.3s
             all        239        454      0.616       0.59      0.639      0.506

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
48/50      4.97G     0.8776     0.6931       1.11         24        416: 100% ━━━━━━━━━━━━ 146/146 6.2it/s 23.5s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 6.6it/s 1.2s
             all        239        454      0.656      0.574      0.639      0.511

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
49/50      5.03G     0.8809     0.6886       1.106         29        416: 100% ━━━━━━━━━━━━ 146/146 5.7it/s 25.6s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 6.5it/s 1.2s
             all        239        454      0.683      0.567      0.634      0.505

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
50/50      5.05G     0.8788     0.6854       1.107         25        416: 100% ━━━━━━━━━━━━ 146/146 6.1it/s 24.0s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 6.5it/s 1.2s
             all        239        454      0.622      0.612      0.633      0.504
```

Figure 9.
Last five epochs representing the final stage of the yolov8 model training process.

Figure 10 shows the confusion matrix for 38 plant disease classifications performed by the YOLOv8 model. There is a strong dark blue diagonal of the matrix and it indicates high accuracy of correctly identifying true positives. The classes Strawberry leaf and Tomato leaf yellow virus are the best classified (darker blue). Some misclassified light-colored clusters appear on the off-diagonal. The most meaningful misclassification factors are Tomato and Potato leaf disease, due to the high similarity between the lesions in the two classes.

The large number of samples misclassified as “Background” indicates that YOLOv8 sometimes confuses very small disease spots on the leaf. Whilst YOLOv8 provides a strong baseline for object detection, the variation present in the predictions shows that fine-grained classification is still a difficult problem for this dataset.

In terms of performance, YOLOv8 achieved a Precision of 0.558, Recall of 0.622, mAP@0.5 of 0.643, and mAP@0.5:0.95 of 0.513. These findings show that for plant disease object recognition, YOLOv8 strikes a good balance between localization precision and detection accuracy.

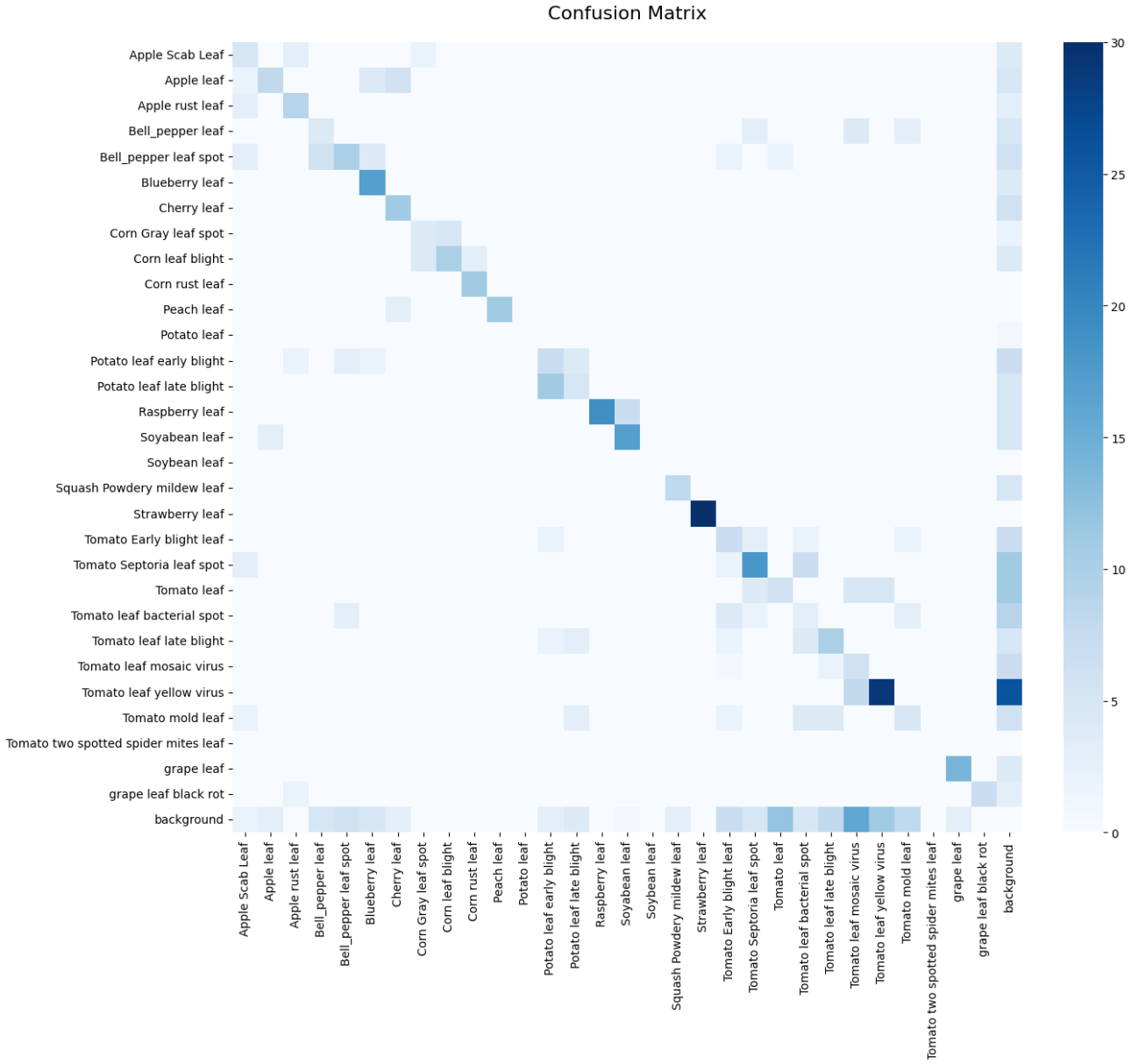


Figure 10.
Confusion Matrix of the YOLOv8 model.

The YOLOv8 model performed exceptionally well in real-time as shown in Figure 11. The results show that for each image the model needed approximately 0.1 ms for preprocessing, 1.9 ms for inference, 0.0 ms for loss computation, and 2.1 ms for post-processing. This high speed of inference of YOLOv8 makes it extremely applicable in real-time applications of agricultural monitoring systems and field level disease detection in agriculture where quick response time is essential.

```
Speed: 0.1ms preprocess, 1.9ms inference, 0.0ms loss, 2.1ms postprocess per image
```

Figure 11.
Inference speed of YOLOv8 model.

4.6. Yolov11 Results

The YOLOv11 model is considerably more complex, it has 358 layers with around 25.33 million parameters and 87.4 GFLOPs which is a very expressive model to detect complex patterns in plant leaves as presented in Figure 12. The final model, which is about 51MB in size after optimization and weight fusion, can be deployed in situations with moderate resource constraints, such as field-level monitoring systems and edge devices.

```
YOLO11l summary: 358 layers, 25,333,610 parameters, 25,333,594 gradients, 87.4 GFLOPs
```

Figure 12.
YOLOv11 model architecture's specifications.

The training behavior as illustrated in Figures 13 and 14 shows a steady decrease in loss values, with all loss components stabilizing after approximately 40 epochs.

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 1/50      4.52G      1.355      3.337       1.52         59        416: 100% ━━━━━━━━━━━━ 146/146 1.9it/s 1:17
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 1.2s/it 9.4s
             all        239        454      0.25      0.401      0.225      0.165

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 2/50      5.47G      1.322      2.553      1.459         64        416: 100% ━━━━━━━━━━━━ 146/146 2.6it/s 56.7s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.6it/s 2.2s
             all        239        454      0.662      0.221      0.271      0.193

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 3/50      5.47G      1.332      2.492      1.468         60        416: 100% ━━━━━━━━━━━━ 146/146 2.6it/s 55.6s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.5it/s 2.3s
             all        239        454      0.297      0.362      0.233      0.158

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 4/50      5.47G      1.327      2.437       1.47         58        416: 100% ━━━━━━━━━━━━ 146/146 2.7it/s 54.8s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.6it/s 2.2s
             all        239        454      0.258      0.454      0.211      0.153

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 5/50      5.47G      1.322      2.329      1.463         96        416: 100% ━━━━━━━━━━━━ 146/146 2.6it/s 55.2s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.5it/s 2.3s
             all        239        454      0.306      0.452      0.358      0.267
```

Figure 13.
Top five epochs performance during yolov11 model training.

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
46/50      5.47G     0.8734     0.7138      1.188         48        416: 100% ━━━━━━━━━━━━ 146/146 2.7it/s 54.0s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.6it/s 2.2s
             all        239        454      0.638      0.663      0.697      0.559

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
47/50      5.47G     0.8561     0.6796      1.184         12        416: 100% ━━━━━━━━━━━━ 146/146 2.7it/s 53.9s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.7it/s 2.2s
             all        239        454      0.668      0.646      0.714      0.571

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
48/50      5.47G     0.8443     0.6632      1.174         24        416: 100% ━━━━━━━━━━━━ 146/146 2.7it/s 54.3s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.6it/s 2.2s
             all        239        454      0.657      0.653      0.713      0.573

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
49/50      5.47G     0.8473     0.6572      1.167         29        416: 100% ━━━━━━━━━━━━ 146/146 2.7it/s 54.0s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.7it/s 2.2s
             all        239        454      0.679      0.656       0.71       0.57

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
50/50      5.47G     0.8353     0.6381      1.163         25        416: 100% ━━━━━━━━━━━━ 146/146 2.7it/s 54.2s
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 8/8 3.6it/s 2.2s
             all        239        454      0.643      0.659       0.71       0.57
```

Figure 14.
Last five epochs representing the final stage of the yolov11 model training process.

Figure 15 illustrate the training loss curves of YOLOv11 model show a constant downward trend which suggests efficient learning and steady model convergence.

The Confusion Matrix of the YOLOv11 model (Figure 16) gives a clear look at the classification accuracy of the model regarding the different categories of plant diseases. It successfully points to the capability of the model to differentiate between 38 different classes and a background class.

There is a strong dark blue diagonal of the matrix and it indicates high accuracy of correctly identifying true positives. In particular the Strawberry leaf, Tomato leaf yellow virus, and Raspberry leaf are the classes with very high confidence scores which are represented by the strong color density on the diagonal line.

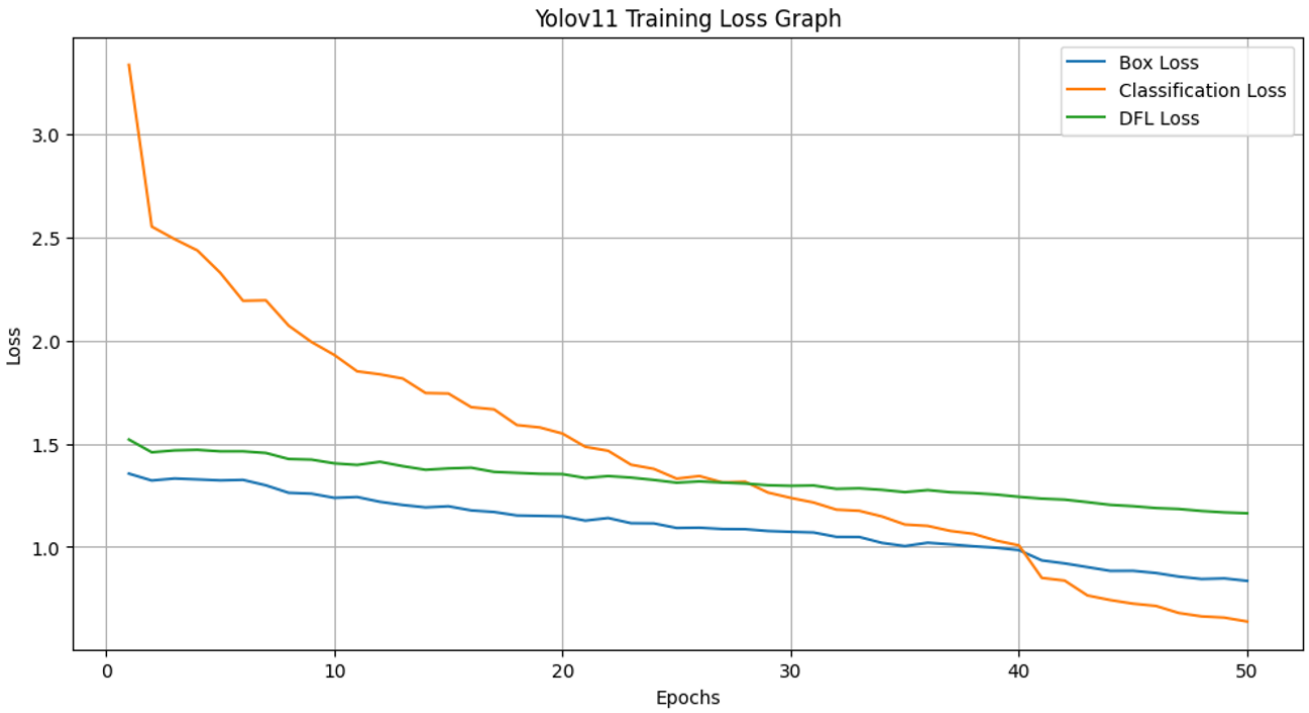


Figure 15.
Training loss curves of YOLOv11 model.

On the one hand the overall performance is strong and on the other hand there are also some off-diagonal patches which display a little confusion between the visually similar classes. As an example, one can see the overlapping in various types of Tomato leaf diseases (e.g., Tomato Septoria leaf spot and Tomato leaf bacterial spot). This is anticipated in fine-grained plant pathology in which lesion textures and colors are visually comparable.

There is a noticeable number of predictions which are given out in the Background column. This means that the model is at times not effective on small-scale features and does not recognize the presence of disease but rather labels the area as background of the healthy leaf.

Regardless of these slight confusions, the concentration of predictions on the diagonal shows that the YOLOv11 model is very reliable in multi-class plant disease detection. The fact that the model can retain high accuracy despite having 38 complex classes makes it well applicable in the real-world agricultural monitoring.

The model outperformed YOLOv8 in every evaluation metric, with Precision of 0.657, Recall of 0.655, mAP@0.5 of 0.713, and mAP@0.5:0.95 of 0.573.

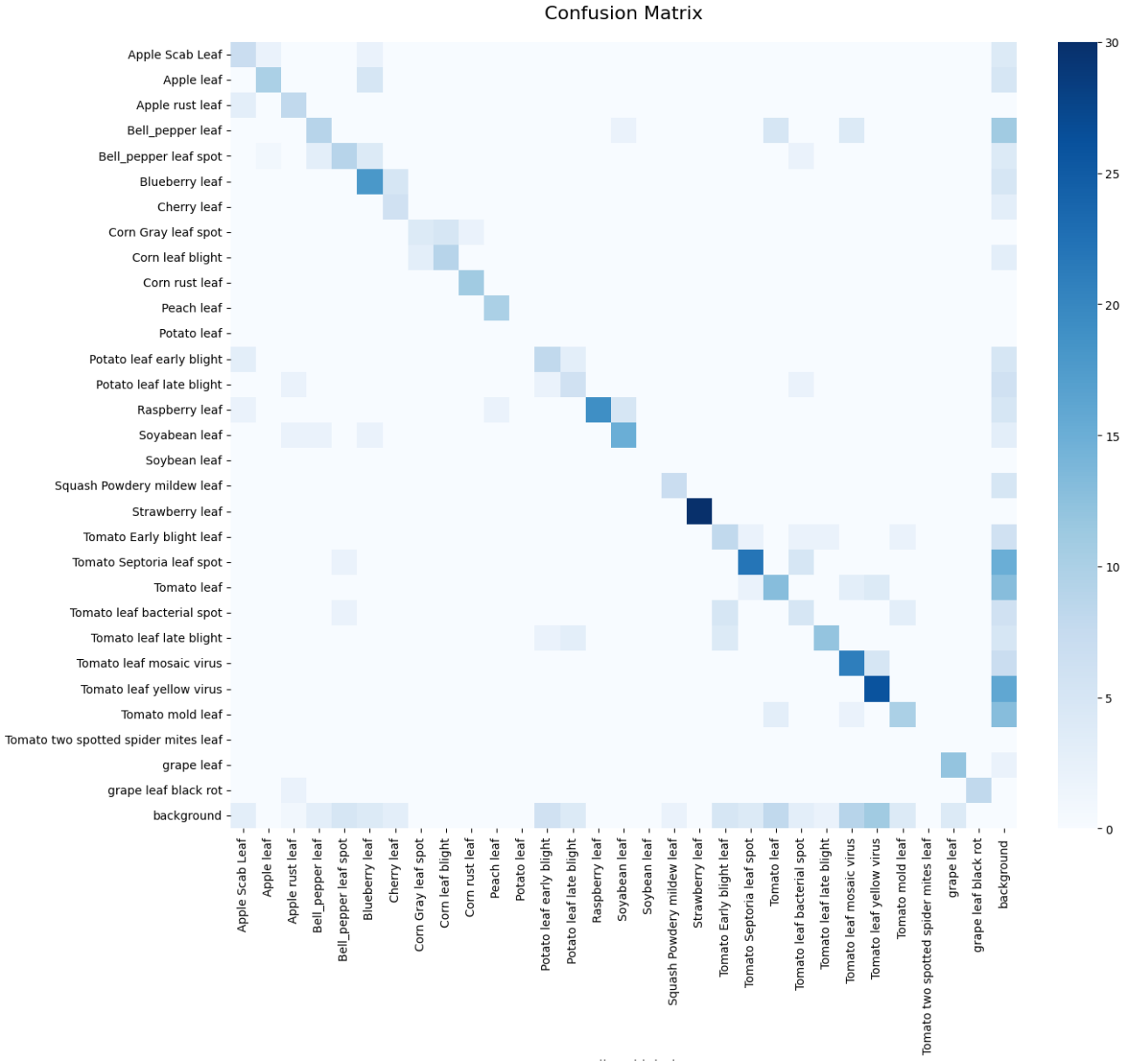


Figure 16.
Confusion matrix of the YOLOv11 model.

YOLOv11 worked well with real time performance as shown in Figure 17. The results shows that preprocessing took 0.1 ms, inference took 6.5 ms, loss computation took 0.0 ms, and post-processing took 2.2 ms per image. Because of its complicated design YOLOv11 requires more inference time than YOLOv8 yet it nevertheless retains a reasonable processing speed for real-world plant disease detection applications.

```
Speed: 0.1ms preprocess, 6.5ms inference, 0.0ms loss, 2.2ms postprocess per image
```

Figure 17.
Inference speed of YOLOv11 model.

4.7. Yolov26 Results

The YOLOv26 model is the most advanced among the evaluated models, consisting of 392 layers and approximately 26.22 million parameters with 93.4 GFLOPs as shown in Figure 18. This deeper architecture enables enhanced feature extraction capabilities.

```
YOLO26l summary: 392 layers, 26,222,604 parameters, 26,222,604 gradients, 93.4 GFLOPs
```

Figure 18.
YOLOv26 model architecture's specifications.

The enhanced depth and the number of parameters enable the network to extract more intricate visual features from plant disease images. Because of this the model is better able to capture subtle variations in leaves textures, discoloration

patterns, and disease spots. However, compared to lighter models the inference time may increase because of the higher processing needs.

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 1/50      4.94G      1.493      3.987    0.01927         59        416: 100% ──────────── 146/146 1.8it/s 1:22
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 1.1s/it 9.1s
             all        239        454      0.293      0.174     0.0895     0.0583

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 2/50       5.9G      1.545      3.317    0.02055         64        416: 100% ──────────── 146/146 2.3it/s 1:04
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 3.1it/s 2.6s
             all        239        454      0.292       0.27      0.169      0.111

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 3/50       5.9G       1.53      3.136    0.02096         60        416: 100% ──────────── 146/146 2.3it/s 1:03
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 3.3it/s 2.4s
             all        239        454      0.145      0.284      0.127     0.0844

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 4/50       5.9G      1.509      3.004    0.02054         58        416: 100% ──────────── 146/146 2.3it/s 1:03
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 3.3it/s 2.4s
             all        239        454      0.439      0.227      0.181      0.117

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 5/50       5.9G      1.503      2.851    0.02022         96        416: 100% ──────────── 146/146 2.3it/s 1:03
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 3.3it/s 2.4s
             all        239        454      0.207      0.322      0.191      0.127
```

Figure 19.
Top five epochs performance during yolov26 model training.

Effective learning and convergence are demonstrated by the training process shown in Figures 19 and 20 which gradually reduces loss values.

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
46/50      5.91G      1.052     0.9694    0.01508         48        416: 100% ──────────── 146/146 2.3it/s 1:03
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 3.0it/s 2.7s
             all        239        454      0.613      0.644      0.646      0.522

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
47/50      5.91G       1.04       0.93    0.01512         12        416: 100% ──────────── 146/146 2.3it/s 1:03
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 3.3it/s 2.4s
             all        239        454      0.608      0.646      0.649      0.529

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
48/50      5.91G      1.035       0.92    0.01489         24        416: 100% ──────────── 146/146 2.3it/s 1:03
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 3.3it/s 2.4s
             all        239        454      0.628      0.642      0.665      0.544

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
49/50      5.91G      1.034     0.9215    0.01467         29        416: 100% ──────────── 146/146 2.3it/s 1:03
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 3.3it/s 2.4s
             all        239        454      0.582      0.661      0.657      0.537

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
50/50      5.91G      1.031     0.9099    0.01473         25        416: 100% ──────────── 146/146 2.3it/s 1:03
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 2.9it/s 2.7s
             all        239        454      0.739      0.571      0.658      0.535
```

Figure 20.
Last five epochs representing the final stage of the yolov26 model training process.

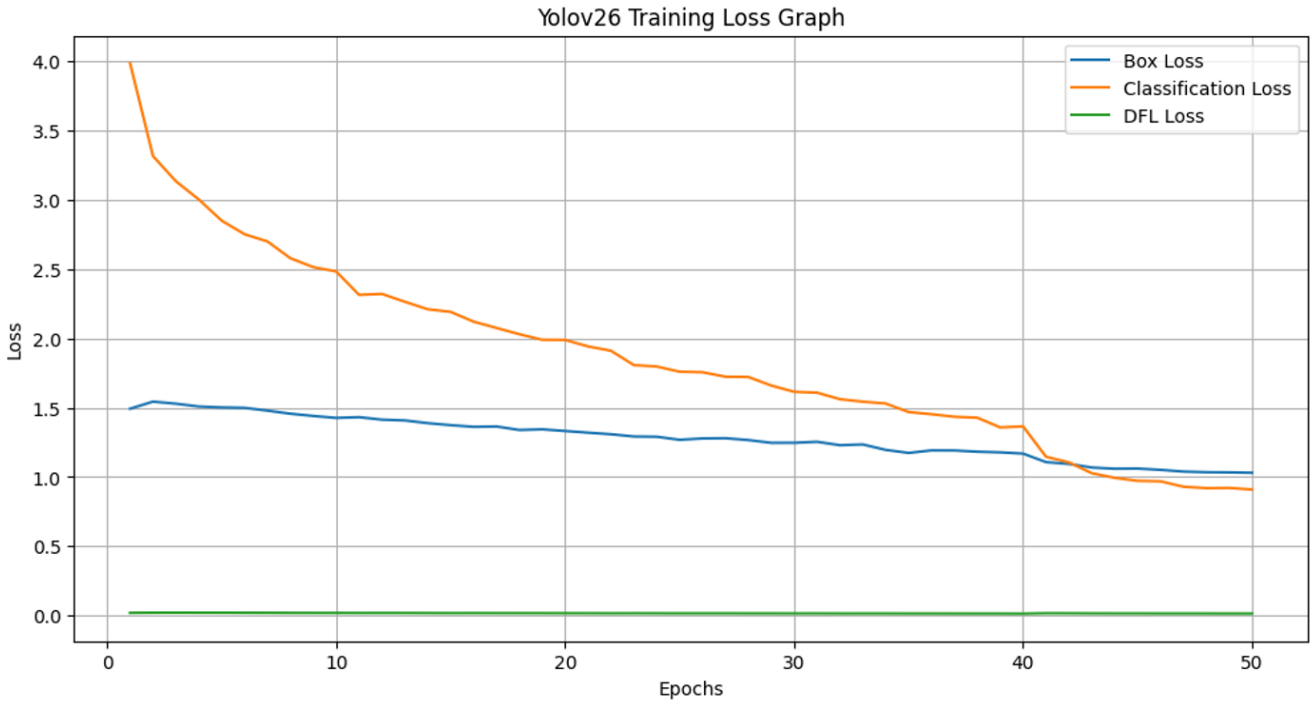


Figure 21.
Training loss curves of YOLOv26 model.

Figure 21 depicts the YOLOv26 model's training loss curves which demonstrate stable optimization of classification and localization tasks.

Figure 22 displays the confusion matrix of the YOLOv26 model that represent the classification performance of the model in the multi-class plant disease dataset. The most remarkable element of this matrix is the sharp and dark blue diagonal. With a complicated dataset of 38 classes the intensive density of these diagonal cells (in particular, in the classes such as Strawberry leaf and Tomato leaf yellow virus) shows that YOLOv26 has reached a higher degree of feature extraction with the correct recognition of the overwhelming majority of the diseased samples.

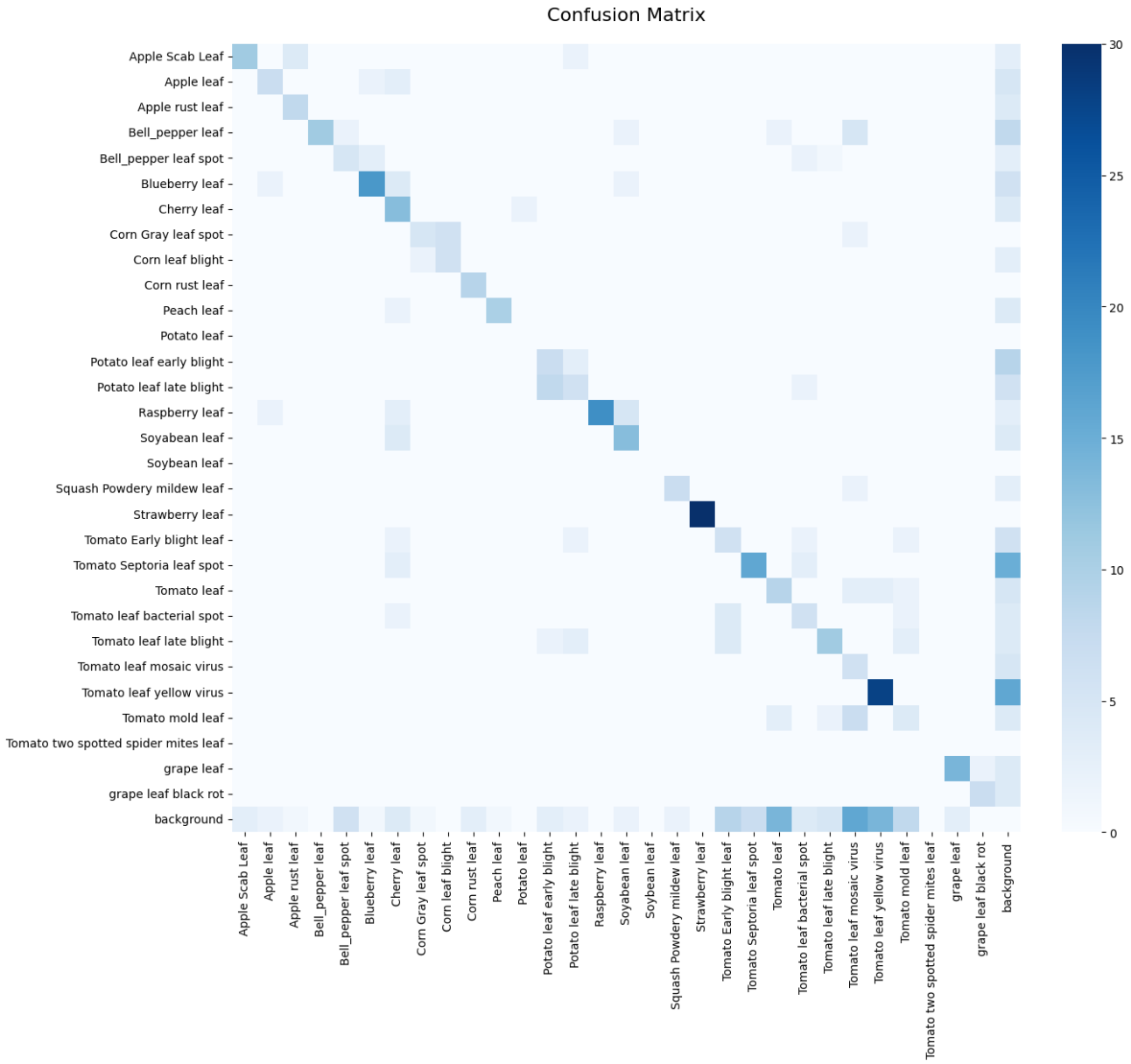


Figure 22.
Confusion matrix of Yolov26 model.

The “off-diagonal” noise is lower than in previous architectures. The confusion between unrelated species (e.g., Apple vs. Grape) is very minimal which proves the high potency of the model to initially recognize the type of plant and then diagnose the disease.

However, in the Tomato and Potato subgroups some light-colored clusters are still discernible (e.g., Tomato Septoria vs. Tomato Bacterial Spot). This implies that although the YOLOv26 is very sophisticated, the very high visual similarity of the necrotic patterns in these particular diseases is still a slight problem.

In term of the performance, YOLOv26 model achieved a Precision of 0.626, Recall of 0.640, mAP@0.5 of 0.665, and mAP@0.5:0.95 of 0.543.

In order to determine whether the YOLOv26 model is appropriate for real-time detection applications, its inference performance was assessed. The average processing time per image is summarized below in Figure 23.

```
Speed: 0.1ms preprocess, 6.8ms inference, 0.0ms loss, 0.3ms postprocess per image
```

Figure 23.
Inference speed of YOLOv26 model.

These findings suggest that because of its larger architecture and higher computational complexity YOLOv26 has slightly higher inference time than YOLOv8 and YOLOv11. Nevertheless, the model retains effective processing capability which makes it appropriate for real-world plant disease detection systems where speed and accuracy both are crucial factors.

5. Discussion

In this section a comparative analysis of the three YOLO-based object detection models used in this study: YOLOv8, YOLOv11, and YOLOv26 is presented. The training of all models was done with the same plant disease data, training hyperparameters, and computation environment so that a fair comparison could be made. Key performance parameters including as precision, recall, mAP, inference time, and model complexity are the focus of the comparison. The overall performance of the proposed models is shown in Table 4.

Table 4.
Performance comparison of the proposed YOLO models.

Model	Layers	Precision	Recall	mAP@0.5	mAP@0.5:0.95	Inference Time
Yolov8	129	0.558	0.622	0.643	0.513	1.9 ms
Yolov11	358	0.657	0.655	0.713	0.573	6.5 ms
Yolov26	392	0.626	0.64	0.665	0.543	6.8 ms

Among the three models, YOLOv11 had the highest precision, at 0.657, indicating the fewest false-positive detections. YOLOv26 had a moderate level of precision of 0.626, and the lowest level of precision was 0.558 with YOLOv8. This implies that the YOLOv11 and YOLOv26 designs make it possible to identify plant disease patterns with better precision.

Yolov11 achieved the highest recall value of 0.655 in this paper, which means that it could have identified a greater percentage of plant disease cases than the other models. YOLOv26 came at a recall of 0.64, and YOLOv8 came at a marginally lower recall of 0.622, implying that some disease instances were missed during detection.

In terms of mAP@0.5, YOLOv11 has the best detection performance having the highest score of 0.713 showing better detection performance than other models. The score of YOLOv26 was 0.665 while YOLOv8 recorded 0.643. Similarly, for the more stringent metric mAP@0.5:0.95, YOLOv11 again achieved the best performance with 0.573, then YOLOv26 (0.543) and in last YOLOv8 (0.513). These findings show that YOLOv11 offers the best-balanced detection and localization work on plant disease detection.

With 1.9 ms per image, YOLOv8 showed the fastest inference speed, making it ideal for real-time applications. YOLOv11 took 6.5 ms and YOLOv26 took slightly more time 6.8 ms to infer, because of its deeper design and greater computational complexity.

With 129 layers, YOLOv8 is the lightest model, which contribute to its lower computational cost and quicker inference time. Compared to it, the YOLOv11 has 358 layers, and the deepest architecture is YOLOv26, which has 392 layers. Although deeper models typically have better feature extraction capabilities but they also demand more processing power.

Overall, the comparison research shows that each model has unique advantages when it comes to identifying plant diseases. It can be observed from the comparison that YOLOv8 is the most efficient model in terms of both speed and computational cost which makes it ideal for edge-device deployment and real-time agricultural monitoring systems. Whereas with the highest precision, recall, and mAP values, YOLOv11 attains the best overall detection performance demonstrating exceptional capacity to precisely identify and localize plant disease patterns. YOLOv26, on the other hand, although slightly slower due to its complex architecture but still maintains competitive performance with balanced precision, recall, and mAP scores. According to these findings more sophisticated models like YOLOv11 and YOLOv26 provide better detection accuracy for in-depth plant disease investigation whereas lightweight models like YOLOv8 are better for speed-critical applications. Thus, the model selection is dependent on the specific application requirements, particularly the balance between detection accuracy and processing speed.

6. Conclusion and Future Work

This study presented a deep learning-based approach for automated plant disease detection using YOLO-based object detection models. This study investigated YOLO-based deep learning models for automated plant disease detection using YOLOv8, YOLOv11, and YOLOv26 under identical experimental settings. The results show that all models are effective for detecting and localizing plant diseases, with differences in accuracy, speed, and computational cost. YOLOv11 achieved the highest overall performance in terms of precision, recall, and mAP, making it the most accurate model. YOLOv26 showed strong detection capability but required higher computational resources, while YOLOv8 was the fastest and most lightweight, making it suitable for real-time applications. Overall, the study confirms that YOLO-based object detection models provide an efficient and practical solution for plant disease detection in precision agriculture by enabling both classification and localization.

In future work, the performance can be improved by using larger and more diverse real-world datasets to enhance model generalization. Deployment of the system as a mobile or web-based application would support real-time use in farming environments. Further optimization for low-power devices such as smartphones and drones can also enhance practical applicability. Additionally, integrating disease severity estimation and treatment recommendation features would make the system more useful for smart agricultural decision-making.

References

[1] S. Savary, L. Willocquet, S. J. Pethybridge, P. Esker, N. McRoberts, and A. Nelson, "The global burden of pathogens and pests on major food crops," *Nature ecology & Evolution*, vol. 3, no. 3, pp. 430-439, 2019, https://doi.org/10.1038/s41559-018-0793-y.

[2] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70-90, 2018, https://doi.org/10.1016/j.compag.2018.02.016.

[3] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016, https://doi.org/10.3389/fpls.2016.01419.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.

[5] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, p. 2674, 2018, https://doi.org/10.3390/s18082674.

[6] A. K. Singh, S. Sreenivasu, U. Mahalaxmi, H. Sharma, D. D. Patil, and E. Asenso, "[Retracted] Hybrid Feature-Based Disease Detection in Plant Leaf Using Convolutional Neural Network, Bayesian Optimized SVM, and Random Forest Classifier," *Journal of Food Quality*, vol. 2022, no. 1, p. 2845320, 2022.

[7] S. K. Sahu and M. Pandey, "An optimal hybrid multiclass SVM for plant leaf disease detection using spatial Fuzzy C-Means model," *Expert Systems with Applications*, vol. 214, p. 118989, 2023, https://doi.org/10.1016/j.eswa.2022.118989.

[8] Y. Alhwaiti *et al.*, "Early detection of late blight tomato disease using histogram oriented gradient based support vector machine," *arXiv preprint arXiv:2306.08326*, 2023, https://doi.org/10.48550/arXiv.2306.08326.

[9] S. M. Sai, G. Gopichand, C. V. Reddy, and K. Teja, "High accurate unhealthy leaf detection," *arXiv preprint arXiv:1908.09003*, 2019, https://doi.org/10.48550/arXiv.1908.09003.

[10] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep neural networks based recognition of plant diseases by leaf image classification," *Computational Intelligence and Neuroscience*, vol. 2016, no. 1, p. 3289801, 2016, https://doi.org/10.1155/2016/3289801.

[11] A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition," *Sensors*, vol. 17, no. 9, p. 2022, 2017, https://doi.org/10.3390/s17092022.

[12] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311-318, 2018, https://doi.org/10.1016/j.compag.2018.01.009.

[13] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Computers and Electronics in Agriculture*, vol. 161, pp. 272-279, 2019, https://doi.org/10.1016/j.compag.2018.03.032.

[14] E. A. Aldakheel, M. Zakariah, and A. H. Alabdalall, "Detection and identification of plant leaf diseases using YOLOv4," *Frontiers in Plant Science*, vol. 15, p. 1355941, 2024, https://doi.org/10.3389/fpls.2024.1355941.

[15] U. Ali, M. A. Ismail, R. A. Habeeb, and S. A. Shah, "Performance evaluation of YOLO models in plant disease detection," *Journal of Informatics and Web Engineering*, vol. 3, no. 2, pp. 199-211, 2024.

[16] A. Shill and M. A. Rahman, "Plant disease detection based on YOLOv3 and YOLOv4," in *2021 international conference on automation, control and mechatronics for industry 4.0 (ACMI)*, 2021: IEEE, pp. 1-6.

[17] R. Ahmed and E. H. Abd-Elkawy, "Improved tomato disease detection with YOLOv5 and YOLOv8," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 13922-13928, 2024, https://doi.org/10.48084/ETASR.7262.

[18] G. Jocher *et al.*, "ultralytics/yolov5: v6. 2-yolov5 classification models, apple m1, reproducibility, clearml and deci. ai integrations," *Zenodo*, 2022, https://doi.org/10.5281/zenodo.7002879.

[19] H. M. Zayani *et al.*, "Deep learning for tomato disease detection with YOLOv8," *Engineering, Technology & Applied Science Research*, vol. 14, no. 2, pp. 13584-13591, 2024, https://doi.org/10.48084/etasr.7064.

[20] X. Wang and J. Liu, "Vegetable disease detection using an improved YOLOv8 algorithm in the greenhouse plant environment," *Scientific Reports*, vol. 14, no. 1, p. 4261, 2024, https://doi.org/10.1038/s41598-024-54540-9.

[21] E. Önler and N. D. Köycü, "Wheat powdery mildew detection with YOLOv8 object detection model," *Applied Sciences*, vol. 14, no. 16, p. 7073, 2024, https://doi.org/10.3390/app14167073.

[22] M. S. Z. Abid, B. Jahan, A. Al Mamun, M. J. Hossen, and S. H. Mazumder, "Bangladeshi crops leaf disease detection using YOLOv8," *Heliyon*, vol. 10, no. 18, 2024.

[23] A. Ghafar, C. Chen, S. Atif Ali Shah, Z. Ur Rehman, and G. Rahman, "Visualizing plant disease distribution and evaluating model performance for deep learning classification with YOLOv8," *Pathogens*, vol. 13, no. 12, p. 1032, 2024, https://doi.org/10.3390/pathogens13121032.

[24] S. Yang, J. Yao, and G. Teng, "Corn leaf spot disease recognition based on improved YOLOv8," *Agriculture*, vol. 14, no. 5, p. 666, 2024, https://doi.org/10.3390/agriculture14050666.

[25] M. Na, L. Yanwen, X. Miao, and Y. Hongwen, "IMPROVED YOLOv8-BASED AUTOMATED DETECTION OF WHEAT LEAF DISEASES," *INMATEH-Agricultural Engineering*, vol. 71, no. 3, 2023.

[26] A. Abulizi, J. Ye, H. Abudukelimu, and W. Guo, "DM-YOLO: Improved YOLOv9 model for tomato leaf disease detection," *Frontiers in Plant Science*, vol. 15, p. 1473928, 2025, https://doi.org/10.3389/fpls.2024.1473928.

[27] Z. Chen, J. Feng, K. Zhu, Z. Yang, Y. Wang, and M. Ren, "YOLOv8-ACCW: Lightweight grape leaf disease detection method based on improved YOLOv8," *IEEE Access*, vol. 12, pp. 123595-123608, 2024, https://doi.org/10.1109/ACCESS.2024.3453379.

[28] J. Wang *et al.*, "Improved lightweight yolov8 model for rice disease detection in multi-scale scenarios," *Agronomy*, vol. 15, no. 2, p. 445, 2025, https://doi.org/10.3390/agronomy15020445.

[29] H. Zhu, D. Wang, Y. Wei, P. Wang, and M. Su, "YOLOV8-CMS: A high-accuracy deep learning model for automated citrus leaf disease classification and grading," *Plant Methods*, vol. 21, no. 1, p. 88, 2025, https://doi.org/10.1186/s13007-025-01396-3.

[30] Kaggle. "Plant Diseases Detection Dataset." https://www.kaggle.com/datasets/kamipakistan/plant-diseases-detection-dataset (accessed 2026).

[31] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680-1716, 2023, https://doi.org/10.3390/make5040083.

[32] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, "Yolov8 to yolo11: A comprehensive architecture in-depth comparative review," *arXiv preprint arXiv:2501.13400*, 2025, https://doi.org/10.48550/arXiv.2501.13400.

[33] G. Jocher, Chaurasia, A., Qiu, J. . "YOLO by Ultralytics." https://github.com/ultralytics/ultralytics (accessed.

[34] Ultralytics. "Ultralytics YOLO11." https://docs.ultralytics.com/models/yolo11#overview (accessed 2026).

[35] Q. Zhao and J. Zhu, "An Improved YOLOv11 architecture with multi-scale attention and spatial fusion for fine-grained residual detection," *Results in Engineering*, vol. 27, p. 107061, 2025, https://doi.org/10.1016/j.rineng.2025.107061.

[36] P. Hidayatullah and R. Tubagus, "YOLO26: A comprehensive architecture overview and key improvements," *arXiv preprint arXiv:2602.14582*, 2026.

[37] R. Sapkota and M. Karkee, "Ultralytics YOLO evolution: An overview of YOLO26, YOLO11, YOLOv8 and YOLOv5 object detectors for computer vision and pattern recognition," *arXiv preprint arXiv:2510.09653*, 2025, https://doi.org/10.48550/arXiv.2510.09653.

[38] D. S. R. Sukhdeve and S. S. Sukhdeve, "Google colaboratory," in *Google cloud platform for data science: A crash course on big data, machine learning, and data analytics services*: Springer, 2023, pp. 11-34.

[39] Ultralytics. "Mean Average Precision (mAP)." https://www.ultralytics.com/glossary/mean-average-precision-map (accessed 2026).

[40] M. Gschwind, "AI: It's all about inference now: Model inference has become the critical driver for model performance," *Queue*, vol. 23, no. 2, pp. 40-78, 2025, https://doi.org/10.1145/3733701.

[41] P. Villalobos, J. Sevilla, T. Besiroglu, L. Heim, A. Ho, and M. Hobbhahn, "Machine learning model sizes and the parameter gap," *arXiv preprint arXiv:2207.02852*, 2022, https://doi.org/10.48550/arXiv.2207.02852.

[42] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI conference on artificial intelligence*, 2020, vol. 34, no. 07, pp. 12993-13000.

[43] P. K. Sekharamantry, F. Melgani, and J. Malacarne, "Deep learning-based apple detection with attention module and improved loss function in YOLO," *Remote Sensing*, vol. 15, no. 6, p. 1516, 2023, https://doi.org/10.3390/rs15061516.

[44] X. Li *et al.*, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21002-21012, 2020, https://doi.org/10.48550/arXiv.2006.04388.