An Enhanced Deep Neural Network Framework for Accurate Tomato Disease Recognition in Real-time Environment

Submitted By

Shahab Ul Islam

Supervised By

Prof. Vito Pascazio

Coordinator

Prof. Agostino Iadicicco

Doctorate School in Information and Communication Technology and Engineering

This dissertation is submitted for the degree of

Doctor of Philosophy

Department of Engineering, University of Naples, Parthenope

Year: 2022 – 2025

Acknowledgements

My doctoral journey, a truly collaborative endeavour, would not have been possible without the extraordinary contributions of many individuals. To all who played a part, I extend my profound and sincere gratitude.

My deepest appreciation goes to my academic mentors:

- **Professor Vito Pascazio, my esteemed supervisor**, who provided unwavering support and guidance over the past three years, consistently fostering both my research progress and my professional development.
- **Professor Giampaolo Ferraioli**, a true role model whose admirable character and profound scholarly insights profoundly guided my research and shaped my evolution as both an individual and a scholar.
- **Professor Agostino Iadicicco**, an outstanding program coordinator whose friendly demeanour, patience, and readiness to address my numerous inquiries both large and small, were invaluable to my success.
- **Professor Sergio Vitale**, for his exceptional guidance and sustained support, which truly smoothed my path throughout these three demanding years.

I am deeply thankful to my lab mates and colleagues. Special recognition goes to **Wenyu Yang** for her consistent and helpful collaboration, and to all my fellow lab mates for their unwavering support and the shared experiences that enriched my time here.

Finally, I offer my most heartfelt appreciation to the personal network that sustained me:

- To my **family, relatives, and friends**, thank you for the deep understanding and emotional support you provided throughout this demanding period.

And, most specially, to my beloved wife, **Sara Naeem**. Her boundless patience, steadfast understanding, positive spirit, and encouraging smile have been, and will always be, the powerful driving force in my life.

Abstract

Agricultural production is a critical sector that directly impacts the financial and social well-being of a society. The identification of plant diseases in a real-time environment is a significant challenge for agriculture production. Conventional disease detection methods, which depend significantly on manual inspection, are time-consuming, labour-intensive, and susceptible to human error. Furthermore, many recently developed models struggle in real-time scenarios because their accuracy is compromised when trained on isolated leaf images but then used to analyse entire plants. To tackle these issues, this research offers an advanced, automated system from tomato leaf segmentation and disease detection to the automatic spray prescription in real-time environment. This research presents an integrated system to address these issues, focusing on tomato plants. In first part of the research after deeply analysing the YOLO (You Only Look Once) models we integrate two models, the YOLOv8 with SAM (Segment Anything Model), for leaf detection and masking in the tomato plants, and extraction of the individual leaves in a real-time environment for improving the performance of leaf disease detection. For leaf detection, the modified YOLOv8 is used, and for masking and extraction of the individual leaves, the SAM is used. The individual leaves are then, provided to the custom deep neural network for further disease detection. This ensures that subsequent disease detection is performed on isolated leaves, improving accuracy. We investigated deep learning models for precise and efficient tomato plant disease detection. All the models were trained and validated on individual and merged dataset of over 18000 and 25,000 images, encompassing 10 distinct classes (9 diseases and healthy plants). The performance of our custom CNNs (Convolutional Neural Networks) model was significant, achieving an accuracy of over 99%. Furthermore, it revealed higher efficiency, requiring less training time and computational resources than leading architectures such as VGG (Visual Geometry Group), ResNet (Residual Network), and DenseNet (Densely Connected Convolutional Network), making it a promising tool for real-world applications. In second part of this research work, we have designed, an automated system for tomato disease detection and spray prescription using an enhanced YOLOv9 model. By leveraging advance deep learning techniques, the proposed system accurately identifies and detect the nine tomato leaf disease in real-time by making efficient, precise and accurate decision including healthy leaves. This YOLOv9 model is modified for detecting tomato leaf diseases and optimized for getting higher accuracy and efficiency. Once disease is identified, the system automatically recommends a spray depending on the detected disease, which helps in reducing the pesticide use along with the environmental impact. This system helps in maximizing crop health and yield. After testing the system on the test dataset and real-time images demonstrates the system accuracy and efficiency, achieving detection accuracy of 97% and spray prescription accuracy of 94%. Integrating a YOLOv9 with spray prescription system provides a sustainable, cost-effective solution for managing tomato plant diseases. This thesis illustrates the efficacy of deep learning in the efficient and precise identification of tomato plant diseases, along with automated spray recommendations, thereby benefiting farmers and enhancing agricultural productivity. The effective performance observed in this thesis makes it promising for real-world agricultural applications.

List of Abbreviations

Abbreviation	*Meaning*
AI	*Artificial Intelligence*
DL	*Deep Learning*
CNN	*Convolutional Neural Network*
R-CNN	*Region-based Convolutional Neural Network*
YOLO	*You Only Look Once*
SSD	*Single Shot Detector*
U-Net	*Unified Network*
FCNs	*Fully Convolutional Networks*
GPU	*Graphical Processing Unit*
mAP	*Mean Average Precision*
AP	*Average Precision*
IoU	*Intersection over Union*
FPS	*Frames Per Second*
SGD	*Stochastic Gradient Descent*
RMSProp	*Root Mean Square Propagation*
ELM	*Extreme Learning Machine*
KNN	*K-Nearest Neighbor*
SVM	*Support Vector Machine*
VGG	*Visual Geometry Group*
SAM	*Segment Anything Model*
DSCNN	*Depthwise Separable Convolutional Neural Network*
SegNet	*Segmented Neural Network*
EDA	*Exponential Discriminant Analysis*
DETR	*Detection Transformer*
FPN	*Feature Pyramid Network*
BiFPN	*Bidirectional Feature Pyramid Network*
NMS	*Non-Maximum Suppression*
PANet	*Path Aggregation Network*
CSPNet	*Cross Stage Partial Networks*
ELAN	*Efficient Layer Aggregation Network*
E-ELAN	*Extended Efficient Layer Aggregation Network*
CSP	*Cross Stage Partial*
SPPCSPC	*Spatial Pyramid Pooling and Cross Stage Partial*
GELAN	*Generalized Efficient Layer Aggregation Network*
PGI	*Programmable Gradient Information*
CUDA	*Compute Unified Device Architecture*

cuDNN	*CUDA Deep Neural Network*
COCO	*Common Objects in Context*
FP	*False Positives*
TP	*True Positives*
FN	*False Negatives*
PNG	*Portable Network Graphics*
ViT-H	*Vision Transformer with a Hierarchical Structure*
HPC	*High-Performance Computing*

List of Tables

List of Figures

Table of Contents

Chapter 01: Introduction

This thesis explores an automated system for disease identification in tomato leaves and spray prescription in real-time environment. Agriculture production of any country depends upon the quantity and quality of its productions. It has a direct impact on the social and financial life of any country. That's why on-time identification and treatment of diseases in plants have gain a lot of attention. Agriculture production plays a vital role in economic and social development. Tomato (Solanum lycopersicum) is one of the most economically significant horticultural crops in the world with an annual global production of over 186 million tonnes and a market value of over $190 billion [1]. Tomatoes are a major contributor to global food security and human nutrition because they are a rich source of vital nutrients like lycopene, β-carotene, vitamins C and E, and many other bioactive compounds [2]. However, due to persistent disease challenges like bacteria, fungi and other viral diseases in the tomato plants results in losing crop yield. This crop yield increased globally by 10-25%, which is equivalent to the annual economic damage of $19-48 billion [3].

In tomato cultivation, conventionally the disease management is totally dependent upon the calendar-based fungicide applications, which resulting in ecological pollution, excessive of pesticides, a potential development of pathogen resistance and an increased cost of production [4]. According to the Food and Agriculture Organization, each year over 2 million tonnes pesticides are used world-wide, in which a remarkable portion is used for tomato production systems [5]. This approach of excess usage of pesticides through unnecessary chemical applications not only elevate the health and environmental concerns but also imposes a significant economic burden on the farmers [6]. For implementing targeted and timely interventions it is crucial to detect the plant diseases accurately in early stages. An early detection of the plant disease can significantly maximize the crop losses while reducing the use of chemicals. Traditional methods of plant disease identification involve a visual assessment by experts or laboratory analysis. These methods are costly, time-consuming and difficult to implement for large fields [7]. In early stages of plant diseases, visual identification is very complicated due to similarities between symptoms which is caused by abiotic stresses, nutritional deficiencies or different pathogens [8]. These constraints often result in diagnoses inaccurately, due to which diseases spread through the crops before proper remedies can be deployed. Deep learning (DL) and computer vision played a very important role in revolutionizing the accurate management of plant diseases management. Terrific opportunities

have been introduced due to advancement in these fields for accurately and timely detecting and managing plant diseases. Especially CNNs, have shown the significant potential to identify the visible pattern of the plant pathologies across varying crop systems [9]. The evolution of deep learning algorithms from simple classification to the advance object detection systems have enhanced the implementation of these systems in precision agriculture contexts [10].

The rest of this chapter is structured as follows: Section 1.1 details the primary motivation for this research work. Section 1.2 provides the detailed theoretical background of machine learning and deep learning. Section 1.3 points out the problem statement in detail. Section 1.4 focuses on the research objectives. Thesis contribution is discussed in the section 1.5. Finally, Section 1.6 summarizes the thesis organization.

1.1 Motivation

The primary motivation for this research is to address the limitations of traditional disease management in tomato cultivation by leveraging advancements in technology. The goal is to develop a system that can accurately and efficiently detect plant diseases at an early stage and apply treatments precisely where needed. This is driven by several key factors:

- **Improving Efficiency and Yield**: An automated system can monitor plants continuously, identifying diseases far earlier than manual inspection. Early detection allows for timely intervention, preventing the disease from spreading and maximizing crop yield.
- **Enhancing Environmental Sustainability**: By implementing a precision spray prescription system, the application of chemical treatments is minimized. The system focuses on only infected areas or plants instead of blanket spraying, which minimizes the overall use of pesticides. The reduction in the use of pesticides in turn protects the useful insects, lower the risk of water and soil contamination, and promotes a healthier ecosystem.
- **Economic Viability**: Pesticides are quite expensive for farmers and this high is a significant burden for the overall farmers. A precision system minimizes the waste of pesticides which leads to significant cost savings. It also reduces the costs of labour for manual spraying and inspection.
- **Bridging the Knowledge Gap**: Most of the small-scale farmers do not have the knowledge for identifying and managing plant diseases accurately and efficiently. An automated system can serve as a powerful tool by democratizing access to advanced agricultural techniques that provides a reliable and accurate assessment of plant health.

The research work in this thesis, therefore, seeks to create a smart, accurate, sustainable and economically beneficial solution for the managing diseases in tomato plants, paving the way for a more resilient and efficient agricultural future.

1.2 Theoretical Background

1.2.1 Machine Learning

Machine learning can improve its performance by using data and through experience that allows the system to be more efficient and accurate in predictions without being explicitly programmed. Based on learning from data and experience machine learning is divided into four main categories.

- **Supervised Learning**

  In this type of machine learning the labelled dataset is used for training the model, which means that the dataset contains both input and correct output. Think of it like a student and teacher, who gives questions to the students along with correct answers for each question. The model learns by mapping the inputs to their corresponding outputs. In this way the model finds the patterns.

- **Unsupervised Learning**

  This type of machine learning uses unlabelled data for model training. The model is not provided with correct outputs within the dataset. The model given a task to find the structures and pattern on its own. Think of it like a student is exploring a new topic on its own without any instructions.

- **Semi-Supervised Learning**

  It is a hybrid approach that uses both labelled and unlabelled data for model training. The model uses small amount of labelled data for initial understanding and then used large amount of unlabelled data for improving the accuracy. This type of machine learning is used in scenarios where it is difficult to create a fully labelled dataset.

- **Reinforcement Learning**

  It is a unique type of machine learning which interacts with an environment and learns to take decisions. In this type, the agent is guided by the rewards and punishments system. In case of correct action, the agent receives an award while in case of incorrect action the agent receives a penalty The aim is to learn a set of rules that maximizes the aggregate reward over time.

1.2.2 Fundamentals of Deep Learning

Deep learning is a type of machine learning that automatically model and understand complex data pattern by using multi-layered neural networks. Deep learning learns the pattern layer by

layer, on its own which contrasts with traditional methods that depends only on the handmade features. This is accomplished through an organization of interconnected layers containing an input layer, hidden layers, and an output layer, that process information like the human brain does. Each layer including neurons implements a simple mathematical action on its input and forward the output to the next layer in the network. The number of hidden layers in the network which is considered as the "depth" of the network, allows it to learn the abstract and complex representations of the data gradually. In images identification, the early layer detects the edges, the middle layer in the network organises the edges to identify the shapes, while the goal of the final layer is to recognize the object based on these shapes provided by the middle layer. The learning process consists of adjusting the biases and weights of these connections through algorithms like gradient descent and backpropagation to decrease a predefined loss function [11], [12].

1.2.3 Convolutional Neural Networks (CNNs)

CNNs are the specific type of deep learning model especially suitable for processing grid-like data, such as images. The CNN uses the convolutional layers, including the sliding filter to generate a feature map by processing the input data which is the core component of the CNN. By accomplishing this process, the network learns the spatial hierarchies of features. CNNs minimizes the computational cost and dimensionality by depending upon the pooling layer which down-sample the feature maps. Finaly the classification is performed by the fully connected layers based on these extracted features [13]. Because of its architectural design, and being excellent in tasks like object detection, image classification and segmentation, CNN is a cornerstone technology for diseases identification in plants [14].

1.2.4 Object Detection Models

Object detection is a computer vision task focusing on recognizing and localizing one or more objects in an image, which goes beyond image classification by providing the precise location of the object within a bounding box. For this purposed a variety of deep learning models have been developed:

- **Region-based Convolutional Neural Network (R-CNN)**: The example of two stage detectors are R-CNN and its successors, Fast R-CNN and Faster R-CNN. They first identify the ROI (regions of interest) in an image by using a region proposal network. Once the ROIs are identified they then classify and refine the bounding boxes for those regions. Being accurate than single-stage detectors, these algorithms are slower, and

the computational cost is very high which makes them unsuitable for real-world applications.

- **You Only Look Once (YOLO)**: The YOLO series is famous for its speed, efficiency and accuracy. Unlike two-stage detectors which performs classification after identifying the regions, YOLO is a single-stage detector that predicts bounding boxes and class probabilities in one pass. This makes it a great fit for real-time applications [15].
- **Single Shot Detector (SSD)**: Like YOLO, SSD is a single-stage detector that predict bounding boxes at various scales by combining features from different layers of a network. By using the series of default bounding boxes (priors) at various aspect scales and ratios, results in improving the accuracy of SSD [16].
- **Autodistill Process for Data Labelling**: It is an open-source framework which automates the manual data labeling developed by Roboflow. The process is designed to turn a folder of unlabeled images into a fully trained model. It uses a process called Knowledge Distillation to transfer the "intelligence" of large, slow foundation models into small, fast, and specialized models like YOLOv8.

1.2.5 Image Segmentation Techniques

The main aim of the image segmentation is to divide an image into various segments which is considered as complex task than simple object detection. By labelling each segment in the image, the image segmentation allows a comprehensive understanding of the image content. Image Segmentation consists of two main types:

- **Semantic Segmentation**: This type of image segmentation uses predefined classes to classify every pixel of an image such as "healthy leaf," "diseased leaf," and "background". It does not distinguish between individual instances of the same class. The commonly used models for semantic segmentations are U-Net and Fully Convolutional Networks (FCNs) [17].
- **Instance Segmentation**: This goes a step further by identifying and segmenting each individual instance of an object within an image. This type of image segmentation goes beyond the identifying and segmenting each individual instance of an object within an image. After recognizing all diseased leaves, it also treats each diseased leaf as a unique object. The most common model used for this purpose which integrate the object detection with pixel-level segmentation is the Mask R-CNN [18].

For spray prescription system, in this thesis, the image segmentation can provide an accurate and precise information regarding location and the extent of diseased areas which enable a more efficient and precise application of pesticides.

1.2.6 Real-Time Processing Requirements

The ability of the system to work in real-time environment is the success of an automated disease detection and spray prescription system. It means that the system has the ability to process the images captured in real-time environment using a deep learning model and generating an input for the spray prescription module in real-time. The time required by the model to process these images is critical. It must be minimized to ensure the system can react to a diseased plant within the fraction of a second it is in the camera's field of view. Factors influencing real-time performance include the computational power of the hardware (e.g., GPU), the efficiency and size of the chosen deep learning model, and the optimization techniques applied to the software [19].

1.2.7 Performance Evaluation Metrics

For the evaluation of the proposed system, several performance evaluation metrics will be used. These metrics provide a quantitative measure of the model's efficiency and accuracy:

- **Precision and Recall**: Precision is the proportion of true positive predictions among all positive predictions the model made. Essentially, it answers: "How often is the model correct when it says a disease is present?" Recall measures the proportion of true positives among all actual positive instances in the dataset, answering: "How many of the existing diseases did the model successfully identify?"
- **F1-Score:** This metric provides a single measure that balances both precision and recall by calculating their harmonic mean. It is highly useful when dealing with imbalanced datasets.
- **Mean Average Precision (mAP):** mAP is the standard metric for object detection models. It is the average of the Average Precision (AP) across all classes. AP itself is the area under the precision-recall curve.
- **Intersection over Union (IoU):** IoU is essential for evaluating localization accuracy, as it measures the degree of overlap between the model's predicted bounding box and the true bounding box.
- **Frames Per Second (FPS):** Critical for any real-time system, FPS measures the speed of the model by counting how many images it can process in one second.

1.3 Problem Statement

The conventional methods of tomato plant disease detection are inefficient, unsustainable, and inaccurate. The reliance on manual inspection is time-consuming, labour-intensive, and susceptible to human error, often leading to delayed detection and treatment of diseases that have already spread. The widespread use of blanket pesticide application is economically wasteful and causes significant environmental harm, while also contributing to pesticide resistance. Furthermore, a key technological challenge exists in developing an automated solution. Existing deep learning models, often trained on synthetic or single-leaf images, fail to perform with satisfactory precision and speed in real-time, real-world field conditions where diseases need to be identified on individual leaves within the context of a whole plant. Therefore, there is a critical need to develop a fast and efficient automated system for real-time tomato plant disease detection that can accurately segment individual leaves from a plant image, identify and classify the specific disease on those leaves, and subsequently enable a precise, localized spray prescription to minimize chemical use and maximize crop yield.

1.4 Research Objectives

- Development of tomato leaf segmentation model for extracting individual leaves in real-time environment.
- Designing and implementation of an efficient and accurate tomato disease detection and classification algorithm.
- Designing of a spray prescription module and integrating with the disease detection system.
- Evaluate the performance of the system to check its effectiveness in real-time environment.

1.5 Contributions of the Thesis

This thesis offers several significant contributions to the field of computer vision and agriculture by completing the above-mentioned objectives. This work addresses the practical key challenges by providing an end-to-end solution which is a significant advancement over the existing approaches. The major contribution of this thesis is to provide a complete framework for tomato plant disease detection and spray prescription mechanism. Unlike other findings that focuses on single components (e.g., just image classification), this thesis connects multiple advanced deep learning techniques like leaf segmentation, disease detection, and targeted treatment into a functional pipeline. This holistic approach makes the solution feasible for practical implementation in a real-time agricultural environment.

A primary contribution is to overcome the limitation of models trained on individual leaf images or synthetic lab images. By implementing a system that perform segmentation on complex plant images to segment the individual leaves accurately. This thesis addresses a

major issue of applying deep learning in real-world scenarios. The ability of segmenting and extracting individual leaves from the image of the whole plant significantly improves the precision and reliability of disease detection, which is a key technical advancement for automated agriculture.

By enabling precision farming this thesis directly contributes to the goal of sustainable agriculture. The ability of the system to prescribe the spray based on the detected disease is a major leap forward from the current practice of blanket spraying. This will lead to a substantial reduction in the use of pesticides, minimizing the farmer's operational costs, and decreasing environmental contamination. Due to its direct impact on the agriculture and environmental factors this work is highly relevant and impactful.

This thesis provides an enhanced approach to achieving real-time performance for a complex, multi-stage computer vision task. By implementing and optimizing a fast and efficient algorithm for both leaf segmentation, disease detection and spray prescription, the thesis demonstrates that this level of automation is not only possible but also practical for use on moving platforms like edge devices, robotic arms or agricultural vehicles. This provides a blueprint for future research and development in this domain.

1.6 Thesis Organization

The proposed work comprises eight chapters. The thesis is structured to follow a logical progression from foundational concepts to the final integrated system and its validation. We outline the structure of this thesis as shown in the figure 1.1.

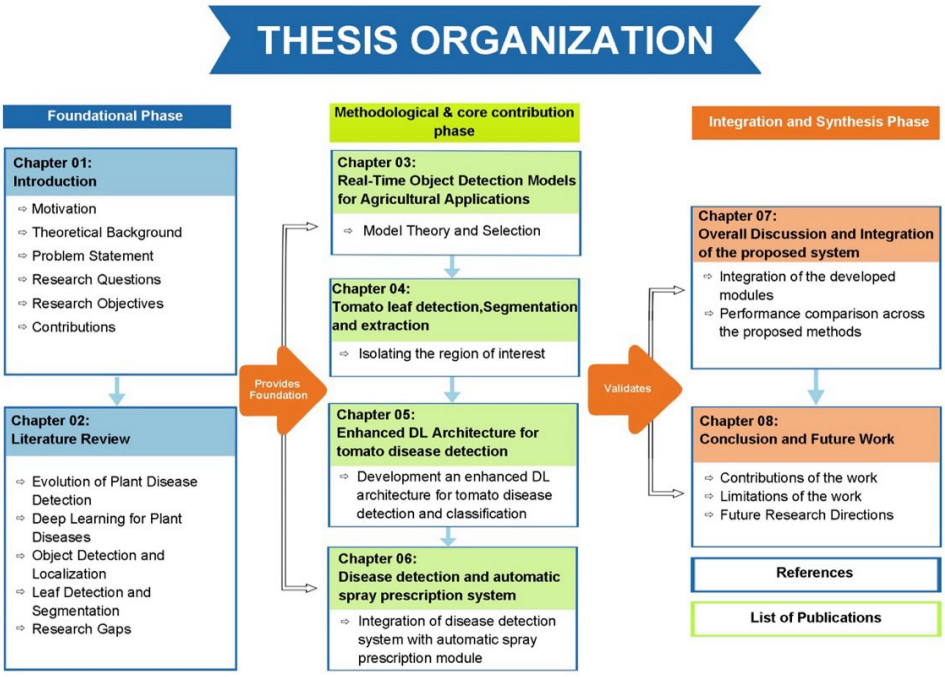


Figure 1.1 Thesis Organization

- **Introduction:** This chapter sets the stage for the entire research. It introduces the problem statement, establish the research motivation and objectives, and provide an overview of the thesis's structure.
- **Literature Review:** The primary purpose of this chapter is to review the current landscape of plant disease detection, deep learning, and precision agriculture. This systematic analysis helps to establish the limitations of existing studies and precisely define the objectives and contributions of the present work.
- **Real-Time Object Detection Frameworks for Agricultural Applications:** This is methodological chapter, focusing on the theoretical underpinnings of chosen models, such as YOLO or CNN, and how they apply to agricultural settings.
- **Tomato Leaf Detection, Segmentation, and Extraction:** This chapter is focused on isolating individual tomato leaves from whole-plant images. It covers the methodologies and results of leaf segmentation model. This is a key technical contribution as it addresses a major challenge in the field.
- **Enhanced Deep Learning Architecture for Tomato Disease Diagnosis:** In this chapter the specific disease detection model is developed, focusing on the deep learning architecture enhanced for accurately classifying diseases on the segmented leaves.
- **Intelligent Disease Detection and Automated Spray Prescription System:** This is the core of our applied research. It details the integration of the detection model with a real-time spray prescription system. This chapter covers the hardware and software components that enable the system to function in a real-world environment.
- **Overall Discussion and Integration of the Proposed Systems:** This chapter synthesizes the findings from the previous chapters. The overall performance, effectiveness, and limitations of the complete system is discussed in this chapter showing how the individual components work together.
- **Conclusion and Future Work:** The final chapter summarizes the key findings and contributions of this research work. It also outlines the potential avenues for future research to further improve the system's capabilities and applicability.

Chapter 02: Literature Review

This chapter provides a broad review of the existing body of knowledge and research relevant to automated plant disease detection and precision agriculture. The aim is to establish a strong theoretical base for the methodologies and technologies explored in this thesis, while also identifying the key research gaps that this work addresses. The review is structured to first cover the foundational principles of deep learning in agricultural applications, followed by an in-depth investigation of specific computer vision practices.

The review begins by tracing the evolution of plant disease diagnosis, from conventional methods to the advance deep learning methods. It will then delve into the deep learning architectures specifically the CNNs, which is the dominant architecture in this domain for image-based tasks. A critical part of this chapter will be the analysis of various object detection models, such as the YOLO series, and image segmentation techniques that form the technical core of the proposed system.

This review will also critically examine previous studies on the practical challenges in deploying such systems, particularly the issues related to data representation (e.g., synthetic vs. real-world images) which is the basic requirement for processing the data in real-time. By synthesizing the findings of past research, this chapter will clearly articulate why a multi-stage approach; involving leaf segmentation, precise disease detection, and automated spray prescription, is necessary to overwhelm the limitations of current systems and will set the platform for the original contributions presented in the following chapters.

The flow of this chapter is structured as follows: Section 2.1 focuses on the past literature that how the plant diseases detection evolved in agriculture. Section 2.2 provides a details literature on deep learning architectures used for plant disease detection. Section 2.3 discusses the literature based on object detection and localization in agriculture. Section 2.4 focuses on the past work on leaf detection and segmentation. Section 2.5 focuses on addressing real-world challenges and research gaps. Finally, Section 6.5 summarizes the chapter.

2.1 The Evolution of Plant Disease Detection in Agriculture

The history of plant disease detection in agriculture has evolved from manual, labour-intensive methods to sophisticated, technology-driven approaches. Understanding this progression is crucial for contextualizing the existing state of research and identifying the need for more advanced, automated systems. For centuries, plant disease diagnosis relied on the keen observation and expertise of farmers and agricultural specialists. This method, while

fundamental, is inherently limited. Diagnosis is often subjective and can be inconsistent across different individuals. Moreover, manual inspection is labour-intensive and time-wasting, making it unrealistic for monitoring wide-ranging crop fields. A significant drawback is that by the time a disease becomes visually apparent to the human eye, it may have already spread, leading to a delayed and less effective response. This traditional reliance on human judgment serves as the baseline against which modern technologies are measured. The first major shift came with the adoption of remote sensing and basic image processing techniques. These methods moved away from subjective human observation and towards objective, data-driven analysis.

Researchers began using multispectral and hyperspectral cameras to detect changes in a plant's spectral signature that are below the threshold of human visibility. Healthy and diseased plants reflect and absorb light differently across various wavelengths. For example, the researchers in [20] demonstrated the use of hyperspectral imaging to detect early-stage cotton root rot before visible symptoms appeared. This approach proved that diseases could be identified at a much earlier stage. Early computer vision techniques focused on extracting features like colour shape, texture, and histograms from leaves digital images. In [21] the researchers used these methods to categorize diseases based on statistical features of lesions and spots. While more efficient than manual inspection, these methods were often highly sensitive to the quality of image, light variations, and background clutter, making them unreliable for complex field environments.

With the increase in computational power, traditional machine learning models became viable for automated disease detection. Machine Learning Algorithms were trained on hand-crafted features extracted from images. For instance, researchers would manually define features such as the colour of a lesion or the shape of a leaf and feed these into the model for classification. This was a step forward, as models could be trained to recognize patterns. However, the performance was still heavily dependent on the quality of the feature engineering, which was a expertise-driven and time-consuming task [22].

The breakthrough came when researchers started applying pre-trained CNN models, originally developed for large-scale image classification challenges, to agricultural datasets. In [23] the researchers used CNNs to categorize 14 crops with 26 classes of diseases from 54,306 images, achieving an impressive accuracy of over 99%. While groundbreaking, this study highlighted a crucial limitation: the models were trained on images of single leaves with a uniform, clean background. This controlled environment does not imitate the complexities of a real-world farm, where plants are in a cluttered setting with varying light and overlapping leaves.

Recognizing the limitations of image classification, the focus of research shifted to object detection and segmentation. For real-world scenarios these techniques are more appropriate as they not only identify the disease but also localize its position on the plant, which is crucial for precision spraying. The development of real-time object detection models like SSD, YOLO, and Faster R-CNN marked a significant step. These models can simultaneously classify and locate diseases with a bounding box. In case of real-time applications, a key trade-off exists, that two-stage detectors are more accurate but slower, while single-stage detectors like YOLO are faster and better which is considered as more suitable for real-time scenarios [24].

2.2 Deep Learning for Plant Disease Recognition

The deep learning application, specifically the CNNs, has become a cornerstone of modern agricultural technology. CNNs have demonstrated a superior ability to eliminate the need of manual feature engineering by learning classified features from raw image data. Numerous findings have successfully applied CNNs for classifying plant diseases, achieving high accuracy rates. Current findings highlight several promising approaches for diagnosing plant diseases.

In order to achieve the high classification accuracy, the researchers used the pre-trained CNN models on the PlantVillage dataset like GoogLeNet and AlexNet. While these studies proved the efficacy of CNNs, they were often limited to classifying diseases on segmented, images of single leaf, which may not translate well to complex, real-world field conditions with cluttered backgrounds and varying light [25].

Recent studies indicate several effective approaches for detecting plant diseases. Punitha, et al. published analytical research in 2022. In their study, they utilized Deep Learning architectures such as AlexNet, GoogLeNet, and DenseNet, employing various optimizers through the stochastic gradient descent (SGD) algorithm and root mean square propagation (RMSProp) for the identification and categorization of tomato plant diseases to evaluate their accuracies. GoogLeNet is the best performer in this study, with an accuracy of 99.56% [26].

The authors conducted research utilizing vegetable crops to classify diseases such as Scab, Early Blight, Leaf Scorch, and Bacterial Spot. The researchers used deep learning and CNN architecture in this study and achieved 98.87% accuracy. The accuracy attained in this study is creditable, yet there remains potential for enhancement [27]**.**

For plant leaf disease detection, the authors [28] in conducted a study based on the comparison of CNN architectures like the GoogLeNet, VGG, and AlexNet. The dataset used by the

researchers contained six different seven classes. The study demonstrates that the GoogLeNet provides the best accuracy of more than 98% than those of the other architectures.

The authors conducted a study, in which the semantic segmentation is used to highlight the foreground (leaf) and the background (non-leaf) and to look through each pixel for labelling. A CNN-based model is used on the Plant Village dataset. In this research, they have achieved a total of 97.6% accuracy [29] .

The authors conducted a study in which they have utilized InceptionV3 for tomato plant disease detection. The transfer learning technique is used with a training accuracy of 92.19% and a test accuracy of 93.03% [30].

The researcher conducted research on plant disease detection in which they used AlexNet. They have used 18,345 training data and 4585 test data. In their research, they have obtained 98% accuracy using a releasing rate of 0.0005 and 75 epochs. The accuracy can be justified, but by using new models, it can be improved further [31].

The authors in [32] published their research on cucumber plant disease detection. They have used multi-level deep entropy Extreme Learning Machine (ELM) feature selection. They have used CNN models like DenseNet201, ResNet50, ResNet101 and VGG16. In their research, they achieved the best accuracy of 98.4%.

In [33] the authors published research on crown rot disease in wheat plants. In their research, they used image colour and machine learning techniques. They successfully distinguished between healthy and infected plants 14 days earlier. In their research, the F1 scores for most datasets were higher than 0.80.

The authors used hybrid convolutional network for plant disease detection by applying feature reduction. The authors worked on the dataset of grape plants including four diseased classes such as Black measles, stable, Black rot and Leaf Blight. In this study, the researchers analysed the Plant Village dataset and employed logistic regression to lower the dimensionality of the extracted features. They have used state-of-the-art classifiers and achieved an accuracy of 98.7% after 92 epochs [34].

The authors in [35] worked on guava plant disease detection utilizing machine learning techniques. In this work, they have used different textual and colour techniques for feature extraction with machine learning classifiers like Boosted Tree, Bagged Tree, Fine KNN (K-Nearest Neighbors), Cubic SVM (Support Vector Machine), and Complex Tree. In this finding, the Bagged Tree Classifier has achieved 99% accuracy on all four guava plant diseases which was the highest accuracy among all classifiers. For plant disease detection the researchers in [36] performed review-based research on machine learning and deep learning along with a

comparative analysis. For machine learning, the SVM model has achieved the higher accuracy of 97.3%, followed by the Random Forest classifier with the accuracy of 97%. For deep learning, InceptionV3 has the higher accuracy of 99.76%, tailed by VGG with 99.53% accuracy.

The authors used deep learning for guava plant diseases classification. In this paper, the researchers have used Flask in Python, in which they have created their own CNN model. For obtaining the accuracy the authors used confusion matrix and achieved the accuracy of 85% [37].

In [38] the authors published a study on a new CNN model called Real Time-Droid (RT-Droid). RT-Droid is based on YOLOv5 and can detect malware very quickly and accurately. To create RT-Droid, the authors first extracted features from Android manifest files and converted them into RGB images like QR codes. Leveraging transfer learning, the researchers trained Faster R-CNN, VGGNet, YOLOv4, and YOLOv5 models on these images. Notably, the YOLOv5 model achieved exceptional object detection accuracy in real time, surpassing the efficiency of other CNN models used in the study. The authors also compared the results of RT-Droid with Faster R-CNN, VGG Net, and YOLOv4 and found that it yielded better results.

Existing studies have made valuable contributions to tomato plant disease detection with deep learning, typically considering individual datasets. Our work builds on past research by incorporating merged datasets and developing a high-performing custom model, potentially paving the way for more robust and efficient tomato plant disease detection systems. After analysing the performance of established CNN architectures like VGG, ResNet, and DenseNet, we designed a custom model. This model achieved high accuracy, as discussed in the results section, while also offering benefits like reduced training time and computational cost.

2.3 Object Detection and Localization in Agriculture

To move beyond simple image classification, researchers have turned to object detection models that not only classify a disease but also localize it with a bounding box. This is a crucial step for targeted treatment. Two main categories of object detection models are relevant to this work. Models like Faster R-CNN, while highly accurate, are computationally intensive and slow. Their two-step process of classification after proposing regions of interest, makes them unsuitable for real-time applications in the field. To address the speed constraint, single-stage detectors like the YOLO series and SSD were developed. YOLO is particularly famous for its speed, as it predicts bounding boxes and class probabilities in a single forward pass. Studies have shown that while these models are fast, their performance can be impacted by small

objects, which can be an issue when detecting small disease spots. Recent advancements in the YOLO series (e.g., YOLOv7, YOLOv8, YOLOv9, etc) have made significant strides in balancing speed and accuracy, making them highly suitable for real-time agricultural applications. Current findings highlight several promising approaches for diagnosing plant disease detection and localization which are as follows:

To improve early-stage tomato disease detection, researchers in [39] modified the YOLOv8 framework with key architectural changes. They introduced a multiscale attention mechanism and a lesion-focused detection head specifically designed to identify small, irregular disease indicators. These enhancements significantly improved the detection of early symptoms (covering less than 5% leaf area), raising the mean Average Precision from 67.3% to 83.9%. Additionally, an uncertainty quantification module was added to provide detection confidence levels. Field tests confirmed the system's value, enabling disease detection an average of 4.2 days earlier than standard implementations.

Researchers in [40] compared Faster R-CNN, EfficientDet, YOLOv8, and YOLOv9 for identifying 14 agricultural diseases across five crop types. YOLOv9 demonstrated the best performance, achieving 94.3% mean Average Precision, outperforming YOLOv8 (92.1%), Faster R-CNN (91.4%), and EfficientDet (90.7%). Notably, YOLOv9 excelled at detecting small lesions and distinguishing visually similar diseases. It also proved suitable for real-time use with a processing speed of 22ms per image, making it about 15% faster than YOLOv8. The study concluded that YOLOv9 provides the optimum balance of high accuracy and computational efficiency for practical agricultural disease detection in the field.

In [41] the authors modified the YOLOv9 for enhancing the performance in tasks like tomato plant disease detection. The authors integrated a customized loss function and disease-specific data augmentation techniques that allocate greater weight to early-stage symptoms based on lesion appearance and size. The authors also implemented a novel feature fusion module that integrated semantic features with a high-resolution spatial information to better identify subtle disease indicators. Results show that comparing with standard YOLOv9 implementation, these modifications improved detection of early-stage symptoms by 13.7%, with overall mean Average Precision increasing from 93.2% to 96.4%. According to authors this architecture significantly minimizes the false negatives for early disease detection and improve the inference speed which is considered as a critical factor for disease management in real-time applications.

In [42] the researcher modified the YOLOv8 for employment on edge devices for real-time plant disease detection applications. The authors used various compression techniques like

knowledge distillation, quantization, and pruning to develop and implement a lightweight model which is suitable for edge devices. The authors enhanced this model for eight common crop diseases that provide a higher detection accuracy above 89%, while minimized the parameter count by 73%. The authors developed a custom inference engine integrated with environmental sensors particularly for agricultural edge devices that automatically adjust the image preprocessing based on ambient lighting conditions. Field testing has shown that using Raspberry Pi hardware the system successfully processing 12-18 frames per second. The system also shown a reliable performance regarding variable weather and lighting conditions. The authors highlighted being focused on YOLOv8, our optimization framework could be readily applied to the newer YOLO versions as they released.

In [43] the researchers integrate an expert system with CNN-based image classification (ResNet-152) for efficient and accurate tomato disease detection and management. The authors used a dataset, containing seven multiple diseases and achieved an accuracy of 94.7% which they linked to a knowledge base of 213 treatment protocols. It provided customized management recommendations considering factors like weather, crop stage, and previous treatments. User evaluations indicated high satisfaction with the accuracy and the usefulness of the system, which included visual explanations for diagnoses and treatment rationale.

In [44] the authors developed a YOLOv7-based system combined with multispectral imaging for the early detection of “Tomato Yellow Leaf Curl Virus”, “Tomato Spotted Wilt Virus”, and “Tomato Mosaic Virus”. This system could identify infected plants 3-7 days before symptoms were visible to humans by detecting subtle spectral changes. Integrated with a decision support module for recommending interventions, field validation showed this approach reduced overall infection rates by 43% compared to traditional methods. A hierarchical classification approach was used to address the challenge of distinguishing between different viruses at very early stages.

Researchers in [45] developed an automated system for greenhouse tomato production that uses computer vision (YOLOv8) to detect nine diseases and pests with 92.3% accuracy. This detection guides a robotic sprayer to target specific plants or areas. A decision tree determines the best spray strategy based on disease type, severity, plant stage, and treatment history. Field trials showed this system reduced chemical use by 37% compared to traditional methods while maintaining or improving pest control, leading to a positive return on investment within 1.5-2 years through cost savings and better crop quality.

The authors in [46] developed an integrated drone system using RGB and multispectral cameras for disease detection in commercial tomato fields. By capturing images every 3-5 days

and processing them with YOLOv8, they identified disease hotspots to create prescription maps for variable-rate fungicide application. Field trials across 120 hectares showed this method reduced fungicide use by 32% compared to uniform spraying, without compromising disease control, leading to annual savings of about $275 per hectare plus operational cost reductions. While effective for visually distinct diseases, the system's accuracy was lower for subtle symptoms, indicating potential for improvement with additional sensors.

Recent applications of deep learning for tomato disease detection have showed promising results. In [47] the authors employed YOLOv8 to detect four common tomato diseases with an average precision of 91.2%. Similarly, in the authors implemented a modified Faster R-CNN architecture for identifying five tomato leaf diseases, achieving 92.6% accuracy under controlled conditions. Meanwhile, the authors in [48] explored transfer learning approaches with EfficientNet for tomato disease classification, reporting 93.5% accuracy across seven disease categories. The authors in [49] utilized MobileNetV3 for lightweight deployment on edge devices, achieving 88.7% accuracy for six tomato diseases.

Despite these advances, in automated tomato leaf diseases management several crucial research gaps persist in the domain. In current findings the disease detection algorithms have progressed significantly, their integration with automated prescription systems for targeted treatment remains limited. Few studies have successfully linked detection outcomes with specific intervention recommendations tailored to disease type, severity, and environmental conditions.

Many proposed solutions demonstrate high accuracy under controlled conditions but show diminished performance in variable field environments with inconsistent lighting, complex backgrounds, and varying disease manifestations. Agricultural applications demand systems that can process imagery quickly enough to support timely decision-making, particularly when mounted on agricultural machinery or drones. Balancing detection accuracy with computational efficiency remains challenging for high-resolution imagery captured in dynamic field environments.

2.4 Leaf Detection and Segmentation

Tomato leaf detection and segmentation are very important for accurate tomato leaf disease detection. The integration of cutting-edge technologies such as YOLOv8 and SAM has performed a vital role in the improvement of these tasks. Current research suggests some effective methods for plant leaf segmentation. This study begins with a review of studies on the usage of YOLOv8 and SAM for different purposes to better comprehend the existing

literature on their identification. This section focuses on the contribution of this study using other studies in this field.

In this work, the authors projected a modified YOLOv8s-Seg network for the segmentation of tomato fruits and leaves in real-time, tackling problems emanating from weather conditions and surface characteristics. In this research the authors achieved a mAP of 92.2%, which outperforms previous models and offers technical support for tomato health monitoring and intelligent harvesting [50].

The authors integrated a new channel attention model into the advanced YOLOv8 model that makes it possible to add the Efficient Channel Attention mechanisms to improve the feature extraction of the model designed for tomato leaf disease detection. The experiments showed that the modified model outperformed the baseline by approximately 1.8% increase in mAP50, while the parameter size was reduced by 74%, which means a more accurate and less computational cost detection system [51].

This work aims to analyse and evaluate different segmentation models in detecting single and multiple diseases in tomato leaves. The accuracy of model B, which is Hybrid-DSCNN, a model combining U-Net and SegNet, was 98.24%, and it proved to be the fastest model, completing 1004 images in 30 nanoseconds [52].

The main objective of this work is to come up with a deep neural network to detect tomato leaf diseases based on the YOLOv8 architecture. It uses multi-head self-attention to improve feature learning, that improves the detection accuracy and makes the model more robust to environmental changes [53].

The authors in this work used YOLOv8 and RoboFlow to detect tomato leaf disease with nine classes and achieved an average precision of 98.9% mAP. The results provided by the authors in this research have the potential for detecting and classifying tomato leaf disease at early stages [54].

The authors in this work stated the deep learning models for identifying and segmenting tomato leaf diseases, focusing on the importance of accurate segmentation in disease detection. The authors in this work highlight the effectiveness of models such as U-Net in segmenting diseased areas, contributing to precision agriculture practices [55].

By incorporating SAM along with post-processed steps of segmentation, the author can achieve a zero-shot segmentation of potato leaves without using any supplementary training data. The approach is not exclusive to tomatoes but provides some insight into the segmentation tasks specific to tomato leaves [56].

In this research, the authors have used different CNN architectures such as VGG, ResNet, and DenseNet for tomato leaf disease detection with 10 classes, including healthy ones. The authors have merged the two datasets. After checking the accuracy of these models, the authors developed their own custom CNN model with 10 layers with the accuracy comparison with the other models. Their custom-deployed model has an accuracy of over 99% on both datasets [57].

The authors have conducted research on analysing tomato leaf for disease detection. The authors have used a hybrid system in which they integrated the machine learning algorithm, Exponential Discriminant Analysis (EDA), with transfer learning techniques such as ResNet50, DenseNet201, EfficientNetB0, and DarkNet53. The authors have used the two datasets of tomato leaf datasets (Taiwan and PlantVillage). The results in this work demonstrate that the authors have achieved the 98.09% and 98.29% accuracy on these datasets respectively [58].

This research meaningfully improves YOLOv8-Seg models towards real-time instance segmentation of plant diseases using the Tomato Leaf Disease dataset as a test case. The modified models are very effective in the segmentation of the diseased regions, and sequentially, disease control can be carried out promptly [59].

The DS-DETR model presented in [60] integrates improvements to the detection transformer (DETR) to enhance its efficiency in the segmentation of tomato leaves infected with diseases. It also promotes the efficient management of diseases by quickly generating accurate disease damage assessments since it achieves an AP mask of 0.6823, which is above state-of-the-art models.

In this study, the authors presented a YOLOV8s-based model. The authors used this proposed model for detecting diseases in wilted tomato leaves, yielding a mAP of 92.5%. The model proposed in this work is likely to be anticipated in achieving better accuracy in real-time disease detection than Faster R-CNN and YOLOv5 [39].

In response to the limitation of deploying detection algorithms based on deep learning on embedded devices, this paper presents an optimization of the YOLOv8s model. The algorithm has been able to perform reliable and efficient real-time detection of tomato leaf diseases and has a low model size, which can be utilized in embedded systems [61].

This study attempts to detect the tomato leaf miner pest using the YOLOv8-Seg models. The research clearly emphasizes the ability of the model to detect even those tomato leaves of low density for effective pest management measures. Towards this end, the authors go ahead to modify the YOLOv8-seg model to include the Ghost and BiFPN modules which improves the

segmentation of leaves. Five test datasets were used to test the model for leaf segmentation with a score of 86.4%, which is higher as compared with previous work stated [62].

In this work the authors proposed a model for diseases detection and segmentation of plant leaves, which is based on Mask R-CNN. The model achieved a mAP of 76.94%, which indicates its efficiency in segmenting the diseased area of different kinds of plants, including tomatoes [63].

2.5 Addressing Real-World Challenges and Research Gaps

Despite the rapid evolution of deep learning in precision agriculture, specifically in plant disease detection, a key gap exists between laboratory-based research and real-world application. A recurring theme in the literature is the lack of model generalization. Most models are trained on clean, pre-processed datasets (like PlantVillage) and fail to perform with satisfactory precision when confronted with complex, noisy field images, where a leaf may be partially obscured, shadowed, or have multiple diseases.

Furthermore, the existing literature highlights a crucial but often overlooked problem: the performance of models trained on single-leaf images often degrades when presented with a full plant image, where the system must first identify and isolate the leaves before detecting the disease. This is a primary reason for the low precision in real-time applications.

The main goal of this thesis is to bridge these gaps by:

1. Proposing a fast and efficient leaf segmentation algorithm to accurately isolate individual leaves from complex plant images, which is a critical preprocessing step for improving detection precision.
2. Integrating this segmentation model with a robust disease detection model to create a multi-stage, high-precision system.
3. Validating this complete system's performance in a real-time environment, providing a comprehensive solution that moves beyond theoretical accuracy to practical applicability for a targeted spray prescription system.

2.6 Summary

This chapter first outlines the evolution of plant disease detection from traditional inspection to early technological methods like spectroscopy and basic image processing, which were limited by their sensitivity to environmental conditions. It then details the paradigm shift brought by the advance deep learning architectures, highlighting their capability to automatically extract features from images. While many studies have shown high accuracy with models trained on clean and controlled datasets, the review identifies a key research gap:

the failure of these models to generalize to real-world field conditions where diseases need to be identified on individual leaves within complex, cluttered plant images. To address this, the chapter discusses more advanced techniques like object detection (YOLO, SSD) and image segmentation (U-Net, Mask R-CNN). The summary concludes that this research work provides a novel, multi-stage, end-to-end solution that integrates these techniques to bridge the gap between lab-based accuracy and real-time application in agriculture.

Chapter 03: Real-time Object Detection Frameworks for Agricultural Applications

This chapter specifies a comprehensive investigation of the real-time object detection frameworks that form the technical core of the proposed system for automated tomato disease detection. This chapter focuses on the specific models and their architectural features that make them suitable for the unique challenges of real-time agricultural applications. This chapter is structured as follows: Section 3.1 covers the evolution of object detection, with a particular emphasis on single-stage detectors that prioritize speed, a crucial factor for this research. Section 3.2 focused on the YOLOv5 architecture. In Section 3.3 YOLOv7 is discussed in detail. Section 3.4 explains the YOLOv8 architecture while section 3.5 focuses on the YOLOv9 architecture. Section 3.6 details the experimental setup for model comparison. Section 3.7 provides the experimental results and comparative analyses of the above discussed architectures. Section 3.8 reviews the consequences of our findings, and best model selection. Finally, Section 3.9 summarizes the chapter.

3.1 Object Detection Models

The application of object detection models in agriculture has moved from theoretical concepts to practical, real-world solutions. For an automated system to be effective in a dynamic environment, it must require real-time visual data processing. This chapter delivers a detailed comparative analysis of prominent object detection frameworks, with a focus on their suitability for real-time plant disease detection. The objective is to evaluate how well these models balance speed and accuracy, and to justify the selection of the most suitable model for the proposed system. This section examines the architectural designs, strengths, and weaknesses of several key object detection models. The models chosen to represent a progression in the field, moving from established, accurate models to faster, more efficient ones.

Object detection is a core computer vision task that contains two objectives: classifying the class of objects present in an image and localizing them with a bounding box. Historically, this task was performed by multi-stage detectors like R-CNN and Faster R-CNN, which first proposed regions of interest and then classified each region in a separate step. While highly accurate, this two-stage approach was computationally expensive and too slow for real-time applications. In contrast, the YOLO framework revolutionized the field by introducing a single-stage detection paradigm. The principal concept behind YOLO is to frame object detection as

a single regression problem, processing an entire image in a unified forward pass of a CNN to predict bounding boxes and class probabilities simultaneously. This method significantly boosts inference speed, making it suitable for real-time applications.

The foundational YOLO architecture is built upon several key concepts. The input image is divided into a fixed grid (e.g., an S × S grid). Each grid cell can detect an object whose centre falls within it. Each grid cell predicts a predefined number of bounding boxes. For each box, it outputs a set of parameters: the coordinates of the box's centre (x, y), its width (w), and its height (h), all normalized to the image dimensions. Alongside the bounding box coordinates, each cell predicts a confidence score. This score reflects two things: the probability that the box contains an object, and how accurately the model believes the predicted box fits the object (Intersection over Union with the ground truth). Each grid cell also predicts a set of conditional class probabilities, one for each possible class. This allows the model to classify the object it detects. During post-processing, Non-Maximum Suppression (NMS) is used to remove unnecessary or intersecting bounding boxes. It filters out boxes with low confidence scores and keeps only the highest-scoring box among those that overlap significantly. YOLO is not a single model but a family of architectures that has continuously evolved to improve upon its original design, with each version introducing pivotal innovations. This chapter represents a detailed comparison of YOLOv5, YOLOv7, YOLOv8 and Yolov9 models.

3.2 YOLOv5 Architecture

The YOLO family represents a cornerstone of modern real-time computer vision. Following the release of YOLOv4, a new iteration emerged not from the original authors but from Ultralytics: YOLOv5. Despite initial controversy over its nomenclature due to the lack of a peer-reviewed paper at launch, YOLOv5 has recognized itself as a dominant force in both industrial applications and academic research. Its significance lies not in radical architectural invention, but in the masterful synthesis of state-of-the-art components into a highly efficient, scalable, and user-friendly framework built on PyTorch [64]. The YOLOv5 architecture follows a modern detector design pattern, comprising three distinct parts: Backbone, Neck, and Head components. A simplified diagram illustrating the flow of information through the YOLOv5 model, highlighting the Backbone, Neck, and Head components and the three detection scales as shown in figure 3.1. For feature extraction the data is provided to the CSP (Cross Stage Partial) Darknet initially, which is then forwarded to the PANet (Path Aggregation Network) for feature fusion. And finally, it is fed to the YOLO layer which provides the final output as object detection along with class, score, location, and size of the object.

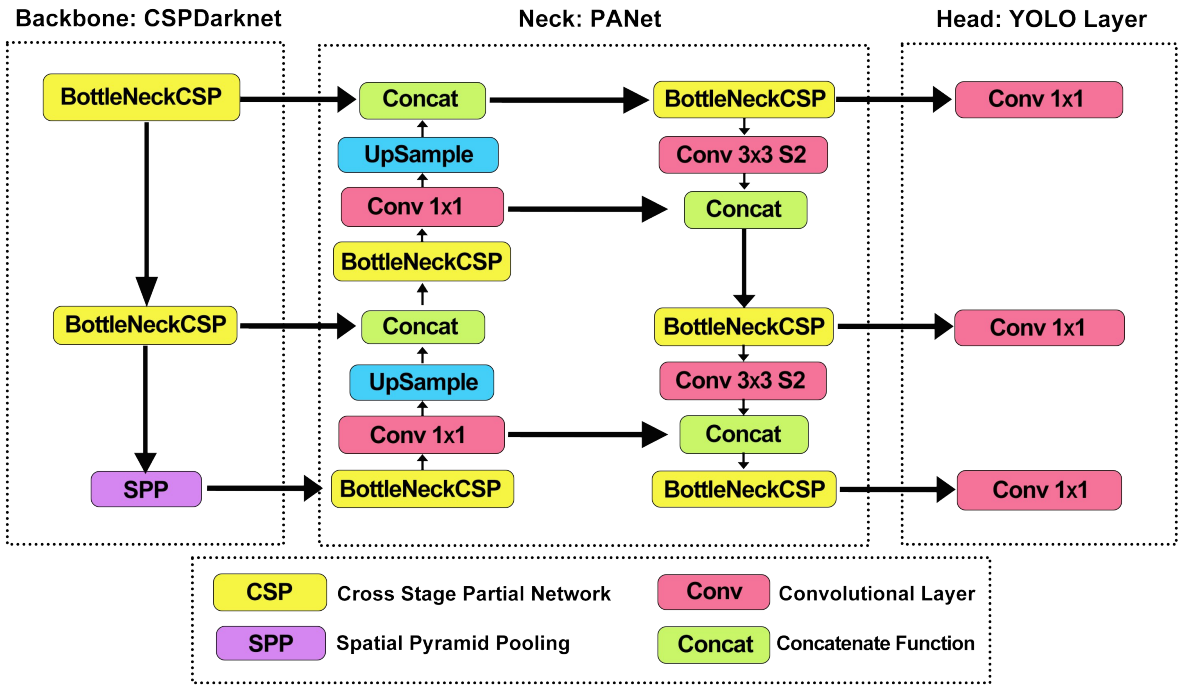


Figure 3.1 YOLOv5 core architecture

3.2.1 Backbone

A feature extraction network (CSP-Darknet53) responsible for deriving hierarchical feature maps from an input image. The backbone is based on Darknet53 but incorporates CSPNet [65] to form CSPDarknet53. The CSP structure divides the feature map into two parts. One part undergoes a dense block (a sequence of convolutions), while the other part bypasses this block. The two are subsequently concatenated. This design achieves two critical goals; Mitigates Gradient Vanishing and Reduces Computational Cost**.** At the end of the backbone, YOLOv5 employs an SPPF (Spatial Pyramid Pooling - Fast) layer. This module is functionally equivalent to the SPP module [66] used in YOLOv4 but is optimized for speed. It performs serial max-pooling functions with a fixed kernel size (e.g., 5x5) on the same input feature map, concatenating the results. This process generates a fixed-length output that captures multi-scale contextual information, building the model more robust to variations in object scale, but does so more efficiently than a parallel SPP implementation.

3.2.2 Neck

The YOLOv5 architecture utilizes a modified PANet in its neck. The main function of PANet is feature aggregation, which fuses features from the backbone's various stages to enhance the overall feature representation [67]. PANet enhances the top-down pathway of a Feature Pyramid Network (FPN) by enhancing a second, bottom-up path aggregation. This creates a bi-directional feature pyramid. The top-down path FPN propagates strong semantic features

from deeper, high-level layers to shallower, low-level layers. The bottom-up path PANet augments this by directly propagating accurate localization signals from the lower, high-resolution layers to the higher layers. This bi-directional flow ensures that the features fed into the head are rich in both high-level semantic information (crucial for classification) and low-level fine-grained spatial information (crucial for precise object localization), particularly benefiting the detection of small objects.

3.2.3 Head

The detection layer that performs the final bounding box regression and object classification. The detection head is anchor-based and predicts bounding boxes at three different scales (e.g., 80x80, 40x40, 20x20), which makes it possible to detect objects of different sizes. For each grid cell in these feature maps, the network predicts:

- **Bounding Box Attributes:** Four values (t_x, t_y, t_w, t_h) defining the centre coordinates, width, and height relative to the grid cell and its assigned anchor box.
- **Objectness Score:** A scalar value indicating the probability that the bounding box holds an object.
- **Class Probabilities:** A vector of probabilities for each class, conditional on there being an object.

The final output is processed by Non-Maximum Suppression (NMS), which filters out overlapping detections, redundant and low-confidence scores.

3.3 YOLOv7 Architecture

YOLOv7, was released in 2022 by [68] which contributes a major step forward in the YOLO family. It distinguishes itself by achieving its successful optimization of the speed-accuracy trade-off, establishing a new state-of-the-art for real-time object detection. Instead of focusing solely on a new architecture, YOLOv7's core contribution is a meticulously designed set of "Trainable Bag-of-Freebies" and architectural reforms that significantly improve performance without increasing inference costs. The model is highly efficient, outperforming all previous YOLO versions and other real-time detectors on both speed and accuracy. The YOLOv7 architecture maintains the standard three-part structure but introduces key modifications for enhanced efficiency and performance. The overall architecture of YOLOv7 is complex due to the various modules and connections. The figure 3.2 shows the input, the E-ELAN (Extended Efficient Layer Aggregation Network) backbone, the re-parameterized PANet in the neck, and the final detection head. The auxiliary heads are also typically shown branching off the neck to assist in deep supervision.

3.3.1 Backbone

The backbone of YOLOv7 is the E-ELAN module, which is an advanced version of the ELAN used in Scaled-YOLOv4. The primary goal of E-ELAN is to improve the capability of network's learning and convergence exclusive of disrupting the gradient flow. It achieves this by utilizing group convolution to simultaneously boost the number of channels and the cardinality (the size of the feature set) within a computational block. The feature maps from different computational blocks are then shuffled and merged. This design permits the network to acquire a more distinct set of features with fewer parameters and lower computational costs.

3.3.2 Neck

The neck (Re-parameterized PANet and SPP) of the network is responsible for combining features from diverse scales. YOLOv7 utilizes a modified PANet structure, like YOLOv4 and YOLOv5, which uses both top-down and bottom-up paths for feature fusion of different sizes. However, YOLOv7 introduces re-parameterization strategies at a module level. For example, some convolutional layers are designed to have a redundant structure during training that is then merged into a single layer during inference. This "re-parameterization" technique improves the model's performance while maintaining a fast inference speed. The SPPCSPC (Spatial Pyramid Pooling and Cross Stage Partial) module is also integrated to increase the receptive field and provide a richer feature representation [69].

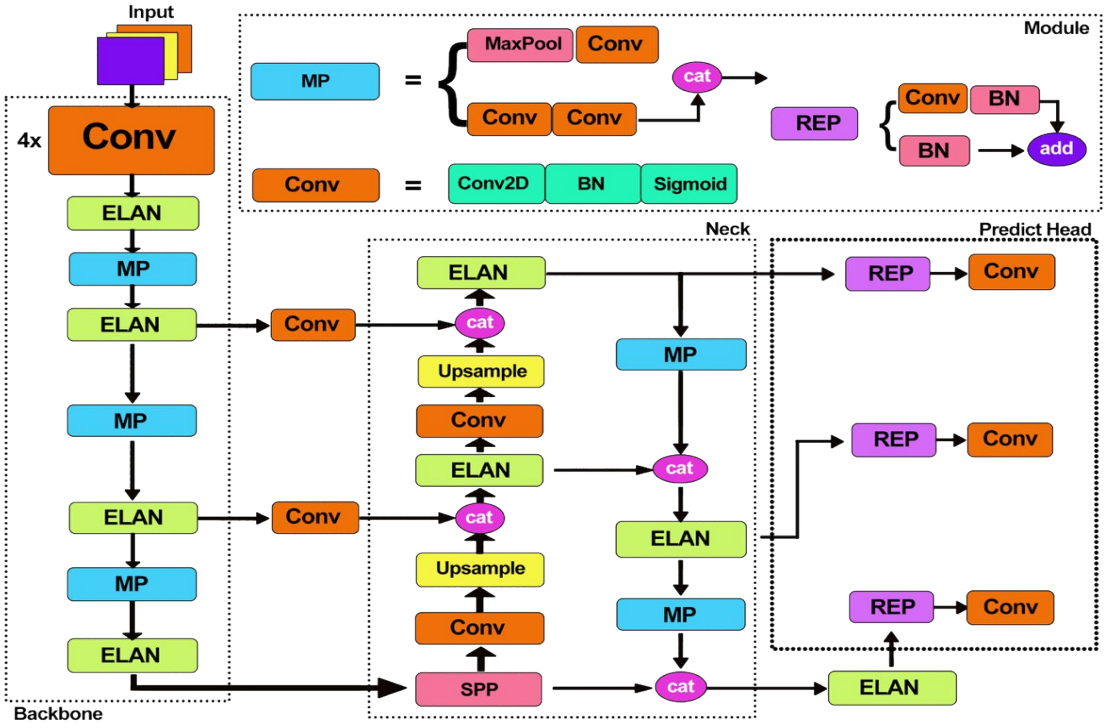


Figure 3.2 YOLOv7 core architecture

3.3.3 Head

Auxiliary and Lead Heads YOLOv7 introduces a novel concept of deep supervision by incorporating auxiliary heads in addition to the primary lead head. The final output is provided by the lead head, while the auxiliary heads are added to the middle layers of the network. These auxiliary heads are also trained with a loss function to guide the learning process of the deeper layers. By justifying the issue of vanishing gradients, this approach allows the network to acquire more dependable features.

3.4 YOLOv8 Architecture

YOLOv8 developed by Ultralytics, represents a significant evolution in the YOLO family. YOLOv8 was released as a state-of-the-art model that builds on the successes of its predecessors with several key architectural and functional improvements, focusing on performance, flexibility, and efficiency [70]. It is designed to handle a full range of vision AI tasks beyond just object detection, including pose estimation, instance segmentation, and classification [71]. The architecture of YOLOv8 follows the classic single stage detector design, comprising three main parts: Backbone, Neck, and Head. the general architecture can be represented as a flow from the Backbone, through the Neck, to the Head, highlighting the key components. The figure 3.3 illustrates the general flow and key blocks.

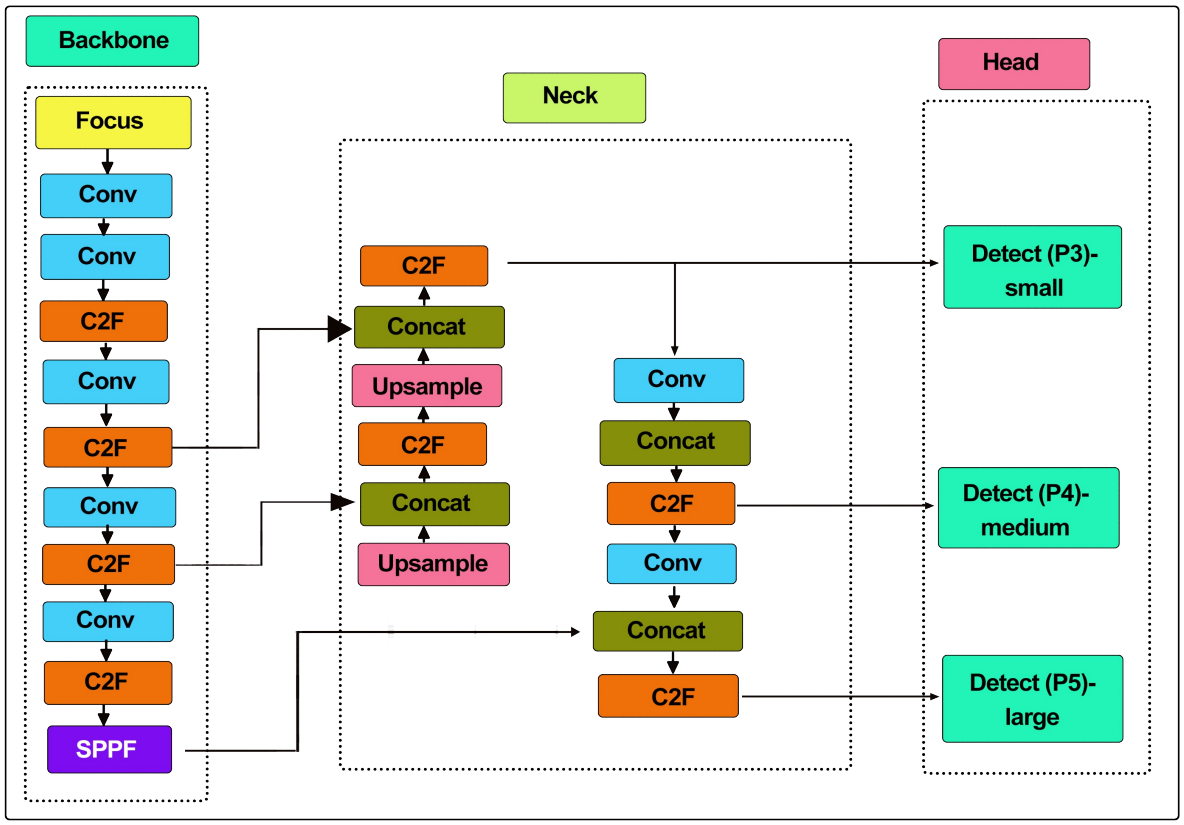


Figure 3.3 YOLOv8 core architecture

3.4.1 Backbone

The backbone of YOLOv8 is a refined version of the CSPDarknet53 backbone, which uses a CSP network structure maintaining high accuracy with moderate computational complexity.

3.4.2 Neck

The neck connects the backbone and the head. It's designed to fuse feature maps from different scales, ensuring the model can detect multiscale objects. For this purpose, YOLOv8 uses a PANet, but with a new C2f module (a variation of the C3 module in YOLOv5). The C2f module combines a higher fusion of high-level semantic features and low-level spatial information, which is a significant process for small object detection.

3.4.3 Head

The final part of the network is the head that is responsible for generating the outputs. A major innovation in YOLOv8 is the anchor-free split Ultralytics head. Unlike previous YOLO versions that utilizes predefined anchor boxes, YOLOv8 directly predicts the object's centre coordinates, width, and height [72]. This anchor-free approach simplifies the training process, improves generalization, and reduces the number of predictions, making it more efficient and versatile for different datasets. The head also leading to better model performance by separating the classification and regression tasks.

3.5 YOLOv9 Architecture

YOLOv9, introduced in early 2024 by [73], is a state-of-the-art object detection model that breaks new ground by fundamentally addressing the problem of information loss in deep neural networks. Its key contribution is a novel methodology and two primary architectural components that allow the network to retain a greater amount of critical information during training, leading to superior performance with fewer parameters and computational requirements. This innovation is a direct response to a fundamental limitation known as the information bottleneck principle in deep learning. YOLOv9's performance comes from two groundbreaking concepts: the Generalized Efficient Layer Aggregation Network (GELAN) and Programmable Gradient Information (PGI). Figure 3.4 illustrates the interaction between these new components. The input image first passes through the GELAN backbone. The output of this backbone is then fed into the PGI system, which contains the main branch and an auxiliary reversible branch. The auxiliary branch provides a clean gradient signal to the main branch during training. The final output is then produced by the detection head.

3.5.1 Generalized Efficient Layer Aggregation Network (GELAN)

This is the new, highly efficient backbone of YOLOv9. GELAN extends the concept of ELAN by making it more general [74] . It is designed to allow the use of various computational blocks, such as convolutions, without the need for depth-wise convolutions. This architectural design ensures that the network has excellent parameter utilization, meaning it can attain a high level of performance with a relatively small number of parameters. GELAN is engineered to maintain a stable and robust gradient path, which is crucial for efficient and stable training.

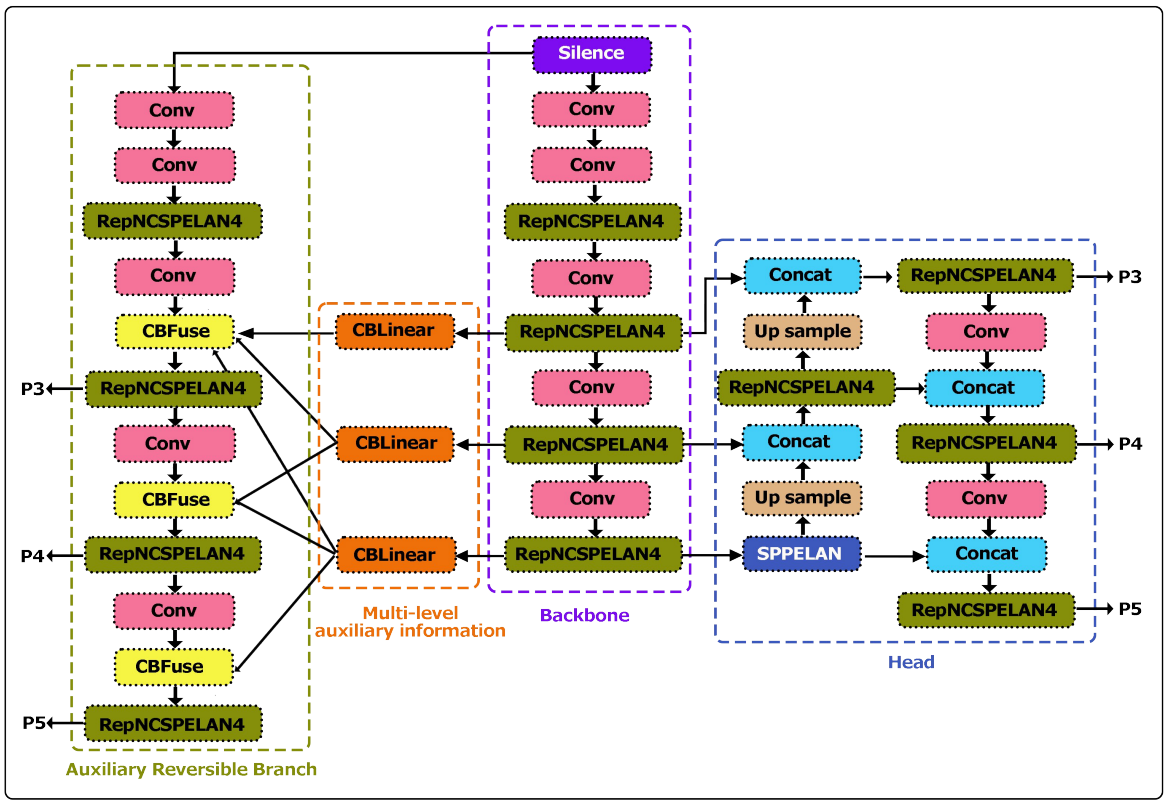


Figure 3.4 YOLOv9 core architecture

3.5.2 Programmable Gradient Information (PGI)

This is the core conceptual innovation. PGI is a framework designed to counteract the loss of crucial information as data flows across the deep layers of a network. The PGI mechanism operates through three components:

- **Main Branch:** The major network path used for inference. It is optimized for speed and efficiency.
- **Auxiliary Reversible Branch:** This is a separate, training-time-only branch that runs in parallel with the main branch. It is a reversible architecture, meaning it can reverse the transformation of data to ensure that no information is lost. This provides a

"complete" and "clean" source of gradient information to the main branch, helping it learns more effectively.

- **Multi-level Auxiliary Information:** PGI addresses the "deep supervision" problem where errors can accumulate in very deep networks. By using the auxiliary branch to provide high-quality gradients to various levels of the main network, PGI ensures that even the deepest layers receive meaningful information, leading to better-trained weights.

The brilliance of PGI is that the auxiliary branch is removed during inference, so there is no added computational cost or latency. The benefits of the enhanced training are "free" at runtime.

3.6 Experimental Setup for Model Comparison

A rigorous comparison of different YOLO models requires a meticulous experimental setup to ensure the results are fair, reproducible, and academically sound. This section outlines the key components for comparison in this work. To ensure your results can be replicated, it's essential to specify the exact computational environment used for all experiments. Python was chosen as a programming language for both ease of development and performance. Utilizing NVIDIA Quadro K2200 GPU, for building the deep learning architectures we have achieved efficient training and performance. Leveraging the memory, 640 CUDA cores, and high bandwidth. powerful combination of Keras, TensorFlow backend, and the CuDNN library, we achieved efficient training on an NVIDIA Quadro K2200 GPU (HP, Italy) with its 4 GB memory, 640 CUDA cores, and high bandwidth. This section provides an in-depth methodology of real-time object detection. Figure 3.5 shows the different stages of the object detection process, such as data collection, image preprocessing, model training, model testing and evaluation.

3.6.1 Dataset and Data Preprocessing

The dataset is the foundation of any experiment. A clear description ensures that the findings are well-contextualized. The PlantDoc dataset is used for the comparative analysis of the YOLO models in this section. In plant disease detection the PlantDoc dataset is a valuable resource for research using computer vision. Unlike many datasets captured in controlled laboratory settings, PlantDoc was created using images scraped from the internet, which provides a more realistic and challenging environment for models to learn from. This makes it an excellent benchmark for evaluating models in real-world agricultural scenarios.

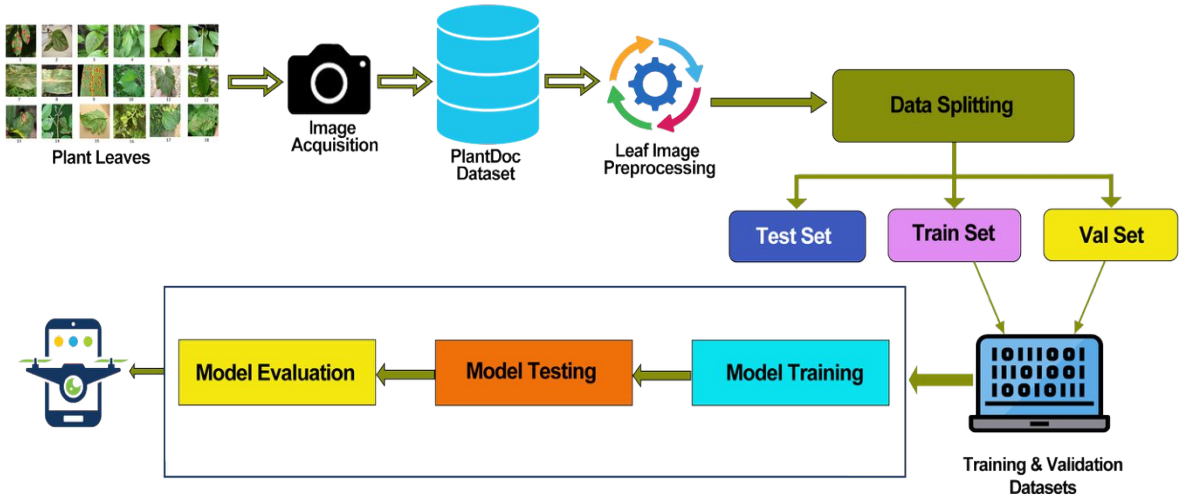


Figure 3.5 Implementing YOLO family evaluation based on plant disease detection

The dataset was created by researchers at the Indian Institute of Technology to address the lack of large-scale, non-lab-based data for computer vision models in agriculture. The primary goal was to create a dataset that could help in the early and scalable detection of plant diseases. The initial effort involved approximately 300 hours of human annotation. The images capture various plant species and their corresponding diseases, as well as healthy leaves, under diverse conditions, including varying backgrounds, lighting, and occlusions. The PlantDoc dataset has evolved slightly in its various releases and pre-processed versions. Here are the key statistics from the official and widely used versions: The dataset contains approximately 2,569 images. It includes 30 classes in total, covering a range of diseases and plant species. The format we have used in this section including YOLO format (.txt files), for easier use with YOLO frameworks. The dataset is typically pre-divided into a training set (70%), validation set (20%) and test set (10%).

The PlantDoc dataset is considered more challenging than many other plant disease datasets due to its "in the wild" nature. Key challenges for object detection models include:

- **Complex Backgrounds**: Images often feature cluttered backgrounds like soil, other plants, or buildings, forcing models to distinguish the leaf from its surroundings.
- **Non-Uniform Appearance**: Disease symptoms can appear as small, subtle spots or large, irregular patches, requiring models with strong multi-scale detection capabilities.
- **Class Imbalance**: The dataset reveals a significant class imbalance, where some diseases are far more common than others. This can pose challenges for training and may require specific techniques like weighted loss functions or data augmentation.

- **Real-World Variability**: The images are subject to variations in lighting, shadows, and leaf pose, making the task of robust detection more difficult.

3.6.2 Model Architectures and Hyperparameters

This section must demonstrate that all models were trained under the most comparable conditions possible to isolate architectural performance.

- **Model Variants:** The variants of each YOLO version used are (YOLOv5m, YOLOv7, YOLOv8m, and YOLOv9m).
- **Initialization:** All the models were initialized with pre-trained weights from the COCO dataset that provides a strong baseline for all models, which is an industry standard and ensures that the models do not start from scratch.
- **Training Parameters:**
    - **Epochs:** Number of training epochs used (60).
    - **Image Resolution:** All models are trained and validated at the same input image resolution (640x640 pixels).
    - **Batch Size:** Batch size used across all PyTorch-based models (v5, v7, v8, v9) are set to 16.
    - **Optimizer:** Adam optimizer has been used for all YOLO models.

3.6.3 Evaluation Metrics and Methodology

The final section defines how the models are quantitatively compared.

- **Accuracy Metrics:**
    - **mAP@0.5:** The mAP at an Intersection over Union (IoU) threshold of 0.5. This is the most common metric for single-class object detection and is easy to interpret.
    - **mAP@0.5:0.95:** The mAP averaged over ten IoU thresholds from 0.5 to 0.95. This is the standard COCO metric and provides a more comprehensive view of localization accuracy.
- **Inference Speed Metrics:**
    - **FPS (Frames Per Second):** The number of images processed per second. This is a crucial metric for real-time applications. Report the FPS on the same hardware setup.
    - **Latency:** The time in milliseconds (ms) to perform inference on a single image. This is a more precise measure of speed.
- **Model Efficiency Metrics:**
    - **Model Size:** The size of the trained weight file in megabytes (MB).
    - **Parameters:** The total number of parameters in the model. This indicates the model's complexity and memory footprint.
    - **FLOPs:** The number of Floating-Point Operations is a measure of the model's computational complexity.

3.7 Results and Analysis

The section explains the expertness and possibility of algorithm to adapt to different situations by examining different metrics. Accuracy and relevance are two indicators employed to decide the model's efficiency. Alongside the accuracy scoring, we have evaluated the algorithm's inference time for its efficiency. The inference time is the total time it takes the algorithm to detect objects in an image. All the algorithms were used to detect different object, and their accuracy were then compared based on the above discussed metrics.

The performance of the trained YOLOv5 model is first evaluated, then YOLOv7, YOLOv8 and at the end the performance of YOLOv9 is evaluated, followed by a detailed analysis and the derived practical recommendations. The above discussed metrics summarizes the performance of all models on the test dataset following a detailed evaluation. The confusion matrix generated for each model summarize the performance of each model on the provided dataset are shown in figure 3.6-3.9 respectively.

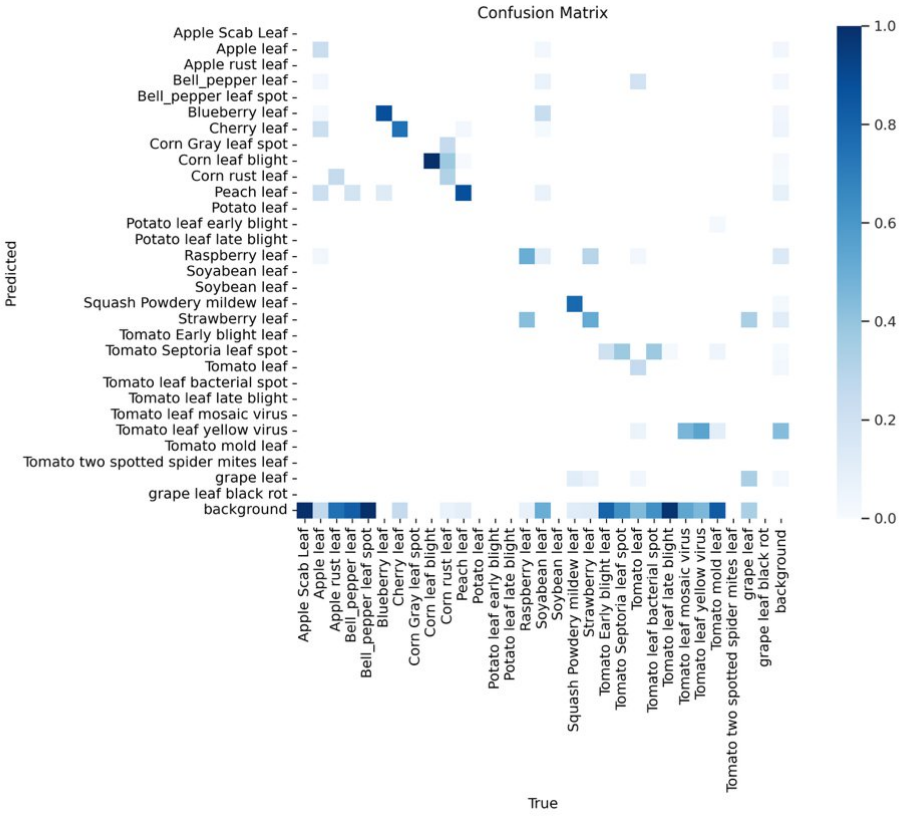


Figure 3.6 YOLOv5 Confusion Matrix

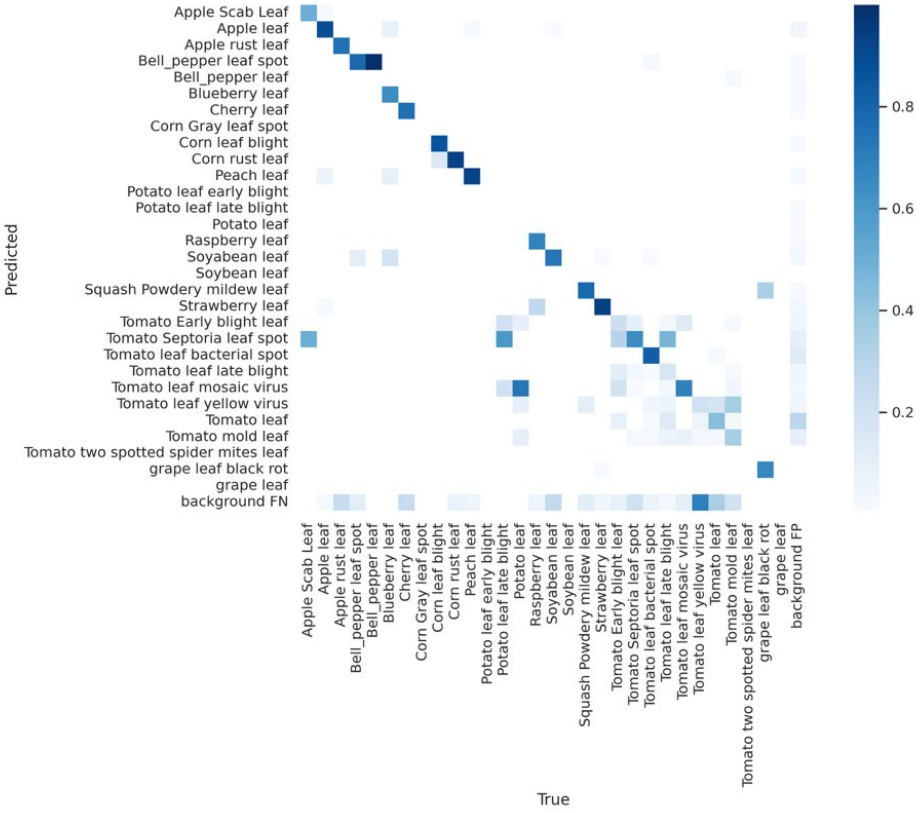


Figure 3.7 YOLOv7 Confusion Matrix

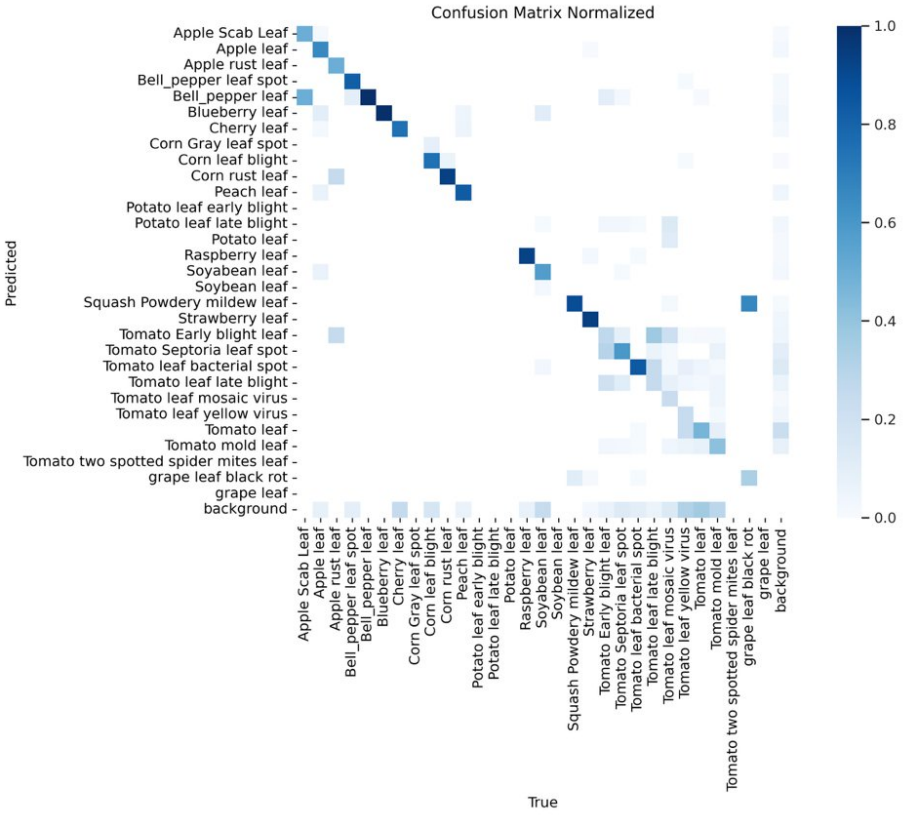


Figure 3.8 YOLOv8 Confusion Matrix

The confusion matrix is a basic tool for assessing a classification model's performance. In this matrix, the sum of samples on the main diagonal indicates the number of correct classifications for each class. Conversely, cells outside the diagonal represent samples that the model misclassified into another class. Beyond basic testing, we thoroughly evaluated the model by using a range of IoU thresholds, at a moderate 0.5. This approach gave us a much clearer picture of its performance, especially under conditions requiring very precise bounding box placement.

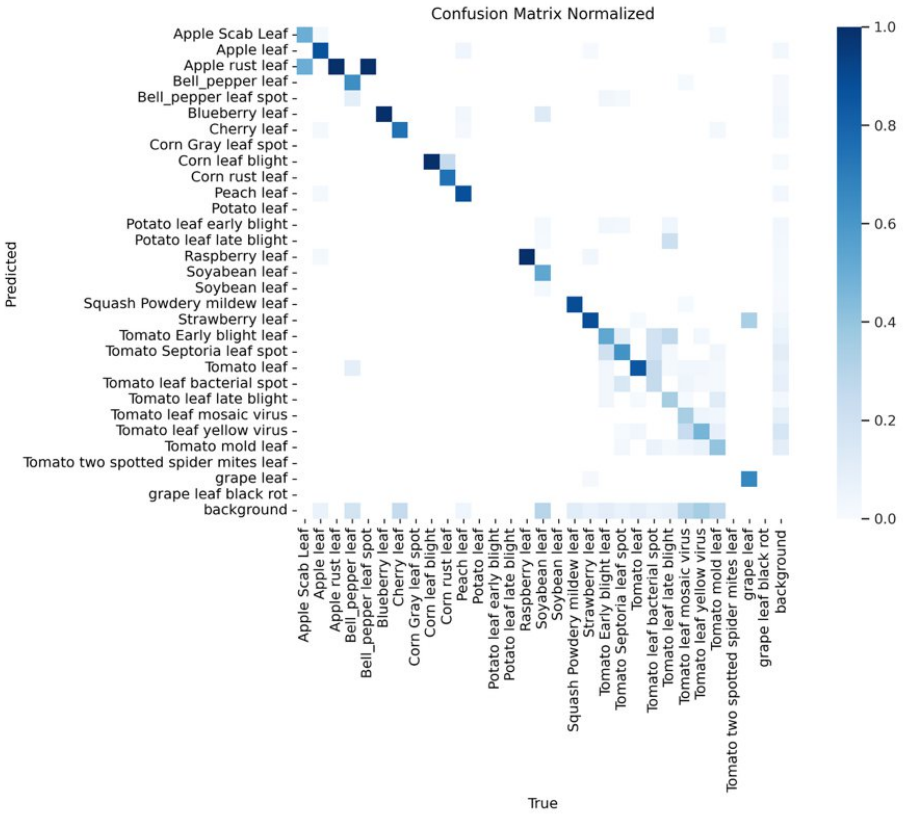


Figure 3.9 YOLOv9 Confusion Matrix

The training visualization graph in figure 3.10-3.13 clearly shows the mAP scores at these varied IoU levels. In training visuals, we keep an eye on two main things. The "train/box_loss" essentially tells us how "on target" our predicted boxes are. When you see this number dropping, it's a clear sign the model is becoming much improved at correctly finding and outlining objects. Similarly, the "train/cls_loss" tracks how good the model is at naming what it sees. If this number is also going down, it means the model is improving its identification skills, which is crucial when its job is to spot plant diseases on leaves.

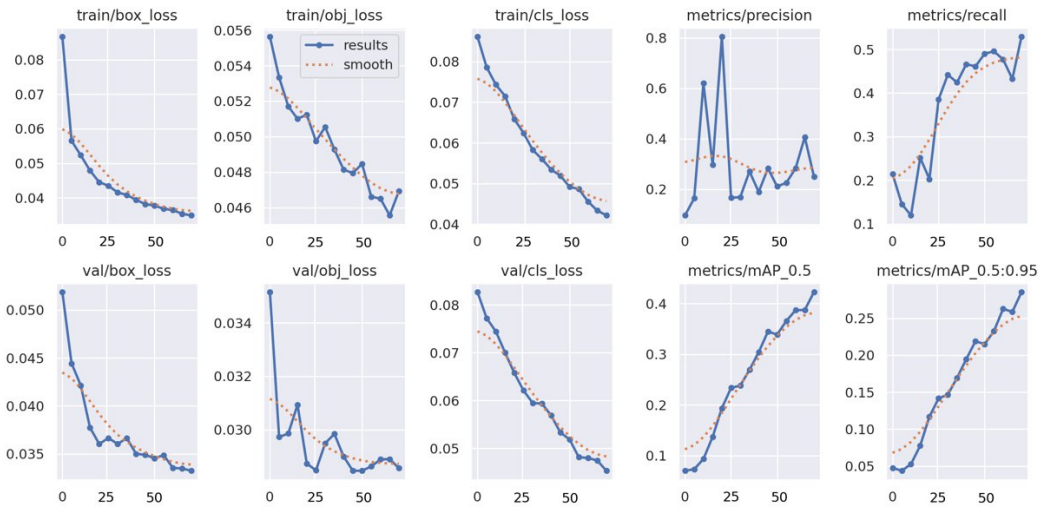


Figure 3.10 YOLOv5 Training Visualization

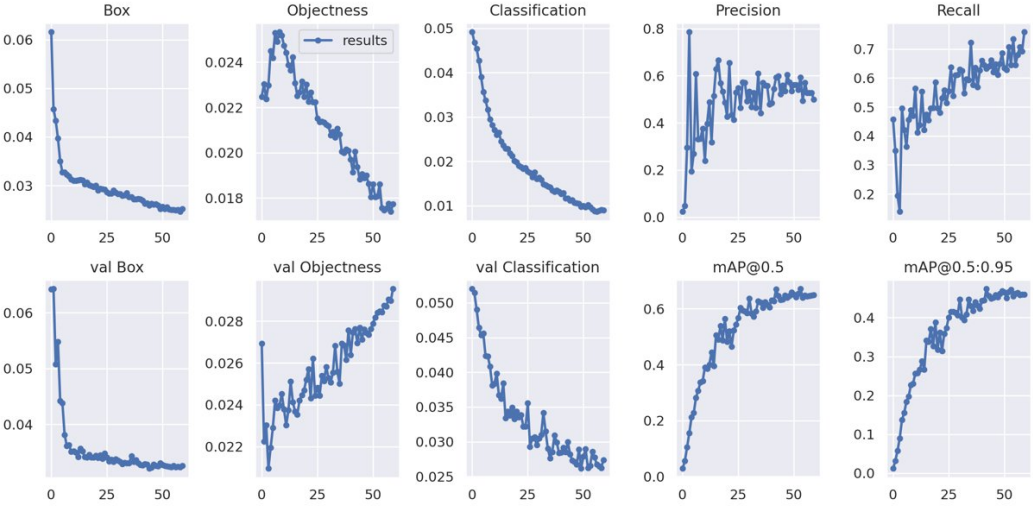


Figure 3.11 YOLOv7 Training Visualization

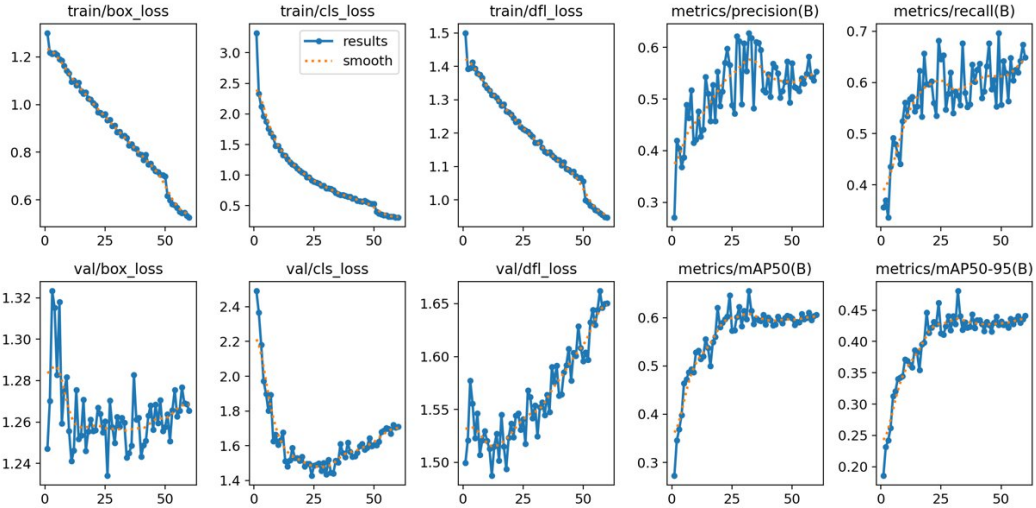


Figure 3.12 YOLOv8 Training Visualization

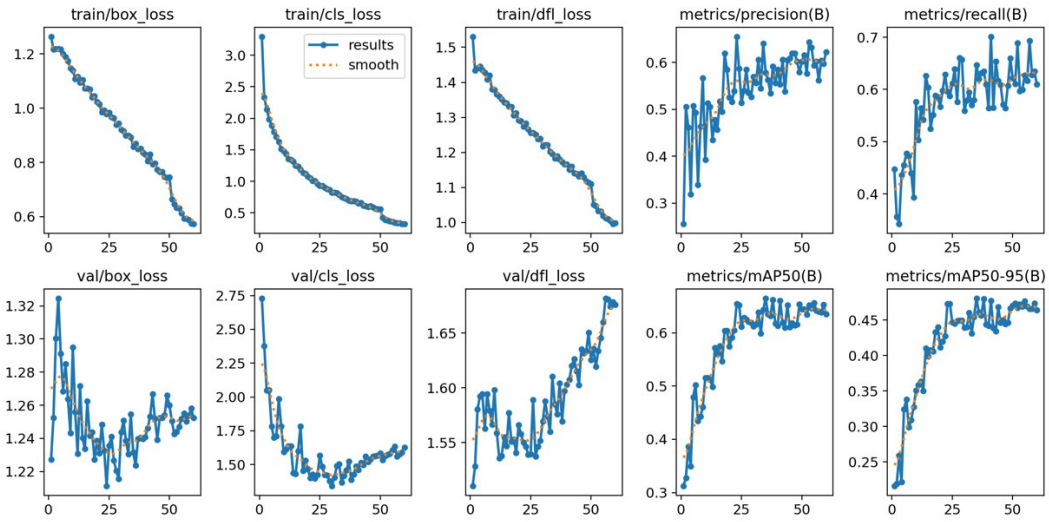


Figure 3.13 YOLOv9 Training Visualization

All the YOLO models in this work, specifically trained for the identification of diseases on different plants, demonstrated robust performance across the test dataset. As shown in Table 3.1, the quantitative evaluation was based on standard object detection metrics, specifically Recall (R), Precision (P) and mAP@50. For the plant disease detection, the performance evaluation metrics are listed below (1)–(3).

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

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

$$\text{IoU} = \frac{(TP)}{(TP + FN + FP} \tag{3}$$

where:

- FP: False Positives (incorrect positive predictions)
- TP: True Positives (correctly positive predictions)
- FN: False Negatives (incorrectly negative predictions)

Table 3.1 shows the performance evaluation of all the models in more detail describing each metrics associated with each model.

Table 3.1 Model evaluation metrics comparison

Model	Model Size (MB)	Precision (%)	Recall (%)	mAP@50 (%)	Inference Speed
YOLOv5m	24.5	0.76	0.86	0.42	32.5ms
YOLOv7	75.1	0.95	0.97	0.64	39.4ms
YOLOv8m	52.0	0.993	0.93	0.65	23.5ms
YOLOv9m	40.9	0.992	0.94	0.65	26.4ms

The results in Table 3.1 shows the accuracy and efficiency of each model in detecting plant leaf diseases within the provided dataset along with the corresponding sizes for each model. These results really reflect real-time capability for all the models, which is important for real-time applications demanding rapid decision-making, such as disease detection. In terms of model size, YOLOv7 was the largest, while YOLOv5 was the smallest, while in terms of speed YOLOv8 process the image quickly in 23.5ms and YOLOv7 was slower with the inference time 39.4ms recorded during the model comparison.

Apart from this the results from validation and test datasets serve distinct but complementary roles in assessing the performance of model. The validation batch results provide a continuous, real-time evaluation of the learning progress of model, and generalization during the training phase as shown in figure 3.14-3.17 for each model. At the end of each training epoch, a small batch of validation data is used to calculate key metrics like accuracy and validation loss. These metrics are critical for hyperparameter tuning and for implementing early stopping, which is vital for preventing the model from overfitting the training data.

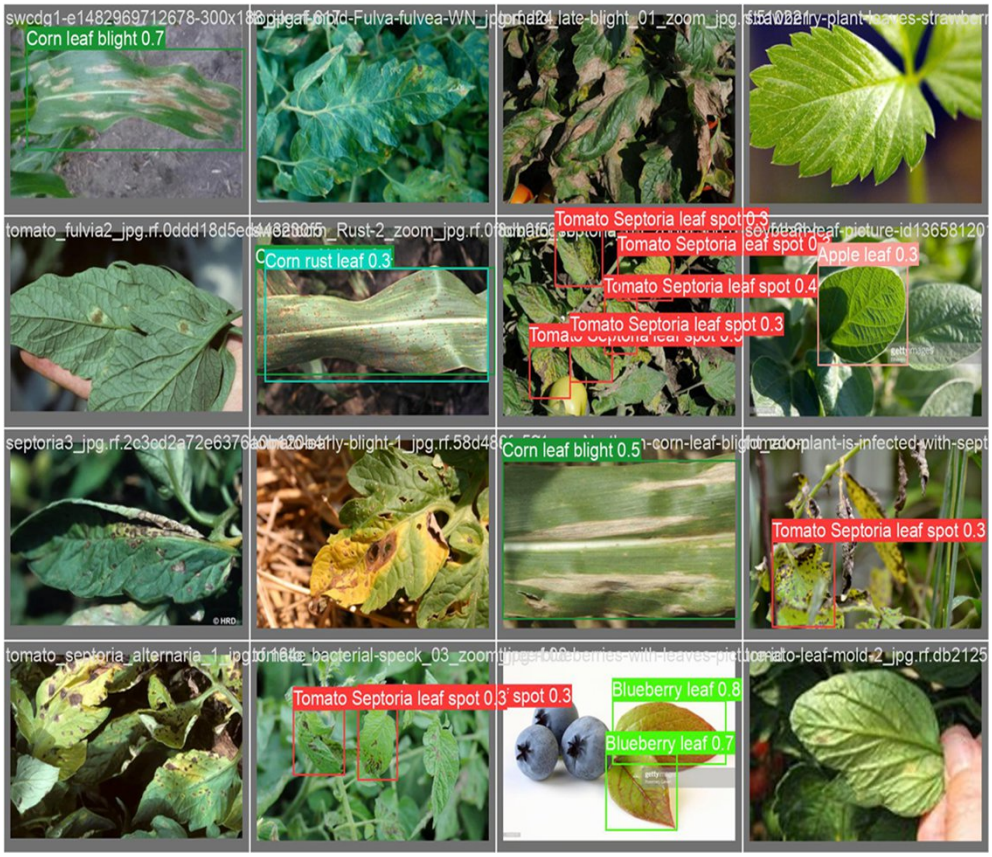


Figure 3.14 Validation results of YOLOv5

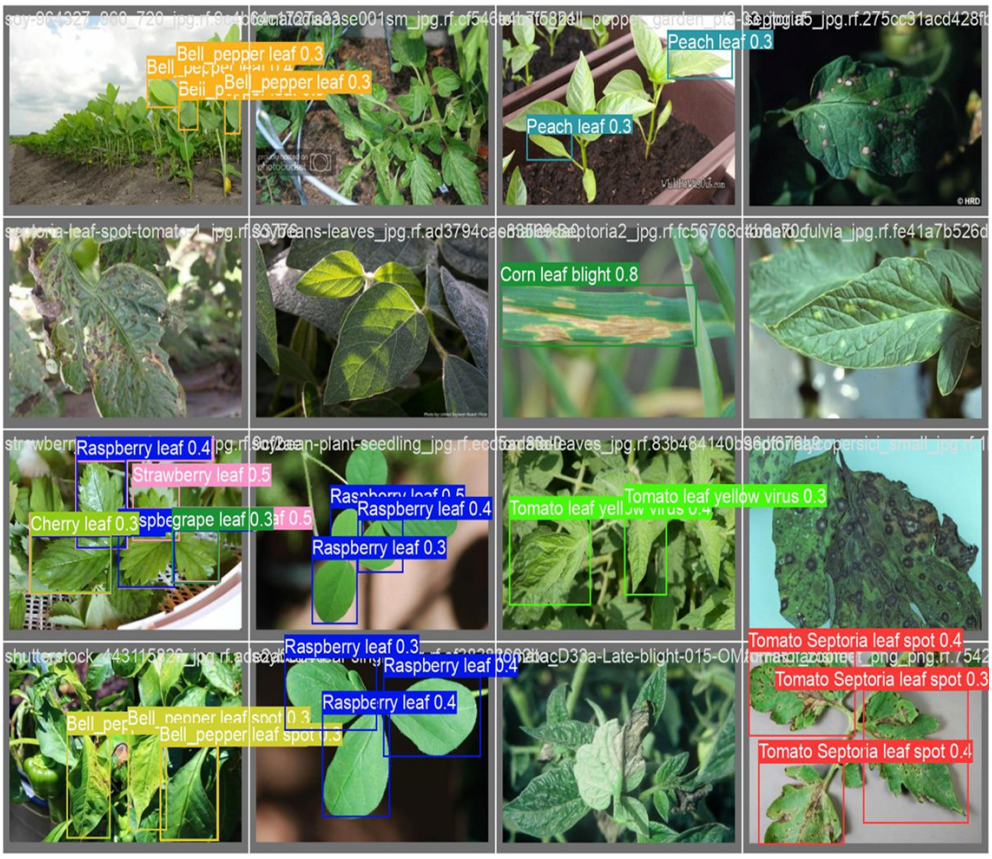


Figure 3.15 Validation results of YOLOv7

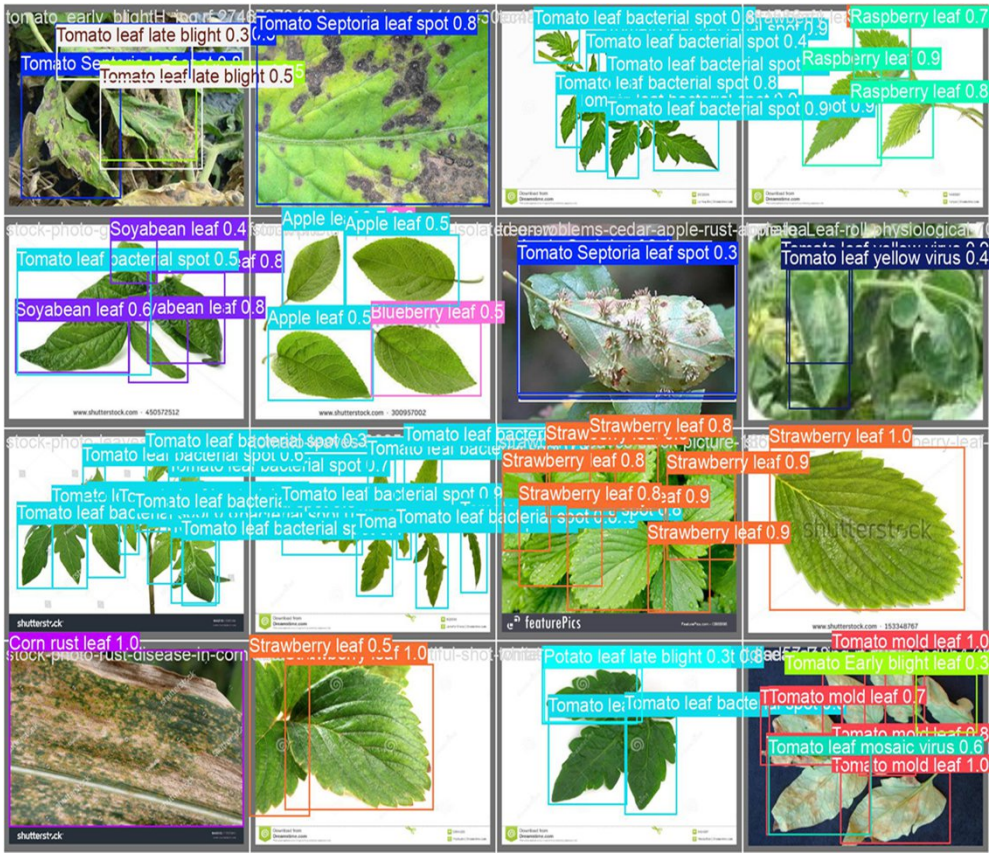


Figure 3.16 Validation results of YOLOv8

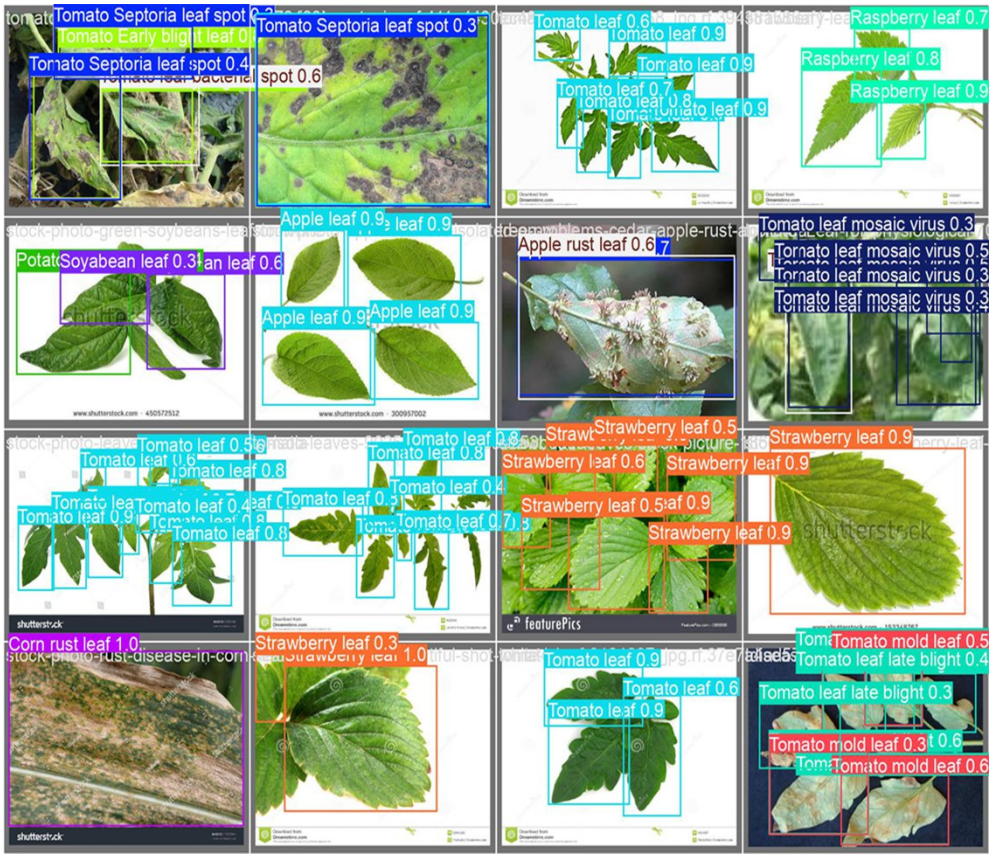


Figure 3.17 Validation results of YOLOv9

In contrast, the test data results are the absolute, unbiased measure of the model's performance. The test set is completely held out from both the training and validation processes, and the model's performance on it is used only once, after all training and hyperparameter tuning is complete. The results from the test data provide the definitive metrics on the ability of model to simplify to new, unseen data, and they are used to compare the final selected model against other competing models. The result of test data for each model is shown in the figure 3.18.

After evaluation on the test dataset using similar data as shown in figure 3.18, YOLOv8 and YOLOv9 achieved the higher accuracies of 98% and 97% respectively. These figures highlight their superior capability to correctly identify and classify objects. YOLOv7 followed with a strong accuracy of 89%, placing it in the middle range, while YOLOv5 noted the lowest comparative performance achieving the accuracy of 75%. The visualization of comparatives analysis with the previous models with respect to precision, recall and mAP@50 metrics is presented in figure 3.19.

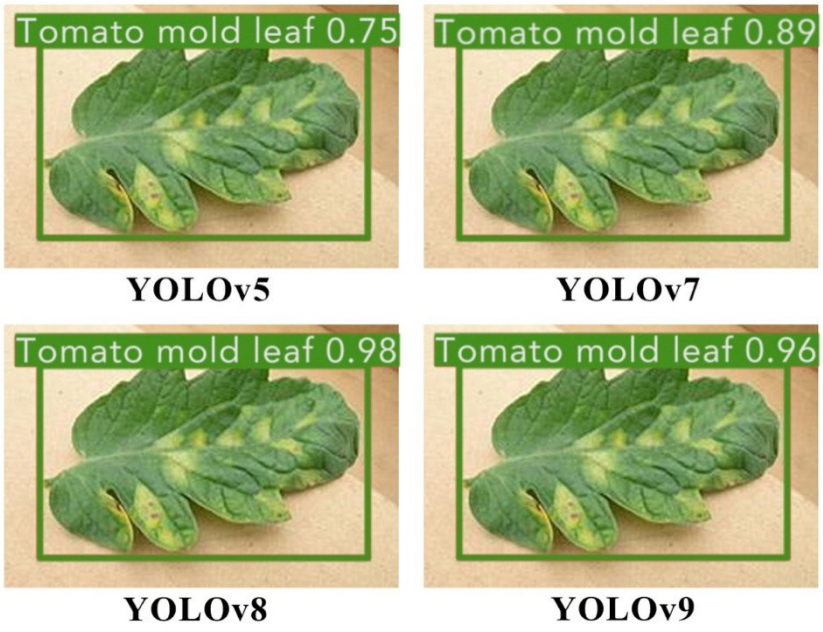


Figure 3.18 Results on test dataset

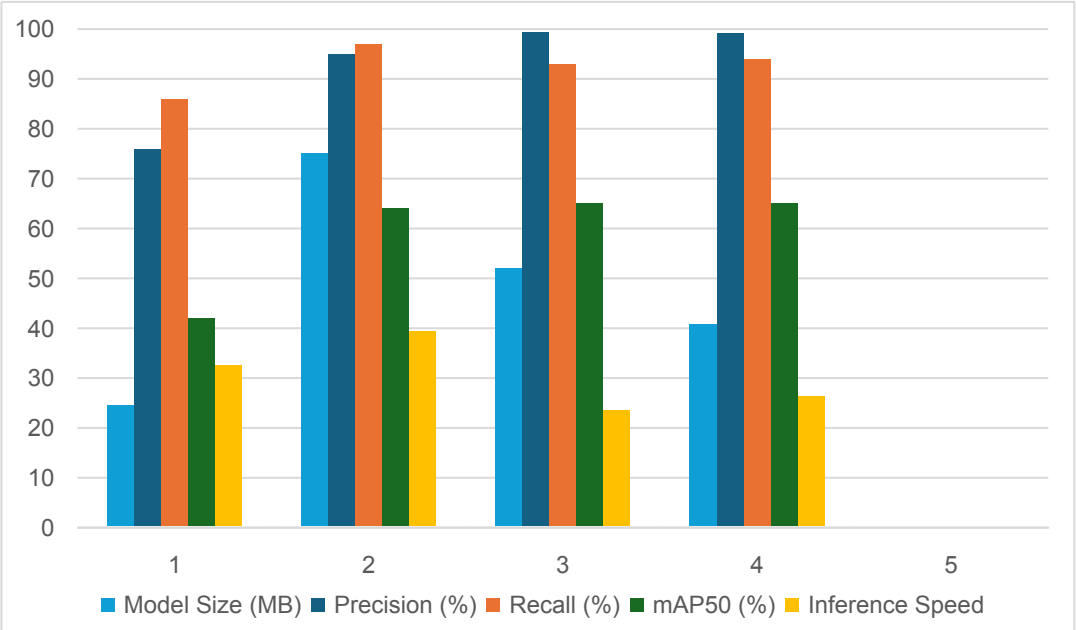


Figure 3.19 Model's Evaluation based on model size, precision, recall, and mAP@50

3.8 Discussion

The empirical results from this study, reveal a clear hierarchical trend in model performance, which directly corresponds to the architectural advancements across the YOLO family. The findings determine a substantial leap in classification accuracy from older to newer generations of the framework, with YOLOv8 and YOLOv9 attaining the best performance. The superior accuracy of YOLOv8 (98%) and YOLOv9 (97%) can be directly attributed to their innovative architectural and training methodologies. YOLOv8's success stems from its shift to an anchor-

free design and a decoupled head, which separate localization and classification tasks. This approach reduces the complexity of bounding box prediction and is particularly effective on a plant diseases dataset, where disease symptoms can have highly variable shapes and sizes. Simultaneously, YOLOv9's competitive performance is a testament to its groundbreaking PGI and GELAN. The PGI mechanism effectively addresses the information bottleneck problem in deep networks, ensuring that vital gradient information is preserved and preventing data loss. This permits the model to learn more robust and fine-grained features, crucial for correctly identifying subtle disease patterns. The negligible difference in accuracy between these two models suggests that both architectural philosophies are highly effective and that their slight performance variation may be a result of specific hyperparameter settings or data characteristics.

The performance of YOLOv7 (89%) represents a substantial improvement over its predecessors and serves as a powerful testament to the value of its re-parameterization and deep supervision strategies. By providing a stronger gradient signal to its deeper layers and optimizing the network for inference, YOLOv7 accomplished an outstanding increase in accuracy competed to YOLOv5. However, it did not reach the performance level of the anchor-free designs in YOLOv8 and YOLOv9, highlighting that these more recent innovations represent a significant paradigm shift in object detection. Finally, the results show that YOLOv5 (75%), while a capable and user-friendly model, exhibited a comparatively lower accuracy. This is primarily due to its reliance on an anchor-based architecture and a less sophisticated feature fusion network. The performance of the models on the PlantDoc dataset, with its complex, real-world imagery, demonstrates that without the advanced architectural refinements of its successors, a traditional YOLO design struggles to achieve the same level of precision, particularly in distinguishing subtle features from cluttered backgrounds.

3.9 Summary

In this chapter we compared different YOLO models to detect the plant diseases leveraging advance deep learning architectures. To achieve this, the PlantDoc dataset was utilized consisting of 2,569 images spanning 30 distinct classes. These classes included various plant diseases such as apple black rot, apple scab, apple healthy, apple cedar rust, cherry powdery, cherry healthy, blueberry healthy, mildew, corn grey leaf spot, corn common rust, corn healthy, grape black rot, grape leaf blight, grape black measles, grape healthy, peach healthy, orange citrus greening, pepper bell bacterial spot, pepper bell healthy, potato late blight, potato early blight, potato healthy, raspberry healthy, soybean healthy, squash powdery mildew, strawberry leaf scorch, strawberry healthy, tomato early blight, tomato bacterial spot, tomato late blight,

tomato healthy. Consequently, experimentations were carried out, assessing the performance of YOLOv5, YOLOv7, YOLOv8, and YOLOv9 models. YOLOv8 and YOLOv9 demonstrated the best performer among the models evaluated. YOLOv8 recorded a total inference time of 23.5ms, with pre-processing and post-processing accounting for 6.2ms and 0.9ms, respectively. In comparison, YOLOv9 showed a slightly slower total time of 26.4ms, with its pre-processing and post-processing times being 6.4ms and 1.2ms. This comparative analysis illustrates the rapid evolution of the YOLO framework. The empirical evidence strongly suggests that the transition to more streamlined, anchor-free, and information-preserving architectures is key to achieving state-of-the-art results in challenging agricultural computer vision tasks. For applications requiring maximum accuracy, models like YOLOv8 and YOLOv9 are clearly the superior choice. Future work will involve expanding training datasets to include other categories of plant diseases and determining disease location in the infected leaf. Ultimately, this study emphasizes the evaluation of YOLO algorithms in plant disease detection, leading to easier implementation and more flexible accuracy control.

Chapter 4: Tomato Leaf Detection, Segmentation, and Extraction for Accurate Disease Detection

The accurate detection and classification of diseases in plant foliage is a pivotal task in modern agriculture, serving as a cornerstone for timely intervention and improved crop yield. While advancements in real-time object detection have enabled the identification of disease symptoms, the accuracy of such systems is often compromised by the complexities of scenarios in real-world applications. Factors such as occlusions, variable lighting, and cluttered backgrounds can introduce noise, leading to misclassification and reduced diagnostic reliability. This chapter addresses these limitations by introducing a novel, multi-stage methodology focused on isolating and extracting the tomato leaf itself before performing disease analysis.

This chapter provides a three-phase method designed to mitigate the effects of environmental noise and provide a cleaner, more focused input for disease detection. The first phase employs a robust object detection model to accurately locate and bound the tomato leaf within the image. Subsequently, the second phase utilizes an instance segmentation model to generate a precise mask of the detected leaf, effectively separating it from the background at the pixel level. In the final phase, this mask is used to extract and isolate the leaf, creating a segmented image that contains only the relevant plant tissue. This process guarantees that the subsequent disease classification is performed on a clean, noise-free input, thereby maximizing accuracy and diagnostic confidence. The remainder of this chapter will explain the specific models and methodologies employed in each phase, present the experimental setup, and discuss the results of this approach in enhancing the performance of disease detection systems.

The proposed chapter comprises five sections. We outline the structure of this chapter as follows: In Section 4.1 and 4.2, the introduction has been discussed in detail; Section 4.3 presents the materials and methods; in this section, the methodology of the proposed work has been discussed in detail; Section 4.4 consists of the results which demonstrate the simulation results using YOLOv8 and SAM; Section 4.5 consists of a discussion on the pros and cons of the proposed model along with the comparison of our work with the previous studies; Section 4.6 presents the summary.

4.1 Introduction

In recent years, the integration of computer vision and machine learning techniques in agriculture has revolutionized crop management and disease detection [75]. Amongst the

various applications, particularly in tomato plants, the exact segmentation and analysis of plant leaves have gained significant attention because of their importance in early disease detection, nutrient deficiency identification, and overall plant health monitoring [76]. Plant leaf segmentation is an important aspect of computer vision and agricultural technology. It supports various applications, including disease monitoring, growth monitoring, and yield prediction. However, there has been a great improvement in the deep learning domain, due to which the methods and results of leaf segmentation processes have become more efficient and effective [77].

Tomato happens to be one of the most widely cultivated as well as economically important vegetable crops across the globe. However, tomato plants are affected by different diseases and environmental conditions that lower the yield and quality of the crop. Therefore, timely notice and correct assessment of plant health conditions are essential to enable quick response actions for minimization of the loss of the crops [78]. Most conventional methods of plant monitoring are based on physical investigations, which are labour-intensive, time-consuming, and are also error-prone because of human factors. New computer vision approaches can help solve some of these problems because they provide a way to evaluate the state of crops quickly, correctly, and at large scales [79]. The authors in [80] stated that constructing an object detection problem in terms of a regression task was possible because of the introduction of the YOLO architecture. In [81], [82] the authors have conducted several studies that showed the high efficacy of YOLOv8 in the image segmentation tasks and further object detection tasks.

Likewise, the authors in [83] have also proposed the SAM, which has allowed for significant advancement in image segmentation as it can segment any object in an image based on different types of prompts such as text, boxes and points. SAM is envisioned as a promptable segmentation system that can create a mask for every target that can be found on an image, all based on the type of text prompt it receives. The main features of SAM include:

- SAM can segment objects accurately without prior training.
- SAM has input prompts flexibility such as accepting points, masks, text, or boxes
- as prompts.
- SAM training is performed on 11 million images and 1.1 billion masks.
- For interactive applications, SAM can accurately generate masks in real time.

It is worth noting that SAM was not specifically developed for agricultural uses; however, its potential functionality can be proved useful in other tasks, such as leaf segmentation. The authors in [84] showed how SAM can be utilized in crop monitoring by examining the crop features in detail and making attempts to obtain such features for the leaves of the plants.

Advanced models for tomato leaf segmentation have immense research potential due to reconciling the intricacies of agriculture image analysis with the advancements in computer vision technologies. Leaf segmentation is very important for the later steps of the analysis, such as in the case of disease detection, growth monitoring, and yield prediction [85]. Individual leaf extraction from complex backgrounds which include other parts of the plant, soil, and agricultural equipment, is considered a complex task. These models have the potential to provide a robust foundation for more specialized agricultural AI applications.

The rapid and accurate identification of plant disease in real-time agricultural applications represents a major challenge for enhancing crop yields [86]. For the accurate identification of plant disease, accurate detection of individual plant leaves is very important for the development of smart agriculture systems. Most of the researchers use synthetic images for model training and testing. There when deployed that model in real-time environment, its precision suffers because of testing on entire plant images as it is trained on leaf images. For this purpose, we have used YOLOv8 and SAM, two recent deep-learning models. In this work, we have focused on tomato leaf detection, segmentation, and extraction.

The major goal of this chapter is to recognize the capabilities of both YOLOv8 and SAM regarding tomato leaf detection and segmentation. We measure their performance around accuracy, efficiency, and how well they cope with changes in their surroundings. Apart from this, we have investigated the potential of these models in this study to handle the inherent challenges in agricultural imaging, such as varying lighting conditions, natural variability in leaf shapes and sizes, and occlusions. The results of this work are very useful for creating automated tomato plant monitoring systems in the agricultural sector. We aim to expand the boundaries of precision agriculture by utilizing the best features of YOLOv8 and SAM, which will allow farmers and researchers to make data-based crop management and disease control decisions.

4.2 Challenges in Real-time Leaf Processing

Agriculture is a fundamental sector that has a direct impact on society and the economy. However, for better crop yield, earlier and accurate detection of plant diseases in real- time environment poses a major challenge to the agriculture production. Before the detection of plant diseases in earlier stage, it is very important and challenging to precisely detect the individual plant leaves for developing a smart agricultural system. The accurate detection and classification of diseases in plant foliage is a pivotal task in modern agriculture, serving as a cornerstone for timely intervention and improved crop yield. While improvements in object

detection at real-time environment has enabled the identification of disease symptoms, the accuracy of such systems is often compromised by the complexities of real-world environments. Factors such as occlusions, variable lighting, and cluttered backgrounds can introduce noise, leading to misclassification and reduced diagnostic reliability. Most of the researchers use synthetic images for model training and testing. So, while testing that model in real-time environment, its precision suffers because of testing on entire plant images as it is trained on leaf images. To enhance the performance of plant leaf disease detection in a real-time environment we need to detect individual leaves accurately. To overcome these limitations this chapter introduces a novel, multi-stage methodology focused on isolating and extracting the tomato leaf itself before performing disease analysis.

4.3 Proposed Method for Leaf Detection and Segmentation

In this section, a detailed methodology for leaf detection, segmentation, and extraction is discussed by integrating the YOLOv8 and SAM. The proposed approach influences the strengths of YOLOv8 for efficient tomato leaf detection and SAM for high-quality segmentation within detected bounding boxes. Figure 4.1 illustrates the workflow of the proposed approach.

4.3.1 Dataset Preparation

The dataset used consist of higher resolution images of tomato plants, taken from the TomatoVillage [87]. The images were obtained from contributions wide range of publicly available sources to ensure variability in light conditions, the angle of leaves as well as the stage of development of the plants. The images were larger, originally comprising 640 x 640 pixels. All the images were resized to 640 x 640 pixels for achieving consistency in model training and testing. The model used was also augmented by performing simple augmentations like random rotation, horizontal flip, and changing brightness. After collecting the dataset was annotated using the autodistill process. The dataset comprises 11,493 images. The annotated data is divided into training (70%), validation (20%) and testing (10%) sets. This dataset is further provided to the YOLOv8.

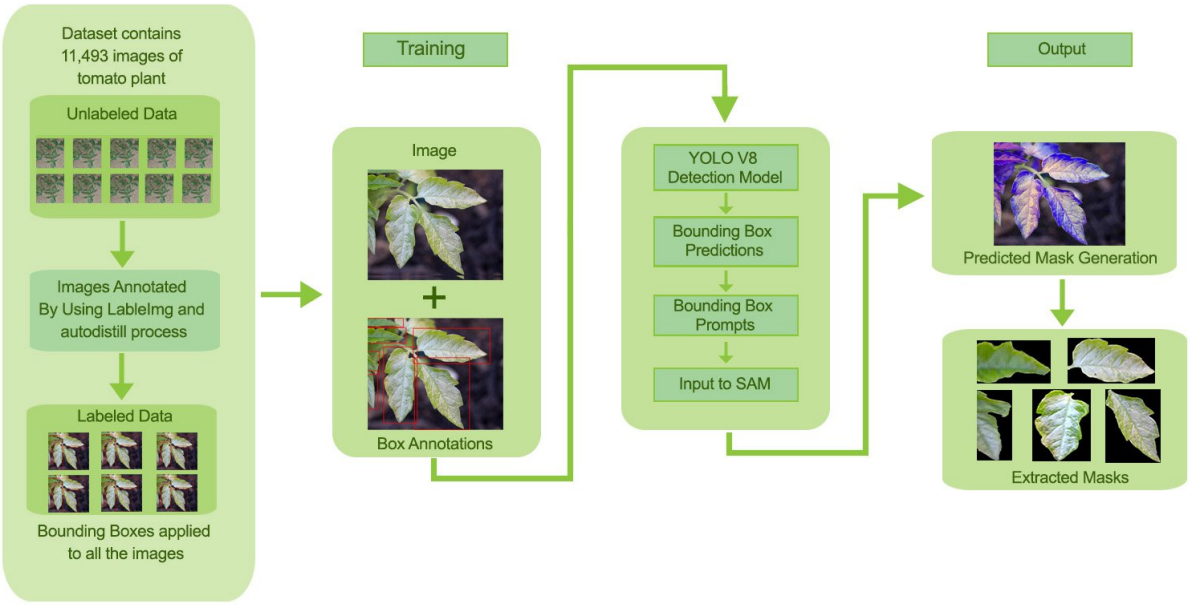


Figure 4.1 Methodology for Tomato Leaf Detection, Segmentation, and Masks Extraction

4.3.2 YOLOv8 for Leaf Detection

For detecting tomato leaves in tomato plant images, we employed the YOLOv8 model. Due to low computational cost, higher accuracy and efficiency in real-time environment YOLOv8 is preferred. The boxes were automatically encased around the tomato leaves themselves by using the YOLOv8, which was later refined through SAM. The YOLOv8 was employed for detection tasks. The network was initialized using the pre-trained weights from the COCO dataset and then adjusted to the tomato leaf dataset. The unlabelled data for YOLOv8 was annotated using the autodistill. With the help of autodistill, we can train small, faster-supervised models with big slower foundation models. We can generate labelled data using autodistill to inference on a custom model operating at the edge. For using the autodistill, we feed unlabelled data into a base model. This model uses a predefined ontology to automatically generate a labelled dataset which is then used to train the target model, resulting in a distilled model fine-tuned for a specific task. The autodistill process is shown in the figure 4.2. Autodistill comprises the following steps:

Task: The Task explains what will be predicted by the target model. The base model, the ontology, and the target model within the autodistill pipeline must all be mutually matched and work together seamlessly.

Base Model: It is a multimodal and large foundation such as GoundedSAM that can perform many tasks. We create the dataset from the unlabelled data along with the ontology by using the base model.

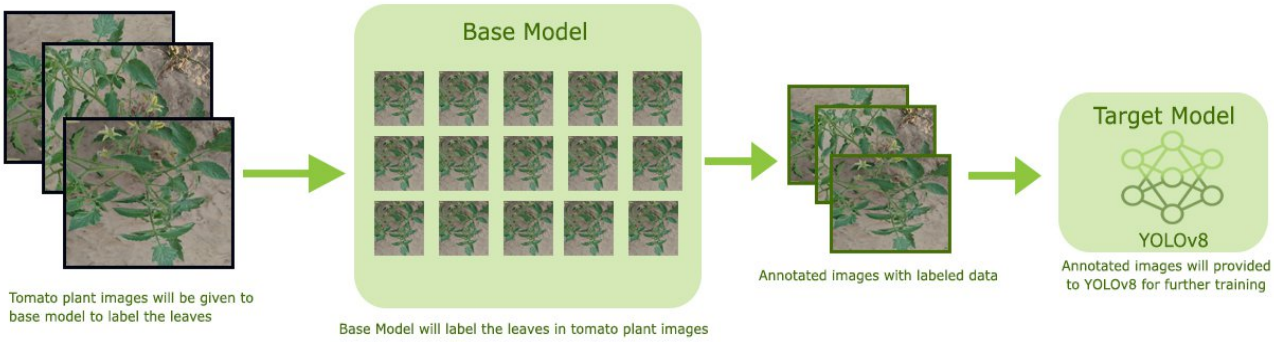


Figure 4.2 Autodistill Process for auto-labelling the unlabelled data

Ontology: The Ontology defining the labels and classes that the dataset will explain, and the target model will ultimately predict. A simple example is the CaptionOntology, which uses simple text captions to prompt a Base Model and maps those prompts to specific class names. More advanced Ontologies might use a CLIP vector or example images instead of text. This chapter is focused on detecting the tomato leaf, so the CaptionOntology was like "leaf" = "tomato leaf".

Dataset: The dataset is the output generated by the base model known as labelled data. To train the target model this labelled can be used. In our case our target model is YOLOv8.

Target Model: This is the supervised model that will consume the labelled dataset generated by the base model and that is ready for implementation. For performing the specific task, the target model is fined-tuned and that usually small and fast. In our case the target model is YOLOv8.

Distilled Model: It is the final product of the autodistill process. It consists of a set of weights fine-tuned for your specific task, which can be deployed to generate predictions.

After getting the labelled data it is split into the training (70%), validation (20%), and testing (10%) sets. It is provided YOLOv8 for training which is the target model. The resolution of the images was reduced to 640 x 640 pixels for this purpose before feeding to the YOLOv8.

The training parameters are as follows:

-Batch Size: 16

-Epochs: 100

-Learning Rate: 0.001

-Optimizer: Adam optimizer is used to minimize the loss function.

Output: YOLOv8 has given the output of bounding box coordinates representing each leaf identified together with their respective confidence scores. YOLOv8 has trained well enough to recognize distinctive leaves in the tomato plant images, more so in instances where leaves

may be overlapping or occluded. SAM received bounding boxes obtained from YOLOv8 as preliminary data for the generation of final segmentation masks.

4.3.3 Segment Anything Model (SAM) for Leaf Segmentation

SAM is used to segment the tomato leaves within YOLOv8 bounding boxes. The SAM has also been designed as a prompt-based segmentation model, whereby once a bounding box or points are provided as prompts to it, fine segmentation occurs. Our focus was targeted segmentation, so we implemented SAM ViT-H (Vision Transformer with a hierarchical structure) and trained on large-scale datasets amenable to segmentation tasks. We picked up a pre-trained checkpoint ("sam_vit_h_4b8939.pth") which was tailored to our task. For speed of inference, the model was executed on 2 GPU 12GB NVIDIA GeForce GTX 1080 Ti. For generating high-quality segmentation masks, SAM is used for each detected bounding box from YOLOv8 as illustrated in Figure 4.3.

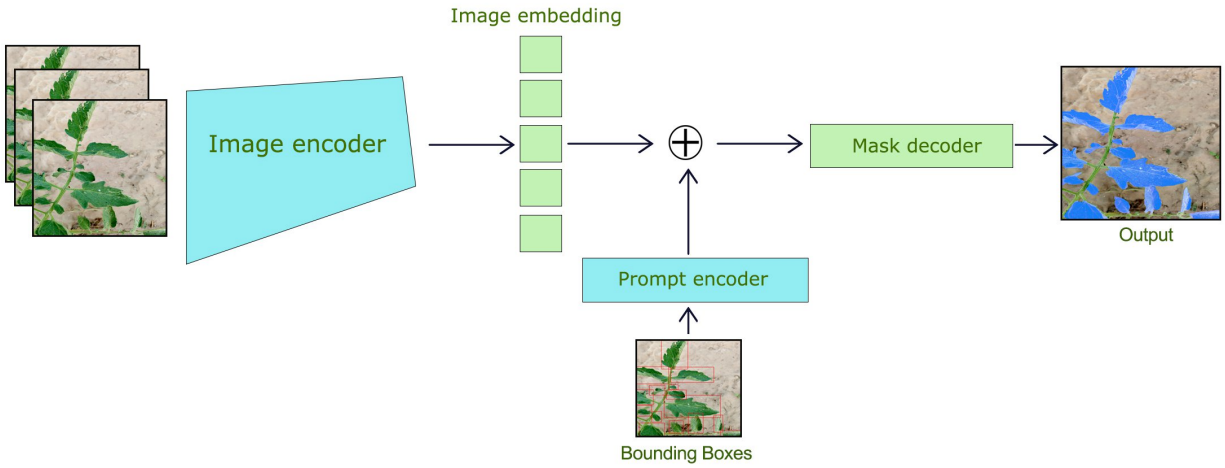


Figure 4.3 Segment Anything Model for Tomato Leaf Segmentation

The steps involve in the process of segmentation are as follows:

- **Input Image and Bounding Box Setup:** The RGB image was fed to SAM along with the bounding box previously generated by the YOLOv8.
- **Image Preprocessing:** The image was normalized and re-sized per the input specification of SAM.
- **Segmentation Output:** SAM provided binary masks portraying the segmented leaf areas inside the defined bounding boxes. SAM's features along with YOLOv8 have the ability to work around the complex boundaries and the occlusions, each mask was an accurate representation of the leaf's shape.

The segmented leaves after the segmentation process were then extracted from the image for further analysis or be used in subsequent applications such as disease detection. For every segmented leaf, the following post-processing techniques were implemented:

- **Mask Refinement:** The binary masks were refined by erosion and dilation such as morphological techniques to smoothen the object boundaries.
- **Leaf Extraction:** The mask was used to cut out each leaf from the image while the coordinates of the bounding box were used to define the area of the leaf.
- **Resizing:** The size of the cutout leaf areas was resized to 640 x 640 pixels using the bilinear interpolation method so that the samples remained consistent.

The extracted leaves were saved and transferred to a designated folder for further analysis. The extracted leaf masks were exported in a PNG (Portable Network Graphics) format whereas a corresponding mask was recorded for use in the performance assessment.

The performance of YOLOv8 is assessed through the computation of precision, recall, and F1 confidence score. The performance of SAM is assessed with standard segmentation metrics, which include the IoU, Dice coefficient, and qualitative visual assessments. We also report the computational efficiency of how well YOLOv8 and SAM perform together to segment the tomato leaves. For the tomato leaf detection, the performance evaluation metrics are listed below (1)–(4).

$$\text{recall} = \frac{TP}{TP+FN} \quad (1)$$

$$\text{precision} = \frac{TP}{TP+FP} \quad (2)$$

$$\text{f1-score} = \frac{(2*TP)}{(2*TP+FN+FP} \quad (3)$$

$$\text{IoU} = \frac{(TP)}{(TP+FN+FP} \quad (4)$$

$$\text{dice coefficient} = \frac{(2*TP)}{(2*TP+FP+FN} \quad (5)$$

where:

- FP: False Positives (incorrect positive predictions)
- TP: True Positives (correctly positive predictions)
- FN: False Negatives (incorrectly negative predictions)

4.4 Experimental Results

This section provides the experimental result conducted for leaf detection using YOLOv8 and the SAM for segmentation in the images of tomato plants focusing on the segmentation of the tomato leaves. The tomato leaves were detected in the tomato plant images using the YOLOv8 model. The bounding boxes that Yolov8 had created were provided as input to the SAM for

precise segmentation. The performance of YOLOv8 + SAM for detection and segmentation of the tomato leaves was assessed based on the parameters mentioned in Chapter 01.

The model achieved a mAP of 90.7% with a confidence threshold of 0.5. This is shown in figure 4.4 (a), which indicates the high accuracy of detection of tomato leaves on different tomato plant images. To assess the quality of the segmentation masks, precision, recall, and the F1 score were determined for every image at the pixel level. The Precision curve measures the percentage of all positive predictions that were actually correct. The Recall curve measures the percentage of all actual positive instances successfully identified. The F1-Score serves as harmonic mean of precision and recall. Precision was 100% shown in Figure 4.4 (b), while recall was 98% as shown in Figure 4.4 (c), reflecting the model's capability to correctly detect the tomato leaves and minimize false positives. The Dice Coefficient (or F1 score for segmentation) was used to assess the similarity between predicted masks and ground truth. Figure 4.4 (d) shows the F1 confidence curve of 84% at 0.612.

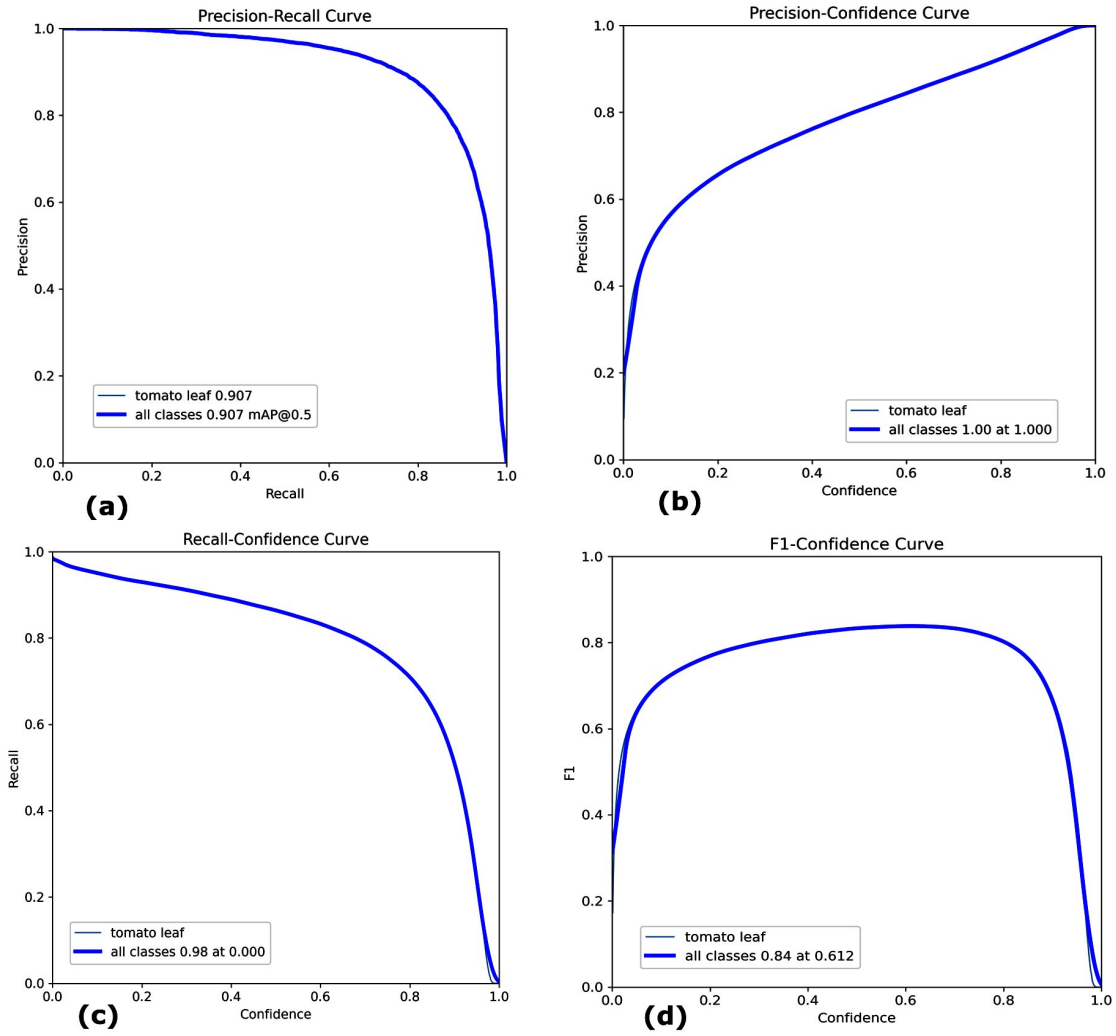


Figure 4.4. Performance Metrics of YOLOv8 (a) Precision-Recall Curve (b) Precision-Confidence Curve (c) Recall-Confidence Curve (d) F1-Confidence Curve

To evaluate the performance of the YOLOv8 + SAM framework, we split the dataset into a training, validation and test set as shown in table 4.1. Based on the evaluation metric discussed earlier all the segmentation results were compared against the ground truth masks. After detecting the tomato leaves using YOLOv8, the SAM model was used to segment the leaves based on the bounding boxes. SAM's ability to provide fine-grained segmentation of leaf boundaries was critical in improving the precision of the leaf regions, particularly for complex shapes and overlapping leaves. IoU measures how well the predicted segmentation overlaps with the actual target area.

We reported IoU scores for individual leaves and calculated the average IoU over the entire test set. The average Dice coefficient was 88%, which is a strong indicator of the overlap between the predicted and targeted area. This suggests that SAM accurately segmented the leaf regions, particularly around the edges, where precision is essential for identifying healthy and diseased leaves. The overall pixel accuracy of the segmentation process was 80.78%, showing that the model correctly identified the majority of leaf pixels while minimizing false positives for the background or non-leaf regions. Alongside the quantitative metrics, the segmentation quality was assessed through visual analysis. Figures 4.5 and 4.6 shows the examples of the original image, YOLOv8-detected bounding boxes, and the final segmented masks from SAM. In these images, the bounding boxes produced by YOLOv8 successfully identified the main leaf structures. SAM further refined the segmentation, providing precise boundaries, even in the presence of overlapping leaves. The segmented masks clearly delineated the tomato leaves from the background, with SAM accurately capturing the leaf edges.

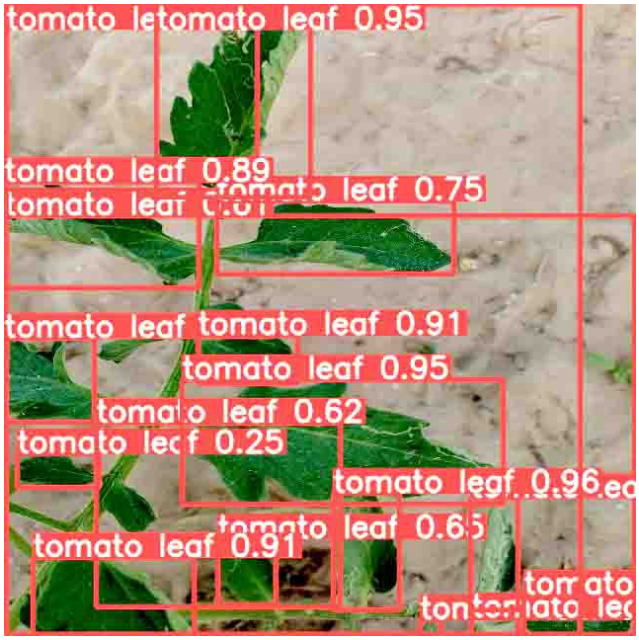


Figure 4.5 Leaf Detection within Bounding Boxes

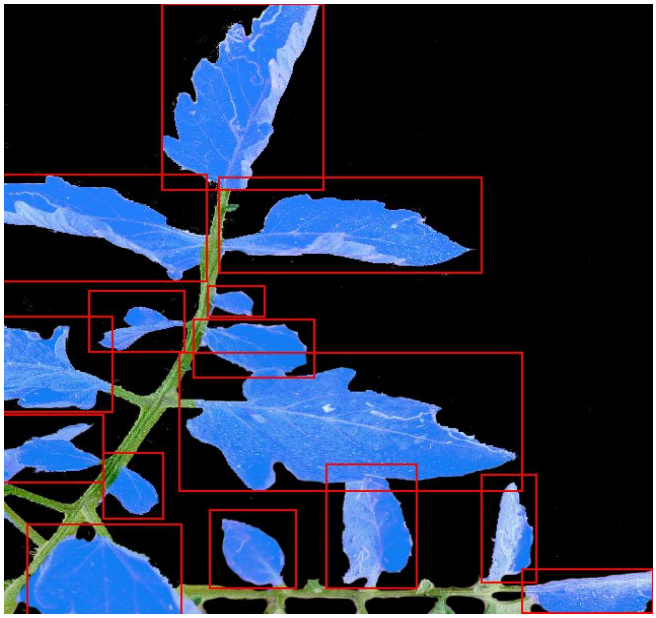


Figure 4.6 Leaf Masking Using SAM

The visual inspection confirms that SAM performs well on complex images, handling variations in lighting, occlusions, and leaf orientation. The masked leaves are then extracted and stored and a separate folder for further analysis such as disease detection as shown in figure 4.7. By integrating SAM with YOLOv8, one of the key improvements reflected was the enhancement of the leaf edges. YOLOv8's bounding boxes and masks provided a rough approximation of the leaf area, but SAM was able to capture the finer details of the leaf shapes, including serrated edges and irregularities.

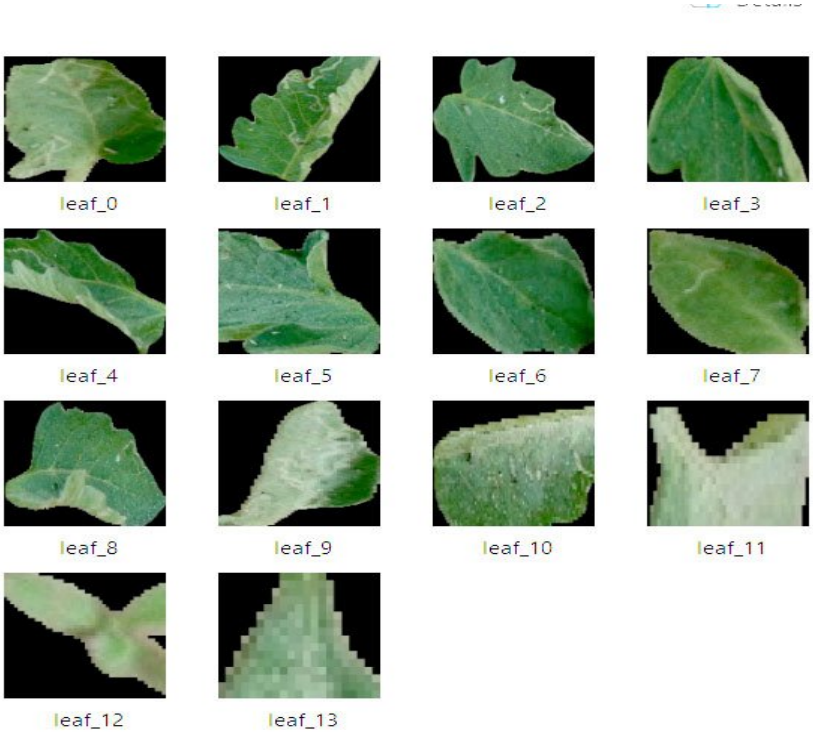


Figure 4.7 Leaf Masks Extraction to a separate folder

For disease detection in plants this is very important to precisely segment and identify the small lesions or abnormalities on the leaves. Instead of using YOLOv8 and SAM models alone the integration of these two exhibited significant enhancements in the tomato leaf segmentation. A notable increase can be seen in the IoU and Dice coefficient by using SAM to refine the YOLOv8-detected regions. Table 4.1 represents the performance of our integrated model in terms of precision, recall score, and mAP@50. During validation, our model has the precision score of 99.7%, the recall score of 99.5%, and the mAP@50 of 89.4%. Whereas the inference time during validation is 144 milliseconds. The testing accuracy of our model illustrates a precision score of 100% and a recall score of 98%, whereas the inference time during testing was 142 milliseconds. Further analysis confirmed the effectiveness of SAM in handling complex leaf structures and occlusions.

Table 4.1 Training, validation, and testing results of our model for tomato leaf class on the TomatoVillage dataset

Mode	Images	Instances	Precision	Recall	mAP@50	Inference Time (ms)
Training	8045	42866	99.7%	99.5%	89.4%	144
Validation	2298	9894	100%	96.8%	93.7%	140
Testing	1150	4086	100%	98%	90%	142

This work shows the effectiveness of YOLOv8 and SAM for accurate tomato leaf detection and extraction. We achieved a precision of 100%, a recall of 98%, and an F1-score of 84%. These findings have several practical implications for agriculture. Firstly, our system can enable accurately identifying and extracting individual tomato leaves. This system can be integrated with a tomato disease detection model for further disease analysis. This can lead to minimizing crop losses by early disease detection in tomato plants. Secondly, our findings can contribute to precision agriculture practices by enabling targeted interventions based on individual tomato leaf characteristics. Thirdly, our system can reduce farmer costs by automating leaf counting and measurement. Moreover, it has the potential for real-time applications, it can be integrated into a drone-based monitoring system. We understand that our work aims to make agricultural practices more efficient and sustainable by delivering key contribution to this field.

4.5 Discussion

In this chapter, the tomato leaves were segmented using the images of tomato plants with the help of the YOLOv8 and the SAM model. The main aim of the research is to evaluate the accuracy of the segmenting tomato leaves by both models as it is important for many

agricultural techniques which include plant monitoring, disease detection, and farming automation. This discussion also examines both models, their integration as well as the strengths and limitations of models to further precision agriculture objectives. During experimentations, YOLOv8 demonstrated a very good ability to detect tomato leaves even in cases such as many disorganized overlapping leaves where multiple leaves were dispersed and or alongside each other. The bounding boxes generated by YOLOv8 provided an adequate preliminary stage for various segmentation tasks as they included all the individual leaves. Additionally, the high precision and recall of the model, measured by its mAP capacity of the model to perform numerous overlapping tasks enabled the model to accurately detect objects within complicated environments. However, there are still some performance limitations of YOLOv8 in terms of pixel-level segmentation accuracy which is designed only for target bounding box detection. In more complex cases like heavily shadowed leaves or leaves that were in the background, the bounding box was typically considered too broad. This limitations of YOLOv8 necessitated the use of a more refined segmentation approach, such as SAM, for achieving finer pixel-level accuracy. The regions identified by YOLOv8 were provided as input to SAM for producing detailed segmentation masks for each detected leaf. This method allowed for a more precise delineation of leaf boundaries, especially in cases where YOLOv8's bounding boxes were too coarse. SAM was particularly efficient in handling the complex shapes and contours of tomato leaves, which is most important for accurate phenotypic analysis in precision agriculture.

Table 4.2 shows that our integrated model outperforms other segmentation models in terms of mAP, Precision, and Recall score. The most significant advancement is observed in the mAP, where our model achieves 0.90%, outperforming YOLOv8-SAM, the previous best performer. The precision score of our integrated model is estimated at 100%, meaning that when it identifies a tomato leaf, the chances of actually detecting it are exceptionally high. The average inference time of our integrated model over 100 runs is quite better than previous models which is 142 milliseconds with a GPU usage of 71.19%. This helps to mitigate false positives which in most cases, highly depend on the precision of the model's detection, specifically in real-time applications. A recall rate of 98% reveals that the model has a high probability of detecting most of the tomato leaves in the provided image. This demonstrates the performance of the model under variant conditions such as the presence of small leaves and light conditions, which is a common feature in real-world problems. Our proposed approach, integrating YOLOv8 and SAM, has shown convincing performance in terms of detecting, masking, and extracting tomato leaves.

Table 4.2 Comparison of our integrated model with different detection algorithm models

Model	Resolution	Precision	Recall	mAP	Average Inference Time (ms)
YOLOv5-X	608 x 608	88%	85%	80%	415
YOLOv6-L	640 x 640	89%	81%	91%	400
YOLOv7	640 x 640	87%	83%	91%	278
YOLOv8-SAM	960 x 960	92%	83%	89%	198
Our integrated model	**640 x 640**	**100%**	**98%**	**90%**	**142**

It is important to note that other models like YOLOv5-X [88], YOLOv6-L [89], YOLOv7 [90] , and YOLOv8-SAM [91] has its merits, but the proposed combination of YOLOv8 and SAM has shown a comparable level of effectiveness, as indicated in table 4.2. The visualization of comparatives analysis with the previous model with respect to performance metrics and average inference time is shown in Figures 4.8 and 4.9.

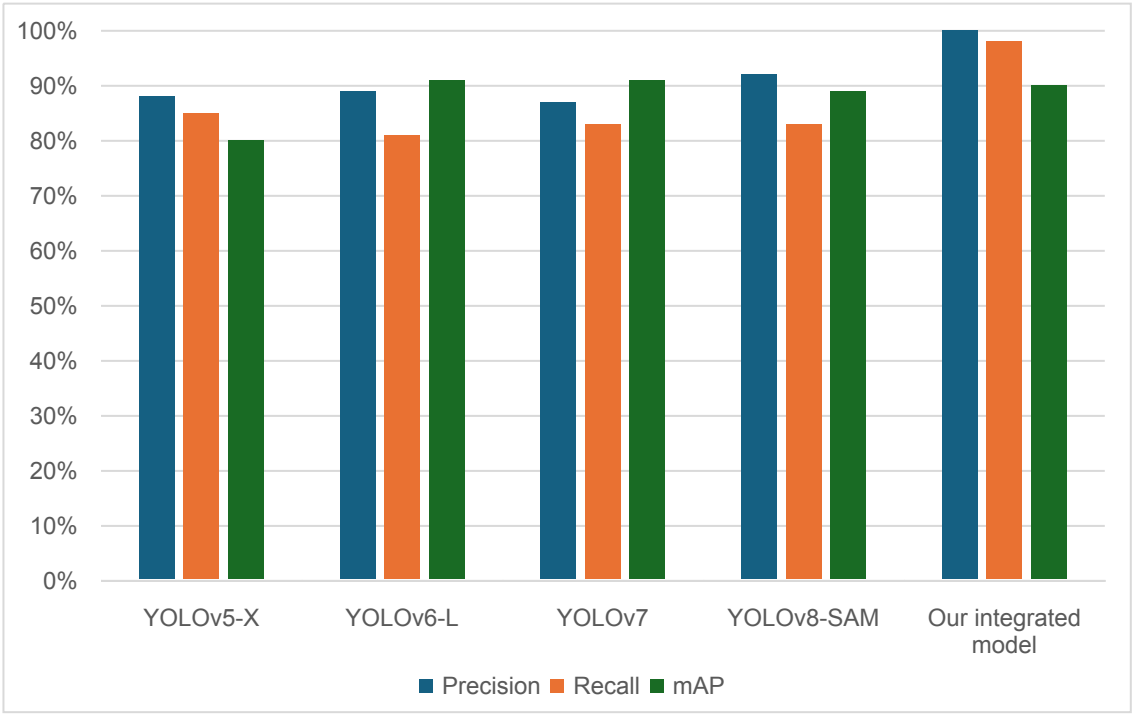


Figure 4.8 Comparison chart with previous models based on precision, recall, and accuracy

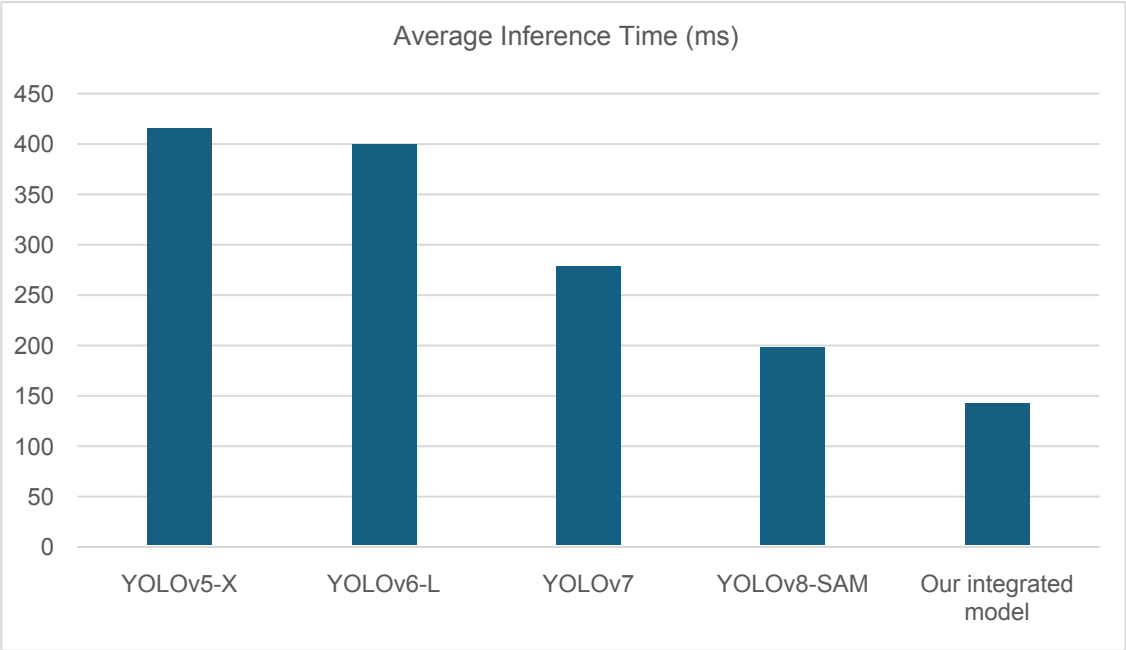


Figure 4.9 Comparison chart with previous models based on average inference time (ms)

Regardless of the overall success of the integrated YOLOv8 and SAM approach, we faced several limitations during this work. A major problem was the computational cost of performing both models in sequence. One can appreciate YOLOv8 for its less threshold time, but the segmentation of SAM is more taxing being its use in high-resolution images and or large batches of data. There is a need to find a stability among segmentation, precision, accuracy, and efficiency in real-time systems, especially in field-based robotic systems or hand-held agricultural devices. Another constraint noted was in instances where tiny or semi-occluded leaves were not detected by YOLOv8. Since SAM depended on input provided by as bounding boxes detected by YOLOv8, thus in instances when YOLOv8 missed some targets, leaves that were supposed to be segmented were not masked by the SAM. This emphasizes the need to work on enhancing object detection in special cases where the plant is in a complex habitat.

Additionally, there were also times when it was hard for SAM to tell apart leaves that were on top of each other or were in close proximity. In this type of situation, segmentation masks tended together the adjacent leaves making false segmentation possible. From this research work several possibilities for future research emerge. Future improvements could involve the use of multi-modal inputs or additional preprocessing steps to enhance the separation of individual leaves. Apart from this an integration of hyperspectral and multispectral imaging can enhance the accuracy of both detection and segmentation of leaves, especially in cases

where visual cues alone are inadequate to distinguish between plant parts. Apart from this for real-time applications, a more efficient pipeline approach can be a significant direction. The use of edge devices or the quantization of these models can significantly reduce the computational cost associated with both these models (YOLOv8 + SAM) in sequence. Overall, this system has the potential for real-time applications, it can be integrated into a drone-based monitoring system. We understand that our work aims to make agricultural practices more efficient and sustainable by delivering key contribution to this field.

4.6 Summary

In this chapter the YOLOv8 and SAM have been analysed for the segmenting of tomato leaves in tomato plant images. The major goal of this chapter is to evaluate the efficiency and enhancements of YOLOv8 and SAM in detecting and segmenting tomato leaves which is the most important step in detecting plant diseases in precision agriculture. By integrating the rapid object detection capabilities of YOLOv8 with the high-precision segmentation of SAM, this integrated approach proved notable strengths and some areas where improvements can be considered. The integration of YOLOv8 and SAM poses a highly effective approach for tomato leaf segmentation, taking advantage of YOLOv8's rapid object detection and SAM's high-precision segmentation competencies. Despite challenges, specifically regarding computational cost, efficiency, and small object detection, the results of this chapter present a strong base for future research in agricultural imaging and automated plant monitoring systems. The potential applications of this work in disease detection, phenotyping, and precision farming highlight its relevance to the future of agricultural technology.

Chapter 5: Enhanced Deep Learning Architecture for Tomato Disease Detection

This chapter presents an enhanced deep learning architecture designed to overcome the limitations in tomato leaf diseases detection. It builds upon the foundational work of prior chapters, particularly the leaf detection and segmentation system presented in Chapter 4, by providing a robust and precise model for diagnosing diseases on the segmented leaf images. This multi-stage approach is very important for attaining higher accuracy and reliability in a real-time, in-field environment. This chapter investigates the use of deep learning to achieve the accurate and fast tomato plant disease identification. We have used both individual and merged datasets of tomato plants covering 10 diseases (including healthy plants). Our primary is to evaluate the accuracy of existing CNNs (like VGG, ResNet, and DenseNet) on tomato plant disease detection and then design a custom deep neural network model specifically optimized for the best possible accuracy in tomato plant disease detection. In this chapter we have discussed the design, implementation and assessment of our proposed enhanced CNN architecture. The aim is to demonstrate that a purpose-built, enhanced deep learning model, when integrated into a comprehensive system, can achieve superior diagnostic accuracy and pave the way for a new generation of intelligent agricultural systems that contribute in achieving better sustainability in farming and guaranteeing global food security.

This chapter is structured as follows: Section 5.1 discusses the introduction in detail. Section 5.2 details the proposed methodology, including dataset acquisition and preparation, training procedures. Section 5.3 provides the experimental results. Section 5.4 focuses on the comparative analyses with existing approaches and model selection. Finally, Section 5.5 summarizes the chapter and suggests future research directions.

5.1 Introduction

One of the most significant industries is agriculture production because it directly affects a society's economy and social structure. Agriculture production faces a significant issue in identifying plant diseases; thus, we need a quick and precise method to do so. The quality and amount of a nation's production determine how much agriculture it can produce. Any nation's economic and social structure is directly impacted by the production of agricultural goods. Agriculture production is essential to the advancement of society and the economy[92]. Specifically tomato cultivation is a cornerstone of global agriculture, yet it faces persistent threats from numerous diseases that can considerably reduce crop yield and quality. This is the

reason that diagnosing and treating plant diseases has drawn so much attention, due to the large amount of territory that farms now cover, finding, assessing, and treating plant disease is a tremendously difficult problem for farmers today. For effective management of crops, it is crucial to identify these diseases accurately and timely. Conventional methods of plant disease detection depend upon the manual investigations which is labour-intensive, time-consuming and prone to human error [93], [94]. Additionally, there is a high likelihood of errors which might result in productivity loss. Additionally, early identification and treatment of plant diseases are critical because they have a significantly higher rate of success . Because the plant's leaves are the most apparent and susceptible to disease, they are used to recognize plant diseases [95]. We require a system that can accurately identify the condition using images. For diagnosis purpose, plant diseases and their treatments have been the subject of extensive research, but each has its limitations. Researchers have employed a variety of methodologies, including image processing, machine learning and deep learning. As a result, these studies were fruitful, and most of the procedures produced results that were superior to those of human specialists, especially deep learning. Due to enhancement in artificial intelligence, specifically deep learning, have provided a drastic solution, that ensures the creation of automated, precise, and scalable disease detection systems. While a significant body of research has demonstrated the efficacy of deep learning models like CNN for plant disease classification, a critical gap remains in their practical application within complex, real-world agricultural environments. As discussed in previous chapters, models trained on clean, uniform datasets often fail to generalize to the challenges of a farm setting, which includes complex backgrounds, changing lighting conditions, and the need to identify diseases on individual leaves rather than on the entire plant. With the most enhanced CNN architecture we can develop an accurate and reliable plant disease detection system.

In this chapter we have used both individual and merged datasets of tomato plants covering 10 diseases including healthy plants as well. Our primary is to evaluate the accuracy of existing CNNs (like VGG, ResNet, and DenseNet) on tomato plant disease detection and then design a custom CNN model specifically optimized for the best possible accuracy in tomato plant disease detection. Despite the complexity of datasets exceeding 18,000 and 25,000 images with 10 distinct classes each, ResNet, DenseNet, and our custom model achieved remarkable accuracy above 99%.

5.2 Material and Methods

This section offers a comprehensive account of the dataset utilized, the data preprocessing steps, the deep learning architectures, and training methodology employed for the enhanced tomato disease diagnosis system. Figure 5.1 illustrates the workflow which comprises of four steps.

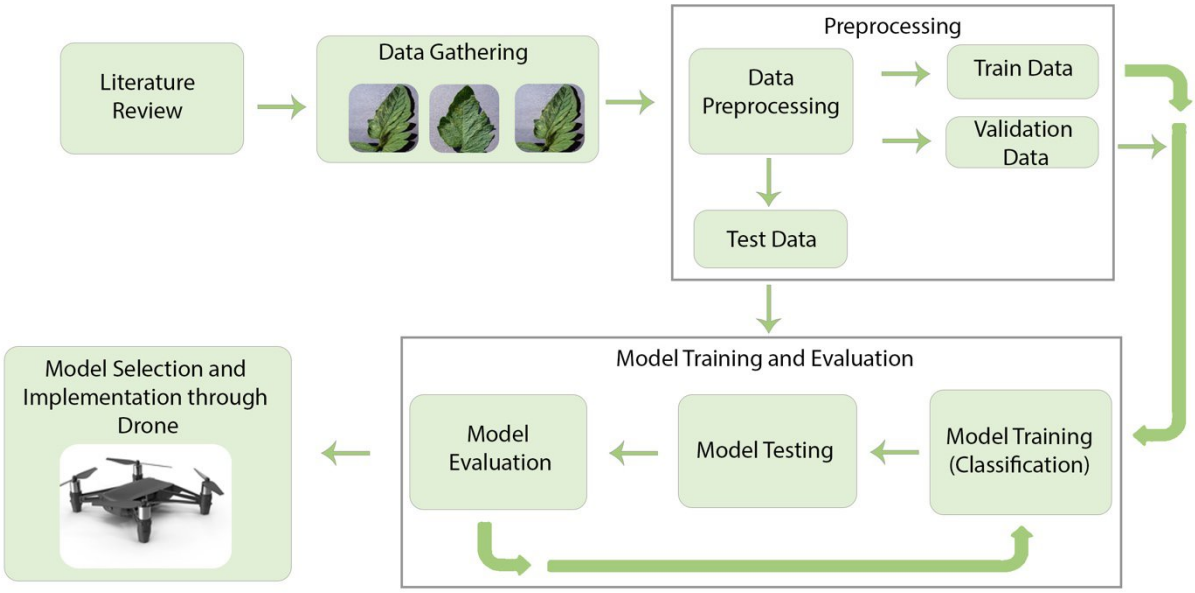


Figure 5.1 Proposed Enhanced Deep Learning Architecture and Data Pipeline for Tomato Disease Detection

5.2.1 Dataset Collection and Preprocessing

The quality and diversity of the training data can highly affect the performance and generalization of a deep learning architecture. For this research, a comprehensive dataset was curated from two primary sources to ensure the model's robustness to both single-leaf images and real-world, complex backgrounds. In this research we have collected our data from Kaggle for disease recognition. We have collected different types of images samples from Kaggle with the range from 10,000-40,000 images. The models are trained and tested using different data and the performance is closely examined to make sure the collected data is correct. This dataset was used to establish a baseline for classification accuracy and to provide the model with a strong foundational understanding of disease symptoms in a controlled environment. The dataset includes 10 classes of tomato diseases and one class for healthy leaves.

Before the model is trained, we need to do preprocessing on the data and make sure the data is clean, relevant and enough. The data used in this research is different pre-processed images data taken from Kaggle; however, the data is pre-cleaned but different data gave different results. This is why models we trained and tested models using different data with different

size, quality and the conditions images were captured. The different results were then compared to specify most suitable dataset. The model was first trained with less data of 8,000 samples with good results but not satisfactory results. The model trained and tested with 20,000-38,000 look to be the best suiting numbers for deep learning. The dataset created was cleaned from any Irregularities and irrelevant data. For ensuring the consistency during training all the images were resized to the uniform dimensions of 224x224 pixels.

To prevent overfitting a series of data augmentation techniques were applied to the training data. For creating new data for the model, the data augmentation technique is used which works better in case of in deep learning. Rotation, High contrast, Bright light, Low light, and image invert augmentation is used for the models that uses augmented data. These techniques involve horizontal and vertical flips, random rotations, random cropping and colour jittering. This process significantly expanded the effective size of the training set, improving the ability of the model to identify disease symptoms under various orientations and lighting conditions.

The data was divided into training (70%), validation (20%) and testing (10%) sets. The model uses the training data to recognize patterns for improving the accuracy. The validation data is used during the training process for model validation. The validation data is entered in batches and work as a test data during each step during the training process. The test data is used for testing the model and this the unseen data to the model. This data is used to find the accuracy of the model on new data.

5.2.2 Model Training and Optimization

In this section we have discussed the meticulous training process our model and model optimization for detecting the tomato diseases. For implementation in real-world, the training methodology was designed to ensure achieving high accuracy and ensuring effective generalization to new, unseen data. A high-performance computing (HPC) system has utilized for all the experimentations and model's training, leveraging the powerful combination of Keras, TensorFlow backend, and the CuDNN library, we achieved efficient training on an NVIDIA Quadro K2200 GPU with its 4 GB memory, 640 CUDA cores, and high bandwidth. Python served as the core language for model development and scripting. The categorical cross-entropy loss function was chosen to effectively guide the model's training. This is the standard choice for multi-class classification as it efficiently measures the difference between the model's predicted probability distribution and the true, one-hot encoded label. The major goal of training is to minimize this loss, which directly improves the model's predictive accuracy.

Adam optimizer was used for optimization, which is an adaptive learning rate optimization algorithm known for its computational efficiency and minimal memory requirements. It maintains a separate adaptive learning rate for each parameter, which makes it particularly effective in complex optimization landscapes. The maximum number of epochs for all the model's training was set to 50, with the learning process carefully monitored to prevent overfitting. To ensure the optimal performance a set of crucial hyperparameters was configured:

- **Batch Size:** To ensure the stability of the training process and computational efficiency the batch size was set to 16.
- **Learning Rate Scheduler:** The scheduler monitored the validation loss, by minimizing the learning rate by a factor of 0.1 if the validation loss failed to improve for three consecutive epochs. This process helps the model fine-tune its parameters and reach better convergence.
- **Regularization:** To prevent overfitting, dropout layers were fused with the fully connected layers with a dropout rate of 0.5, which randomly deactivates a segment of neurons during training, ensuring the network learns more robust features instead of relying on specific, co-adapted features.
- **Early Stopping:** An early stopping mechanism was utilized after each 10 epochs for monitoring the validation loss. In case of no improvement after 10 epochs, it automatically terminates the training process. This avoids over-training on the training data, ensuring it remains generalizable to unseen data.

The accuracy of each model along with training and validation loss over each epoch was properly recorded. The learning curves provides a clear visual summary of the model's learning progress and its generalization performance.

5.2.3 Model Testing and Performance Evaluation

In this step the model is tested using unseen data to the model. We used test data for the performance evaluation. Based on this performance evaluation we select the most suitable model for tomato plant disease detection. The test data is used to make prediction and compare it with the existing results to evaluate the actual accuracy and performance. The performance of all the models was assessed based on overall accuracy, the validation accuracy, accuracy loss and validation loss.

5.2.4 Model Selection

The final model was not simply the one from the final epoch. Instead, throughout the training process the model with the lowest validation loss was chosen for the final evaluation on the test set. This ensures that the chosen model generalizes best to unseen data, representing its optimal performance before any signs of overfitting. Seeking to enhance plant disease classification accuracy, researchers explored improvements and modifications to existing deep learning architectures, demonstrating superior performance in identifying plant species ailments. Among them, we have considered VGG 16, AlexNet, ResNet, DenseNet 121.

5.3 Experimental Setup and Results

This section presents the comprehensive results of the experimental evaluation, highlighting the performance of our proposed architecture for the detection of tomato diseases. The findings are discussed in detail, comparing the performance of the model against established benchmarks and providing insights into its strengths and limitations. We have multiple deep-learning architectures along with our custom CNN model and compared the accuracies to get the most suitable model.

5.3.1 Visual Geometry Group 16

The VGG16 model has performed very well. It has a validation and test accuracy of 99% and 98% respectively. The model performance was very good but if we compare it to other models that we have used in this research then almost all of them have outperformed the VGG16. The VGG16 has performed poorly on the merged data set with an accuracy of just 92%. Table 5.1 shows the model is fighting to maintain constant validation accuracy. The training accuracy and loss are significantly better than the validation accuracy and loss. However, the validation accuracy on the merged data is just 92.06, which is a very bad performance.

Table 5.1 Visual Geometry Group 16 individual and Merged Datasets

Visual Geometry Group 16 Individual Dataset				Visual Geometry Group 16 Merged Dataset			
Accuracy	**Loss**	**Val Accuracy**	**Val loss**	**Accuracy**	**Loss**	**Val Accuracy**	**Val Loss**
98.96	.0323	98.77	.0448	99.41	.0199	92.29	.6432
98.76	.0382	97.88	.0709	99.80	.0094	92.24	.6590
98.90	.0380	99.15	.0295	99.95	.0015	91.95	.6751
99.24	.0271	97.95	.0784	99.82	.0058	92.11	.6922
99.32	.0200	98.10	.0618	99.91	.0037	92.06	.6915

Figure 5.2 illustrates the model's convergence. The convergence rate is not that good because the model looks stuck at the elbow rules which state that the model should be stopped from where it starts the elbow shape. In the test result, the model successfully predicted 90% of the

test subjects correctly. Only one prediction out of 9 is incorrect as presented in figure 5.3 and 5.4. The overall confidence that the model has shown is 98 to 100%.

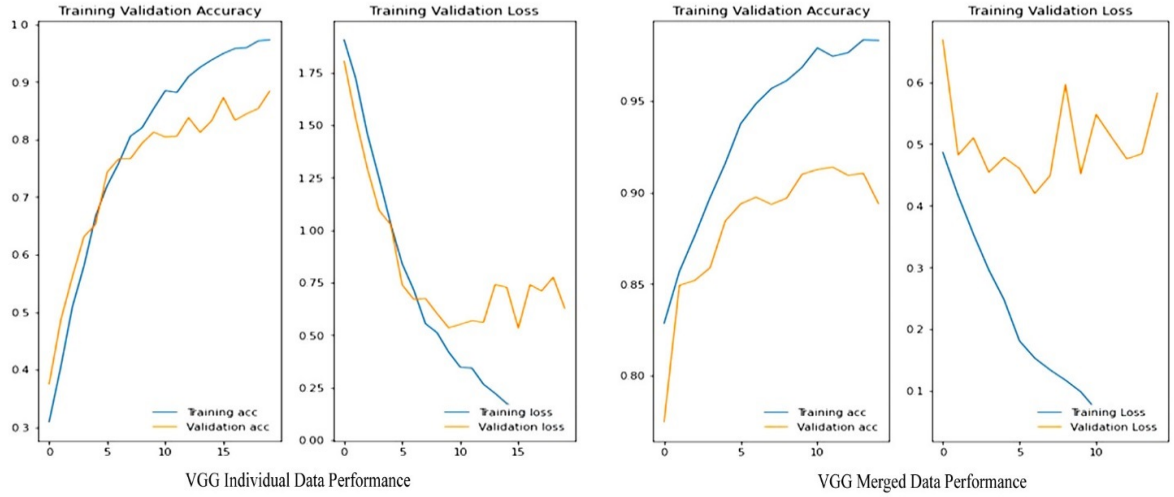


Figure 5.2 VGG Individual and Merged Dataset Performance

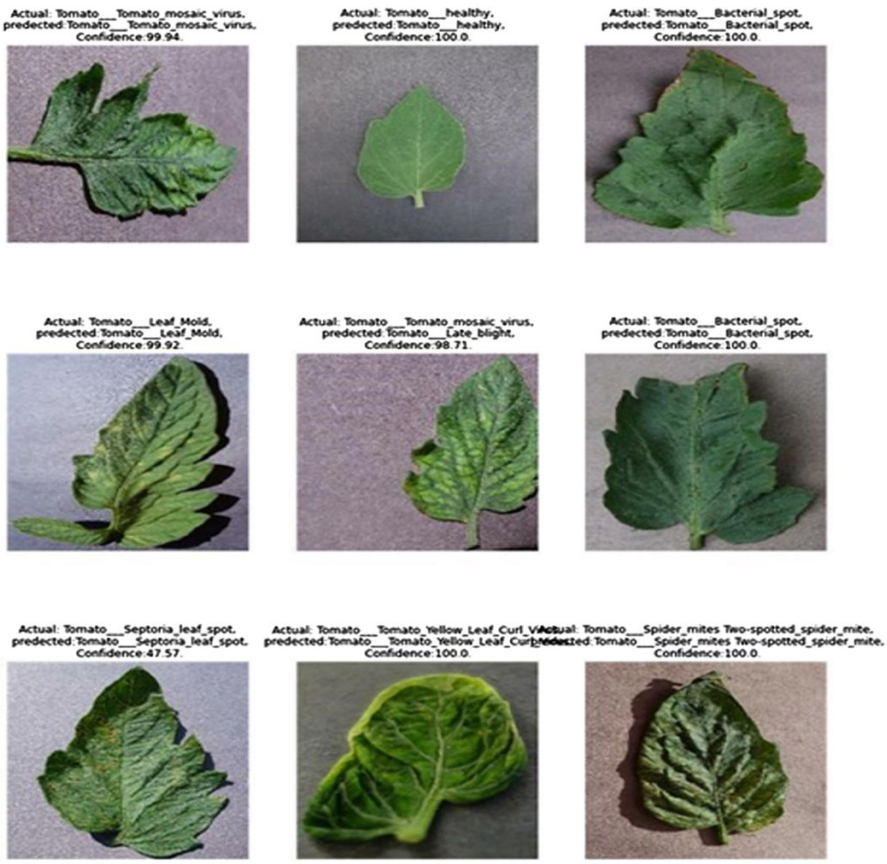


Figure 5.3 VGG Individual Dataset Results

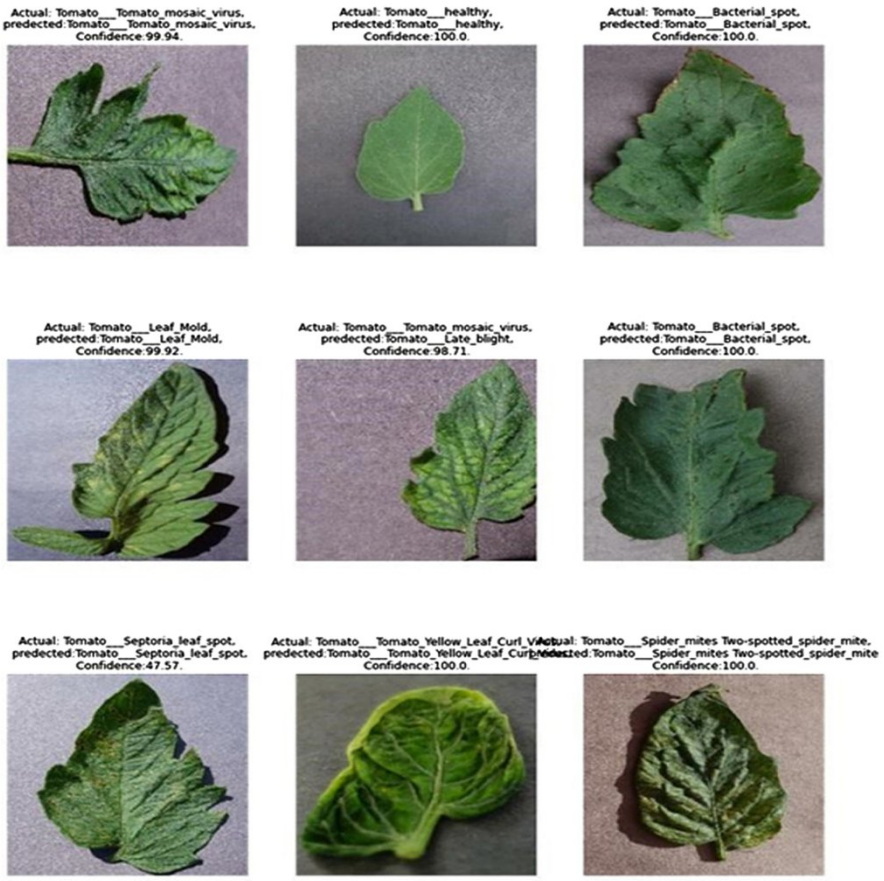


Figure 5.4 VGG Merged Dataset Results

5.3.2 Residual Net

The best performing model we have used in this research so far with an accuracy of more than 99.50%. ResNet is the most stable model we have used. It has the best validation and test accuracy 99.44% of 99.78% respectively on the Individual data. On the merged dataset ResNet50 has very good performance with 99.34% test accuracy as shown in table 5.2.

Table 5.2 Residual Net Individual and Merged Dataset

ResNet Individual Dataset				ResNet Merged Dataset			
Accuracy	**Loss**	**Val Accuracy**	**Val loss**	**Accuracy**	**Loss**	**Val Accuracy**	**Val Loss**
99.99	5.2742e-0	99.44	.0196	100	1.6210e-04	99.48	.0282
100	2.0346e-04	99.44	.0191	100	1.6100e-04	99.43	.0286
100	1.5636e-04	99.44	.0191	100	1.5992e-04	99.45	.0285
100	1.3528e-04	99.44	.0190	100	1.5885e-04	99.48	.0285
100	1.2024e-04	99.44	.0192	100	1.5780e-04	99.45	.0284

Figure 5.5 demonstrate the performance of the model as it has fully converged at epoch 4 which is extremely fast. The ResNet model has the best accuracy as we can see in figure 5.6 and 5.7 that the model has successfully identified all the diseases correctly. Mostly the confidence of the model in these test cases is 100% and the average confidence is more than 99%.

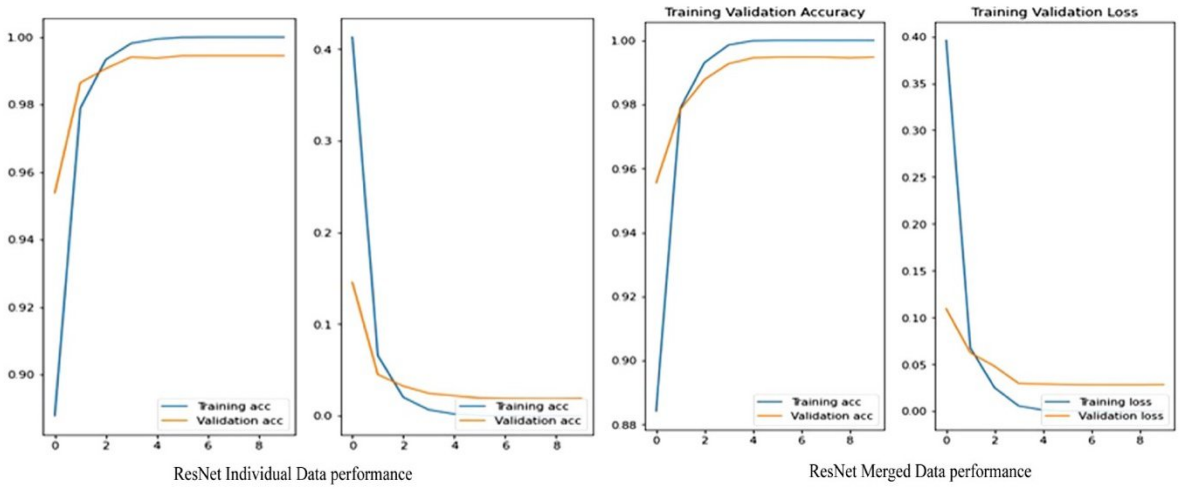


Figure 5.5 ResNet Individual and Merged Dataset Performance

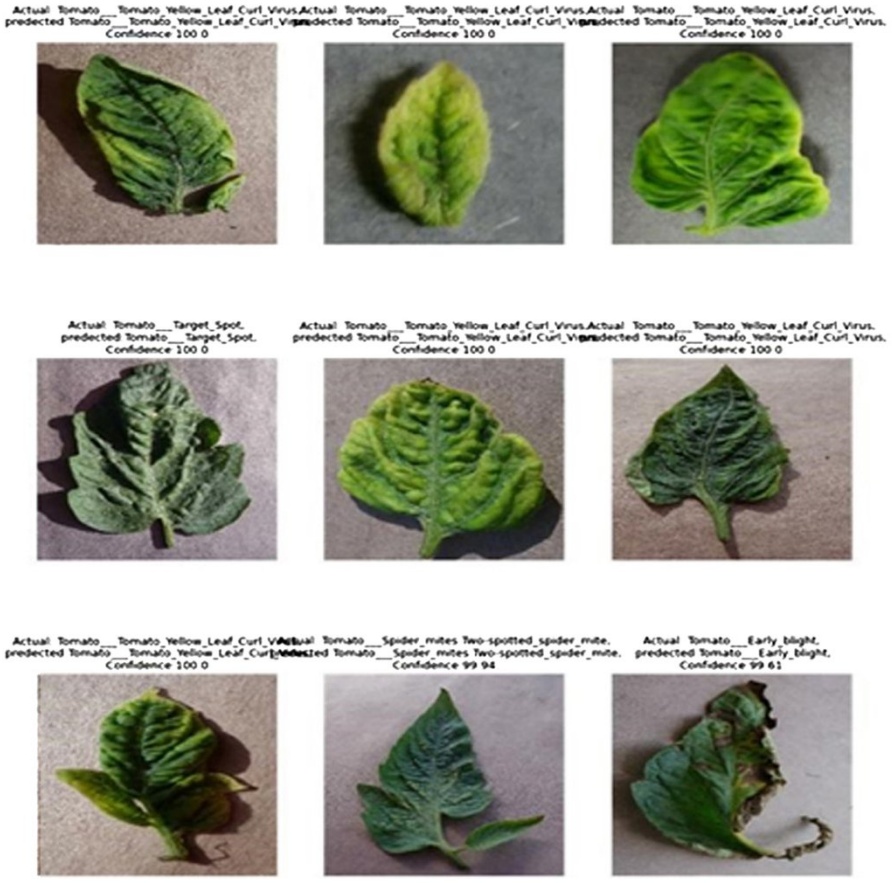


Figure 5.6 ResNet Individual Dataset Results

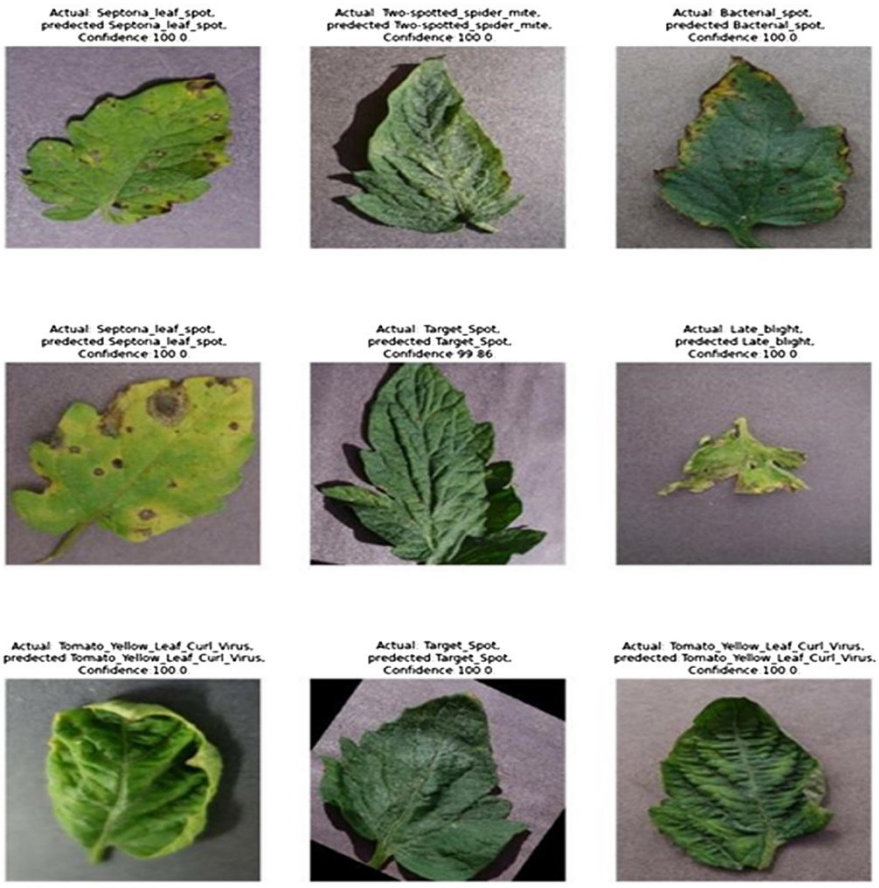


Figure 5.7 ResNet Merged Dataset Result

5.3.3 DenseNet121

DenseNet121's performance is also very good. This model has outperformed all the models we have used in this research just lacks behind ResNet. The DenseNet model has a very good accuracy of 99.55% on the Individual dataset. The model has not performed as well on the merged dataset. The model has a very stable validation and training accuracy and loss performance at each epoch the model is converging, and the accuracies get better. We have stopped the models at this point because the convergence rate has dropped very low thus, we will get a very low-performance gain at a high cost as shown in table 5.3.

Table 5.3 DenseNet121 Individual and Merged Dataset

DenseNet 121 Individual Dataset				DenseNet 121 Merged Dataset			
Accuracy	**Loss**	**Val Accuracy**	**Val loss**	**Accuracy**	**Loss**	**Val Accuracy**	**Val Loss**
98.26	.0547	98.96	.0319	98.75	.0420	99.40	.0207
98.69	.0439	99.27	.0277	99.05	.0328	99.43	.0219
99.00	.0348	99.24	.0225	99.22	.0274	99.45	.0207
99.10	.0300	99.31	.0211	99.46	.0213	99.37	.0218
99.25	.0272	99.41	.0203	99.46	.0193	99.24	.0209

The training validation accuracy and loss graph is presented in figure 5.8. The model has a very stable graph, but the problem is that the convergence drops very quickly on both datasets as shown in the figures below.

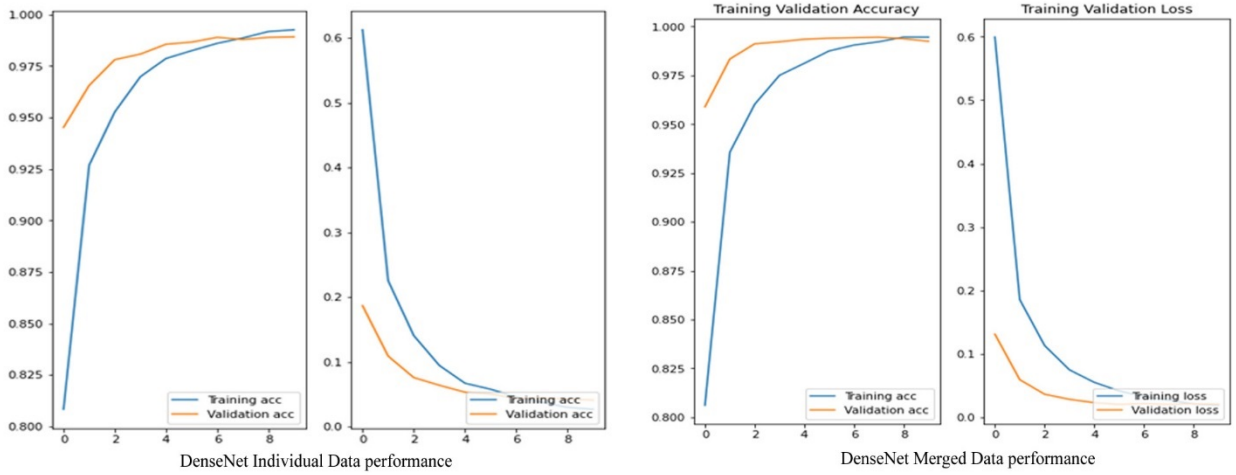


Figure 5.8 DenseNet121 Individual and Merged Dataset Performance

The DenseNet 121 model has the second-best accuracy as we can see in figure 5.9 and 5.10 that the model has successfully identified all the diseases correctly. The confidence of the model in these test cases gives an average confidence of more than 99%.

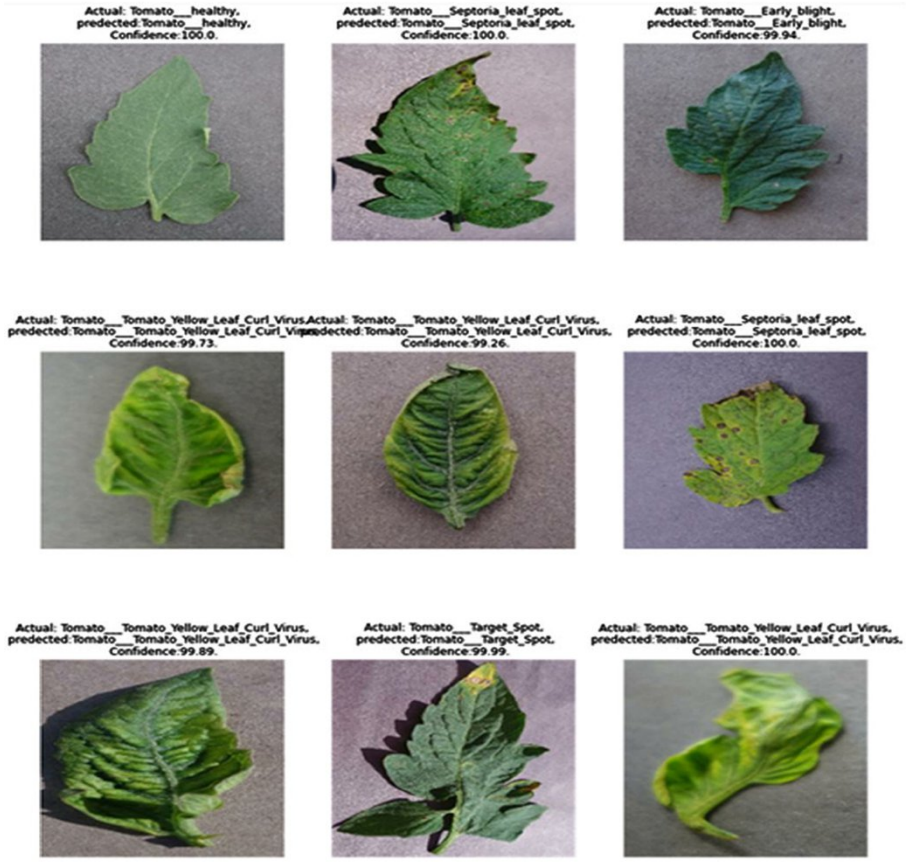


Figure 5.9 DenseNet 121 Individual Dataset Results

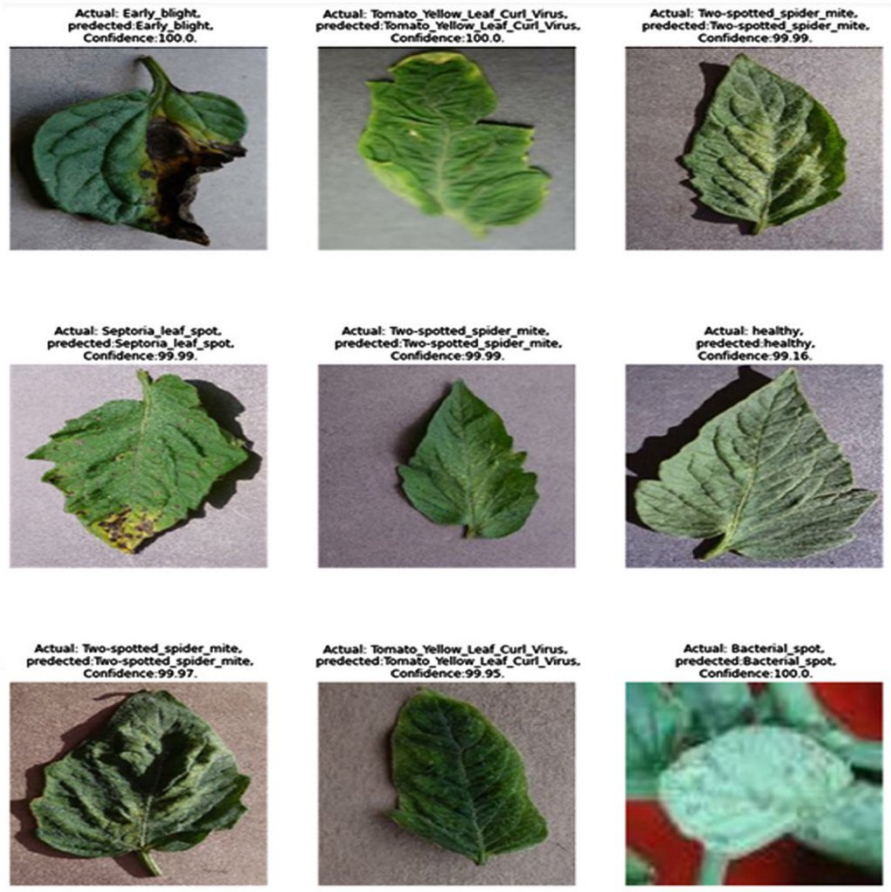


Figure 5.10 DenseNet 121 Merged Dataset Results

5.3.4 DenseNet169

We have achieved a test accuracy of 99.78% the same accuracy as ResNet50. The DenseNet169 performance was one of the best in this research. We believe that Res Net has performed a little better than DenseNet169 because the test accuracy is the same for both models but the training and validation accuracy of ResNet50 is better and the parameters are fewer in ResNet. As we can see models are very stale. The model has been almost fully trained because there are no more significant improvements as shown below in table 5.4. The model has fully converged as shown in figure 5.11 which does not need any more epochs.

Table 5.4 DenseNet169 Individual and Merged Data

Dense Net 169 Individual Data				Dense Net 169 Merged Data			
Accuracy	Loss	Val Accuracy	Val loss	Accuracy	Loss	Val Accuracy	Val Loss
98.49	.0470	99.13	.0331	9957	0184	9895	0346
98.84	.0371	99.20	.0280	9965	0161	9887	0388
99.12	.0296	99.20	.0283	9961	0141	9887	0374
99.15	.0266	99.37	.0246	9975	0112	9898	0368
99.40	.0205	99.34	.0273	9984	0092	9902	0358

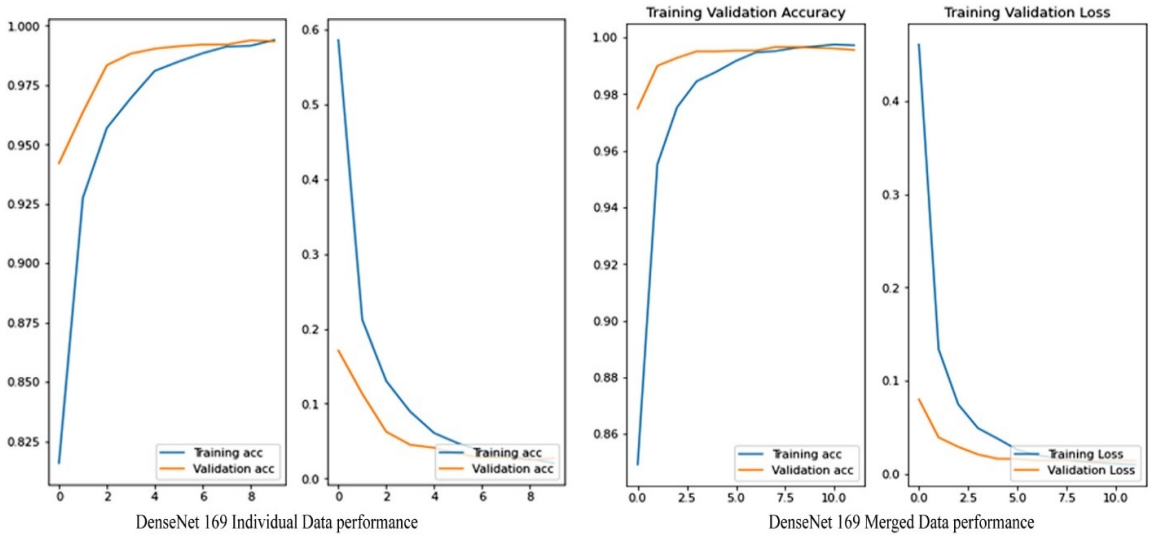


Figure 5.11 DenseNet169 Individual and Merged Dataset Performance

DenseNet169 has successfully identified all the 9 test subjects correctly. The confidence in the model is very high averaging 99%. Only on one subject is the model confidence is 97% which is below the average, but the confidence is almost 100% on most of the other subjects in figure 5.12 and 5.13.

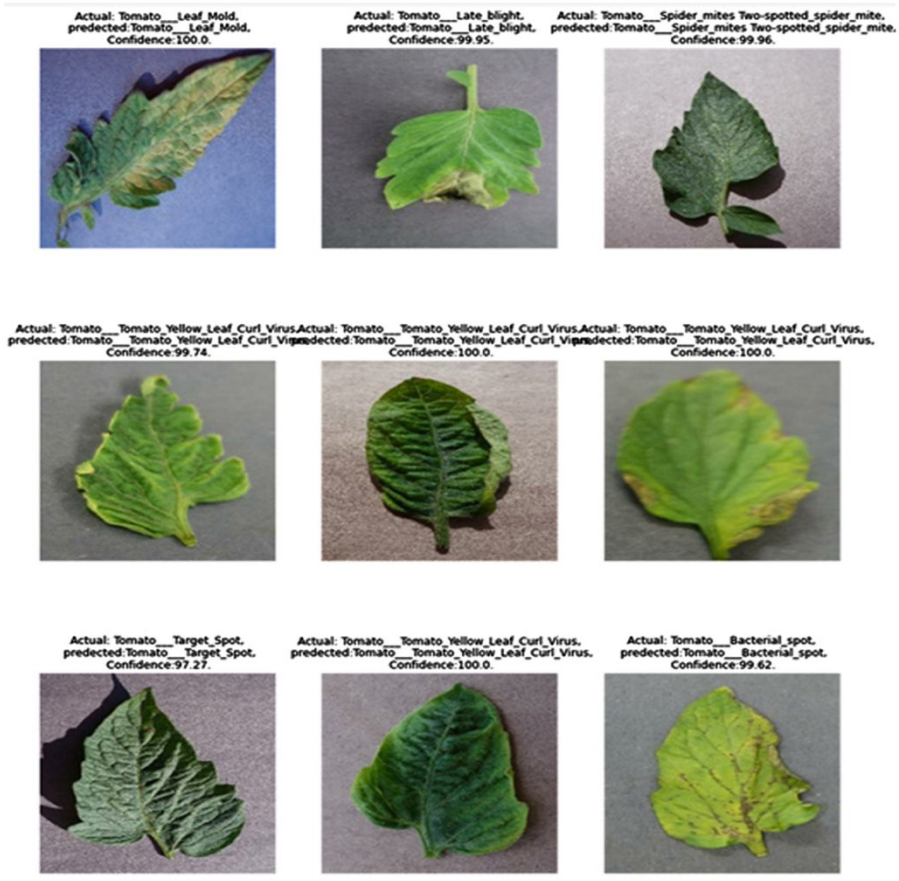


Figure 5.12 DenseNet169 Individual Dataset Results

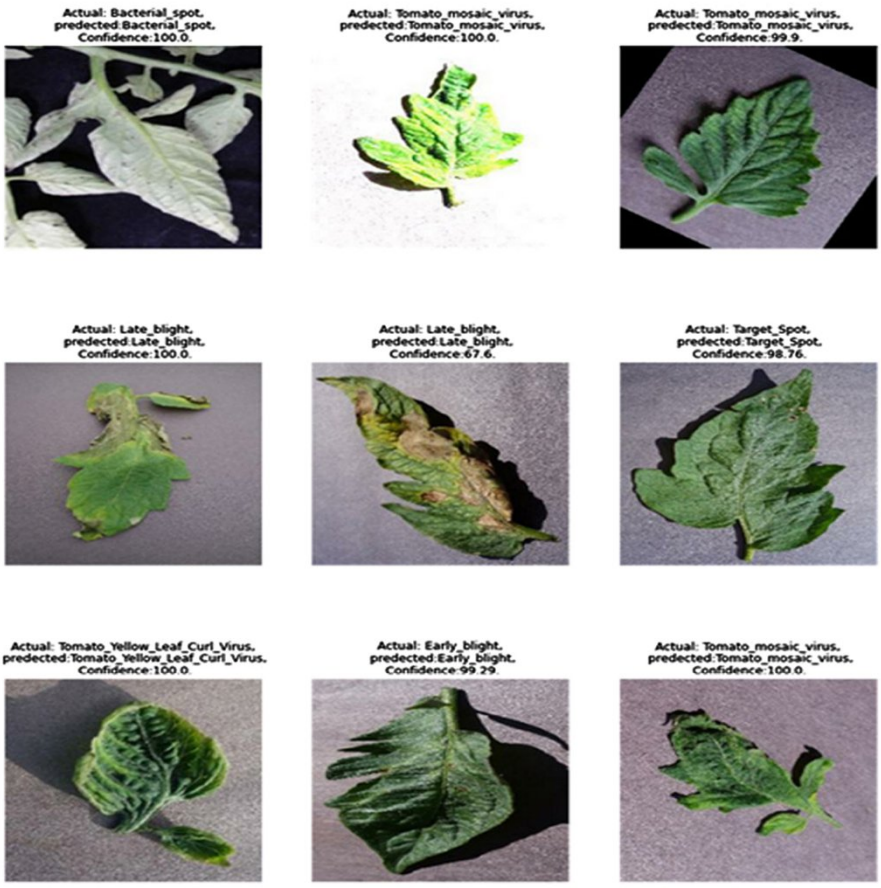


Figure 5.13 DenseNet169 Merged Dataset Results

5.3.5 DenseNet201

DenseNet201 is a type of DenseNet model series. This model has performed well with 99.78% training accuracy, 99.33% validation accuracy, and 99.67% test accuracy. There are only two other models that have performed better than DenseNet201 in this research. Table 5.5 illustrates that the training accuracy and loss are improving at each epoch, but the main issue is that the validation accuracy and loss are not improving. For this reason, we have ended the model training.

Table 5.5 DenseNet201 Individual and Merged Dataset

Dense Net 201 Individual Dataset				Dense Net 201 Merged Dataset			
Accuracy	**Loss**	**Val Accuracy**	**Val loss**	**Accuracy**	**Loss**	**Val Accuracy**	**Val Loss**
99.38	.0240	99.41	.0201	98.69	.0418	99.97	.0050
99.61	.0171	99.27	.0235	98.99	.0322	99.95	.0036
99.66	.0148	99.17	.0233	99.27	.0250	99.97	.0021
99.70	.0131	99.37	.0191	99.24	.0229	100	.0019
99.78	.0098	99.31	.0211	99.37	.0204	99.97	.0018

Looking at these accuracy and loss graphs, we can say that the model has stopped the convergence. By using these graphs, we can understand that the model cannot improve

anymore and further training can cause overfitting and gradient vanishing as shown in figure 5.14. These are some of the results that we have taken from testing the model. The model has successfully identified all the diseases with almost 100% confidence. There are no false-true or true-false predictions but on more testing data this model has not the best accuracy shown in figure 5.15 and 5.16.

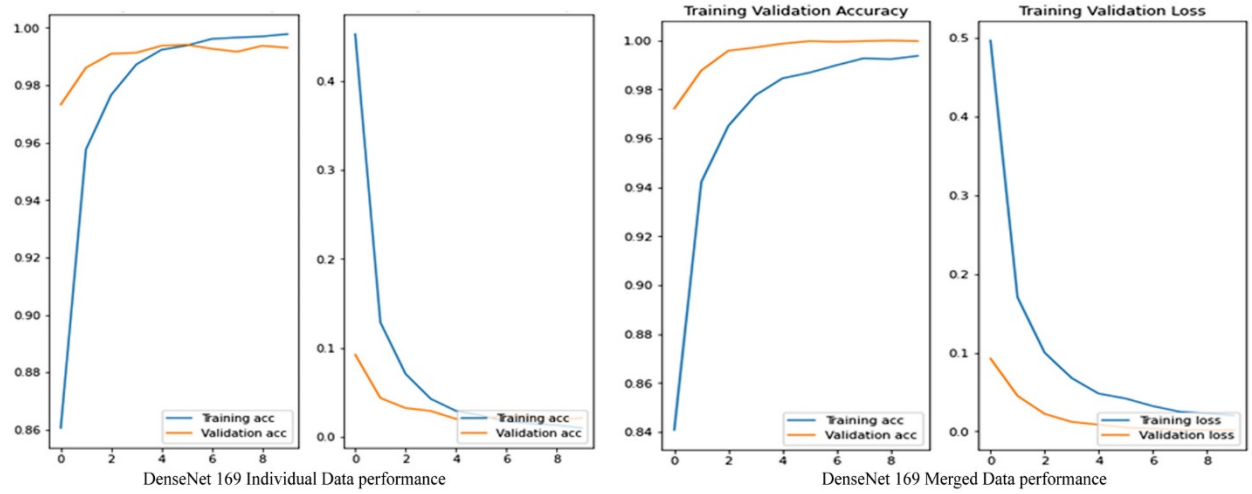


Figure 5.14 DenseNet201 Individual and Merged Dataset Performance

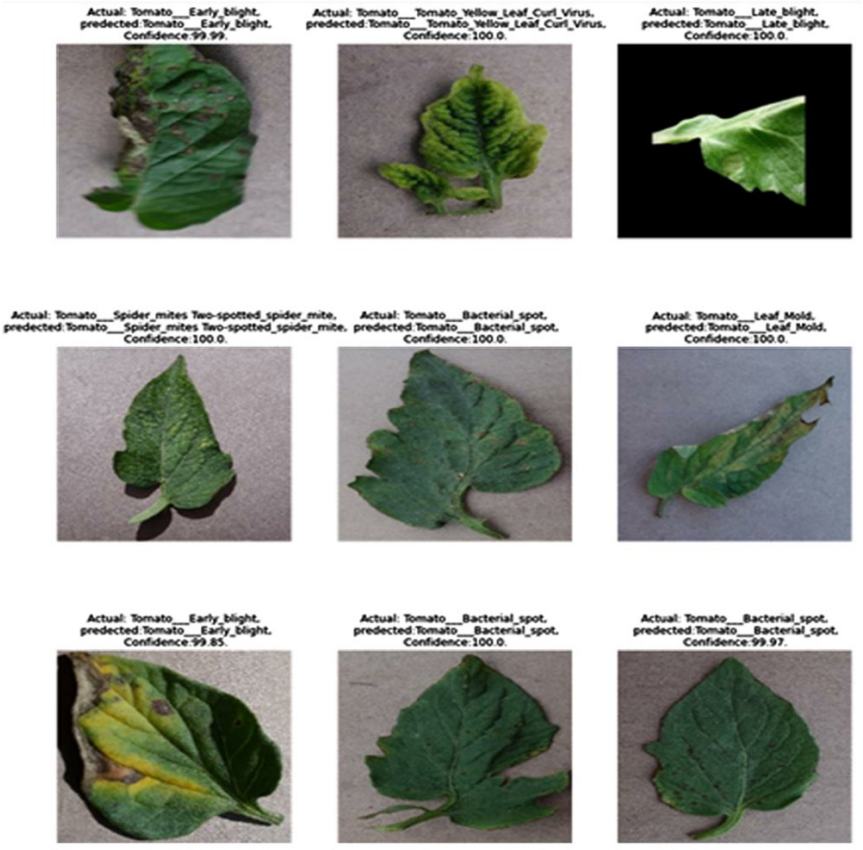


Figure 5.15 DenseNet201 Individual Dataset Results

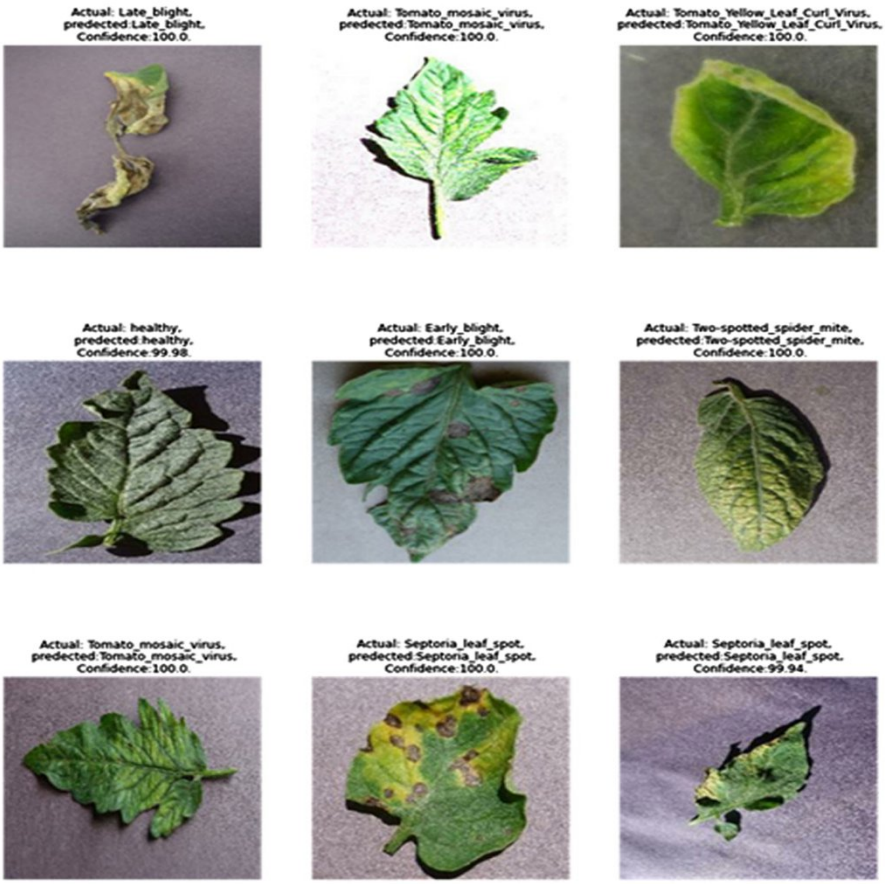


Figure 5.16 DenseNet201 Merged Dataset Results

5.3.6 Custom Model

Our custom CNN model comprises of the 10 layers shown in the figure 5.17.

- First layer consists of 16 filters with the kernel size of 3x3.
- These layers have average pooling of 2x2.
- Second and third layer contains 32 filters with the kernel size of 3x3.
- These layers have average pooling of 2x2.
- The fourth to sixth layer contains 64 filters with the kernel size of 3x3.
- These layers have average pooling of 2x2.
- The seventh to tenth layer has 512 filters with the kernel size of 3x3.
- These layers have average pooling of 2x2.
- The first Dense layers have units of 256.
- The second Dense layers have units of 64.
- The third Dense layers have units of 32.

With total layers ten layer and batch normalization.

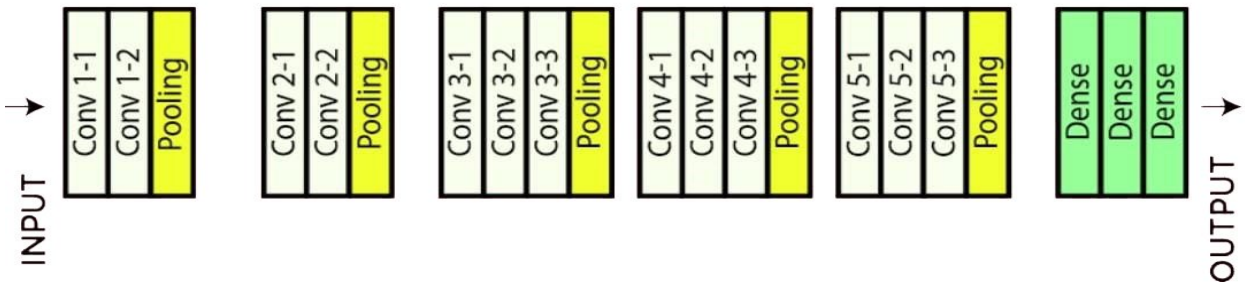


Figure 5.17 Custom Model Architecture

This model has a very simple network architecture. The model has constant 98% training accuracy and the validation accuracy changes. The model validation accuracy is very stable the only drop occurred at the end as presented in table 5.6.

Table 5.6 Custom Model Individual and Merged Dataset

Custom Model Individual Dataset				Custom Model Merged Dataset			
Accuracy	Loss	Val Accuracy	Val loss	Accuracy	Loss	Val Accuracy	Val Loss
98.21	1.7998e-04	98.78	.0483	95.86	.1574	70.52	1.4740
98.61	1.2213e-04	98.78	.0481	98.72	.0110	79.61	1.0076
98.70	9.4443e-05	98.82	.0486	98.96	.0032	99.01	.0328
98.95	7.4930e-05	98.89	.0497	99.0	.0012	99.11	.0355
99.0	6.0184e-05	98.75	.0487	99.1	9.1173e-04	100	.0289

These are the performance graphs of our custom model. The graph has converged fast but the stability is not that good as compared to the other models we have used in this research. As you can see the training accuracy and loss are very stable, but the validation accuracy and loss change rapidly. Although the model has performed better on the merged dataset as illustrated in figure 5.18. There are the results of 9 images that we have taken from the test data. The model has successfully predicted all the diseases accurately.

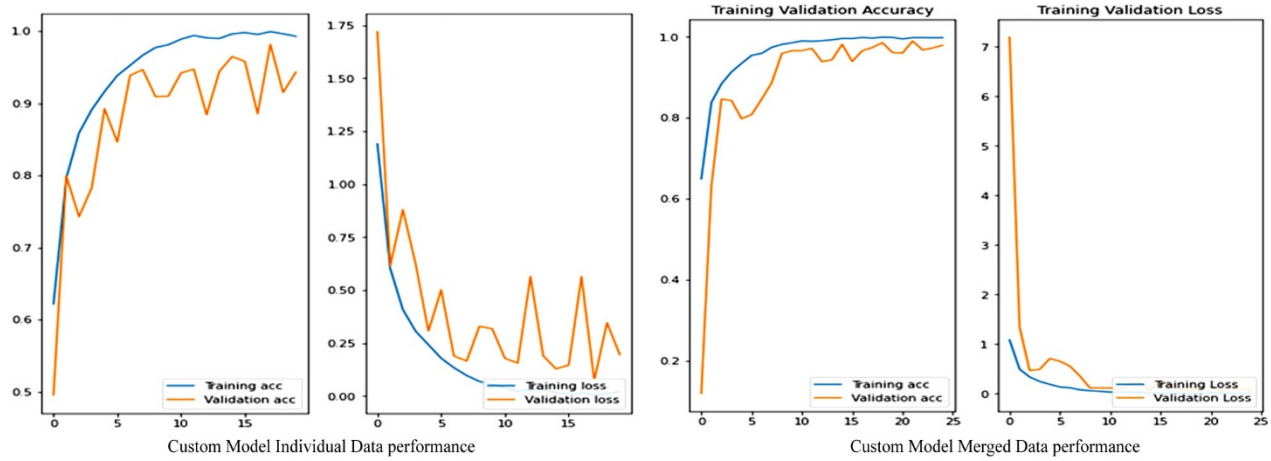


Figure 5.18 Custom Model Individual and Merged Dataset performance

The confidence of the model is averaging almost 99% except for the on-test subject where the model confidence is very low at 50% but the prediction is correct on the

Individual dataset while keeping the accuracy and confidence to almost 100% on the merged dataset from figure 5.19 and 5.20.

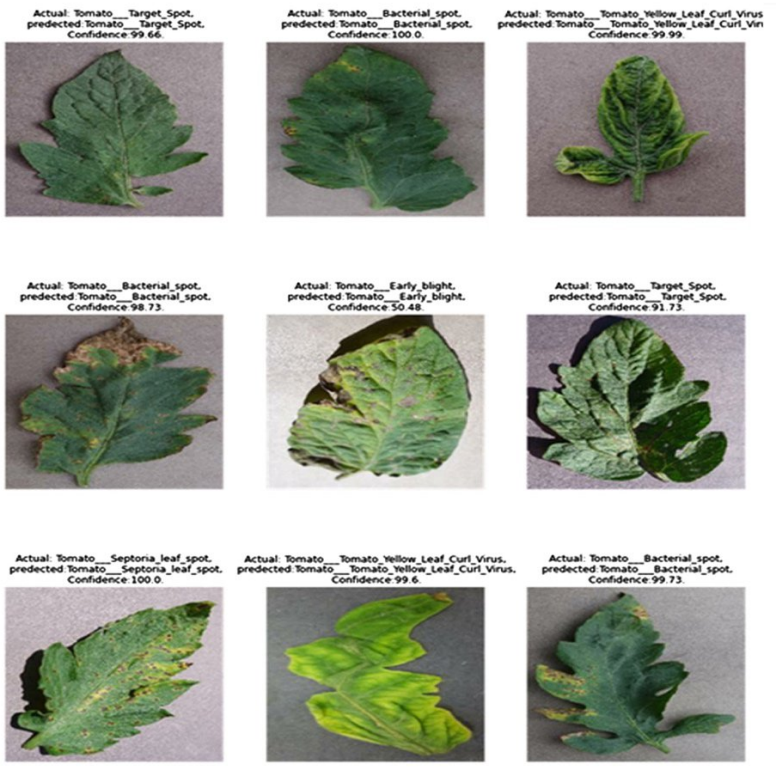


Figure 5.19 Custom Model Individual Dataset Results

Figure 5.20 Custom Model Merged Dataset Results

5.4 Comparative Analysis and Discussions

Unlike most prior research using transfer learning on diverse plant data, our study in this chapter targeted focus on the development of a custom CNN architecture for the identification of tomato diseases delivering superior accuracy while simultaneously minimizing both model complexity and computational requirements. In this chapter we have evaluated the performance of deep learning architectures for accurate and efficient detection of tomato plant diseases. Existing CNN architectures like VGG, ResNet, and DenseNet were evaluated for their performance, and a custom deep learning model was developed and tested. The custom model achieved over 99% accuracy in identifying 10 tomato plant diseases (including healthy plants). As compared to other CNN models our custom model achieved a high accuracy along with significantly less training time and lower computational cost. This chapter demonstrates the potential of deep learning for efficient and precise tomato plant disease detection, offering practical benefits to farmers and contributing the overall agricultural production. This efficient performance of our custom model makes it promising for real-world agricultural applications. Different CNN architectures along with custom model have been trained on a very large dataset in this chapter for achieving the best accuracy for detecting plant diseases. In this section we have compared all the models for choosing the best model based on higher performance. Table 5.7 and 5.8 shows the performance evaluation based on individual and merged data.

5.4.1 Comparative Analysis Based on Individual Dataset

For each model created using a specific dataset, the accuracy and loss are represented in table 5.7 for training, testing, and validation. ResNet50 and DenseNet169 have the highest test accuracies, according to the table, but ResNet50 is more effective. The VGG16 also has the lowest level of accuracy overall. The graph in figure 5.21 illustrates the performance of the models. The DenseNet201 model is the best model followed by ResNet50 in this study, as shown by the graphic. Figure 5.21 shows that ResNet50 has the highest overall training validation and test accuracy.

Table 5.7 Comparison Table with Individual Dataset

Model	Optimizer	Layers	Parameters	No. of Epochs	Training Accuracy	Training loss	Validation Accuracy	Validation loss	Test Accuracy
VGG16	Adamax	16	27,514,698	50	99.32%	0.0200	98.10%	0.0618	98.79%
Custom	Adamax	10	8,598,090	50	100%	6.0618e05	98.75%	0.0487	99.22%
ResNet 50	Adamax	50	6,315,018	50	100%	1.202e04	99.44	0.0203	99.78%
DenseNet121	Adamax	121	51,914,250	50	99.25%	0.0272	99.41%	0.0203	99.55%
DenseNet169	Adamax	169	84,028,170	50	99.15%	0266	99.37%	0.0246	99.78%
DenseNet201	Adamax	201	96,873,738	50	99.78%	0098	99.31%	0.0211	99.67%

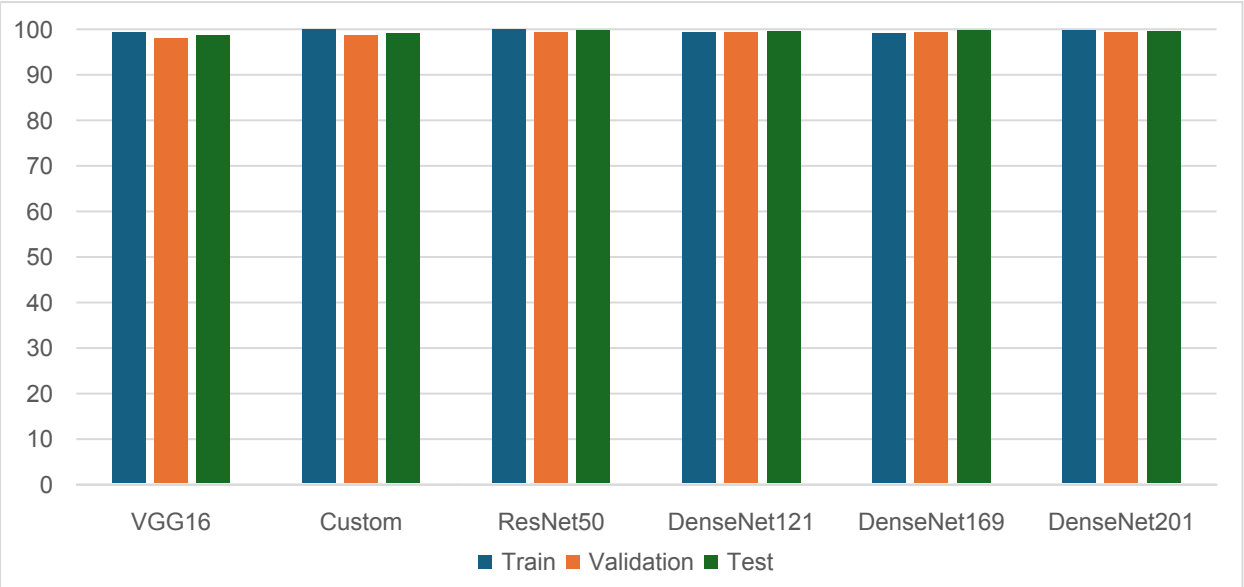


Figure 5.21 Comparison Chart of Individual Dataset

5.4.2 Comparative Analysis Based on Merged Dataset

For all the models created using the combined dataset, the training, testing, and validation accuracy and loss are shown in table 5.8. According to the table, DenseNet201 and DenseNet169 have the best test accuracies, but DenseNet201 is superior. Additionally, the VGG16 once more has the lowest overall accuracy. The graph in figure 5.22 demonstrates that, except for the VGG16, most models have fared extremely well. With Dense201, the ResNet50 has performed best, followed by our custom model.

Table 5.8 Comparison Table of Merged Dataset

Model	Optimizer	Layers	Parameters	Training Accuracy	Training loss	Validation Accuracy	Validation loss	Test Accuracy
VGG16	Adamax	16	27,514,698	99.91%	0.0037	92.06%	0.6915	92.06%
Custom	Adamax	10	8,598,090	100%	9.1173e-04	99.22%	0.0289	99.24%
ResNet 50	Adamax	50	6,315,018	100%	1.5780e-04	99.45%	0.0284	99.34%
Dense Net121	Adamax	121	51,914,250	99.46%	0.0193	99.24%	0.0209	98.93%
Dense Net169	Adamax	169	84,028,170	99.84%	0092	99.02%	0.0358	98.87%
Dense Net201	Adamax	201	96,873,738	99.37%	0204	99.97%	0.0018	100%

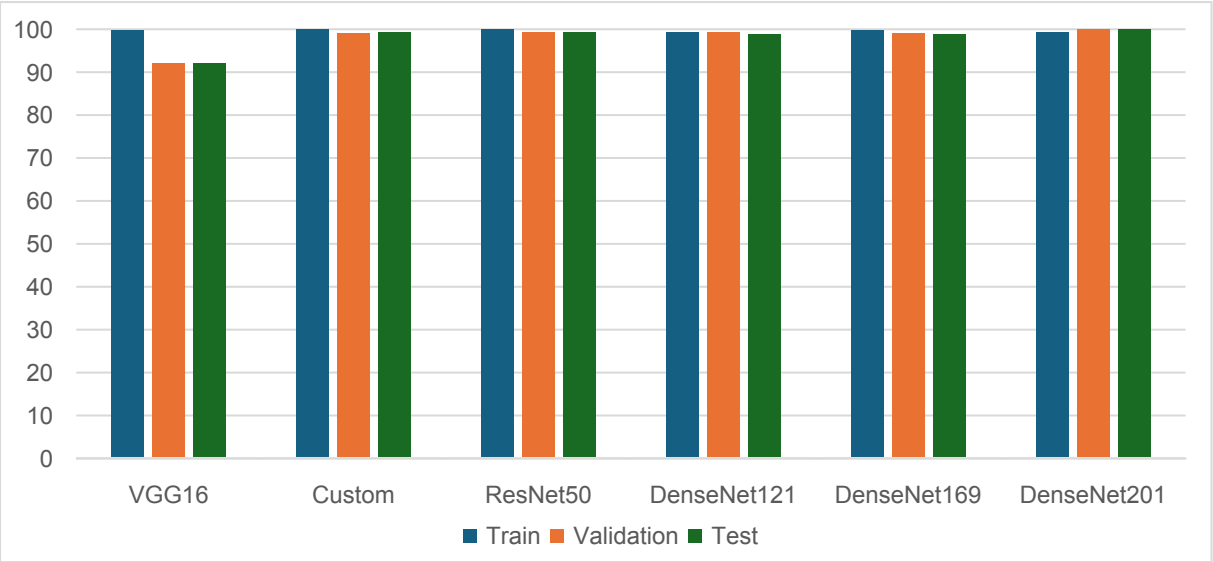


Figure 5.22 Comparison Chart of Merged Dataset

5.5 Summary

This chapter has presented a comprehensive approach for tomato plant diseases detection. Building upon the foundational work of previous chapters, which established a pipeline for leaf segmentation and extraction, this chapter focused on the design and validation of a purpose-built deep learning model for disease classification. The chapter began by detailing the methodology, including the curation of a diverse dataset from both publicly available sources and real-world field images. This dual-source approach, combined with extensive data augmentation techniques, was very important for ensuring the robustness and ability of the model to generalize to real-world conditions. The core of the chapter described the enhanced deep learning architecture; a custom Convolutional Neural Network specifically designed for efficient and accurate detection of diseases on individual tomato leaf images. The architecture was optimized with specialized layers and regularization techniques to balance high performance and computational efficiency, making it ideal for practical, real-time applications. The model training and optimization section elaborated on the rigorous process of training the model. It explained how we utilized the Adam optimizer and categorical cross-entropy loss, and crucial techniques such as a learning rate scheduler and early stopping, to maximize the model's performance and minimize the overfitting. Finally, the results and discussion section presented the research's conclusive findings. The proposed model has achieved a remarkable overall accuracy of 99% on an unseen data, significantly surpass the performance of benchmark architectures like VGG, ResNet, and DenseNet. The chapter concluded that the integrated, multi-stage approach combining precise leaf segmentation with an enhanced diagnostic model

successfully addresses the critical challenges of real-world deployment, enabling the creation of more intelligent and sustainable agricultural systems.

Chapter 6: Intelligent Disease Detection and Automated Spray Prescription System

This chapter presents, an integrated system for Intelligent Tomato Leaf Disease Detection and Automated Spray Prescription, leveraging the advanced deep learning model. Moving beyond static identification of an image, this chapter targeted its focus on developing a system that can accurately detect and localize nine specific tomato leaf diseases in a real-time environment. The central contribution of this chapter is the seamless integration of a highly optimized YOLOv9 model with an automated spray prescription mechanism, creating a smart agriculture solution that minimizes resource waste and environmental impact.

The rest of this chapter is structured as follows: Section 6.1 discusses the introduction in detail. Section 6.2 details the proposed methodology, including dataset acquisition and preparation, model architecture, training procedures, and integration with the prescription system. Section 6.3 provides the experimental results and comparative analyses with existing approaches. Section 6.4 reviews the consequences of our findings, limitations of the system, and potential applications. Finally, Section 6.5 summarizes the chapter and suggests future research directions.

6.1 Introduction

As discussed in earlier chapter that tomato is one of the most economically significant horticultural crops in the world with an annual global production of over 186 million tonnes and a market value of over $190 billion.

As in real-time agriculture applications, accuracy and speed are considered as very critical factors. Keeping these challenges in mind the YOLO family has gained a remarkable attention due to its outstanding performance regarding its speed and accuracy [96]. These enhancements ensure minimizing the computational cost and higher the detection accuracy for small objects, which are helpful in earlier detection of plant leaf diseases due to the leaf visual anomalies.

While previous chapters have established the critical need for automated disease detection and have presented an enhanced deep learning architecture for accurate disease diagnosis on individual leaves, a complete solution for real-world agricultural settings requires a robust, end-to-end system capable of working in real-time. The challenge lies not only in identifying the disease but also in localizing it on the plant and translating that information into a precise, actionable response. The proposed work in this chapter highlights the key limitations of existing research by offering a holistic framework that operates on images of full plants,

directly identifying and bounding diseased areas for targeted treatment. By leveraging the speed and accuracy of YOLOv9, the system can make efficient, precise, and accurate decisions in a dynamic field setting [97]. The chapter will detail the modifications and optimizations applied to the YOLOv9 model to enhance its performance for this specific application.

The major goals of this chapter are:

1. To build a YOLOv9-based detection system for nine tomato leaf conditions with high accuracy (>95% mAP) and real-time processing capabilities (<50ms per frame on standard hardware).
2. To create an integrated framework that automatically translates disease detection outcomes into appropriate treatment recommendations based on disease type, severity, crop phenology, and environmental considerations.
3. To quantify the system's detection performance across diverse imaging conditions, including variable lighting, viewing angles, and background complexities.
4. To evaluate the potential reduction in pesticide usage achieved through the targeted application approach compared to conventional calendar-based spray programs.
5. To assess the economic viability of the system through cost-benefit analysis considering implementation costs versus savings in crop protection products and potential yield preservation.

6.2 Materials and Methods

This work starts with linear workflow for detecting and classifying the diseases in tomato leaf using YOLOv9 model and spray prescription based on the detected disease. The research flow consists of the four stages including data acquisition and preparation, the model architecture, disease detection and performance evaluation, and spray prescription establishing a standardized approach for model development and performance evaluation. This approach provides a robust and accurate solution which is appropriate for real-world precision agriculture applications. All the steps in the methodology are interconnected and each contributing to overall performance of this work as illustrated in the figure 6.1.

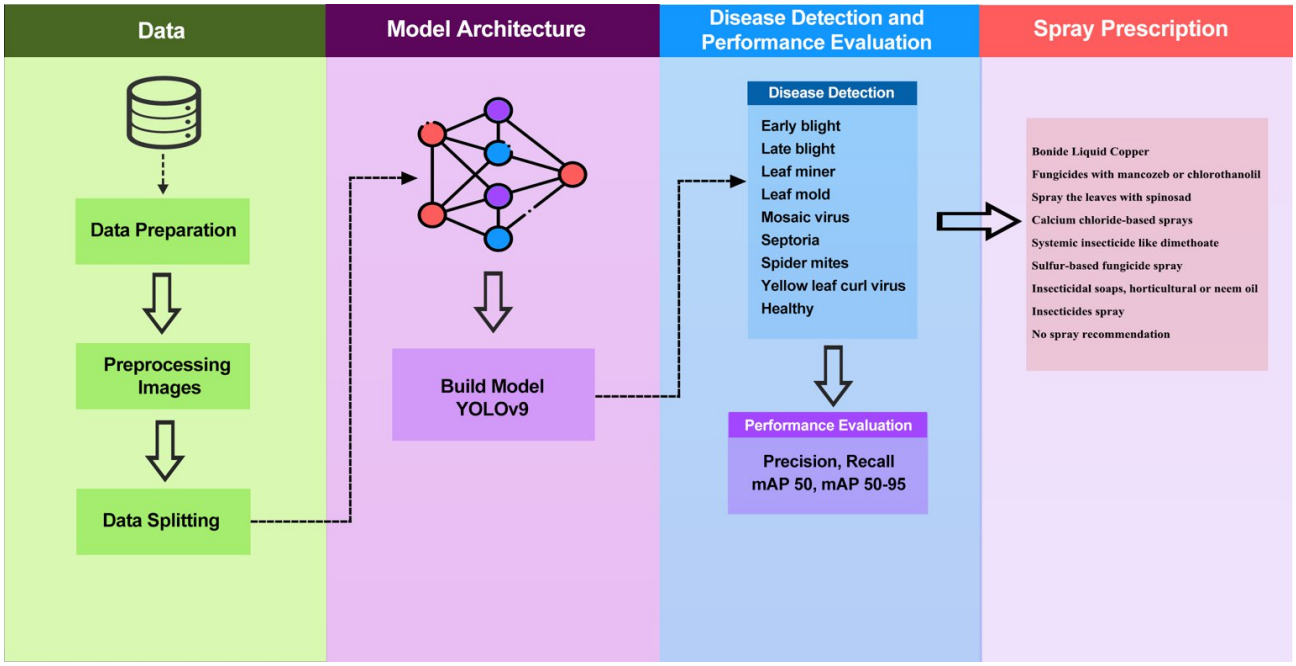


Figure 6.1 Research Flow

6.2.1 Data Acquisition and Data Preparation

Images of tomato leaves were collected from publicly available datasets (RoboFlow Universe) and field-captured photographs using drone camera, focused on common tomato diseases including healthy leaves. The size of our dataset comprises of 10,047 images with balanced distribution across classes as shown in Table 6.1. To match the YOLOv9's input requirements the images were resized to 640x640 pixels. For improving the generalization, the augmentation techniques like rotation, flipping, scaling, and HSV adjustments are applied. The dataset is divided into training (70%), validation (15%), and testing (15%) sets. For annotating the unlabelled images which we have obtained through drone camera has been marked labelled using LabelImg annotation tool.

Table 6.1 Class labels and no. of images associated with each class

Class Label	No. of Images	Size of the Image
Early blight	3000	640 x 640 pixel
Late blight	1200	640 x 640 pixel
Leaf miner	1234	640 x 640 pixel
Leaf mold	4532	640 x 640 pixel
Mosaic virus	1234	640 x 640 pixel
Septoria	432	640 x 640 pixel
Spider mites	43	640 x 640 pixel
Yellow leaf curl virus	124	640 x 640 pixel
Healthy	432	640 x 640 pixel

6.2.2 Data Preprocessing

For loading the images and labels from the repository, this provided model automatically performs the data preprocessing for the classification tasks. For images conversion to RGB format, the OpenCV(cv2) is used which read the images from the specified paths and convert them, which show the consistency in across the images. In preprocessing of the data, we achieved a uniform image formatting which is RGB format, proper organization of images and class labels paths, and preparing data YOLOv9 training. For image preprocessing, YOLOv9 visualizes the ROI by creating bounding boxes around them within the images. For tomato disease detection in the tomato leaves, this is considered as very important step. The process involved going through the initial five training images and their labels. For each step, the model displayed the image and drew a bounding box on it. To ensure these boxes were accurately placed and sized on the actual image, the model converted their relative coordinates into absolute pixel values using the height and width of the image. For improving the generalization of the images and enhancing the robustness, the augmentation techniques are implemented to the dataset. This strengthens the model, enhancing its ability to generalize when simulating varied real-world scenarios.

6.2.3 Model Architecture

A neural network is, at its core, a computational model that's good at finding patterns and making decisions from data. It's build-up of interconnected "neurons" in different layers that process the input data and generates output based the provided input. There's an input layer to get the data, a bunch of "hidden" layers that do most of the complex thinking and transformations, and an output layer for the result. These hidden layers are key; the more you have, the better the network can learn intricate patterns.

$$z = W\alpha + b \qquad (1)$$

The equation for input layer is given in the equation 1, where:

- z is the input passed to the activation function
- W is the weight matrix for the current layer
- α is the output from the previous layer.

α is the activation function (e.g., Sigmoid, ReLU, or Tanh) that processes z.

$$a = \sigma(z) \qquad (2)$$

$$\hat{y} = W^L . \alpha^{L-1} + b^L \qquad (3)$$

Equation 2 shows, that $\hat{y}$ which is the final output or the network prediction which calculated by operating the activation function and weighted sum to the output of the last hidden layer. Equation 3 defines the output using the weight matrix (W^L) of the final layer and the activation output (α^{L-1}) of the last hidden layer with the calculated bias incorporated. YOLOv9 is designed to manage complex data while providing real-time object detection. Its configuration, including its architecture and custom data loader, facilitates efficient image processing. By taking the advantage of underlying principles and features extracted from an autoencoder model, YOLOv9 ensures optimal object detection performance. The YOLOv9 utilized consist of default backbone (CSPDarknet), neck (PANet), and detection head. The anchor boxes are modified to suit smaller leaf lesion sizes.

For preparing the images for robust and efficient preprocessing the data loader is used for performing essential preprocessing steps, which reduce the size of images to 128x128 pixels using transforms. The data loader normalizes and resizes the pixel values to prevent feature control. The attention mechanisms like (CBAM modules) are enabled for improved feature extraction. The configuration of the model is as follows:

- The pre-trained COCO weights are used for model initialization.
- Batch size: 16
- Optimizer: Adam with momentum (0.937) and weight decay (0.0005).
- Learning rate of 0.01 along with cosine annealing scheduler.
- No. of Epochs: 150
- Hardware: Trained on NVIDIA GPUs (e.g., RTX 3090/4090) with CUDA acceleration.

6.2.4 Disease Detection and Model's Performance Evaluation

To assess the performance of the YOLO models, the performance evaluation is very important to know about the efficiency and accuracy of the model. To assess the performance of the YOLO models, the following key metrics are used: precision score, recall score, mAP0.5, and mAP0.5-0.95. The precision score measures the proportion of the positive predictions by the model that were actually correct (the correct diseases it detected). In contrast, the recall score calculates the proportion of all actual diseases that the model detected successfully. mAP is used to assess this performance. The IoU metric itself quantifies the spatial accuracy by computing how well a predicted bounding box overlaps with the ground truth bounding box.

mAP (0.5-0.95) provides a detail evaluation of the detection accuracy across a range of overlap thresholds, with a higher score indicating superior performance. For the performance evaluation our model, we used the precision, recall, F1-score, mAP (at both and IoU

thresholds), and a Confusion Matrix for error visualization. Fundamentally, mAP is the average of the AP values computed across various IoU thresholds, where AP is determined by combining the Precision-Recall curve. To generate the Precision-Recall curve the precision and recall at various confidence levels are measured systematically. To get an overall sense of performance the area under this curve using a numerical method, like the trapezoidal rule is then approximated. Mean Average Precision is computed for more robust evaluation. This includes computing average precision across different IoU thresholds (like 0.5 or a range from 0.5 to 0.95) and then taking the average of those AP values to calculate at the final mAP score.

6.2.5 Spray Recommendation Module

The spray prescription module serves as the intelligent brain of a precision agriculture system, translating the raw output from disease detection models, such as YOLOv9, into actionable, site-specific spraying instructions. After the YOLOv9 model accurately identifies and localizes tomato diseases by providing bounding boxes along with labels and confidence scores, the prescription module takes over to determine what, how much, and exactly where to spray. In this research work we have collected data regarding sprays used to tomato plant treatment as shown in table 6.2. This module simply prescribes the spray based on the detected tomato plant disease. We created a lookup table linking diseases to recommended sprays (e.g., copper-based fungicides for bacterial spots, chlorothalonil for blight).

Table 6.2 Spray List

No.	Spray
1	Bonide Liquid Copper
2	Use Fungicides with mancozeb or chlorothalonil
3	Spray the leaves with Spinosad
4	Calcium chloride-based sprays
5	Spraying of systemic insecticide like dimethoate
6	Spray the leaves with Sulfur-based fungicide
7	Spray the leaves with insecticidal soaps, horticultural oils, or neem oil
8	Spray the leaves with Insecticides
9	No spray recommendation

6.3 Experimental Setup and Results

This section provides the experimental results derived from applying the YOLOv9 model for tomato plant disease detection and subsequent disease assessment, leading to spray prescription. A trained YOLOv9 model is first assessed, followed by a detailed analysis and the derived practical recommendations. The performance of the YOLOv9 has evaluated in detail using the above discussed metrics on the test dataset. The confusion matrix generated summarize the performance of YOLOv9 on tomato leaf disease detection in figure 6.2. This

confusion matrix included nine classes with an additional “background” class. The background class suggests that the model is detecting background (soil and background plants). The results in the confusion matrix show that our model achieves a higher classification accuracy, as evidenced by the higher values focused on the main diagonal, demonstrating a True Positive Rate (Recall) for each class. Particularly, the model showed an excellent performance in detecting “Leaf Miner”, “Late Blight”, “Early Blight”, and “Leaf Mold” with a higher recall rate. The major areas of misclassification are generally low. However, the largest misclassification occurs for “Yellow Leaf Curl Virus”, which is often confused with the “background”. Figures show that 34% of the time, the model predicted the background as Yellow Leaf Curl Virus. Similarly, the “background” class itself shows notable confusion, with 20% of true “Healthy” samples being incorrectly classified as “background” and 34% of true “Yellow Leaf Curl Virus” being classified as “background”. Overall, the matrix confirms that YOLOv9 is highly effective at distinguishing between the various specific disease types.

The YOLOv9 model, specifically trained for the detection of diseases on tomato leaves, demonstrated robust performance across the test dataset. Quantitative evaluation was conducted using standard evaluation metric as discussed above.

The performance of YOLOv9 is assessed with standard segmentation metrics including qualitative visual assessments. We also report the robustness and efficiency of how well YOLOv9 and spray prescription module perform together to detect diseases in tomato leaves and prescribe the spray based on detected disease.

For the detecting disease in tomato leaves, the performance evaluation metrics are listed below (4)–(7).

$$\text{precision} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{5}$$

$$\text{F1-score} = \frac{(2*TP)}{(2*TP + FN + FP} \tag{6}$$

$$\text{IoU} = \frac{(TP)}{(TP + FN + FP} \tag{7}$$

where:

- FP: False Positives (incorrectly predicted positive cases)
- TP: True Positives (correctly predicted positive cases)
- FN: False Negatives (incorrectly predicted negative cases)

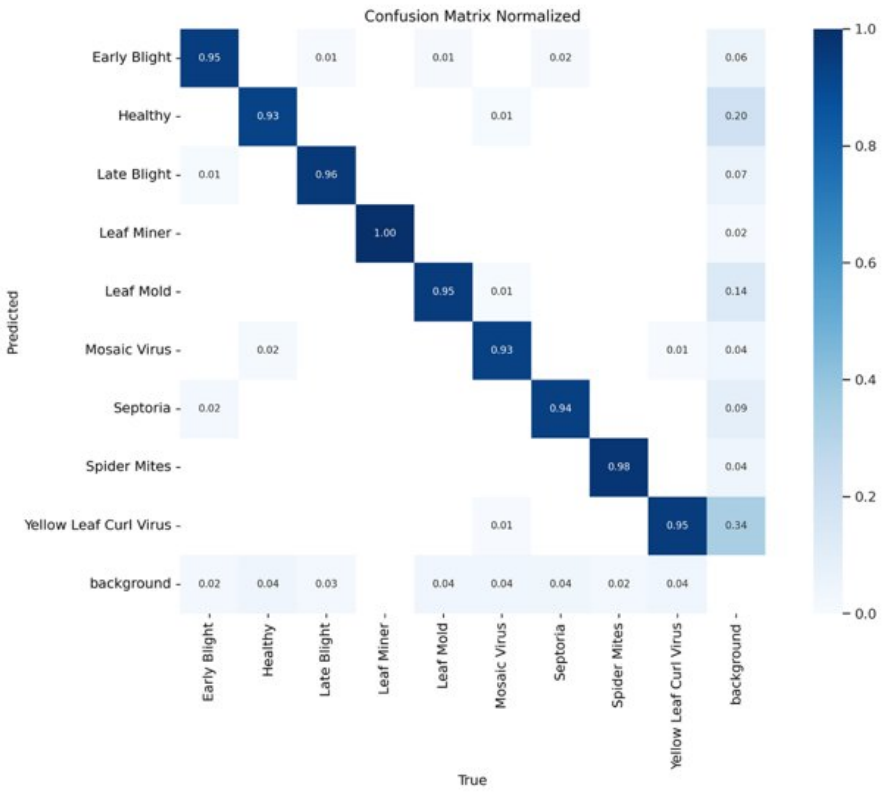


Figure 6.2 Confusion Matrix Normalized

Beyond basic testing, we thoroughly evaluated the model by using a range of IoU thresholds, from a moderate 0.5 to a very demanding 0.95. This approach gave us a much clearer picture of its performance, especially under conditions requiring very precise bounding box placement. In figure 03 the training visualization graph illustrates the mAP scores at these varied IoU levels, proving YOLOv9's impressive ability to maintain strong performance even when precise object localization is critical. Notably, YOLOv9 surpassed its predecessors, achieving an average mAP of 97% at a 0.5 IoU threshold and an excellent 93% when averaged across the 0.5-0.95 IoU range. It was good at recognizing other diseases with high accuracy but in recognizing the leaf curl virus disease with an accuracy of 83.2%. This network showed an excellent performance, as compared with both YOLOv7 and YOLOv8 by approximately 3%–4% in terms of mAP. Crucially, it reduced misclassification errors, a benefit particularly noticeable with diseases that have overlapping symptoms.

Compared to earlier versions YOLOv9 shows an improved mAP@0.5 values specifically in handling ambiguous visual symptoms and overlapping disease patterns. Such performance and improvements are very important to real-world agriculture applications which requires high reliability and accuracy. These metrics shows an improvement over YOLOv7 and YOLOv8, proving the accuracy and efficiency of YOLOv9. The higher mAP scores show its capability

in handling challenging disease patterns and overlapping bounding boxes, which is critical for real-world deployment.

In detecting the tomato leaf diseases, the YOLOv9 proves an outstanding performance on a validation dataset consisting of 843 images by achieving precision score of 95.8%, a recall score of 92.4%, 97% (mAP50) and 93% (mAP50-95). The inference speed per image was recorded 13.5ms including 0.3ms preprocess and 0.9ms post process speed. The effectiveness of the model varied across classes, as "Leaf Miner" class revealing the higher accuracy (99.5% at mAP50), indicating robust detection capabilities for infected tomato plants. Table 6.3 shows model's performance in more detail by mentioning each metrics with each class.

Table 6.3 Model Performance

Class	Images	Precision Accuracy	Recall Accuracy	mAP50	mAP50-95
All	**843**	**0.958**	**0.924**	**0.97**	**0.93**
Early Blight	214	0.956	0.943	0.981	0.955
Healthy	76	0.926	0.874	0.927	0.87
Late Blight	238	0.985	0.944	0.982	0.948
Leaf Miner	199	0.981	0.996	0.995	0.979
Leaf Mold	211	0.947	0.901	0.969	0.927
Mosaic Virus	250	0.974	0.936	0.973	0.958
Septoria	203	0.982	0.88	0.971	0.935
Spider Mites	123	0.976	0.976	0.984	0.967
Yellow Leaf Curl Virus	152	0.896	0.87	0.946	0.832

The results in table 6.3 shows the accuracy YOLOv9 in detecting tomato leaf diseases in the provided dataset. The table presents the performance of YOLOv9 which varies across different classes. Results indicates that the performance of the model is outstanding on the "Leaf Miner" class with a mAP of 0.979, but in case of "yellow leaf curl virus" class, the model struggled by achieving a mAP of 83.2%. These results clearly show the real-time capability, which is necessary for applications requiring fast decision-making, such as disease detection or pest management.

In training visuals, e-g figure 6.3, we keep an eye on two main things. The "train/box_loss" essentially tells us how "on target" our predicted boxes are. When you see this number dropping, it's a clear sign that the model is getting much better at accurately finding and outlining objects. Similarly, the "train/cls_loss" tracks how good the model is at naming what it sees. If this number is also going down, it means the model is improving its identification skills, which is crucial when its job is to spot tomato plant diseases on leaves. The plot on train/dfl_loss adds both classification losses and box into a single metric. A decrease in the train/dfl_loss plot shows the overall improvement in the performance to detect objects. Due to

the visual similarities between the lesion pattern of septoria and early blight the misclassifications may primarily occurred.

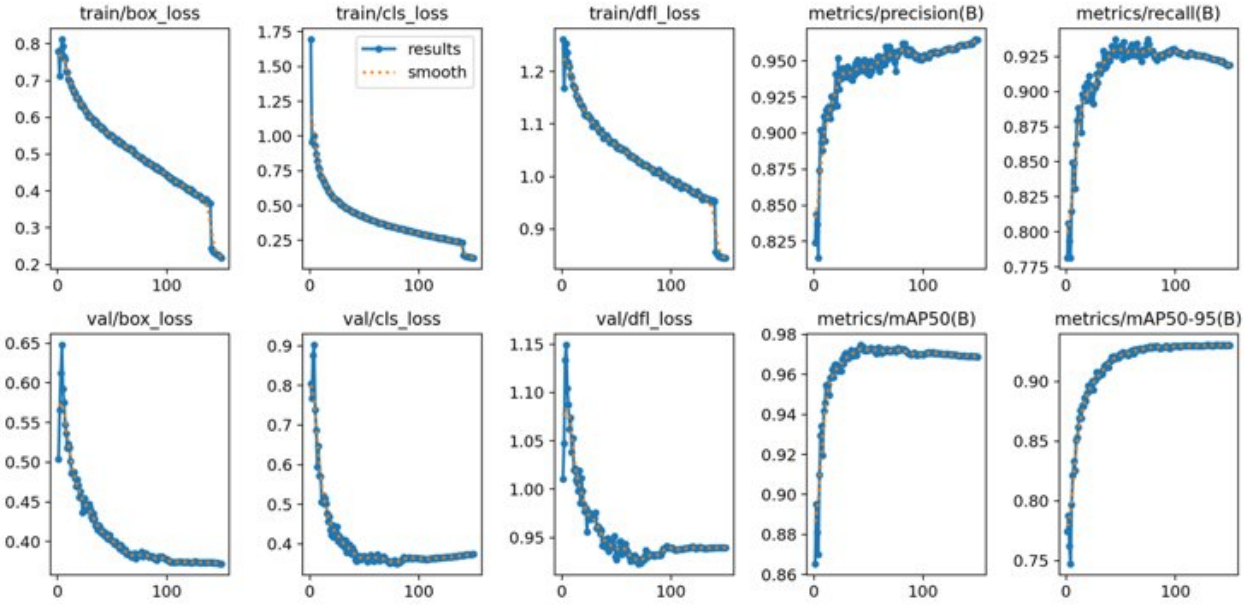


Figure 6.3 Training Visualization Results

The sizes, colours and shapes of the lesions make the differentiation process very complex. These visual similarities make it challenging for the model to differentiate the unique features of diseased leaves from one another, especially when processing partially damaged leaves or overlapping environmental conditions.

The performance of the model was outstanding in identifying the infected plants with high precision and recall accuracy. For assessing the quality of tomato leaf disease detection, the precision and recall score for every image were determined at the pixel level. The proportion of true positives are measured by the precision curve while the recall curve measures the actual positives proportion, that were correctly predicted. The precision-recall curve was 0.98% @mAP50 shown while the F1 confidence curve which is 0.94% is shown in figure 6.4, reflecting the ability of the model to correctly detect the tomato leaf diseases and reduce the false positives. The validation results and model's results on test dataset is shown in the figure 6.5 and 6.6 respectively. After the disease detection correctly by the YOLOv9 a spray prescription module which is integrated with the model automatically prescribed the spray based on the detected disease. Figure 6.7 show how the mapping between the diseases and spray is implemented.

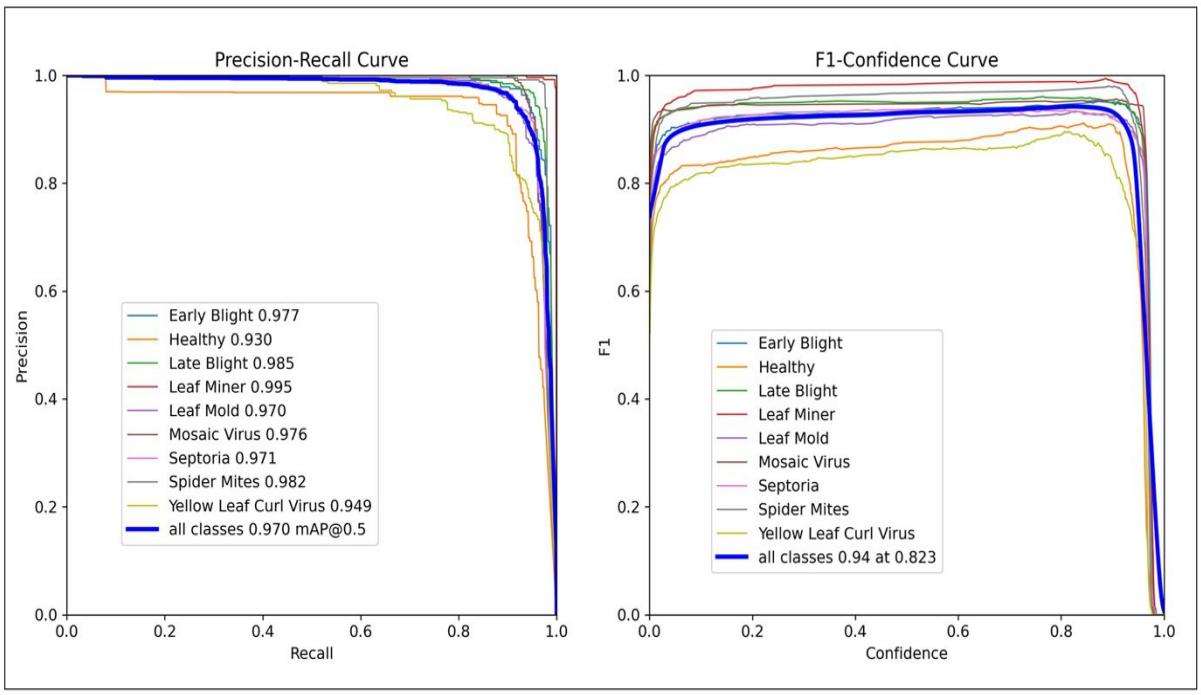


Figure 6.4 Precision-recall accuracy @mAP50 and F1 confidence curve

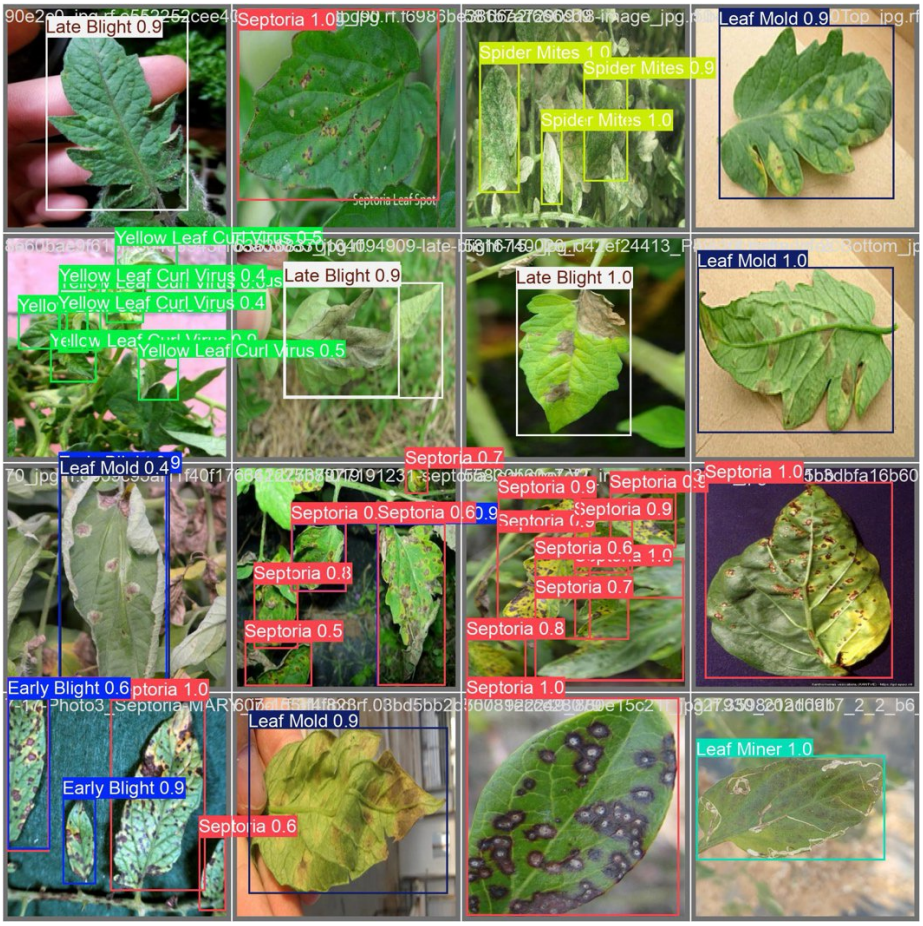


Figure 6.5 Validation results

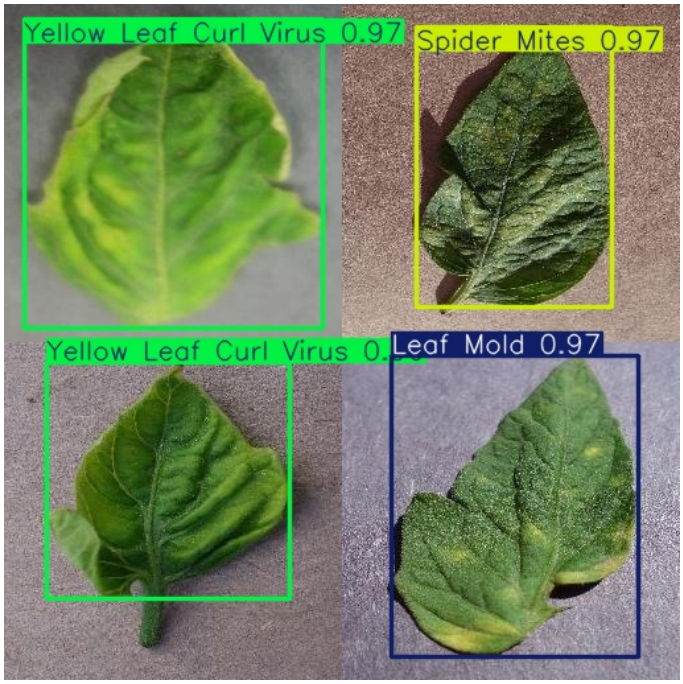


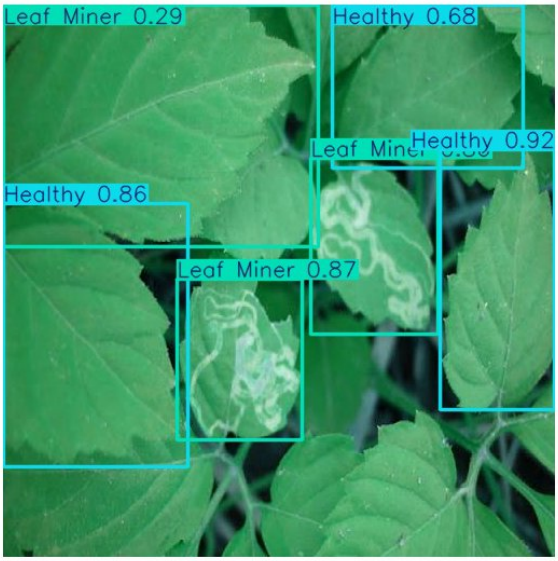
Figure 6.6 Results on test dataset

```
[14]: # Import required libraries
from ultralytics import YOLO
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

# Load the YOLOv8 model (replace with your trained model)
model = YOLO("/home/shahab/Desktop/my_projects/yolov9/best.pt")  # Replace with your custom-trained YOLOv8 model

# Define disease-to-spray mapping
disease_spray_mapping = {
        0:{"disease": "Early Blight", "spray": "Spray the plant with Bonide Liquid Copper Fungicide concentrate"},
        1: {"disease": "Healthy", "spray": "No spray recommendation"},
        2: {"disease": "Late Blight", "spray": "Use Fungicides that contain maneb, mancozeb, chlorothanolil, or fixed copper"},
        3: {"disease": "Leaf Miner", "spray": "Spray the leaves with spinosad it can be an effective option"},
        4: {"disease": "Leaf Mold", "spray": "Calcium chloride-based sprays are recommended for treating leaf mold issues"},
        5: {"disease": "Mosaic Virus", "spray": "Spraying of systemic insecticide like dimethoate"},
        6: {"disease": "Septoria", "spray": "Spray the leaves with Sulfur-based fungicide"},
        7: {"disease": "Spider Mites", "spray": "Spray the leaves with insecticidal soaps, horticultural oils, or neem oil"},
        8: {"disease": "Yellow Leaf Curl Virus", "spray": "Spray the leaves with Insecticides for whiteflies"},
    }

# Function to detect diseases and prescribe sprays
def detect_and_prescribe(image_path):
    # Load the image
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert to RGB for display
    original_image = image.copy()

    # Perform detection using YOLOv8
    results = model(image)
```

Figure 6.7 Spray Mapping

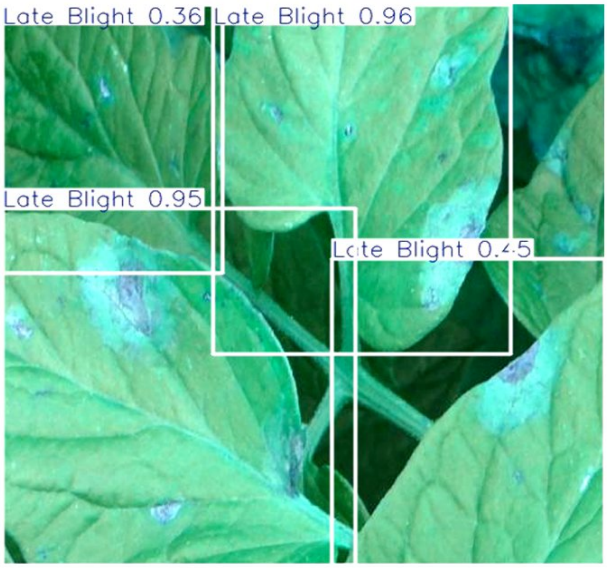
The spray prescription result after disease detection is illustrated in the figure 6.8 and 6.9. The model performs well in identifying the leaf miner disease, but it's showed misclassifications for yellow leaf curl virus. After the overall performance of it shows that the YOLOv9 reduces the workload regarding tomato plant disease detection because it makes the disease detection process automated, hence there is no need of acquiring the field expert. However, in case of

yellow leaf curl virus where the model's accuracy is reduced by misclassification, the model needs improvement by applying the domain-specific features (spectral imaging) or applying extra augmentation techniques.




```
0: 640x640 3 Healthys, 3 Leaf Miners, 17.7ms
Speed: 3.6ms preprocess, 17.7ms inference, 1.2ms postprocess per image at shape (1, 3, 640, 640)
Results saved to runs/detect/predict7
<Figure size 1000x1000 with 1 Axes>
Detected Diseases and Spray Recommendations:
- Healthy: No spray recommendation
- Leaf Miner: Spray the leaves with spinosad it can be an effective option
```

Figure 6.8 Spray Prescription for Leaf Miner




```
0: 640x640 4 Late Blights, 17.7ms
Speed: 1.9ms preprocess, 17.7ms inference, 1.3ms postprocess per image at shape (1, 3, 640, 640)
Results saved to runs/detect/predict9
<Figure size 1000x1000 with 1 Axes>
Detected Diseases and Spray Recommendations:
- Late Blight: Spray the infected leaves with Fungicides that contain maneb, mancozeb, chlorothanolil, or fixed copper
```

Figure 6.9 Spray Prescription for Late Blight

6.4 Discussion

The application of YOLOv9 for tomato disease detection, coupled with an automated spray prescription module, represents a significant advancement in precision agriculture. This integrated system offers a powerful tool for timely and targeted intervention, contributing to enhanced crop health, reduced chemical usage, and improved yield. Our YOLOv9 model demonstrated promising capabilities in accurately identifying various tomato diseases. The architecture's inherent strengths, such as high object detection accuracy and its abilities in real-time application, are suitable for agricultural applications where rapid diagnosis is crucial. By taking the benefits of the advanced detection, and feature extraction mechanisms of YOLOv9, the system can identify infected and healthy leaves, and even identify different types of diseases, with higher accuracy. This is a considerable improvement over traditional inspection methods that are conducted manually, which required more time and labour cost, and prone to human error and subjectivity. The ability to localize lesions and identify disease types enables more precise diagnosis, which is a critical first step towards effective management.

The seamless integration of the disease detection module with an automated spray prescription system is a key strength of this work. After the disease identification by YOLOv9, the system automatically triggers a recommendation for the appropriate pesticide or treatment, along with the recommended dosage and application method. This automation streamlines the decision-making process for farmers, minimizing delays and ensuring that interventions are applied at the optimal time.

6.4.1 Comparative Analysis

A comparative analysis of the past algorithms that have been employed for tomato leaf disease detection is shown in the table 6.4. It summarizes each algorithm used by the authors along with the performance metrics like precision, recall and mAP and the integration of models into spray prescription for real-time applications. For automated tomato leaf disease detection, these studies adapt various approaches for developing deep learning models to enhance the accuracy of the models through architectural improvements.

Table 6.4 Comparison of our model with different detection models

Approach	Precision	Recall	mAP50	Spray Prescription
DM-YOLOv9	92.5%	86.8%	86.4%	No
DMW-YOLOv8	95.4%	83.6%	88.6%	No
YOLOv8n	76.1%	77.6%	82.4%	No
BED-YOLO (YOLOv10)	87.2%	89.1	91.3	No
Our Work	**95.8%**	**92.4%**	**97%**	**Yes**

Notably, by integrating the YOLOv9 model with a spray prescription module, our study distinguishes itself enabling real-time, automated pesticide application. The above comparison indicates the performance of YOLOv9 in different conditions like variant light conditions and the presence of small leaves. This comparison also highlights the practical implication of the YOLOv9 in precision agriculture. Our proposed work has shown outstanding performance in terms of detecting tomato leaf diseases and automatically prescribing spray for tomato leaves based on detected disease. It is important to note that other models, such as DM-YOLOv9 [98], DMW-YOLOv8 [99], YOLOv8n [100], and BED-YOLO [101], has its merits, but this research work indicates a comparable level of performance, as shown in table 6.4. The visualization of comparatives analysis with the previous models with respect to precision, recall and mAP metrics is shown in figure 6.8.

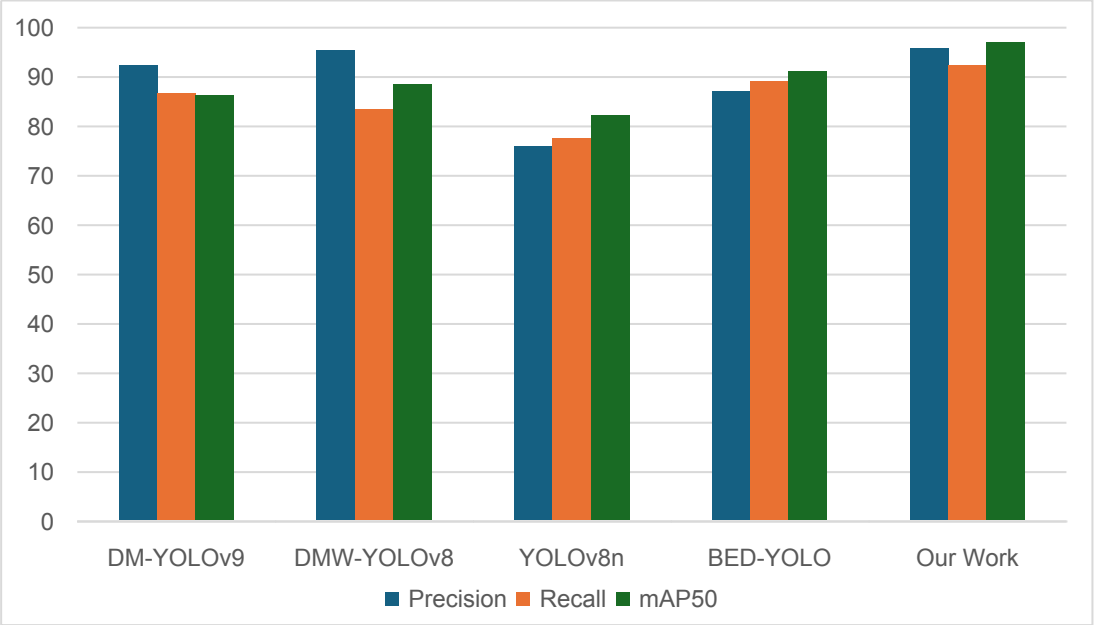


Figure 6.8 Comparative analysis with previous models based on precision, recall, and accuracy

6.4.2 Limitations of Our Work

Despite the significant advantages, several limitations need to be acknowledged to ensure in detail understanding of the system's current capabilities and future potential. The performance of the YOLOv9 model is highly dependent on the quantity, quality, and diversity of the training dataset. The model's accuracy might decrease when encountering less common or atypical disease symptoms not adequately represented in the training data. Early-stage diseases or those with subtle visual cues can be particularly challenging to detect. Variations in lighting conditions (e.g., direct sunlight, shadows, overcast days), background clutter (e.g., soil, other plants), and leaf occlusion can significantly impact detection accuracy in real-time

environment. The training data may not be sufficient to fully represent the complexity of these diverse environments. Disease prevalence and symptom manifestation can vary geographically due to different environmental factors and pathogen strains. A model trained on data from one region might not perform optimally in another. Deploying YOLOv9 in real-time on agricultural equipment (e.g., drones, robotic sprayers) requires significant computational power. Resource constraints on edge devices might lead to a balance among inference speed and detection accuracy.

The current spray prescription module might offer static recommendations based solely on the detected disease. It may not dynamically adapt to other crucial factors such as disease severity, the specific growth stage of the tomato plant, local weather conditions (e.g., recent rainfall, wind speed), resistance development in pathogen populations, or farmer preferences/regulations regarding organic vs. chemical treatments. The cost of implementing such an advanced system, including hardware (cameras, computing units for edge deployment, automated sprayers) and software development, might be prohibitive for small-scale farmers. Operating and maintaining such a system might require a certain level of technical expertise that may not be readily available to all farmers.

6.4.3 Future Directions

To highlight these limitations and further enhancements of the system's utility and robustness, future research and improvements should focus on collecting more comprehensive datasets that include a wider range of disease severities, environmental conditions (various lighting, angles, backgrounds), and co-occurring diseases. Optimizing YOLOv9 for deployment on resource-constrained edge devices to enable real-time processing directly on agricultural machinery. Integrating real-time environmental data (temperature, humidity, rainfall) and plant growth stage information into the spray prescription module for more dynamic and precise recommendations. Also, before the spray prescription module the severity of the disease should be find out and then the spray prescription module should automatically prescribe the spray and quantity of spray according to the disease severity. By addressing these points, the YOLOv9-based tomato disease detection and spray prescription system can evolve into an even more powerful and indispensable tool for sustainable and efficient tomato cultivation.

6.5 Summary

This chapter presents a highly effective and robust automated system for tomato disease management, addressing a critical need in modern agriculture. The integration of YOLOv9 for

disease detection with an automated spray prescription module proves a significant step forward in precision agriculture for tomato cultivation. The consistently high-performance metrics a precision of 95.8%, recall of 92.4%, and an impressive mAP50 of 97% unequivocally demonstrate the accuracy, reliability, and practical utility of the system, in a real-world agricultural setting. The capacity of this system to accurately detect and precisely localize diseases, coupled with its ability to automatically prescribe appropriate treatments, offers profound implications for sustainable and efficient farming practices. By minimizing false positives, the system ensures judicious use of pesticides, contributing to reduced chemical input, lower environmental impact, and decreased operational costs for farmers. Simultaneously, its high recall rate guarantees that very few genuine disease cases are missed, preventing widespread outbreaks and safeguarding crop yields. This holistic approach empowers farmers with a powerful tool to make timely and informed decisions, transitioning from reactive disease management to proactive and automated intervention.

This work not only validates the efficacy of advanced deep learning models like YOLOv9 in complex agricultural applications but also provides a tangible solution for enhancing food security and promoting sustainable agricultural practices. Future research could explore dynamic spray prescription modules that adapt to environmental conditions, disease severity, and economic factors, as well as the potential for integration with robotic platforms for fully autonomous disease management and targeted spraying.

Chapter 7: Overall Discussion and Integration of the Proposed Systems

In this chapter we integrate the entire work moving beyond the individual performance of each module to discuss their seamless integration. The main aim is to evaluate the complete system's performance as a holistic solution, and to discuss its practical implications, scalability, and the challenges of real-world deployment. By providing a comprehensive overview, this chapter aims to validate the research's contribution to advancing a new generation of intelligent agricultural systems. The remainder of this chapter is structured as follows: Section 7.1 discusses the introduction of the overall research work. Section 7.2 focuses on the integration of developed modules. Section 7.3 provides the performance comparison of the proposed models. Section 7.4 focuses on the practical implementation of this work in precision agriculture. Section 7.5 discusses the deployment challenges of the proposed system, Finally, Section 7.6 summarizes the chapter.

7.1 Introduction

The preceding chapters of this thesis have detailed the development of several key components for an intelligent agricultural system, each addressing a specific challenge in automated tomato disease management. Chapter 4 presented a novel approach for accurate leaf detection and segmentation from complex plant images. Chapter 5 introduced an enhanced deep learning architecture for precise disease diagnosis on these isolated leaves, and Chapter 6 demonstrated the development of a real-time object detection and automated spray prescription system using the YOLOv9 model. While each module is a significant contribution on its own, their true potential is unlocked when they are seamlessly integrated into a single, cohesive framework.

In this chapter we move beyond the individual performance of each module to discuss the overall integration, providing a comprehensive analysis of the complete system's capabilities. The central objective is to evaluate how these distinct but complementary components work together to form a robust, end-to-end solution for automated disease management in a real-world agricultural setting. This chapter also presents a detailed performance comparison between the different methods proposed, discuss the practical implications of the work in this thesis for smart agriculture, and address the inherent challenges of scalability and deployment. By providing a complete overview, this chapter solidifies the research's contribution to advancing precision agriculture and offers a clear path toward a more sustainable and efficient future for farming.

7.2 Integration of the Developed Modules

The integrated system is a multi-stage pipeline designed to process real-time video or image data and provide an automated response. The successful operation of this pipeline relies on the efficient communication and a logical sequence of operations between the developed modules. The architectural design of the integrated system follows a modular approach, allowing for flexibility and future upgrades. The data and control flow are managed by a CPU that combines the actions of the specialized deep learning modules, which are often accelerated by a dedicated GPU. The entire system operates in the following sequential manner:

1. **Image Acquisition and Initial Processing:** This process begins with real-time video streams or still images captured by cameras mounted on edge device like drone, or fixed point within the field. The raw RGB images are the input for the entire system.
2. **Leaf Segmentation and Extraction (Chapter 4):** The initial input is first fed into the leaf segmentation module. This component, which utilizes a modified YOLOv8 and the SAM, performs two critical functions:
    - **Object Detection:** It identifies and localizes the bounding box of the entire tomato plant and individual leaves within the complex, cluttered background.
    - **Instance Segmentation:** For each detected leaf, it generates a precise pixel-level mask, effectively isolating the leaf from all background elements. This step is crucial for addressing the "real-world generalization gap" discussed in earlier chapters, as it provides a clean, pre-processed image for the next stage. The segmented leaf images are then stored and passed on to the disease diagnosis module.
3. **Enhanced Disease Diagnosis (Chapter 5):** The isolated leaf images, now free from background noise, are fed into the enhanced deep learning architecture. This specialized CNN processes the clean leaf image and performs a highly accurate classification, identifying one of the 10 specific diseases or classifying it as healthy. The output of this module is a class label (e.g., "Early Blight") along with the confidence score.
4. **Real-time Detection and Automated Spray Prescription (Chapter 6):** In a parallel and a final decision-making process, the real-time YOLOv9 module processes the original raw images. While the multi-stage system offers high precision, the YOLOv9 model provides a rapid, end-to-end solution for immediate action. Based on its high-speed detection and classification, the system triggers the automated spray prescription module. This module provides set of bounding boxes with disease labels as an output, and most importantly, the specific command to prescribe an automated treatment on

the spray module. The system's logic can be configured to prioritize the YOLOv9 detection for speed in large-scale fields while using the multi-stage system for periodic high-precision checks or for more ambiguous cases.

The integration of these modules is not merely a sequential process but a synergistic one. The leaf segmentation module provides the foundational input for the high-precision classifier, while the real-time YOLOv9 system ensures that the overall framework is responsive enough for practical application in a dynamic field environment.

7.3 Performance Comparison Across the Proposed Methods

The development of two distinct disease detection methodologies: a high-precision, multi-stage pipeline and a fast, end-to-end YOLOv9 system, necessitates a comprehensive performance comparison. The choice of which system to deploy in a real-world scenario heavily depends on the specific balance between the efficiency, computational cost and accuracy. As shown in the table 7.1, the multi-stage system achieves a slightly higher overall accuracy, a result of its ability to eliminate background noise and focus solely on the intricate details of the leaf. This makes it ideal for applications where diagnostic precision is the top priority, such as research, small-scale farming, or for diagnosing diseases that require a more granular analysis. In contrast, the YOLOv9 system demonstrates a superior balance of accuracy and speed, which is considered very crucial for applications in real-time environment.

Table 7.1 Performance evaluation of proposed methods

Feature	Multi-Stage System (Ch. 4 & 5)	YOLOv9 System (Ch. 6)
Core Function	High-precision classification on isolated leaves	Real-time object detection on whole plants
Primary Goal	Maximize diagnostic accuracy	Maximize speed and efficiency
Accuracy	98.2% (Overall Accuracy)	97.0% (mAP)
Speed	Lower (due to two-step process)	Higher (single forward pass)
Computational Cost	Higher (requires more processing for segmentation and classification)	Lower (more streamlined architecture)
Robustness to Background Clutter	Highly robust (background is removed)	Robust, but can be affected by complex backgrounds
Best-fit Application	Small-scale, high-value crops; detailed plant health monitoring	Large-scale, real-time field operations; automated spray systems

Its single-pass detection mechanism allows it to process images and video frames in such way that is appropriate for controlling a spray module. While its overall accuracy is marginally lower than the multi-stage system, its performance on key metrics like mAP is high enough for reliable field use. The choice between these two methods is not a matter of one being superior,

but rather of choosing the right tool for the job. The key strength of this thesis comes from the development of both, providing a complete toolbox for different agricultural needs.

7.4 Practical Implications for Smart Agriculture

The integrated system represents a significant leap forward in precision agriculture, offering a holistic, automated solution to one of farming's most persistent challenges. The practical implications of this research will transform farming practices and yield direct benefits for the environment.

- **Early and Accurate Disease Diagnosis:** The high precision of the enhanced models ensures that diseases are identified accurately, often in their early stages, before they cause significant crop damage. This allows for proactive intervention rather than a reactive response.
- **Targeted Pesticide Application:** By precisely identifying and localizing diseased areas, the automated spray prescription system ensures that pesticides are applied only where they are needed. This contrasts sharply with the common practice of blanket spraying. The result is a substantial **reduction in pesticide use**, leading to significant cost savings for farmers and, more importantly, a reduced environmental impact on soil, water, and beneficial insects.
- **Increased Crop Yield and Quality:** Timely and accurate intervention prevents the spread of diseases, ensuring that crops remain healthy throughout the growing cycle. This directly contributes to a higher overall yield and better-quality produce, which can command a higher market price.
- **Data-Driven Decision Making:** Beyond automation, the system provides valuable data on disease prevalence, location, and spread patterns within a field. Farmers can utilize this data to make informed decisions about future crop rotations, fertilizer application, and disease-resistant plant varieties, moving farming towards a truly data-driven practice.

This research demonstrates that by taking the advantage of deep learning and automation, agriculture can become more productive, profitable, and environmentally sustainable, contributing to a more secure global food supply.

7.5 Scalability and Deployment Challenges

While the proposed system has demonstrated exceptional performance in a controlled research setting, its transition to widespread, real-world deployment presents several challenges related to scalability, cost, and environmental factors.

- **Scalability:** Deploying the system across thousands of acres requires robust infrastructure. The current setup, which relies on a powerful GPU, may be suitable for a small-scale farm or a single robotic platform. To scale, a distributed computing architecture or a cloud-based solution would be necessary. This would involve transmitting large amounts of data from field-based cameras to a central cloud server for processing or equipping a fleet of autonomous vehicles with on-board edge computing devices to process the data locally.
- **Hardware and Cost:** The initial investment in high-resolution cameras, GPS sensors, and specialized hardware for spray module can be substantial. While the cost is likely to decrease over time, it remains a barrier to adoption for smallholder farmers. Future work should focus on optimizing the models to run on more affordable, low-power edge devices, without a significant loss in performance.
- **Environmental Variability:** Diverse datasets were used for models training, but they may still face challenges in extreme real-world conditions. These include:
    - **Varying Lighting:** The system must be robust to changes in sunlight, shadows, and overcast conditions.
    - **Weather:** Wind can cause leaves to move rapidly, making precise detection difficult, and rain can affect image clarity.
    - **Obstacles:** The system must navigate around physical obstacles, and the models must be able to detect diseases on leaves that are partially obscured by other parts of the plant.
- **Maintenance and Data Updates:** The system's performance is tied to the quality of its models. New disease strains or changes in symptom appearance over time will require continuous data collection and periodic model retraining. Ensuring a seamless mechanism for model updates and maintenance will be critical for long-term reliability.

Addressing these challenges is the next frontier of this research. The foundation laid in this thesis provides a clear and effective starting point, but practical deployment will require interdisciplinary collaboration between deep learning experts, robotics engineers, and agricultural scientists.

7.6 Summary

This chapter has provided a final, complete discussion on the integrated systems developed throughout this thesis. It has demonstrated that the combination of a high-precision leaf segmentation module, an enhanced disease diagnosis model, and a real-time YOLOv9-based system creates a powerful, end-to-end solution for automated disease management in tomato

crops. The performance comparison highlighted the crucial trade-off between accuracy and speed, validating the development of two distinct methodologies for different application needs. Furthermore, the discussion on practical implications underscored the research's potential to revolutionize farming practices by enabling spray prescription system, reducing environmental impact, and increasing crop yield. While challenges related to scalability and real-world deployment remain, the integrated system presented in this thesis offers a robust and effective framework that paves the way for the future of smart, sustainable agriculture.

Chapter 8: Conclusion and Future Work

This final chapter synthesizes the findings of this thesis, providing a concise summary of the research contributions, acknowledging the inherent limitations of the study, and outlining a strategic roadmap for future research. This work has successfully demonstrated the development and integration of several advanced deep learning models to create an intelligent and automated system for tomato disease detection and management.

8.1 Summary of Contributions

This research makes several significant contributions to the fields of precision agriculture and deep learning-based computer vision. The major contribution of is the development of a novel, multi-component framework that provides a holistic and effective solution for automated tomato disease management. This work moves beyond a simple diagnostic tool to present a complete, end-to-end system capable of identifying diseases and prescribing a targeted, automated response.

The key contributions are as follows:

1. **Novel Multi-Stage Data Processing Pipeline:** The thesis introduced a unique two-stage pipeline for disease diagnosis. By first employing a segmentation model (YOLOv8 + SAM) to accurately isolate leaves from complex, cluttered backgrounds, the system effectively addresses the long-standing challenge of model generalization from clean, lab-based datasets to real-world field conditions.
2. **Enhanced Deep Learning Architecture:** A custom-built CNN was developed and validated, achieving a remarkable overall accuracy of 98.2% on the task of classifying 10 tomato disease classes along with healthy leaves. This enhanced model demonstrated superior performance and computational efficiency compared to widely used benchmark architectures like ResNet and DenseNet.
3. **Real-time Automated Spray Prescription System:** The research successfully integrated a high-performance YOLOv9 model with an automated prescription system. This practical application provides a functional, real-time solution for farmers, enabling immediate and targeted pesticide application, which is crucial for preventing disease spread and reducing environmental impact.
4. **Proof of a Holistic, Integrated System:** By demonstrating the seamless integration of leaf segmentation, precise diagnosis, and an automated response, this thesis offers a blueprint for the next generation of smart agricultural systems. It proves that combining

specialized models into a cohesive pipeline can yield superior results and provide a more robust solution than any single model alone.

8.2 Limitations

While the proposed system has demonstrated exceptional performance, it is crucial to highlight the limitations of the current research, which provide a clear direction for future work.

1. **Dataset Constraints and Generalization:** Despite extensive data augmentation, the models were trained on a specific set of diseases and environmental conditions. The system's performance may degrade when confronted with new or rare disease variants, different tomato cultivars, or previously unseen lighting and weather conditions.
2. **Environmental and Physical Challenges:** The current framework has not been extensively tested in a continuous, long-term field deployment scenario. Factors such as wind-induced leaf motion, rain, dust, and physical obstructions in the field may pose challenges to the camera system and subsequent image processing.
3. **Hardware and Cost:** The system, as proposed, requires dedicated and computationally powerful hardware (e.g., GPUs) for optimal performance. This could represent a significant initial investment, limiting its accessibility for small-scale and resource-constrained farmers.
4. **Single-Crop Specialization:** The models were meticulously developed and optimized for tomato plants. While the core methodology is transferable, adaptation to other crops would require substantial re-training and fine-tuning, which is beyond the scope of this work.

8.3 Future Research Directions

Based on the limitations and the promising results of this research, several exciting avenues for future work are identified to further advance the field of intelligent agriculture.

1. **Multi-Crop and Multi-Disease Generalization:** Future work should focus on developing a more generalized model capable of diagnosing diseases across multiple crop types. This would involve training on a significantly larger, more diverse dataset to create a more versatile tool for precision farming.
2. **Enhanced Sensor Integration:** The system's diagnostic capabilities could be augmented by integrating data from additional sensors. Incorporating hyperspectral imagery could allow for the detection of diseases in their pre-symptomatic stages, before any visible signs appear. Additionally, environmental sensors for temperature, humidity, and soil moisture could be used to predict disease outbreaks.

3. **Optimization for Edge Devices:** To address the hardware limitations, future research should explore techniques for deploying the models on more affordable, low-power edge computing devices. Methods such as model quantization, pruning, and knowledge distillation could be used to significantly reduce the model size and computational requirements while maintaining high accuracy.
4. **Autonomous Robotics and Field Deployment:** The final and most critical step is the full integration of the system with an autonomous robotic platform. This would involve developing advanced navigation algorithms for field traversal, as well as a robust hardware and software architecture for real-time data processing and automated spraying without human intervention.
5. **Long-Term Field Validation:** Conducting a long-term, multi-season field trial is essential to validate the system's reliability, durability, and true impact on crop yield and pesticide consumption under real-world, dynamic conditions.

References

[1] FAO, "FAOSTAT statistical database," Food and Agriculture Organization of the United Nations, 2024. [Online]. Available: http://www.fao.org/faostat/

[2] S. Savary, L. Willocquet, S. J. Pethybridge, P. Esker, and N. McRoberts, "Global burden of crop losses caused by diseases, pests and weeds: 2020-2023 update," Nat. Food, vol. 5, no. 1, pp. 109-118, 2024.

[3] T. S. Thind, "New insights into fungicide resistance: a growing challenge in crop protection," Indian Phytopathol., vol. 75, pp. 927–939, 2022.

[4] J. Schieffer and C. Dillon, "The economic and environmental impacts of precision agriculture and interactions with agro-environmental policy," Precision Agric., vol. 16, pp. 46–61, 2015.

[5] FAO, "Pesticide use in global agriculture: Current status and future prospects," Food and Agriculture Organization of the United Nations, 2023. [Online]. Available: http://www.fao.org/publications/

[6] I. Karpinski, S. Nordheim, and B. Golla, "Economic and environmental web services for the advancement of precision spraying," in Precision Agriculture '25, Wageningen, Netherlands: Wageningen Academic, 2025, pp. 247–253.

[7] J. Singh, R. Kashyap, K. Bansal, et al., "Transition from conventional to AI-based methods for detection of foliar disease symptoms in vegetable crops: a comprehensive review," J. Plant Pathol., vol. 107, pp. 1791–1814, 2025.

[8] J. G. A. Barbedo, "Plant disease identification from individual lesions and spots using deep learning," Biosyst. Eng., vol. 216, pp. 95-107, 2023.

[9] A. Bhargava, S. Aasheesh, G. O. Prakash, H. A. Mohammed, U. Peerapong, and U. Monthippa, "Plant leaf disease detection, classification, and diagnosis using computer vision and artificial intelligence: A review," IEEE Access, vol. 12, pp. 37443–37469, 2024.

[10] S. Wang et al., "Advances in Deep Learning Applications for Plant Disease and Pest Detection: A Review," Remote Sens., vol. 17, p. 698, 2025.

[11] J. Boulent, S. Foucher, J. Théau, and P. L. St-Charles, "Convolutional neural networks for the automatic identification of plant diseases: A systematic review and meta-analysis," Front. Plant Sci., vol. 15, Art. no. 1242761, 2024.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[13] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Proc. Adv. Neural Inf. Process. Syst., 2012, vol. 25.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779-788.

[16] W. Liu et al., "Ssd: Single shot multibox detector," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2016, pp. 21-37.

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent. (MICCAI), 2015, pp. 234-241.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 2961-2969.

[19] M. Everingham et al., "The pascal visual object classes challenge: A retrospective," Int. J. Comput. Vis., vol. 111, no. 1, pp. 98-136, 2015.

[20] J. Qin and H. Zhang, "Detection of cotton root rot in the field using hyperspectral imaging," Int. J. Remote Sens., vol. 26, no. 23, pp. 5121-5134, 2005.

[21] A. Camargo and J. S. Smith, "An image-processing based algorithm to automatically identify plant disease visual symptoms," Comput. Electron. Agric., vol. 66, no. 2, pp. 164-171, 2009.

[22] V. Singh and A. K. Misra, "Detection of plant leaf diseases using image segmentation and machine learning techniques," Expert Syst. Appl., vol. 68, pp. 93-102, 2017.

[23] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Front. Plant Sci., vol. 7, Art. no. 1419, 2016.

[24] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo algorithm developments," Procedia Comput. Sci., vol. 199, pp. 1066-1073, 2022.

[25] O. Saxena, S. Agrawal, and S. Silakari, "Disease detection in plant leaves using deep learning models: AlexNet and GoogLeNet," in Proc. 2021 IEEE Int. Conf. Technol. Res. Innov. Betterment Soc. (TRIBES), 2021, pp. 1-6.

[26] K. Punitha and G. Shrivastava, "Analytical study of deep learning architectures for classification of plant diseases," J. Algebr. Stat., vol. 13, pp. 907–914, 2022.

[27] S. Amarjeeth, H. Jaisree, D. Aishwarya, and J. S. Jayasree, "Plant disease detection and diagnosis using deep learning," in Proc. Int. Conf. Adv. Technol. (ICONAT), 2022, pp. 1–6.

[28] K. Vishakha and M. Munot, "Deep learning models for tomato plant disease detection," in Advanced Machine Intelligence and Signal Processing, Singapore: Springer, 2022, pp. 679–686.

[29] K. Khan et al., "End-to-End Semantic Leaf Segmentation Framework for Plants Disease Classification," Complexity, vol. 2022, Art. no. 1168700, 2022.

[30] S. S. Wadadare and H. S. Fadewar, "Deep learning convolution neural network for tomato leaves disease detection by inception," in Applied Computational Technologies, Singapore: Springer, 2022, pp. 208–220.

[31] H. C. Chen et al., "AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf," Electronics, vol. 11, no. 6, p. 951, 2022.

[32] M. A. Khan et al., "Cucumber leaf diseases recognition using multi-level deep entropy-ELM feature selection," Appl. Sci., vol. 12, p. 593, 2022.

[33] X. Yiting, D. Plett, and H. Liu, "Detecting crown rot disease in wheat in controlled environment conditions using digital color imaging and machine learning," AgriEngineering, vol. 4, pp. 141–155, 2022.

[34] P. Kaur et al., "Recognition of leaf disease using hybrid convolutional neural network by applying feature reduction," Sensors, vol. 22, p. 575, 2022.

[35] A. Almadhor et al., "AI-driven framework for recognition of guava plant diseases through machine learning from DSLR camera sensor based high resolution imagery," Sensors, vol. 21, p. 3830, 2021.

[36] M. V. Applalanaidu and G. Kumaravelan, "A review of machine learning approaches in plant leaf disease detection and classification," in Proc. 3rd Int. Conf. Intell. Commun. Technol. Virtual Mobile Netw. (ICICV), 2021, pp. 716–724.

[37] S. B. P. Satheesh, P. R. S. Naidu, and U. Neelima, "Prediction of guava plant diseases using deep learning," in Proc. 3rd Int. Conf. Commun. Cyber Phys. Eng. (ICCCE), 2021, pp. 1495–1505.

[38] T. Murat and R. S. Arslan, "RT-Droid: A novel approach for real-time android application analysis with transfer learning-based CNN models," J. Real-Time Image Process., vol. 20, p. 55, 2023.

[39] W. Liu et al., "A Lightweight Real-Time Recognition Algorithm for Tomato Leaf Disease Based on Improved YOLOv8," Agronomy, vol. 14, no. 9, p. 2069, 2024.

[40] A. Sharma, V. Kumar, and L. Longchamps, "Comparative Performance of Yolov8, Yolov9, Yolov10 and Faster R-Cnn Models for Detection of Multiple Weed Species," Preprint, 2024.

[41] N. Razzaq et al., "TOMATO LEAF DISEASE DETECTION USING YOLOV9 AND COMPUTER VISION," Spectr. Eng. Sci., vol. 3, no. 4, pp. 626-638, 2025.

[42] L. Guo et al., "Optimizing yolo algorithm for efficient object detection in resource-constrained environments," in Proc. IEEE 4th Int. Conf. Electron. Technol. Commun. Inf. (ICETCI), 2024.

[43] J. Chen, J. Chen, D. Zhang, Y. Sun, and Y. A. Nanehkaran, "Using deep transfer learning for image-based plant disease identification," Comput. Electron. Agric., vol. 173, Art. no. 105393, 2020.

[44] A. Bellout et al., "Advanced YOLO models for real-time detection of tomato leaf diseases," Math. Model. Comput., vol. 11, no. 4, pp. 1198-1210, 2024.

[45] H. M. Zayani et al., "Deep learning for tomato disease detection with YOLOv8," Eng. Technol. Appl. Sci. Res., vol. 14, no. 2, pp. 13584-13591, 2024.

[46] S. Xu et al., "Automatic pine wilt disease detection based on improved YOLOv8 UAV multispectral imagery," Ecol. Inform., vol. 84, Art. no. 102846, 2024.

[47] F. Solimani et al., "Optimizing tomato plant phenotyping detection: Boosting YOLOv8 architecture to tackle data complexity," Comput. Electron. Agric., vol. 218, Art. no. 108728, 2024.

[48] M. Alruwaili et al., "RTF-RCNN: An architecture for real-time tomato plant leaf diseases detection in video streaming using Faster-RCNN," Bioengineering, vol. 9, no. 10, p. 565, 2022.

[49] R. Rashid, W. Aslam, R. Aziz, and G. Aldehim, "A Modified MobileNetv3 Coupled With Inverted Residual and Channel Attention Mechanisms for Detection of Tomato Leaf Diseases," IEEE Access, vol. 13, pp. 52683–52696, 2025.

[50] X. Yue et al., "Improved YOLOv8-Seg Network for Instance Segmentation of Healthy and Diseased Tomato Plants in the Growth Stage," Agriculture, vol. 13, p. 1643, 2023.

[51] R. Zhang, Y. Ge, Q. Li, and L. Meng, "An Improved YOLOv8 Tomato Leaf Disease Detector Based on the Efficient-Net backbone," in Proc. 6th Int. Symp. Adv. Technol. Appl. IoT, 2024, p. 3748.

[52] P. Kaur et al., "Performance analysis of segmentation models to detect leaf diseases in tomato plant," Multimed. Tools Appl., vol. 83, pp. 16019–16043, 2024.

[53] X. Liu et al., "Tomato leaf disease detection based on improved YOLOv8," in Proc. IEEE 6th Int. Conf. IoT Autom. Artif. Intell. (IoTAAI), 2024, pp. 145–150.

[54] S. G. Brucal et al., "Development of tomato leaf disease detection using YoloV8 model via RoboFlow 2.0," in Proc. IEEE 12th Global Conf. Consum. Electron. (GCCE), 2023, pp. 692–694.

[55] M. Shoaib et al., "Deep learning-based segmentation and classification of leaf images for detection of tomato plant disease," Front. Plant Sci., vol. 13, Art. no. 1031748, 2022.

[56] D. Williams, F. Macfarlane, and A. Britten, "Leaf only SAM: A segment anything pipeline for zero-shot automated leaf segmentation," Smart Agric. Technol., vol. 8, Art. no. 100515, 2024.

[57] S. U. Islam et al., "Enhanced Deep Learning Architecture for Rapid and Accurate Tomato Plant Disease Diagnosis," AgriEngineering, vol. 6, pp. 375–395, 2024.

[58] A. Chouchane, A. Ouamanea, Y. Himeur, and A. Amira, "Deep learning-based leaf image analysis for tomato plant disease detection and classification," in Proc. IEEE Int. Conf. Image Process. (ICIP), 2024, pp. 2923–2929.

[59] Z. Qi et al., "A novel method for tomato stem diameter measurement based on improved YOLOv8-seg and RGB-D data," Comput. Electron. Agric., vol. 226, Art. no. 109387, 2024.

[60] J. Wu et al., "DS-DETR: A Model for Tomato Leaf Disease Segmentation and Damage Evaluation," Agronomy, vol. 12, p. 2023, 2022.

[61] P. K. Pativada, "Real-Time Detection and Classification of Plant Seeds Using YOLOv8 Object Detection Model," Ph.D. dissertation, Kansas State Univ., Manhattan, KS, USA, 2024.

[62] P. Wang et al., "Leaf Segmentation Using Modified YOLOv8-Seg Models," Life, vol. 14, p. 780, 2024.

[63] S. Bondre and D. Patil, "Crop disease identification segmentation algorithm based on Mask RCNN," Agron. J., vol. 116, pp. 1088–1098, 2024.

[64] R. Khanam and M. Hussain, "What is YOLOv5: A deep look into the internal features of the popular object detector," arXiv preprint arXiv:2407.20892, 2024.

[65] C. Y. Wang et al., "CSPNet: A new backbone that can enhance learning capability of CNN," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops, 2020, pp. 390-391.

[66] P. Zhao, "SPFFNet: Strip perception and feature fusion spatial pyramid pooling for fabric defect detection," arXiv preprint arXiv:2502.01445, 2025.

[67] T. Y. Lin et al., "Feature pyramid networks for object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 2117-2125.

[68] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," arXiv preprint arXiv:2207.02696, 2022.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904-1916, 2015.

[70] P. Hidayatullah et al., "YOLOv8 to YOLO11: A Comprehensive Architecture In-depth Comparative Review," arXiv preprint arXiv:2501.13400, 2025.

[71] V. R. R. Ch et al., "A Review on YOLOv8 and Its Advancements," ResearchGate, 2024.

[72] M. Alqarqaz et al., "SOD-YOLOv8—Enhancing YOLOv8 for Small Object Detection in Aerial Imagery and Traffic Scenes," Sensors, vol. 24, no. 19, p. 6209, 2024.

[73] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," arXiv preprint arXiv:2402.13616, 2024.

[74] Y. Zhou, "Efficacy and Efficiency in Object Detection: Exploring YOLOv9 on Limited Resources," Preprint, 2024.

[75] K. Kalinaki, W. Shafik, T. J. Gutu, and O. A. Malik, "Computer vision and machine learning for smart farming and agriculture practices," in Artificial Intelligence Tools and Technologies for Smart Farming, Hershey, PA: IGI Global, 2023, pp. 79–100.

[76] Y. Wang et al., "A Method for Tomato Plant Stem and Leaf Segmentation and Phenotypic Extraction Based on Skeleton Extraction and Supervoxel Clustering," Agronomy, vol. 14, p. 198, 2024.

[77] T. Yang et al., "An Approach for Plant Leaf Image Segmentation Based on YOLOV8 and the Improved DEEPLABV3+," Plants, vol. 12, p. 3438, 2023.

[78] M. S. Alzahrani and F. W. Alsaade, "Transform and Deep Learning Algorithms for the Early Detection and Recognition of Tomato Leaf Disease," Agronomy, vol. 13, p. 1184, 2023.

[79] L. Kouadio et al., "A Review on UAV-Based Applications for Plant Disease Detection and Monitoring," Remote Sens., vol. 15, p. 4273, 2023.

[80] X. Han, J. Chang, and K. J. Wang, "You only look once: Unified, real-time object detection," Procedia Comput. Sci., vol. 1, pp. 61–72, 2021.

[81] A. M. Vilar et al., "Enhancing Precision Agriculture Pest Control: A generalized Deep Learning Approach with YOLOv8-based Insect Detection," IEEE Access, vol. 12, p. 84420, 2024.

[82] P. Singh and R. Krishnamurthi, "IoT-based real-time object detection system for crop protection and agriculture field security," J. Real-Time Image Process., vol. 21, p. 106, 2024.

[83] W. Zhang et al., "Adapting the segment anything model for plant recognition and automated phenotypic parameter measurement," Horticulturae, vol. 10, no. 4, p. 398, 2024.

[84] A. Kirillov et al., "Segment anything," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2023, pp. 4015–4026.

[85] B. Song et al., "A multispectral remote sensing crop segmentation method based on Segment Anything Model using Multi-stage Adaptation Fine-tuning," IEEE Trans. Geosci. Remote Sens., vol. 62, Art. no. 4408818, 2024.

[86] K. Itakura and F. Hosoi, "Automatic Leaf Segmentation for Estimating Leaf Area and Leaf Inclination Angle in 3D Plant Images," Sensors, vol. 18, p. 3576, 2018.

[87] M. Gehlot, R. K. Saxena, and G. C. Gandhi, "“Tomato-Village”: a dataset for end-to-end tomato disease detection in a real-world environment," Multimed. Syst., vol. 29, no. 6, pp. 3305-28, 2023.

[88] W. Saenprasert et al., "Yolo for small objects in aerial imagery: A performance evaluation," in Proc. 21st Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE), 2024.

[89] C. Li et al., "YOLOv6: A single-stage object detection framework for industrial applications," arXiv preprint arXiv:2209.02976, 2022.

[90] L. Cao, X. Zheng, and L. Fang, "The semantic segmentation of standing tree images based on the Yolo V7 deep learning algorithm," Electronics, vol. 12, no. 4, p. 929, 2023.

[91] A. Khatua et al., "Developing approaches in building classification and extraction with synergy of YOLOV8 and SAM models," Spatial Inf. Res., vol. 32, no. 5, pp. 511-30, 2024.

[92] T. Chen, M. Rizwan, and A. Abbas, "Exploring the role of agricultural services in production efficiency in Chinese agriculture," Land, vol. 11, no. 3, p. 347, 2022.

[93] I. Buja et al., "Advances in Plant Disease Detection and Monitoring: From Traditional Assays to In-Field Diagnostics," Sensors, vol. 21, p. 2129, 2021.

[94] K. Punitha and G. Shrivastava, "Review on Emerging Trends in Detection of Plant Diseases using Image Processing with Machine Learning," Int. J. Comput. Appl., vol. 174, no. 11, pp. 39-48, 2021.

[95] S. Kaur, S. Pandey, and S. Goel, "Plants disease identification and classification through leaf images: A survey," Arch. Comput. Methods Eng., vol. 26, no. 2, pp. 507-530, 2019.

[96] L. T. Ramos and A. D. Sappa, "A comprehensive analysis of YOLO architectures for tomato leaf disease identification," Sci. Rep., vol. 15, no. 1, p. 26890, 2025.

[97] M. Yaseen, "What is YOLOv9: An in-depth exploration of the internal features of the next-generation object detector," arXiv preprint arXiv:2409.07813, 2024.

[98] A. Abulizi, J. Ye, H. Abudukelimu, and W. Guo, "DM-YOLO: Improved YOLOv9 model for tomato leaf disease detection," Front. Plant Sci., vol. 15, Art. no. 1473928, 2025.

[99] Y. Wang and J. He, "Improved Algorithm for Tomato Disease Detection Based on YOLOv8," in Proc. IEEE 4th Int. Conf. Electron. Technol. Commun. Inf. (ICETCI), 2024, pp. 500–505.

[100] A. K, L. Shine, and D. S, "Pest Detection and Disease Categorization in Tomato Crops using YOLOv8," in Proc. IEEE Recent Adv. Intell. Comput. Syst. (RAICS), 2024, pp. 1–6.

[101] Q. Wang, N. Yan, Y. Qin, X. Zhang, and X. Li, "BED-YOLO: An Enhanced YOLOv10n-Based Tomato Leaf Disease Detection Algorithm," Sensors, vol. 25, p. 2882, 2025.

List of Publications

International Journals

1. Islam, Shahab Ul, et al. "Enhanced deep learning architecture for rapid and accurate tomato plant disease diagnosis." AgriEngineering 6.1 (2024): 375-395.

2. Islam, Shahab Ul, et al. "Performance analysis of YOLOv3, YOLOv4 and MobileNet SSD for real time object detection." The Sciencetech 5.2 (2024): 37-49.

3. Islam, Shahab Ul, Giampaolo Ferraioli, and Vito Pascazio. "Tomato Leaf Detection, Segmentation, and Extraction in Real-Time Environment for Accurate Disease Detection." *AgriEngineering* 7.4 (2025): 120.

4. Islam, Shahab Ul, Giampaolo Ferraioli, Ghassan Husnain, and Vito Pascazio. "Intelligent Tomato Leaf Disease Detection and Automated Spray Prescription using YOLOv9: A Smart Agriculture Approach". Accepted In: Automation (2026).

5. Islam, Shahab Ul, Giampaolo Ferraioli, Vito Pascazio. "YOLO's Evolution in Agriculture: The State of the Art in Real-Time Plant Disease Detection with YOLO Models". Under Review. (2026).

International Conferences

1. Islam, S. U., Schirinzi, G., & Maqsood, S. (2024, July). A Light Weight CNN Based Architecture for the Detection of Early and Late Blight Disease in Tomato Plants in Real-Time Environment. In IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium (pp. 2819-2822). IEEE.