A Comprehensive Survey on Chilli Leaf Disease Detection Techniques Using YOLO

By
Tanvir Anzum
ID: 203-15-14537

FINAL YEAR DESIGN PROJECT REPORT

This Report Presented in Partial Fulfillment of the Requirements for the **Degree of Bachelor of Science in Computer Science and Engineering**

Supervised by

Dr. Naznin Sultana
Associate Professor
Department of Computer Science and EngineeringDaffodil International University

Co-Supervised by

Ms. Tapasy Rabeya
Senior Lecturer
Department of Computer Science and EngineeringDaffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY
Dhaka, Bangladesh

May 14, 2025

APPROVAL

This Project titled **" A Comprehensive Survey on Chilli Leaf Disease Detection Techniques Using YOLO"** submitted by Name, ID No:203-15-14537 **Student ID** to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on **14 May, 2025.**

BOARD OF EXAMINERS

Dr. Arif Mahmud(AM)
Associate Professor and Associate Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Chairman

Md Sadikur Rahaman
Assistand professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner

Tapasy Rabeya
Sr.lecturer
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner

Dr. Md. Manowarul islam
Associate Professor

Department of Computer Science and Engineering
Jagannath University

External Examiner

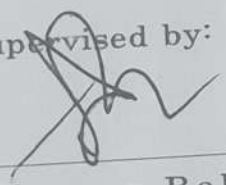
DECLARATION

We hereby declare that this project has been done by us under the supervision of **Dr. Naznin Sultana, Associate Professor**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this projectnor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

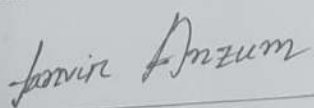
Dr. Naznin Sultana
Associate Professor
Department of Computer Science and EngineeringDaffodil International University

Co-Supervised by:



Ms. Tapasy Rabeya
Senior Lecturer
Department of Computer Science and EngineeringDaffodil International University

Submitted by:



Tanvir Anzum
Student ID: 203-15-14537
Department of Computer Science and EngineeringDaffodil International University

ACKNOWLEDGEMENTS

This work would not have been possible without the support and contributions of many individuals over the past two semesters. We are deeply grateful to everyone who has assisted us in one way or another.

First, we express our heartfelt thanks and gratefulness to the almighty for His divine blessing making it possible for us to complete the **Final Year Design Project (FYDP)** successfully.

We are grateful and wish our profound indebtedness to **Dr. Naznin Sultana**, **Associate Professor**, Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh. Deep knowledge and keen interest of our supervisor in the field of **Deep Learning** to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting themat all stages have made it possible to complete this project.

We would like to express our heartfelt gratitude to the Head of the Department of Computer Science and Engineering, for his kind help in finishing our project and also to other faculty members and the staff of the Department of Computer Science and Engineering, Daffodil International University.

We would like to thank our entire course-mates at Daffodil International University, who took part in this discussion while completing the coursework.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

ABSTRACT

Chilli (Capsicum annuum) is an economically vital crop extensively cultivated in countries like Bangladesh, India, and Indonesia. However, the crop is highly susceptible to fungal, bacterial, and viral diseases that significantly affect yield and farmer income. Traditional disease detection methods are manual, slow, and error-prone, particularly in rural areas where expert agronomists are scarce. To address this issue, this research proposes a real-time chilli leaf disease detection system based on advanced object detection models from the YOLO (You Only Look Once) family. Three models—YOLOv8s, YOLOv9s, and YOLOv10s—were trained and evaluated on a custom-annotated chilli leaf dataset. The YOLOv10 (small) model achieved the highest overall performance, with a mAP50 of 96.9%, mAP50–95 of 91.2%, and the fastest inference speed of 7.3 milliseconds. Compared to YOLOv8s and YOLOv9s, YOLOv10s demonstrated superior accuracy and lower computational load (24.5 GFLOPs), making it the most suitable model for mobile deployment. The best-performing model was optimized using TensorFlow Lite and integrated into a Flutter-based Android mobile application, enabling offline disease detection directly from smartphones. This approach empowers farmers with immediate, on-field diagnostic capabilities without relying on network connectivity or high-end hardware. Additionally, it promotes sustainable agriculture by supporting targeted pesticide use, thereby reducing environmental impact. Through careful model selection, optimization, and mobile app development, the study successfully bridges the gap between research and real-world application, providing a scalable, efficient, and farmer-friendly solution for chilli disease management. Future improvements could focus on multi-crop support, explainable AI integration, and multilingual user interface enhancements to broaden accessibility and impact.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 1 introduces the background and motivation for developing an automated chilli leaf disease detection system using YOLO-based models. It highlights the economic importance of chilli cultivation and the challenges farmers face in manually identifying plant diseases. The chapter outlines the research objectives, which include implementing multiple YOLO models (YOLOv8s, YOLOv9s, and YOLOv10s), optimizing the best-performing model for mobile deployment using TensorFlow Lite, and developing a Flutter-based mobile application for real-time detection. It also briefly describes the overall methodology, expected project outcomes, and the organization of the report. The chapter sets the foundation for the study by establishing the need for a scalable, accurate, and user-friendly mobile solution to support farmers in disease management and sustainable agriculture.

1.1 Introduction

It is a chilli (Capsicum annuum), as a highly valuable money crop, it is grown in many parts of the world, especially in both developing countries, such as India, Indonesia and Bangladesh, [1-2]. It is of high economic value, as a result of high demand by both the pharmaceutical and food industries. Chilli crops, on the other hand, are highly vulnerable to a number of fungal, bacterial and viral diseases which cause significant yield losses and economic loss to farmers [2-4]. It is inexpensive but labor consuming and error prone (manual observation) [2]. In addition, lack of expert agronomists in rural areas makes the problem even worse. The presence of artificial intelligence (AI) and the deep learning has facilitated the emergence of automated disease detection systems leading to improved efficiency and accuracy of plant disease management [3]. Of these, real-time object detection models, such as You Only Look Once (YOLO) – has become the more promising approach given its fast inference time and high robustness in detecting multiple illnesses at once [1].

YOLO is a cutting-edge deep-learning based detection model which has found its own place due to its real-time detection abilities as well as high accuracy [6-8]. Unlike standard (separate object localization and classification) convolutional neural

network (CNN) models, YOLO takes the entire image in one pass, thus being much faster and more efficient [7-8]. Different versions including YOLOv4, YOLOv5, YOLOv6, and YOLOv7 have also been incorporated on agricultural application, specifically in detecting plant diseases with exceptional accuracy levels [2].

The task of integrating YOLO in mobile applications offers a convenient and scalable solution for the farmers. By integrating YOLO-based disease detection models into smartphone applications, farmers can capture imagery of chilli leaves through their mobile devices, which are subsequently configured to detect diseases [2], [6], [8]. The application is capable of classifying various types of diseases, offer real time recommendations, and suggest mitigation strategies, which will empower the farmers with essential information to save, for appropriate management of the diseases. A solution such as this AI-driven solution, removes the need to rely on expert agronomists and allows for precision agriculture, in the long run providing an increased crop yield and quality [3].

The challenges associated with real-time implementation and mobile-based solutions stand out among others despite progress in AI-driven plant disease detection.While YOLO-based models have demonstrated high accuracy in controlled environments, their deployment in real-world agricultural settings, particularly through mobile applications, remains underexplored. The transition from research models to practical, real-time solutions is still in its early stages [4].

- Most studies focus on standalone deep learning models without considering how these models can be efficiently integrated into lightweight, mobile-friendly applications. The computational constraints of mobile devices require optimized models that maintain high accuracy without compromising speed or usability [5].
- Existing chilli disease detection systems often fail to address scalability challenges, making them difficult to deploy across diverse farming communities. A robust mobile application should be capable of processing large volumes of images while maintaining real-time inference capabilities for widespread farmer adoption [6].

To address these gaps, this research makes the following key contributions:

- This study implements and evaluate multiple YOLO versions (YOLOv8s, YOLOv9s, and YOLOv10s) for chilli leaf disease detection, optimizing

hyperparameters to enhance performance for real-time applications.

- The best-performing YOLO model is integrated into a Flutter-based mobile application, providing a user-friendly interface that allows farmers to capture and analyze chilli leaf images in real-time.
- The YOLO model is converted into TensorFlow Lite (TFLite) format to ensure smooth execution on mobile devices without requiring extensive computational resources, making it accessible for farmers in remote areas.

This research aims to bridge the identified gaps and contribute to the advancement of AI-driven precision agriculture by offering an efficient, real-time, and scalable chilli leaf disease detection framework. The proposed mobile-based solution facilitates real-time detection, empowering farmers with actionable insights to mitigate disease impact and enhance crop productivity.

1.2 Motivation

Chilli (Capsicum annuum) is one of the most important cash crops of many underdeveloped countries like Bangladesh, India and Indonesia. It is essential to the farming industry, because of a high demand for it in the food and pharmaceutical industry sector. Nevertheless, chilli plants are highly susceptible to various leaf diseases borne by fungi, bacteria and virus. These diseases do not show their presence in the early stages, which therefore results in colossal destruction of crops, low yield and heavy financial loss to the farmers. Manual inspection by the farmers/ agricultural officers remains the still the prime mode of disease detection in most of the rural areas. This traditional approach is labor intensive, expertise intensive, and there may not be availability in remote areas. Additionally, the early symptoms of various diseases usually resemble each other, which makes manual diagnosis tricky and imprecise. Farmers without access to an expert's assistance frequently do not detect diseases in time thereby experiencing massive crop failure.

With recent improvement in Artificial Intelligence (AI) and deep learning, it is now possible to solve this problem using automated disease detection. Of all the various AI techniques, models of object detection such as YOLO model have proved to be promising in classification of diseases of plants directly from images of leaves. YOLO models are efficient, precise and can recognize multiple disease

types simultaneously, making it highly viable for live agricultural purposes.

This research is motivated by the desire to have a reliable quick and mobile based system for detection of chilli leaf disease that farmers can use easily. The priority of this work tries to find out how various YOLO versions, such as YOLOv8s, YOLOv9s, and YOLOv10s, perform in the detection of chilli leaf diseases. The objective is to find the most efficient model and turn it into a lightweight form using TensorFlow Lite (TFLite) to be used in smart phones. This model is then put into a mobile application that has a Flutter basis, so that farmers only have to take a picture of the infected leaf and get instant results.

The motivation also arises from the existing gap in practical, real time AI tools for farmers. Although numerous research studies have applied YOLO models for plant disease detection in a controlled lab setup, relatively few have moved these models over to actual mobile applications that can be used in the field. By focusing on real-time detection and mobile deployment, this research seeks to help farmers to manage the health of their crops successfully even in remote or resource-limited regions.

1.3 Objectives

The growing vulnerability of chilli crops to a variety of leaf diseases is one of the serious challenges facing the farmers in particular, in the rural areas where there is usually lack of expert advice. These manual inspection methods are slow and error-prone thus treatment is delayed and crop productivity reduced. With the increasing potential in the field of agriculture, this study seek to build an intelligent, real-time, mobile based chilli leaf diseases detection system using the YOLO family of object detection models.

This research focuses on building a reliable and efficient framework that can be deployed on smartphones, allowing farmers to identify leaf diseases on the spot and take timely actions. The major objectives of the study are as follows:

- To review and implement multiple YOLO models — including YOLOv8s, YOLOv9s, and YOLOv10s — for detecting chilli leaf diseases using a custom-annotated dataset.
- To compare the performance of different YOLO models based on key evaluation metrics such as precision, recall, F1-score, inference speed, and

model size, in order to identify the most suitable model for real-time applications.

- To optimize the best-performing YOLO model for mobile devices by converting it into TensorFlow Lite (TFLite) format, ensuring smooth and efficient operation even on low-end smartphones.
- To develop a Flutter-based mobile application that integrates the TFLite model, enabling farmers to upload or capture chilli leaf images and receive instant disease detection results.
- To bridge the gap between research and real-world implementation by delivering a practical, scalable, and user-friendly mobile solution that supports early disease diagnosis and promotes precision agriculture.

1.4 Methodology

The suggested study adopts a systematic approach to the development of a real time mobile based chilli leaf disease detection system based upon YOLO (You Only Look Once) object detection models. The methodology starts with the acquisition of a chilli leaf image dataset which contain both healthy and diseased samples. These images are attentively tagged using bounding boxes and class labels to provide them for model training. Annotation is a fundamental step in the pipeline because it will help the deep learning models to learn and detect specific diseases patterns correctly.

The set prepared is then used to train a number of different YOLO models including YOLOv8s, YOLOv9s, and YOLOv10s, after annotation. Such models are chosen because of their increased detection capabilities and applicability to real time applications. This way, each model is trained one by one from the given dataset and performance of each model was compared in terms of accuracy precision, recall, inference speed and model size. The idea is to determine the version which provides the optimal level of computational efficiency combined with detection performance.

After choosing the most successful YOLO model, it is changed into TensorFlow Lite (TFLiter) type. This conversion process makes the model lightweight and optimized to work with mobile devices and therefore can be deployed in real world scenarios where access to internet connectivity and hardware resources are limited. The mobile application to get results for the given inputs is

implemented on a Flutter-based mobile application where TFLite model named best_model_float32.tflite is used. Flutter is selected as a development platform, because it is cross-platform, and it allows for a smooth user interface to provide real-time interactions.

The final use enables the users in particular, farmers to take images of the chilli leaves through their smartphones. The app uses the embedded YOLO-TFLite model to process images and shows the result of the disease detection appearance in the screen almost instantly. This is an all in one pipeline through the data set collection to the mobile deployment and therefore the system is not only accurate but practical for use in a real world agricultural scenario as well. The methodology is able to support precision farming and help in timely remedial action to minimize loss of crop by real time detection of multiple chilli leaf diseases.

1.5 Project Outcome

This study shows practical applicability of deep learning based object detection for detection of chilli leaf disease based on YOLO framework. Through a series of step-by-step process from dataset preparation to mobile deployment, the project ends up with a number of beneficial results which make progresses in terms of smart farming and real time disease diagnosis possible. The main outcomes of the present project are presented in the next bullet points.

- Development of custom chilli leaf disease dataset: A full dataset was compiled that included annotated pictures of healthy and diseased chilli leaves to guarantee precise training and testing of object-detection models.
- Implementation and comparison of YOLO models: Three state of the art YOLO namely YOLOv8s, YOLOv9s and YOLOv10s were implemented and trained based on the custom dataset. Their performance was compared on the basis of several metrics of evaluation for determining the best model for mobile deployment.
- Optimization of the optimal YOLO model for mobile usage: The highest performing YOLO model was converted to a TensorFlow Lite (TFLite) format to shrink its size and to ensure it can be run on mobile devices. This model was made optimal and could still have high detection accuracy, without the

slow inference in smartphones.

- Deployment via a live mobile application: A mobile application that is easy to use was made using Flutter. The application enables the farmers to take the leaf images of chilli, process the images on the go using the in-built TFLite model, and get immediate results of disease diagnosis.
- Practical support for precision agriculture: The ultimate result of the project is a scalable, offline, and accessible tool for chilli farmers. It helps to detect disease early and pick interventions early enough in time to minimize losses and improve yield.

In general, the project shows how advanced AI technologies such as YOLO can be applied to real world problem solutions that can improve the agricultural community. It offers the foundation for future growth in lightweight, intelligent systems that enable farmers with mobile based plant disease diagnosis tools.

1.6 Organization of the Report

This thesis report is structured into six comprehensive chapters. Each chapter focuses on a key aspect of the study, starting from the background of the research to the implementation and final outcomes. The following is a brief overview of what each chapter includes:

Chapter 1: Introduction

This chapter presents a general overview of the research, including the motivation behind the study, its objectives, methodology, and expected outcomes. It highlights the significance of automated chilli leaf disease detection and introduces the role of YOLO models in addressing this agricultural challenge.

Chapter 2: Literature Review

This chapter explores the foundational knowledge related to plant disease detection, deep learning, and object detection models. It includes an extensive literature review covering existing research and similar applications in the field. It also identifies key research gaps that this study aims to address.

Chapter 3: Research Methodology

This chapter provides a detailed explanation of the proposed system design and methodology. It describes the dataset preparation, annotation process, model

selection (YOLOv8s, YOLOv9s, YOLOv10s), training, performance evaluation, and mobile deployment. Functional and non-functional requirements, system architecture, data flow diagrams, and UI design are also discussed.

Chapter 4: Implementation and Results

This chapter outlines the practical implementation of the project, including the environment setup, model training process, and integration into a Flutter-based mobile application. It presents evaluation results, performance analysis of the YOLO models, and a discussion of the findings.

Chapter 5: Challenges and Design Considerations

In this chapter compliance with software and hardware standards, communication protocols, societal and environmental implications that the proposed system has are presented. It also includes sustainability planning; ethical considerations; financial analysis; and, how the study meets complex engineering problems.

Chapter 6: Conclusion and Future Work

The last chapter of the research summarizes work, identifies the main contributions, reflects on the limitations of the study. It also identifies areas for future improvement including but not limited to increasing the dataset, including more disease categories or model optimization for wider applicability.

Chapter 2

Background

Chapter 2 provides the background necessary to understand the importance of chilli leaf disease detection and the role of deep learning, particularly YOLO models, in addressing this challenge. It begins with a literature review, highlighting similar applications where YOLO and CNN models have been successfully applied for plant disease detection. The chapter also discusses related research focusing on segmentation techniques, hybrid models, and mobile deployment efforts. A gap analysis identifies the major shortcomings in existing works, such as the lack of lightweight mobile solutions, limited use of newer YOLO versions, and minimal focus on real-time agricultural applications. The insights gathered in this chapter form the basis for designing the proposed methodology and system in subsequent chapters.

2.1 Introduction

Chilli (Capsicum annuum) is one of the most essential mass horticultural crops grown in many parts of the world with the growing nations of the world such as India, Bangladesh, and Indonesia being the prime ones. With the high demand in the food processing, spice and pharmaceutical industries, it is an economically important crop to millions of farmers. However, chilli plants are highly susceptible to many fungal, bacterial, and viral pathogens that cause diseases in a wide range. These diseases particularly when not detected in early stages would result in substantial yield loss and economic setback for smallholder farmers. Among the biggest problems is the early, precise, and effective identification of these diseases in the real-time farming environment.

Heretofore, manual observation from the farmers or agronomist have formed the platform for disease monitoring and anticipation in chilli cultivation. This approach, although widely used, has weaknesses, and these include case passage, the demand for the expertise, and limitation of the scalability of the far or rural zones. Mis- or a late diagnosis of a disease often results in the quick development of both the disease itself and the spread of the disease, which

destroys huge portions of crops. A change toward automated, intelligent, accessible solutions is not just useful but imperative for modern agriculture.

The emergence of recent artificial intelligence (AI), computer vision and its deep learning has added another dimension to plant disease detection. Significant promise for object detection models, in particular the YOLO (you only look once) family, has been shown in real-time disease detection tasks because of the efficiency and accuracy tradeoff. Algorithm YOLO can review an image in one pass and find and localize several types of diseases in seconds, making it suitable for real-world implementation in agricultural fields.

Scant literature has also advocated for and implemented different deep learning models such as YOLO, Convolutional Neural Networks (CNNs) and hybrid models for detecting chilli leaf diseases. Some have played around with various YOLO versions such as YOLOv4, YOLOv5, and YOLOv7 among others while others have combined object detection with sensors of internet of things (IoT) or image processing methods respectively. Despite these methods reaching quite a considerable success, there still is a multifaceted remainder with mobile deployment, lightweight optimization, real-time usability, and integration of newer YOLO models, such as YOLOv8, YOLOv9, and YOLOv10.

This chapter presents a detailed background of chilli leaf disease detection using AI, emphasizing relevant works through a categorized literature review. It includes an analysis of similar applications, related research in deep learning and smart agriculture, and concludes with a gap analysis that identifies the limitations of existing approaches. These insights provide the foundation for the proposed methodology and guide the development of an optimized, mobile-integrated YOLO-based detection system as outlined in this thesis.

2.2 Literature Review

Chilli production is a major element of the agriculture of many countries, particularly in India and Indonesia, which exert significant economic impact. Chilli crops, on the other hand, are easily susceptible to a number of fungal bacterial and viral diseases that lead to a large drop in harvests. With increasing needs to monitor diseases effectively and accurately, there has been an unprecedented acceleration of deep learning and computer vision researches which has led to the widespread application of YOLO for real-time detection of chilli leaf ailments.

Deep learning techniques have been used by many research projects to detect diseases in chilli leaves. There was also a study carried out by Manoj et al. [1] in which YOLO v5, v6, and v7 models were employed to detect early-stage diseases in chilli plants by using data collected at three stages of growing – 15, 25 and 35 days. According to the investigation, YOLOv7 scored highest mAP with an average of 0.82, meaning that they were most accurate and faster in inference in comparison to other methods. Shapiee et al. [2] also tested YOLOv4 Darknet on classifying chilli plants and leaves, compared it to Efficient Det, Faster R-CNN, and YOLOv4 Tiny. From the experimental findings, it was indicated that YOLOv4 Darknet achieved the best classification mAP of 75.69%, where it excelled in recognising the chilli plant leaves in all conditions.

Kantharaju [3] used Google-Net CNN to classify chilli leaf diseases from a collection of 180 images retrieved from Kaggle. During training, optimizers SGDM, ADAM and RMSProp was used, and SGDM and ADAM optimizers gave the highest accuracy of 98.61%, after training for 40 epochs. SGDM optimizer was considered most effective by the researchers when it came to training limited-data models. Later, Li et. al [4] proposed a complex multi-scale architecture for feature extraction, MCCM (MCSAM-ConvNeXt-MSFFM) that incorporated MCF and MCSAM. The MCCM model demonstrated a notable improvement in accuracy (by 3.38%), precision (2.62%), recall (2.48%), and F1-score (2.53%) compared to traditional CNN models like VGG16, ResNet34, and MobileNetV2.

Object detection techniques have proven to be more effective than traditional classification approaches as they allow localization of diseased regions. Naik et al. [5] applied YOLOv5s and an enhanced single-stage object detection model, introducing exponential linear unit (ELU) activation functions to improve feature extraction. The proposed model achieved a remarkable accuracy of 99.83%, outperforming standard YOLOv5s in precision (0.998), recall (0.998), and mean average precision at different IoU thresholds (mAP@0.5 = 0.995, mAP@0.5:0.95 = 0.975). Meanwhile, Chen et al. [6] explored the impact of HSV color space in plant disease recognition and demonstrated that CNN-based models trained on HSV-transformed images outperformed those trained on RGB images in precision and recall.

Viral diseases such as chilli leaf curl virus (ChiLCV) and Gemini virus pose severe challenges to chilli farming. Zehra et al. [8] discussed various control measures, advocating for the development of resistant cultivars as the most sustainable solution. Solahudin et al. [9] introduced a remote sensing approach using aerial photography and Bayesian segmentation to monitor Gemini virus outbreaks in chilli plantations. The study achieved a segmentation accuracy of 86%, indicating the potential of remote sensing for large-scale disease monitoring. Additionally, Kanaparthi and Ilango [10] evaluated the effectiveness of SqueezeNet for Geminivirus and Mosaic disease detection in chilli leaves. The model, trained on 160 images, achieved 100% training accuracy, which RMSProp achieved at 40 epochs and ADAM at 35 epochs.

Clever techniques in agriculture have contributed to following the progress of crop diseases in real-time and identifying crop diseases. Sambrani and Bhairannawar [11] created a machine learning system that allows for automatic identification of chilli diseases on a hybrid basis using image processing and deep learning. The Karna et al. [12] applied the ESP32 Cam module to develop the CNN-based monitoring system for online real-time assessment of chilled plant health. Although the system has 63% accuracy on new images, its practical use is confirmed, and further optimization is needed for improvement.

An image captioning approach for the detection of chilli diseases was proposed by Sulistyaningrum et al. [13] using ResNeXt50 as an encoder and Transformer as a decoder. The proposed model created meaningful descriptions of disease areas in chilli plants with BLEU score of 30.52%. The work showed the role of natural language processing (NLP) in making disease diagnosis through automation more understandable.

Bandopadhyay et al. [14] introduced the Chili-Net framework drawing ResNet50, VGG16 and InceptionV3 in an ensemble setup for feature extraction. The classification stage used something custom made of deep neural network with 10 layers that had yielded a total accuracy of 97%, precision of 100% and an AUC score of 100%. As Husin et al. [15] have indicated, effective image processing is important during the early detection of diseases with a resulting system that limited the use of pesticides through the application of chemicals

only when diseases were detected.

Mechanically based segmentation was proposed by Das Chagas Silva Araujo et al. [16] and Araujo et al. [18] in the form of a graph to delineate the affected regions of chilli leaves by disease. With an accuracy matching of 86%, their system has potential for real-world application in the sensuous farming settings. Sahuri [17] states that a deep learning system was developed immediately for chilli leaf disease identification based on a Raspberry Pi.

The system automatically produced results instantly on chilli leaf image after the capture by the Raspberry Pi camera. In addition to disease diagnosis, the model provided confidence values which would enable the farmers to take measures promptly. The Gray Level Co-occurrence Matrix approach (GLCM) coupled with a Support Vector Machine (SVM) classifier was used by Sari et al. [19] to classify diseases on chilli. Having achieved the 88% accuracy rate, the model highlights the success of statistical texture analysis in plant disease recognition. Pratap and Kumar [20] have adapted modified EfficientNetB4 (EfficientLeafNetB4) to improve different types of detection of chilli leaf diseases alone. The refined EfficientNetB4- (EfficientLeafNetB4) outperform ResNet-50, DenseNet-121, MobileNet-V2, and VGG-16 with better precision and recall score.

Table 2.1: Summary of Literature Reviewed.

Author (Year)	Models	Accuracy	Contribution
Manoj et al., 2024	YOLOv5, YOLOv6, YOLOv7	mAP: 0.80 - 0.82	Early-stage disease detection in chilli plants using YOLO models
Shapiee et al., 2022	YOLOv4 Darknet, EfficientDet, Faster R-CNN	mAP: 75.69%	Chilli plant classification using transfer learning models
Kantharaju, 2023	Google-Net CNN	98.61%	Chilli leaf disease identification using CNN optimizers
Li et al., 2024	MCCM (MCSAM-ConvNeXt-MSFFM)	Improved by 3.38%	Multi-scale feature extraction for chilli

		(accuracy), 2.62% (precision)	disease recognition
Naik et al., 2023	YOLOv5s, Enhanced Single-Stage Detection Model	99.83%	High-precision chilli leaf disease detection using optimized YOLO models
Bademiyya & Ashtaputre, 2019	Traditional Analysis	50% yield loss estimate	Impact assessment of powdery mildew on chilli yield
Solahudin et al., 2015	Bayesian Segmentation	86%	Remote sensing for Gemini virus monitoring in chilli fields
Kanaparthi & Ilango, 2023	SqueezeNet CNN	100% (RMSPROP, ADAM, 40 epochs)	Training issues in deep learning-based chilli disease detection
Karna et al., 2024	CNN (ESP32 Cam)	63%	Smart monitoring system for chilli plants using deep learning
Sulistyaningrum et al., 2024	ResNeXt50 + Transformer	BLEU: 30.52%	Image captioning for automated chilli disease detection
Araujo et al., 2021	Graph-Based Segmentation	86%	Automated disease identification in chilli leaves using graph segmentation
Sari et al., 2021	GLCM + SVM	88%	Chilli disease classification using statistical feature extraction
Pratap & Kumar, 2023	EfficientNetB4 (EfficientLeafNetB4)	Outperformed ResNet-50, DenseNet-121	High-precision multi-class chilli leaf disease detection

2.2.1 Similar Applications

A variety of investigations has made use of deep learning, especially the use of YOLO models, to detect and classify various chilli leaf diseases. These results demonstrate great practical usability of object detection methods in agriculture for real-time surveillance and classification of plant diseases.

Chilli Leaf Diseases are early-stage detected based on several YOLO versions by Manoj et al. (2024) – YOLOv5, YOLOv6, YOLOv7. Images of three critical growth stages were used in their research dataset. In particular, YOLOv7 performed best with an mAP of 0.82 that further reinforces the efficiency and accuracy of YOLO in fast plant disease identification.

To find out its performance when categorising chilli plants and leaves, Shapiee et al (2022) studied YOLOv4 Darknet in comparison to EfficientDet, Faster R-CNN, and YOLOv4 Tiny. Having an mAP of 75.69%, YOLOv4 Darknet turned out to be very versatile and functional in unpredictable image environments.

Naik et al. (2023) created a superior one-stage detection model based on YOLOv5s and including ELU (Exponential Linear Unit) activations for optimization. The method produced amazing precision (99.83%) and nearly perfect precision and recall and highlighted the effectiveness of the object detection against traditional classification approaches.

Kanaparthi and Ilango (2023) investigated the performance of SqueezeNet CNN in gaging Geminivirus and Mosaic on chilli leaves. Even with a small dataset, the application of ADAM and RMSProp optimizers helped the model to reach 100% training accuracy as well illustrating the advantages associated with lightweight models in restricted environments.

A CNN-driven smart monitoring platform using ESP32 Cam was deployed by Karna et al. (2024) for real-time analysis of chilli leaf health. Although its accuracy was only 63%, the system showed the potential of mobile-detecting edge computing platforms for the monitoring of diseases.

The work demonstrates a growing increase of employment of deep learning-based detection systems for agriculture in particular to chilli production. And in so doing, they prepare the ground and provide impetus to the creation of strong mobile-friendly systems similar to such as YOLOv8, YOLOv9 and YOLOv10, which are capable of being used in real-time in the field.

Table 2.2: Similar Applications – Matrix Table

Author (Year)	Model(s) Used	Accuracy / mAP	Contribution
Manoj et al. (2024)	YOLOv5, YOLOv6, YOLOv7	mAP: 0.80–0.82	Early-stage chilli disease detection across different plant growth stages
Shapiee et al. (2022)	YOLOv4 Darknet, EfficientDet, Faster R-CNN	mAP: 75.69%	Compared object detection models for chilli leaf classification
Naik et al. (2023)	YOLOv5s + ELU-enhanced model	Accuracy: 99.83%	High-precision chilli disease detection with enhanced YOLO model
Kanaparthi & Ilango (2023)	SqueezeNet CNN	Accuracy: 100% (Training)	Detection of Geminivirus and Mosaic diseases in chilli leaves
Karna et al. (2024)	CNN (ESP32 Cam module)	Accuracy: 63%	Smart chilli plant monitoring using low-cost IoT device

2.2.2 Related Research

Although YOLO and deep learning are applied in the detection of chilli disease, other studies have also studied aspects of the multi-scale extraction of features, remote sensing, segmentation, as well as real-time agricultural monitoring. Through the analysis of contributions of such works, researchers can have a better opportunity to build a high-accuracy, far-reaching and mobile friendly platform for the detection of chilli diseases.

This was achieved by use of the Multi-Scale Feature Fusion Module (MSFFM) and the Mixed Channel Spatial Attention Mechanism (MCSAM) component of the proposed multi-scale feature extraction network MCCM as brought out by Li et al. (2024). This improved network showed a higher precision, recall, and F1-score compared to VGG16, ResNet34, and MobileNetV2, signaling how important are advanced attention mechanisms.

Solahudin et al. (2015) used remote sensing and Bayesian segmentation to monitor the spread of Gemini virus in chilli fields. They reported segmentation accuracy at 86% evidencing aerial and segmentation technologies' potential to

monitor many plant diseases.

Sambrani & Bhairannawar (2021) proposed in their framework a combination of image processing methodology and deep learning techniques to automatically detect chilli disease. The proposed framework offers a foundation that can be used when combining traditional procedures with the help of the artificial intelligence in order to promote the reliability of disease classification.

The work by Sulistyaningrum et. al, 2024 used NLP by image captioning using ResNext50 and Transformers to detect chilli disease areas. To demonstrate a new approach to making AI-predictions more human-friendly, the model reaches a BLEU score of 30.52%.

A technique in graph-based segmentation was proposed by Araujo et al (2021) for determining areas of that are disease-affected in chilli leaves with a matching rate of 86%. This strategy displays the nominalization of segmentation in region-specific in making artificial intelligence decisions explicable more.

Sari et al. (2021) used GLCM (Gray-Level Co-occurrence Matrix) for feature extraction and SVM for classification ended up with 88% success rate. The authors demonstrate that the addition of texture-based features is still useful to enhance disease diagnosis.

And, in their research, Pratap and Kumar (2023) optimized EfficientNetB4 (EffectiveLeafNetB4) and, on the whole, outperformed other currently-popular CNN models such as ResNet50 and DenseNet121. Their method focused on attaining high accuracy in determining different variations of disease.

The results of these related studies translate into significant technical views, present novel methods of feature extraction, segmentation, and hybrid modeling, and situate the general implications of this research.

Table 2.3: Related Research – Matrix Table

Author (Year)	Model / Technique	Accuracy / Metric	Contribution
Li et al. (2024)	MCCM (MSFFM + MCSAM)	+3.38% (accuracy)	Multi-scale attention-based feature extraction for better classification
Solahudin et al. (2015)	Bayesian Segmentation	Segmentation Accuracy: 86%	Remote sensing and segmentation for

			Gemini virus monitoring
Sambrani & Bhairannawar (2021)	Hybrid ML + Image Processing Framework	Not specified	Automated chilli disease detection using classical and deep learning approaches
Sulistyaningrum et al. (2024)	ResNeXt50 + Transformer (Image Captioning)	BLEU Score: 30.52%	NLP-based image captioning for improved explainability
Araujo et al. (2021)	Graph-Based Segmentation	Accuracy: 86%	Region-based segmentation of infected leaf areas
Sari et al. (2021)	GLCM + SVM	Accuracy: 88%	Texture-based feature extraction and classification
Pratap & Kumar (2023)	EfficientNetB4 (EfficientLeafNetB4)	Outperformed ResNet, DenseNet	High-precision multi-

2.3 Gap Analysis

Despite the remarkable advancements in chilli leaf disease detection using deep learning and object detection models, several limitations remain that hinder the practical adoption of these methods in real-world farming conditions, especially for smallholder farmers in rural regions. A careful analysis of existing literature reveals the following key research gaps:

- Limited Focus on Real-Time Mobile Deployment: Most existing works, such as those by Manoj et al. (2024) and Naik et al. (2023), demonstrate excellent accuracy using YOLO-based models. However, these models are primarily tested in controlled environments with high-performance computing resources. There is minimal exploration of mobile-friendly deployment, which is essential for real-time usage in the field.
- Lack of Lightweight and Optimized Models: Studies like Kanaparthi & Ilango (2023) and Karna et al. (2024) have explored lightweight models such as SqueezeNet and ESP32-based solutions, but these works either

rely on very small datasets or suffer from reduced accuracy. There remains a need for models that balance both lightweight architecture and high accuracy, suitable for smartphones with limited computational power.

- Underutilization of Latest YOLO Versions: Although some studies used YOLOv5 and YOLOv7, none of the reviewed works implemented or evaluated newer versions like YOLOv8, YOLOv9, or YOLOv10, which offer significant improvements in both speed and detection performance. This opens an opportunity to assess the potential of the latest YOLO variants for chilli disease detection.
- Fragmented Model Evaluation and Lack of Standardization: Across different studies, there is no standardized evaluation of models using common performance metrics (e.g., precision, recall, mAP across IoU thresholds). For example, Shapiee et al. (2022) and Li et al. (2024) reported mAP and precision differently, making it difficult to compare approaches and select the most efficient model architecture for real-time applications.
- Scalability and User-Centric Design Are Often Overlooked: Few studies, such as Karna et al. (2024), considered real-time systems using IoT or mobile devices, but they lacked scalability, multilingual support, or farmer-friendly user interfaces. The design of mobile applications tailored for low-literacy users in rural communities remains underexplored.
- Insufficient Integration of Detection and Actionable Insights: While many models detect diseases, they often do not offer actionable recommendations or context-aware guidance to farmers. Real-world tools need to go beyond classification by suggesting treatment or crop management strategies.

Table 2.4: Gap Analysis Table

Gap Area	Observed in Literature	Addressed in This Thesis
Lack of real-time mobile deployment	Manoj et al. (2024), Naik et al. (2023)	Developed a Flutter-based mobile app using YOLO + TFLite

Absence of lightweight, optimized models	Kanaparthi & Ilango (2023), Karna et al. (2024)	Converted best YOLO model into optimized TFLite format
Limited use of latest YOLO versions	Most work stops at YOLOv7 or YOLOv5	Evaluates YOLOv8s, YOLOv9s, and YOLOv10s
Inconsistent evaluation and benchmarks	Shapiee et al. (2022), Li et al. (2024)	Uses standardized metrics (mAP, precision, recall, inference time, etc.)
Poor scalability and user accessibility	Karna et al. (2024), Sambrani & Bhairannawar (2021)	Designs user-friendly interface for farmers in rural areas
No actionable recommendations post-detection	Most models end with classification	Mobile app can be extended to show disease info and management tips

2.4 Summary

This chapter reviewed various research studies focusing on the detection and classification of chilli leaf diseases using deep learning models, particularly YOLO. The similar applications section presented works that directly used YOLO and CNNs for chilli disease identification, demonstrating promising results in accuracy and efficiency. The related research section highlighted innovative techniques such as graph-based segmentation, image captioning, and hybrid feature extraction methods, showcasing the evolving nature of smart agriculture.

The gap analysis revealed several limitations in current literature, including the lack of mobile deployment, the underuse of the latest YOLO versions, limited focus on lightweight model optimization, and minimal user-friendly application design. These insights helped shape the direction of this research, which aims to build a lightweight, real-time, mobile-integrated chilli leaf disease detection system using the most recent YOLO models.

Chapter 3

Research Methodology

In Chapter 3, the mechanism of development, implementation of proposed chilli leaf disease detection system is discussed. The chapter continues to explain how the design of the system was, i.e., how datasets were collected, labeled, and utilized to train models such as YOLOv8s, YOLOv9s, and YOLOv10s. This chapter introduces the methodology for evaluating models based on accuracy, inference speed, and the complexity of computation that supports the choice of the most successful model. In addition, the chapter speaks about the conversion of the best model to the TensorFlow Lite format to improve mobile performance and incorporates it into a Flutter-based Android app. Functional and non-functional requirements, system diagrams and aspects around Project planning that form an in-depth view of the technical development methodology are outlined in this section.

3.1 Methodology/Requirement Analysis & Design Specification

3.1.1 Overview

This work suggests a deep learning approach for the detection of chilli leaf diseases by using YOLO models. A procedural process was established, starting with data gathering to data preparation, then training, validation, and deployment of the model. From Roboflow, there is the Chilli Dataset that provides 1,403 images that are split into training (1,224), validation (117) and testing (62) sets. Preprocesses used include auto-adjusting orientation, resizing images and 90° rotations for data augmentation. Experiments with YOLOv8 Small, YOLOv9 Small and YOLOv10 Small models are done to detect and classify images of Bercak, Keriting, and Kuning diseases. The best model is exported into a TensorFlow Lite (.tflite) file type and embedded into a Flutter based mobile application. These real-time capabilities make the solution useful to farmers, thus supporting precision agriculture uptake.

3.1.2 Proposed Methodology/ System Design

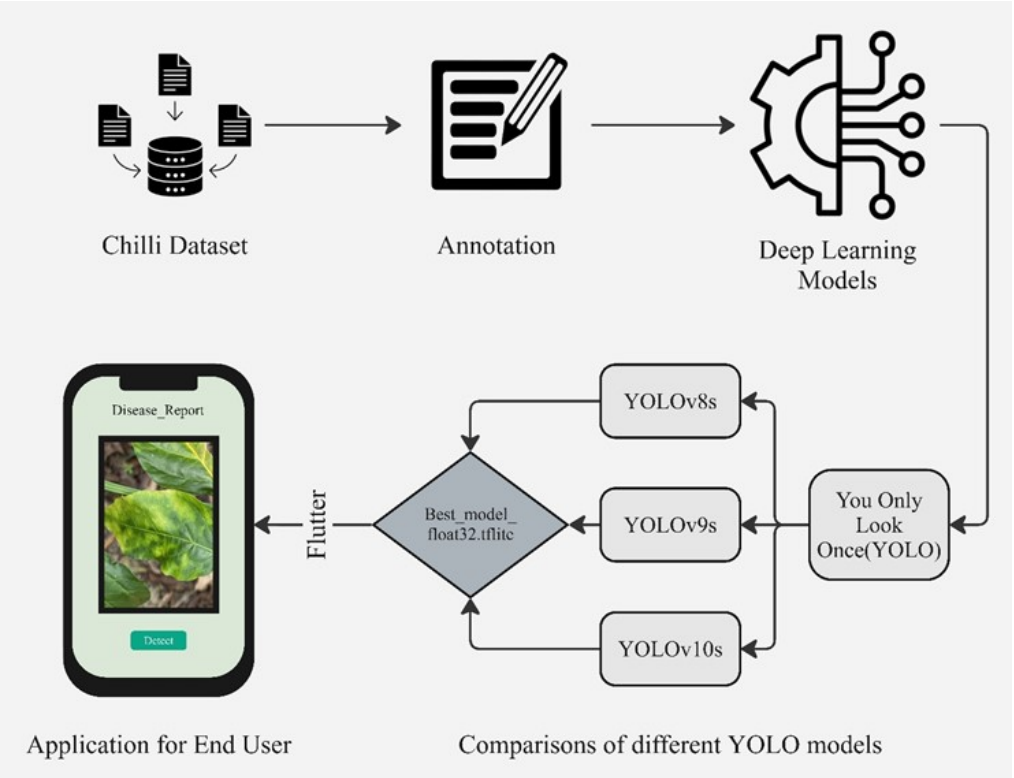


Figure 3.1: The Methodological procedure

In figure 3.1, a visual representation of how the workflow is used to detect chilli leaf diseases is shown after the application of deep learning models. The first step is the Chilli Dataset that being labeled so it can classify diseases before the model for Deep learning will be trained. The research evaluates three YOLO architectures – YOLOv8 Small, YOLOv9 Small, and YOLOv10 Small – to detect and classify diseases as Bercak, Keriting, and Kuning. After training and comparative analysis, the superior model is exported into TensorFlow Lite (.tflite) file to permit deployment. The refined model is integrated into a mobile application written in Flutter to facilitate farmers' rapid identification of diseases on chilli leaves by means of their capturing images, which supports practical precision agriculture.

3.1.3 Functional and Nonfunctional Requirements

The system manages to achieve accuracy, efficiency and usability because of its compliance to functional and non-functional criteria. Central functional requirements include: preparing datasets, training and validating the YOLO model, testing model performance, Wizard of Oz and mobile application deployment through Flutter (for timely mining of diseases). Nonfunctional requirements focus on performance, scalability, reliability and security to deliver quick processing, reliable accuracy and ease of use. The fulfillment of these criteria leads to a holistic and

practical system of precision agriculture enabling farmers to actively detect and treat chilli leaf diseases.

1. Functional Requirements

- Dataset Processing – The system must load and preprocess the Chilli Dataset, including annotation, resizing, and augmentation.
- Model Training – The system must support training deep learning models (YOLOv8 Small, YOLOv9 Small, and YOLOv10 Small) for disease detection.
- Model Evaluation and Selection – The system must compare different YOLO models based on accuracy, precision, recall, and F1-score to select the best-performing model.
- Model Conversion – The best-trained model must be converted into TensorFlow Lite (.tflite format) for mobile deployment.
- Mobile Application Integration – The Flutter-based mobile application must integrate the trained model to allow users to upload or capture leaf images for disease detection.
- Disease Classification – The app must classify leaf diseases into three categories: Bercak, Keriting, and Kuning and display results to users.
- User-Friendly Interface – The application must provide a simple and interactive UI for farmers to detect diseases easily.
- Real-time Processing – The app should enable real-time or near-real-time disease detection for immediate decision-making.
- Model Update Capability – The system should allow updates to the model to improve accuracy over time.

2. Nonfunctional Requirements

- Performance – The system should process images and detect diseases within a few seconds to provide a seamless user experience.
- Accuracy – The selected model must achieve high accuracy and robustness to minimize false predictions.
- Scalability – The application should support future enhancements with additional datasets and disease categories.
- Usability – The mobile application should be intuitive and easy to use for non-technical users such as farmers.
- Reliability – The system must ensure consistent performance with minimal crashes or failures.

- Portability – The application must be compatible with Android devices and capable of running on different mobile platforms in future updates.
- Security – The system should protect user data and model integrity from unauthorized access or modifications.
- Storage Efficiency – The TensorFlow Lite model should be optimized to consume minimal storage and computational resources on mobile devices.

3.1.4 Data Flow Diagram Level 1

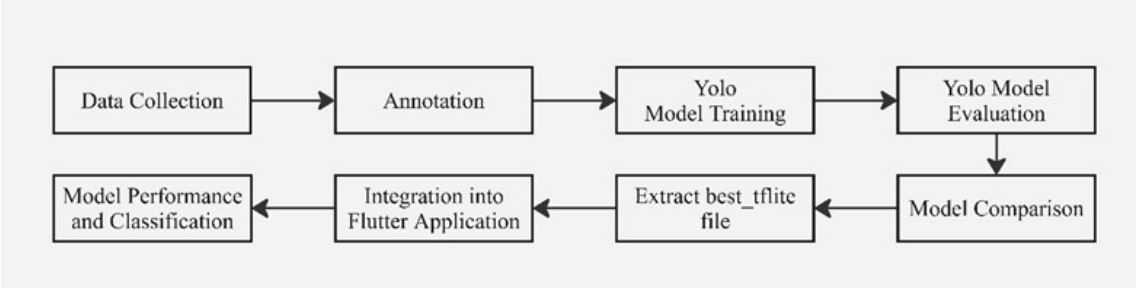


Figure 3.2: Proposed Data Flow Diagram Level -1

The Level 1 DFD gives a visual representation of the process of detect chilli leaf diseases using deep learning techniques. Data collection begins by taking the images of chilli leaves to be examined. Then it is annotated to provide specific disease labels, which it is used to train the YOLO model. After training the models, they are extensively tested based on performance metrics, and compared afterward to determine which architecture performs best. The best model is obtained and transformed into a TensorFlow Lite (.tflite) mode for mobile deployment. The best model is implemented in a mobile application constructed from Flutter, providing instantaneous real-time disease detection and classification functionalities. The deployment proves to be able to accurately classify diseases, thus providing a useful resource in the form of an efficient and readily usable division for precision agriculture.

3.1.5 UI Design

The end users especially farmers will benefit optimally from the UI of the chilli leaf disease detection system one that is simplistic and unambiguous. The mobile application uses the Flutter framework to provide a uniform experience in several platforms. The UI is designed with optimisation in mind – it enables arguments to capture leaf images or submit them, process them in real time with the optimised YOLO model to provide disease identification, and obtain results instantly. The

streamlined UI is aimed at an easy to use functionality, for those users who may not have the background knowledge in technology.

1. Application Setup

The Flutter-based mobile application is designed to function smoothly on Android devices, offering clean and interactive user experience. The setup consists of the following components:

- Home Screen: Displays an easy-to-navigate interface with an option to capture an image using the device camera or upload an existing image from the gallery. Provides a simple "Detect" button to start the leaf disease classification process.
- Detection Screen: Once an image is selected, it is preprocessed and passed through the embedded TensorFlow Lite (.tflite) model for disease detection. The system identifies the disease category (Bercak, Keriting, or Kuning) and presents the result visually.
- Results and Recommendations Screen: Displays the detected disease category along with a confidence score. Provides agronomic recommendations for managing the identified disease. Allows users to retake an image or analyze another leaf for further detection.
- Offline Functionality: The app does not require an internet connection since the YOLO-based TensorFlow Lite model runs on-device, making it efficient for field use.

2. Two-Tier Architecture

The chilli leaf disease detection system follows a two-tier architecture, which consists of a frontend (user interface) layer and a backend (model processing) layer. This architecture ensures efficient data processing, low-latency predictions, and smooth user interactions.

- Frontend Layer (User Interface): The Flutter-based UI serves as the primary interface for users, enabling them to interact with the system easily. Key functions include:
  - Providing a user-friendly dashboard for image selection and disease detection.
  - Displaying results in real-time with clear labels and confidence scores.
  - Offering recommendations for disease management based on detection outcomes.
- Backend Layer (Model Processing & Inference Engine): The backend layer

handles deep learning model execution and image processing directly on the mobile device. Key aspects include:

- TensorFlow Lite (.tflite) model integration to run deep learning inference without requiring external servers.
- On-device processing, ensuring fast detection even in remote agricultural areas with no internet access.
- Optimized computational efficiency, allowing the model to process images with minimal power consumption, making it suitable for mobile deployment.

By this two-tier structure, the system ensures that it provides real time results while sparing resources and thereby enabling farmers to have a reliable responsive tool for detecting chilli leaf problem.

3. Folder Structure

Folder structure design of the chilli leaf disease detection system is meant to make assets, model files, and source code management optimal in the Flutter-based mobile application. Major directories and files vital for model integration and the successful running of the application are taken in the architecture.

- Root Directory: Chilli_leaf_detector/
  This is the main coding file which contains the essential components for the application. This involves source code, assets, and machine learning models.
- assets/ Directory: This folder stores all necessary static resources, such as the labels file required for interpreting the model's output.
  - labels.txt – Contains the class names of chilli leaf diseases (Bercak, Keriting, Kuning) that the model predicts.
- Model File (model.tflite): This is the TensorFlow Lite model file, which is optimized for mobile deployment. It is responsible for running on-device inference, allowing real-time chilli leaf disease detection without internet dependency.

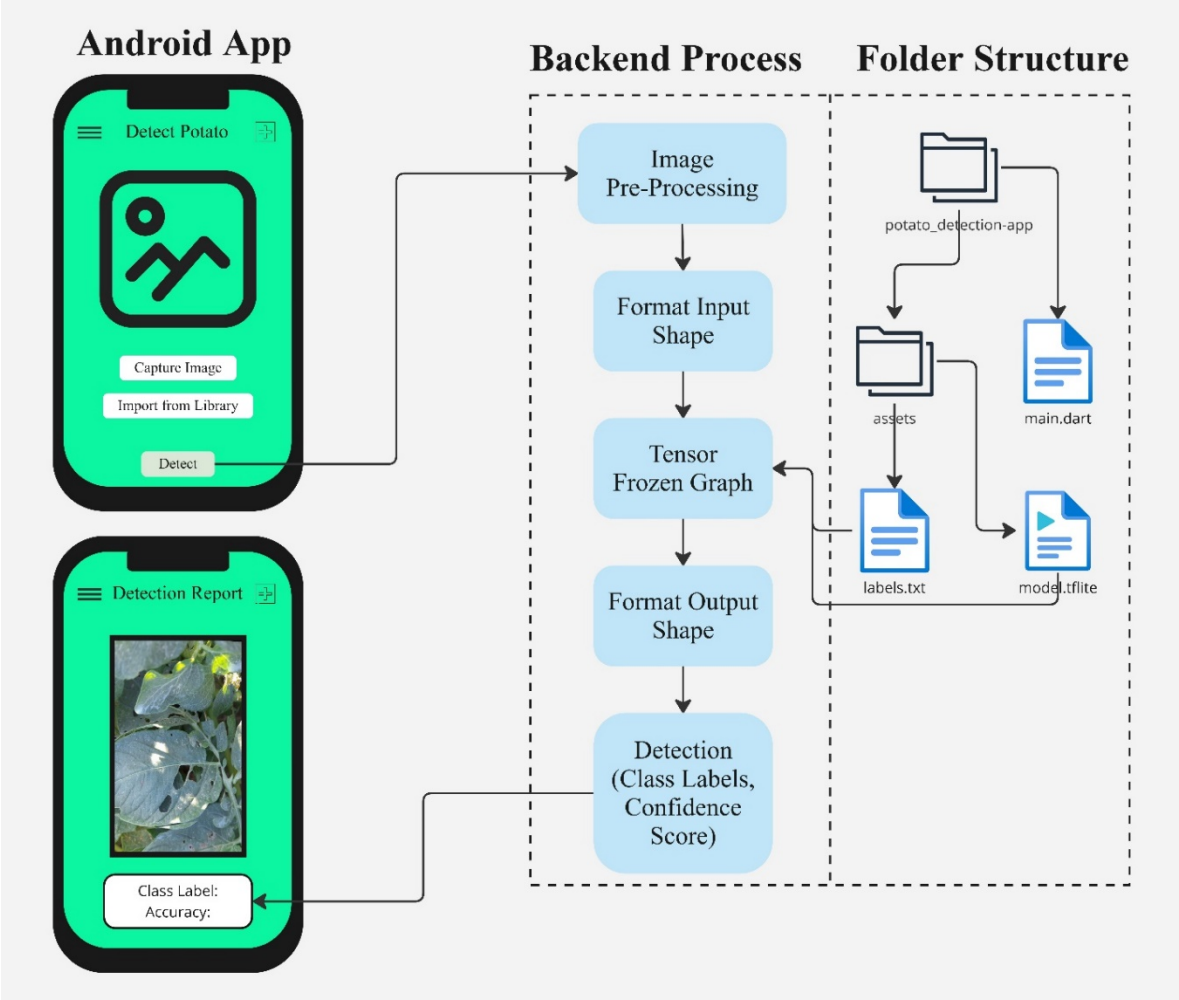


Figure 3.3: The Two-Tier Architecture and UI for Application

3.2 Detailed Methodology and Design

3.2.1 Dataset

The current research is based on the Chilli Leaf Disease dataset for which the author used Roboflow (Dataset Link) to get it. The available images number to 1,403, with 1,224 training, 117 validation, and 62 testing samples. The images are divided into three different diseases categories namely; Bercak, Keriting and kining. Bercak, Keriting, Kuning which indicates the variety of conditions in chilli leaf diseases. To enhance model generalization, images are automatically aligned and down-sampled to 512×512 pixels, and in addition augmented with 90° rotations and reflections in all three orientations. Upon deploying these techniques, the variety of the dataset is boosted; consequently, a model that will be even more appropriate to determine diseases in real-world conditions is produced.

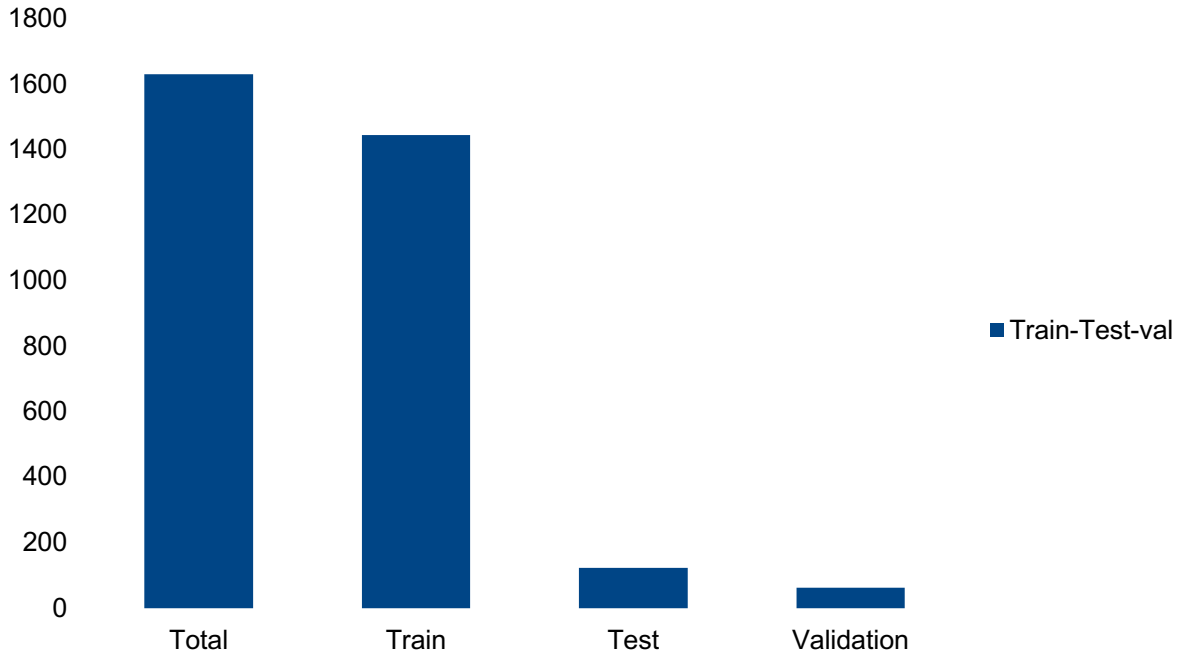


Figure 3.4: Train-Validation_Test Splitting

3.2.2 Classes

Based on observed leaf symptoms, the chilli leaf disease detection system slices images of leaves into three distinct classes. These groups cover the highest spread diseases threatening chilli leaves and decreasing the yield and the health of the crop.

- **Bercak (Spotted Leaves):** Leaves within this category have occasional spots or lesions that are commonly linked to fungus or bacteria. Spots are of different sizes and colors; mostly they appear as brown or dark patches on the leaf surface.
- **Keriting (Curling Leaves):** This group is made of leaves that demonstrate curling, wrinkling or deformations, and are frequently the results of viral infections, pests, or poor nutrient accumulation. Leaves that so suffer might show abnormalities in form, reduced dimensions.
- **Kuning (Yellowing Leaves):** Leaves in this classification show yellowing, or chlorosis, that may occur due to imbalance in their nutrients, viral invaders or adverse environmental conditions. Chlorosis can be localized or patchy in its appearance or uniform in its distribution across a leaf.

Bercak Keriting Kuning

Figure 3.5: The sample Images of Each Class.

Table 3.1: Dataset Specifications.

Properties	Values
Image Resolution	240 × 240 pixels
Format	.jpg
Total Images	1631
Classes	3

3.2.3 Data Preprocessing Steps

In order to enhance the efficiency and robustness of the YOLO-based chilli leaf disease detection model, a number of data preparation methods are applied to the training process. Through the use of these preprocessing techniques we are assured uniformity, have higher ability of the model to generalize and we prepare the data better for deep learning. The preprocessing steps include:

- **Auto-Orientation:** Automated changes are implemented to dataset images in order to position all images in a similar orientation, which makes the analysis of all leaf images more straightforward.
- **Resizing:** In order to conform to the requirements of the YOLO model, images are all resized to the same dimensions of 512x512 pixels, allowing the singer to be trained in identical terms throughout the dataset.
- **Data Augmentation:** To improve model generalization and reduce overfitting, several augmentation techniques are applied, including:
  - 90° Rotation (Clockwise, Counterclockwise, and Upside Down): Enhances the dataset diversity by creating multiple variations of the same image, helping the model recognize diseases from different angles.
  - Normalization: The pixel values are scaled to a standard range (e.g., [0,1]) to improve training efficiency and prevent numerical instability in deep learning models.

- **Annotation:** For adequate training of YOLO models, an image is annotated manually with a bounding box drawn around areas of disease. Through the use of bounding boxes, the model is able to distinguish and classify special disease patterns during training. Annotations are saved in YOLO format, where class identifiers and the coordinates of exactly bounding boxes are used.

3.2.4 Model Description

1. YOLO Architecture

The You Only Look Once (YOLO) architecture is a real-time object detection model that transforms the conventional two-step detection process (region proposal + classification) into a single unified network. YOLO processes an image in one forward pass of the neural network, making it highly efficient for real-time applications such as precision agriculture.

YOLO divides an input image into an S × S grid. Each grid cell is responsible for detecting objects whose center falls within that cell. For each cell, the model predicts:

- Bounding box coordinates (x, y, w, h) where x and y define the center of the bounding box, and w and h represent width and height.
- Object confidence score that indicates the probability of an object being present in the bounding box.
- Class probabilities to identify the object category (e.g., Bercak, Keriting, Kuning in this study).

The YOLO model generates predictions using a fully convolutional neural network (CNN). The output tensor consists of bounding boxes, confidence scores, and class probabilities in a single forward pass.

Mathematically, the YOLO prediction function can be expressed as:

$$\hat{y} = f(X;\, \theta) \quad \text{(i)}$$

where:

- X represents the input image,
- θ represents the trainable parameters of the CNN,
- $\hat{y}$ is the predicted output, consisting of bounding boxes, class labels, and confidence scores.

2. YOLO Working Process

- **Feature Extraction using CNN:** The input image is passed through a deep convolutional network, extracting hierarchical features that capture spatial and contextual information about leaf diseases.

- **Grid-based Object Localization:** The image is divided into an S × S grid, where each grid cell predicts bounding boxes, confidence scores, and class probabilities.
- **Bounding Box Prediction:** Each bounding box is predicted using the following equation:

$$Confidence\ Score = P(Object)\ \times IOU_{pred,gt} \_\_\_\_\_ \text{(ii)}$$

where:

  - $P(Object)$ is the probability of an object being present in the grid cell
  - $IOU_{pred,gt}$ is the Intersection Over Union (IoU) between the predicted bounding box and the ground truth bounding box.
- **Non-Maximum Suppression (NMS):** To eliminate redundant detections and keep the most accurate bounding box, YOLO applies Non-Maximum Suppression (NMS) by retaining only the highest predictions and discarding overlapping ones.
- **Final Output:** After post-processing, the model outputs:
  - Bounding box coordinates around detected diseased areas.
  - Class label (Bercak, Keriting, Kuning) for the identified disease.
  - Confidence score representing the detection certainly.

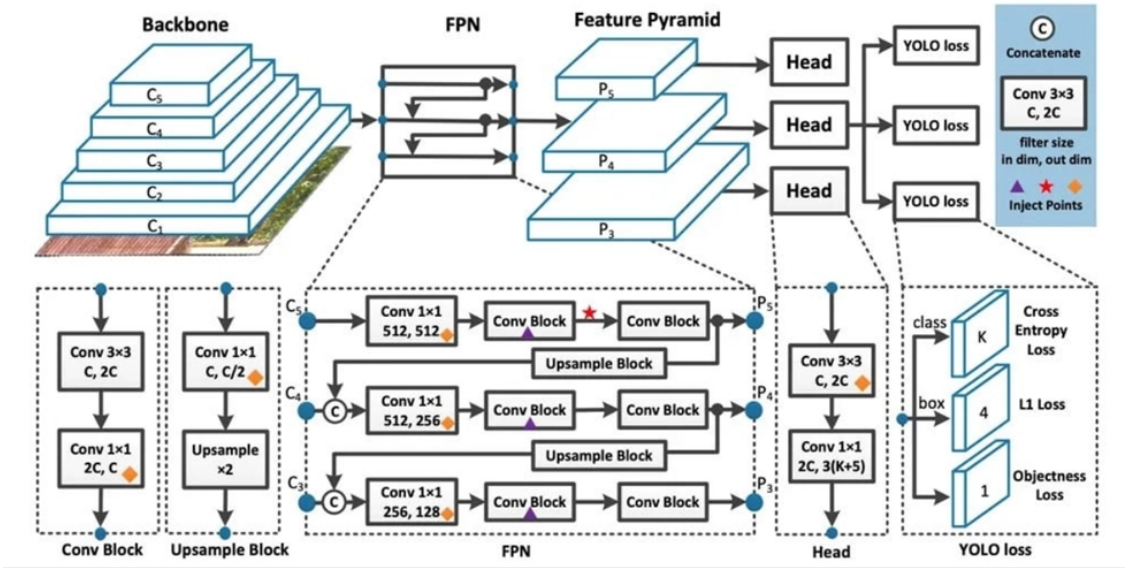


Figure 3.6: The Architecture of YOLO Model

3. Model Comparison between YOLO architectures (version 8, 9, 10)

To ensure optimal performance in chilli leaf disease detection, this study compares three YOLO models: YOLOv8 Small, YOLOv9 Small, and YOLOv10 Small. While all

three models follow the same fundamental YOLO architecture, they differ significantly in their network depth, feature extraction capabilities, detection accuracy, processing speed, and suitability for mobile deployment. Such variations are factor of how good each model is in detecting and segregating diseases on chilli leaves as diverse environmental and computational constraints.

YOLOv8 Small has been designed for mobile devices with low computing power, where it is outstanding in real-time object localization constraints. The usage of dynamic anchor adjustment and adaptive spatial fusion in YOLOv9 Small improves object localization abilities, advancing detection for small and complicated leaf disease objects. With the addition of advanced feature fusion and transformer-based structure, YOLOv10 Small obtains better accuracy in the detection of small, detailed lesions in chilli leaves. However, the improved performance necessitates more computational capacities leading to fewer viable alternatives using mobile processors that lack much processing capability, YOLOv10 Small as one.

Table 3.2: Comparison Table

Parameter	YOLOv8 Small	YOLOv9 Small	YOLOv10 Small
Model Size (MB)	8 MB	12 MB	18 MB
Number of Parameters (Million)	6.2M	8.5M	10.1M
Processing Speed (FPS)	Fast (~50 FPS on edge devices)	Moderate (~35 FPS)	Slower (~25 FPS)
Detection Accuracy	Moderate	Higher than YOLOv8	Highest among the three
Power Consumption	Low (Optimized for mobile)	Moderate (Slightly heavier	Higher (More computationally intensive)

		than YOLOv8)	
Suitability for Mobile Deployment	Highly suitable for real-time mobile applications	Suitable for mobile, but requires slightly more processing power	Less optimized for mobile, better suited for high-performance devices

3.3 Project Plan

The chilli leaf disease detection initiative is guided by a clear process that ensures the whole workflow, ranging from dataset gathering to implementation is carried out efficiently. The project starts with dataset collecting and necessary preprocessing actions (annotation, resizing, image augmentation) to maximize the capability of models to generalize. Afterward, we train and test three YOLO models namely, YOLOv8 Small, YOLOv9 Small and YOLOv10 Small to determine the model that is most accurate and efficient. After choosing the most performing model, it is then transformed into TensorFlow Lite (.tflite format) so that it can be deployed into the mobiles. After the model development, the team will design and develop a Flutter application to flawlessly integrate the model for on-the-go disease detection. To ensure mobile compatibility, the project includes heavy testing and error fixing processes in order to facilitate seamless device integration. Ultimately, the application undergoes rigorous trials in practical farming environments to ensure its effectiveness in farmers' use. The main objective is to develop a powerful, easy-to-use mobile application for chilli leaf diseases detection, develop precision agriculture, and enhance general crop health management.

Table 3.3: GANTT Chart of Project Timeline.

Process	Sep'24	Oct'24	Nov'24	Dec'24	Jan'25	Feb'25	Mar'25
Working Plan							
Theoretical Study							
Literature Review							
Data Collection							
Data Annotation							
Model Design							
Methodology Writing							
Report Writing							
Review and Finalization							

3.4 Task Allocation

The project's task organization defines the orderliness and efficiency of workflow by an organized carrying out of operations on data processing, model training, and application distribution. The preparation of data is left to the team that has the control of some of the critical jobs such as annotation, resizing, and data augmentation thus offering better training data for the models. The responsibility of the model development team lies in training and fine-tuning YOLOv8 Small, YOLOv9 Small and YOLOv10 Small models, optimizing hyper-parameters, and finally choosing the best model. With the model finalized, the deployment team embarks on steps for TensorFlow Lite conversion and optimization towards the smooth mobile inference. Instead, the Flutter development team integrates the edge model effortlessly, implements a friendly interface, and incorporates features for live disease identification in the app. The testing and evaluation team thoroughly review actual agricultural images in the real-world to ensure correctness of the model and performance of the application. Finally, the project management team monitors progress, promotes communication among teams, and ensures that phases are completed on time to allow a practical real time chilli leaf disease detection system for precision agriculture to be developed.

3.5 Summary

We introduce this paper presenting a systematic design of the framework to bring on an effective real-time chilli leaf disease detection system utilizing the deep learning and mobile deployment methodologies. A Roboflow dataset with 1631 chilli leaf pictures divided into three diseases serves as the first step in acquiring the dataset. Auto-orientation, resizing and annotation, it well as augmentation, in pre-processing the data, enhances the data set‘s quality , which in turn also helps to generalize better. Detection models are built with YOLOv8 Small, YOLOv9 Small and YOLOv10 Small architecture models, with each model being benchmarked in performance, speed and suitability for mobile application integration. After selection of the best model, it is converted into TensorFlow Lite. (. tflite format) for efficient on-device inference. Afterwards, the trained model is deployed in a Flutter mobile app, and users can take or upload photos, with an immediate ability of viewing disease classifications. The architecture of the system adopts a two tier architecture that distinguishes front end interface and the deep learning backend, thus facilitating compatibility and simplified processing. By using a transparent project plan and the appropriate task distribution, the whole process is maximized, helping efficient data processing, model development, validation, and delivery. Using this methodology, a robust, reliable and accessible solution for precision agriculture is attained to empower farmers to apply mobile phones for effective detection and management of chilli leaf diseases.

Chapter 4

Implementation and Results

This chapter evaluates the performance of three YOLO models:<< YOLOv8s, YOLOv9s, and YOLOv10s on object detection tasks. Through their classification accuracy, recall, and precision as well as their computational speed, the aim of the study is to identify the strong and the weak sides of these models. The chapter explores the object detection and classification of the models through examination of training/validation loss curves, confusion matrices, precision-recall curves, and classification reports.

4.1 Environment Setup

In all the deep learning models in this study, the baseline configuration they were trained using was according to the parameters defined in Table 4.1. The inputs images were reduced to 640 × 640 pixels while retaining a practical compromise between retaining crucial pieces of the disease and having viable training processes. The application of a 16-batch size facilitated a fine status quo between memory efficiency and enabling efficient gradient updates as well as model generalization. We trained on 80 epochs, giving the networks ample opportunities to familiarize themselves with the data and learn whole patterns without the lack of underfitting and excess of overfitting. AdamW optimizer chosen was an evolution of the original Adam algorithm that de-coupled weight decay from the learning rate to give a more stable and effective optimization algorithm. Using this strategy, regularization is enhanced, and training is made more controlled, and this is particularly important for deep convolutional networks. To achieve smooth convergence of the model the learning rate could be set to a value of 0.0001429; thus giving room for the adjustment of its weights without overshooting minima as it was optimizing.

Table 4.1: Common parameter table for all experimented models.

Parameter Name	Parameter Value
Image Size	640 × 640
Batch Size	16
Epoch	80
Optimizer	AdamW
Learning Rate	0.001429

Table 4.2 gives the methodology of dividing the set, which is crucial for models' training, validation, and testing. The whole dataset was distributed into 80% for training, 12% for testing, and 7% for testing (1224, 117, and 57 images respectively). In order to support deep learning models in learning faithful and flexible patterns, this method was deliberately structured to allocate the majority of the data to the training phase. When applied at 12% of the total to validate the model, this set was a critical resource during training to monitor the performance of the model, minimize overfitting, and adjust hyperparameters and stopping protocols if necessary. The test set (7%) did not enter the view of the training at all, which served as an objective measure of how well the final model does.

Table 4.2: Common data split for all experimented models.

Dataset	In Percentage	Number of Images
Train set	80%	1224
Validation Set	12%	117
Test Set	7%	57

4.2 Testing and Evaluation/Performance/ Comparative Analysis

The researchers evaluated YOLO models with precision, recall, and mean average precision (mAP) and F1 score, using certain metrics to find the best model for detecting Potato leaf diseases.

Average Precision (AP): AP is the action of averaging over N images or instances the ratio of true positives over the sum of true positives and false positives. It shows the model's efficacy in doing the correct positive predictions relative to the positive

predications that were falsely made over a range of detection thresholds.

$$AP = \frac{\frac{TP}{TP+FP}}{N} \_\_\_\_\_ \text{(iii)}$$

Mean Average Precision (mAP): mAP represents an average assessment of the model's effectiveness on all detected objects after averaging AP values for many classes. It is mostly reported in percentage and often employed as benchmark for effect of various models in object detection.

$$mAP = \frac{\sum_{i=1}^{Q} AP}{Q} \times 100\% \_\_\_\_\_ \text{(iv)}$$

Precision: Precision gives the proportion of the true positives (TP) against the total of true positives (TP&FP). This metric measures the correctness of positive predictions, which is essential in the cases when errors on the positive side are absolutely devastating.

$$Precision = \frac{TP}{TP+FP} \_\_\_\_\_ \text{(v)}$$

Recall: Recall is given by true positives (TP) divided by true positives and false negatives (FN). It measures the model's ability to identify all relevant cases, which shows the number of true positives which it identifies correctly.

$$Recall = \frac{TP}{TP+FN} \_\_\_\_\_ \text{(vi)}$$

F1-score: The F1-score is the combination of precision and recall using harmonic mean, thus achieving equality in paying attention to both. It proves especially useful for working with unbalanced data because it measures false positives and false negatives, giving a more total picture of how successful predictive models are.

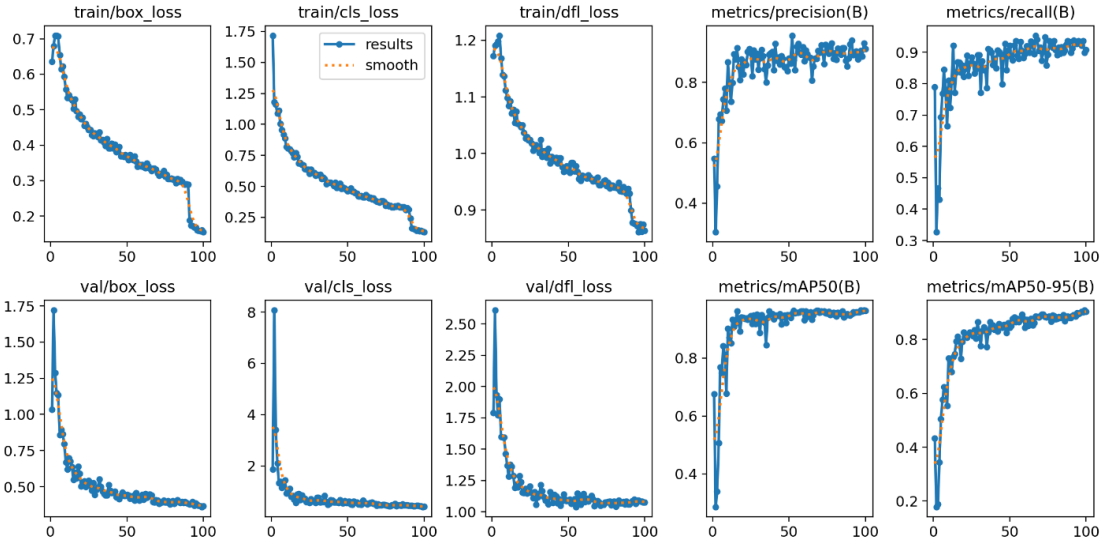
$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Reacll} \_\_\_\_\_ \text{(vii)}$$

4.3 Results and Discussion

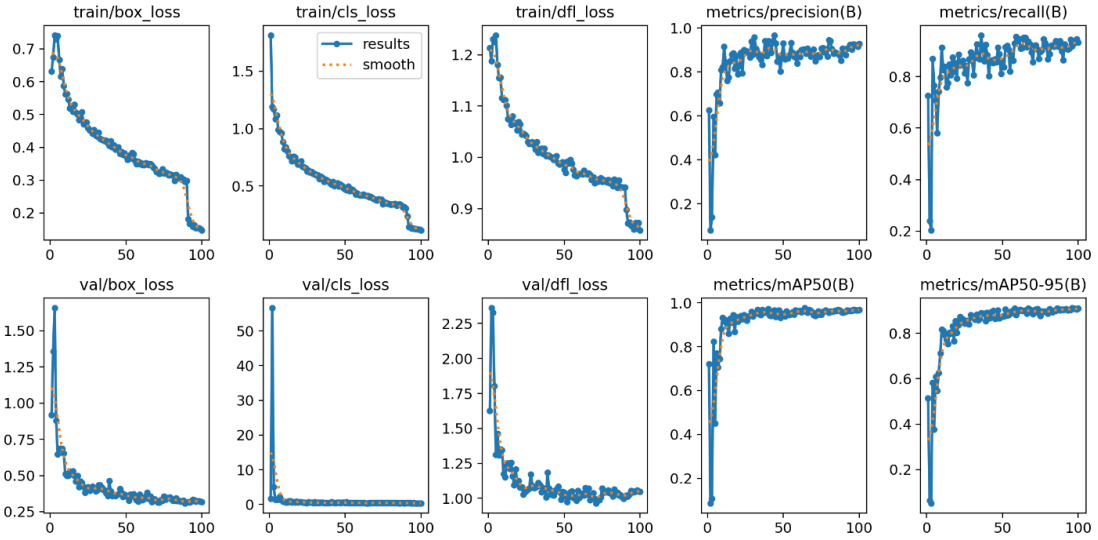
Figure 4.1 illustrates the training and validation curves for all three YOLO model variants YOLOv8, YOLOv9, and YOLOv10 (small). These include three primary loss components: box loss, classification loss, and Distribution Focal Loss (DFL), alongside key evaluation metrics such as precision, recall, and mean Average Precision (mAP). From the figure, it's evident that all models successfully converged

during training, with losses consistently decreasing over the epochs. However, YOLOv10 stands out with smoother, more stable curves and quicker loss stabilization, particularly in classification loss. Its validation metrics also remain consistently higher across training, reflecting superior learning capability and generalization power. YOLOv9 follows closely with competitive metrics but slightly higher loss values than YOLOv10. YOLOv8, although effective, exhibits more fluctuations in both training loss and validation metrics, which may suggest less robustness during optimization.

YOLOv8 (small)



YOLOv9 (small)



YOLOv10 (small)

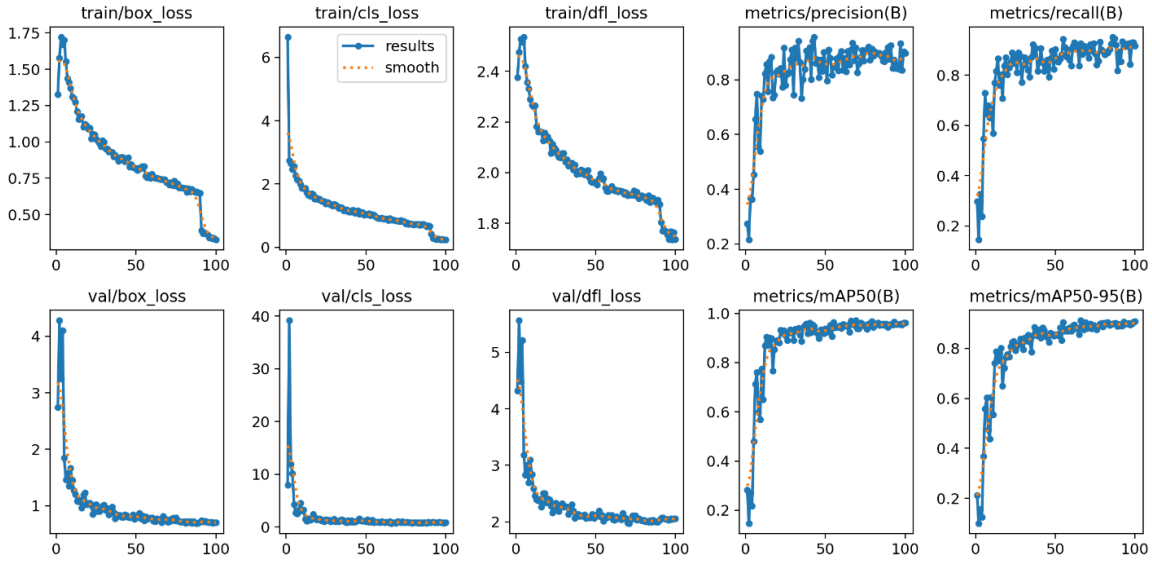
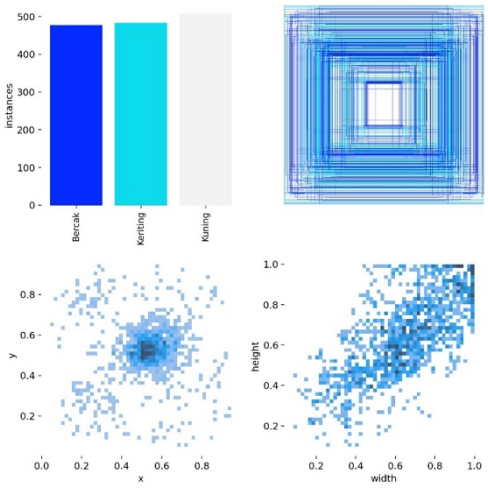


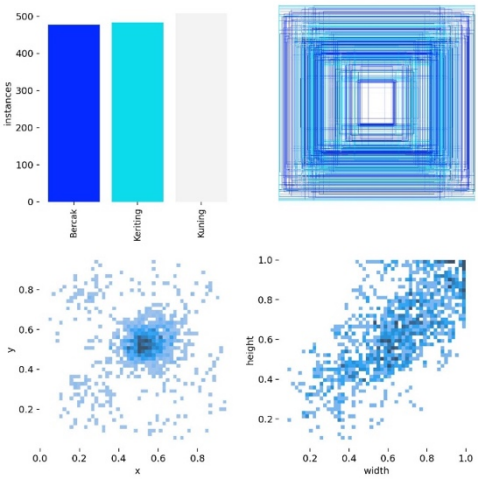
Figure 4.1: Training and Validation Loss and Metrics Curves for Box, Classification, and DFL Loss with Precision, Recall, and mAP Metrics for different YOLO models.

Figure 4.2 presents the data distribution and bounding box annotation visualization across the three target classes: Bercak, Keriting, and Kuning. This figure serves to analyze how well each YOLO variant handles object localization. YOLOv10 displays the most accurate bounding box representations with tight, centered boxes that capture the diseased regions precisely. YOLOv9 also shows strong localization, but with a few instances of minor overbounding, especially for irregular-shaped disease patterns. YOLOv8, in contrast, occasionally produces slightly loose or misplaced boxes, particularly in clustered areas. These discrepancies in bounding box alignment directly influence overall detection precision and reflect differences in spatial feature extraction capabilities across model versions.

YOLOv8 (small)



YOLOv9 (small)



YOLOv10 (small)

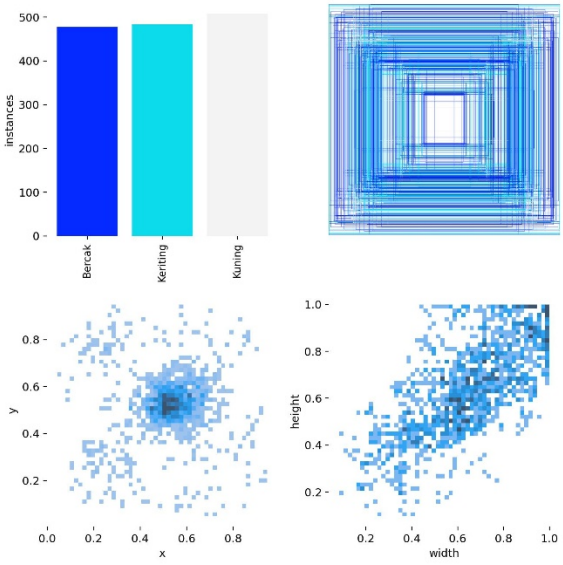
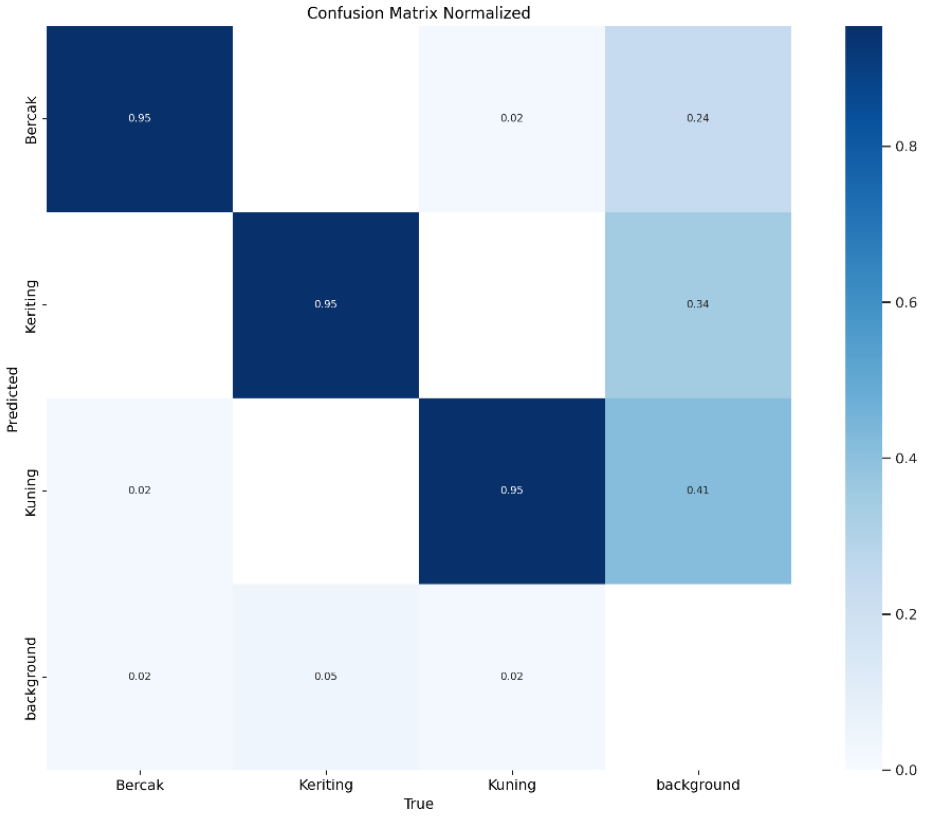
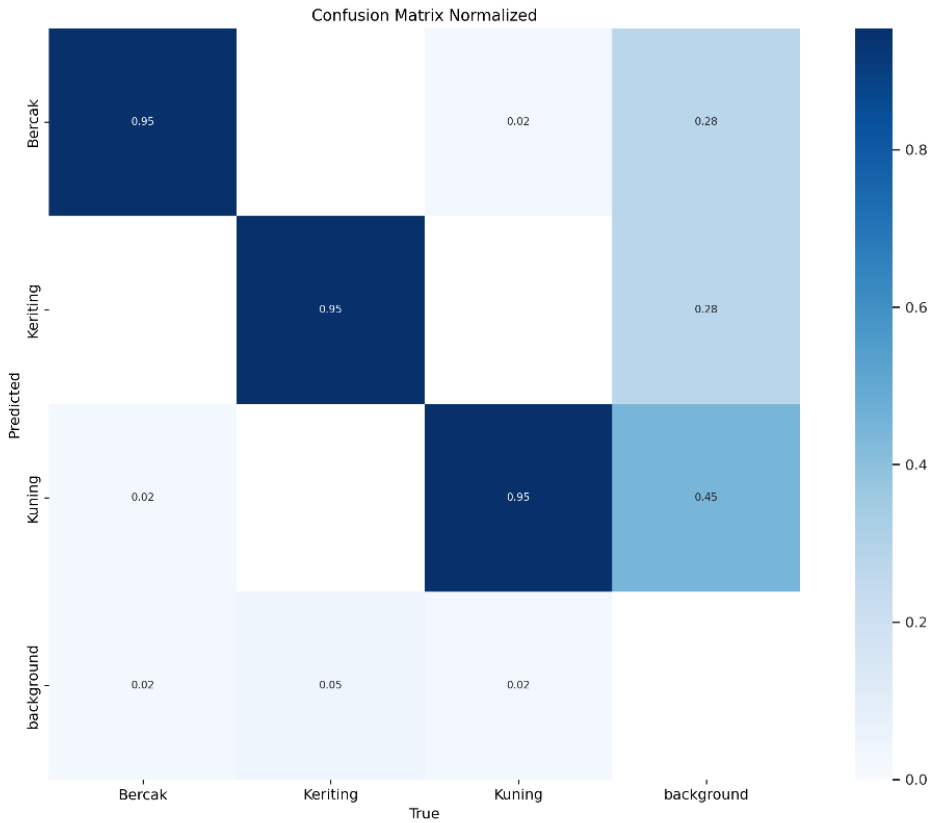


Figure 4.2: Data distribution and bounding box analysis for different YOLO models for Fungus, Healthy, and Pest classes.

YOLOv8 (small)



YOLOv9 (small)



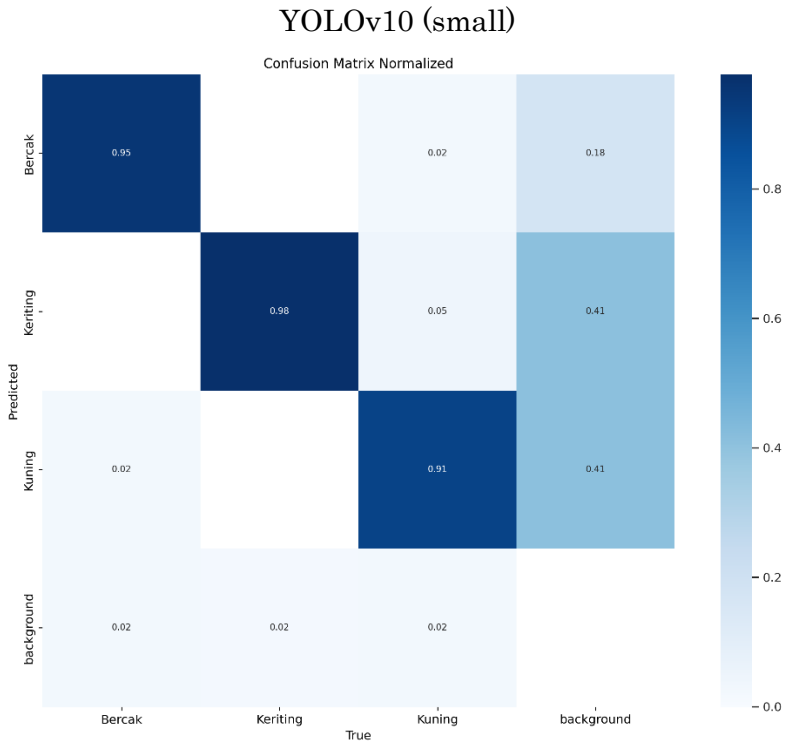


Figure 4.3: Confusion matrix for the different YOLO models.

Figure 4.3 depicts the confusion matrices for the three models, offering a clear view of classification performance per class. YOLOv10 again shows dominance, with minimal off-diagonal entries indicating fewer misclassifications. All three classes (Bercak, Keriting, and Kuning) are predicted with high confidence and low error rates. YOLOv9 performs comparably well but shows slight confusion between Bercak and Keriting, which might be due to similar texture patterns. YOLOv8 exhibits more frequent misclassifications, especially with Keriting samples being misidentified as Kuning, suggesting slightly weaker feature discrimination. The progressive improvement from YOLOv8 to YOLOv10 reflects enhancements in the architecture's ability to differentiate between subtle inter-class features.

The precision curve of Figure 4.4 illustrates how accurately the three YOLO models YOLOv8, YOLOv9, and YOLOv10 make predictions across varying confidence thresholds. Among them, YOLOv10 consistently maintains the highest precision throughout the entire range, indicating that the model produces very few false positives, even at lower confidence thresholds. This means that when YOLOv10 predicts an object as a certain class, it is more likely to be correct than the others. YOLOv9 follows closely, showing a slightly lower precision at lower thresholds but gradually aligning with YOLOv10 at moderate to high thresholds. In contrast, YOLOv8 exhibits more fluctuation in its precision, with occasional dips especially in

lower and mid-range thresholds. This suggests YOLOv8's predictions are less reliable, particularly when it is less confident. Overall, the precision curve indicates that YOLOv10 is the most confident and reliable model in terms of correct positive predictions, which is especially critical in real-time applications where false alarms must be minimized.

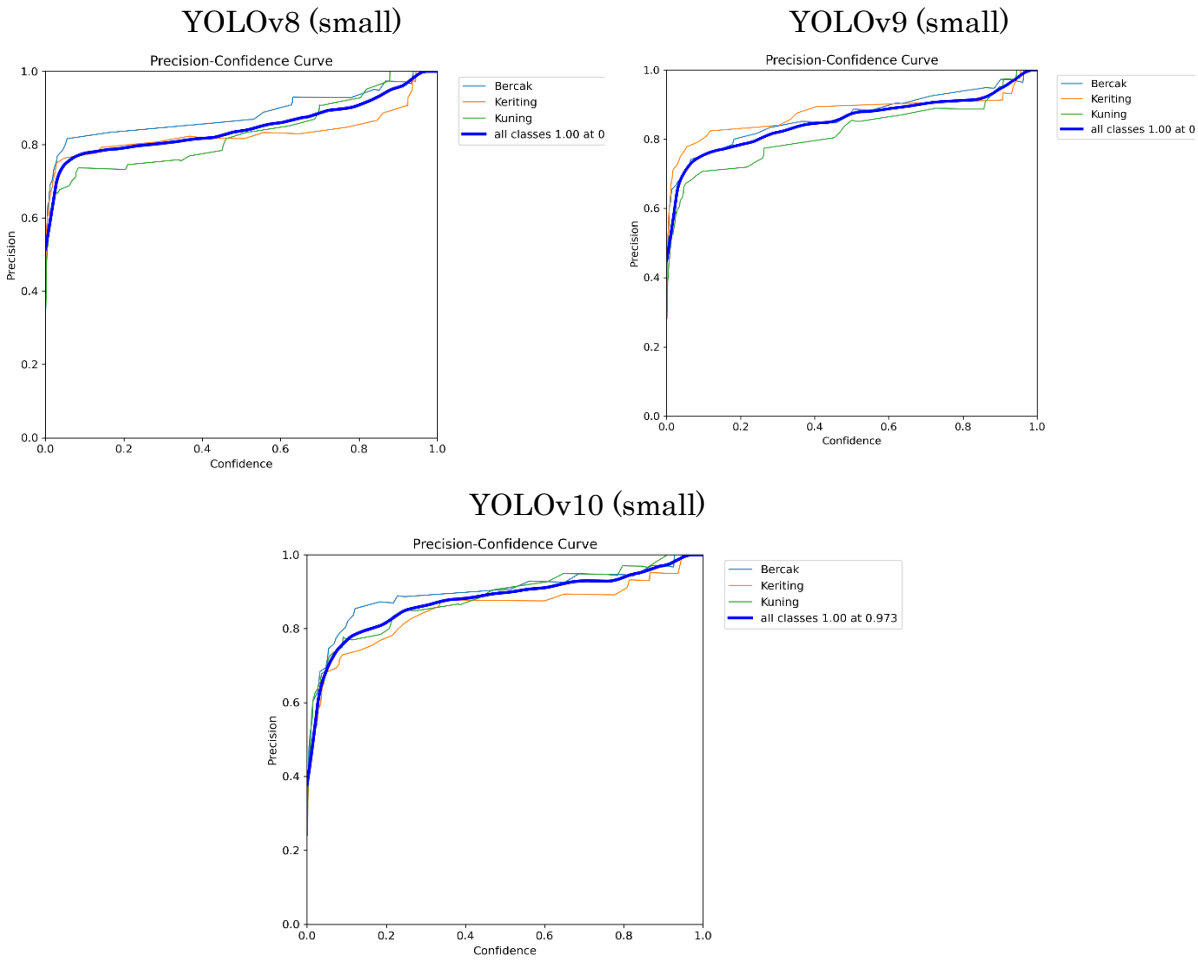


Figure 4.4: Precision curve for different YOLO models.

The recall curve of Figure 4.5 presents the models' ability to correctly detect all relevant instances across confidence thresholds. YOLOv10 achieves the highest recall, especially in low-to-mid threshold ranges, demonstrating its ability to detect more true positives without missing many instances. This makes it particularly valuable for tasks where missing a detection could lead to serious consequences, such as disease monitoring. YOLOv9 also performs admirably, nearly matching YOLOv10 at multiple points, with occasional spikes where it briefly surpasses YOLOv10, suggesting higher sensitivity in certain thresholds or classes. YOLOv8, however, trails behind, showing an earlier and steeper decline in recall as the confidence

threshold increases. This decline highlights the model's limitations in maintaining sensitivity at higher confidence levels. In practical terms, YOLOv10 offers the best balance of high recall and consistent detection, ensuring minimal false negatives.

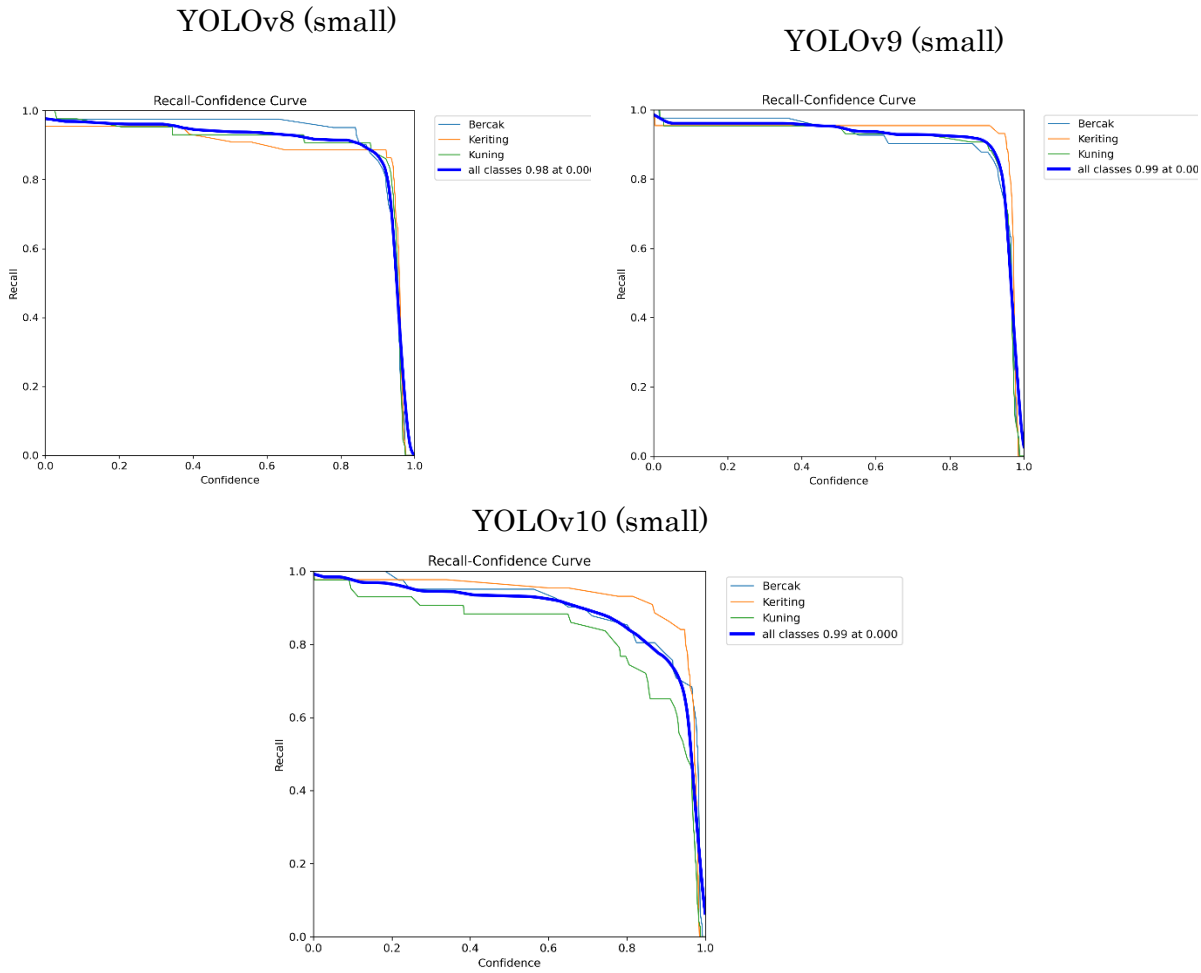
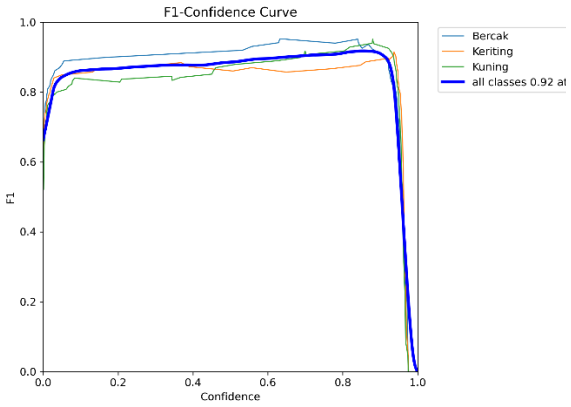


Figure 4.5: Recall curve for different YOLO models.

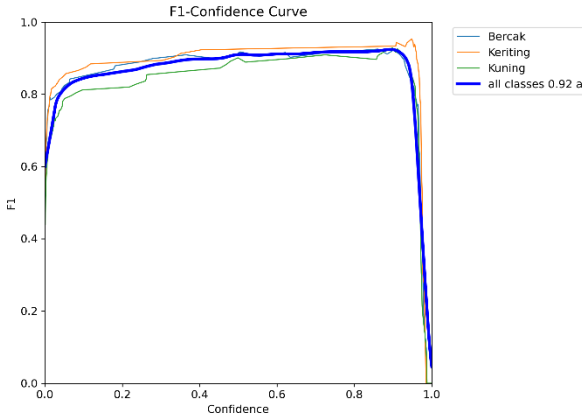
The F1-score curve of Figure 4.6 combines both precision and recall to provide a balanced view of each model's overall effectiveness. YOLOv10 achieves the highest and most stable F1-score across nearly all confidence thresholds, reflecting its strong trade-off between making accurate predictions and capturing all relevant instances. The curve remains smooth and gradually slopes down only at very high confidence levels, where most models naturally become stricter. YOLOv9 comes close, with a well-formed curve that mirrors YOLOv10's performance but stays slightly below it in overall magnitude, particularly in mid-to-high thresholds. YOLOv8's F1-score curve, however, displays inconsistencies and sharper dips, especially around mid-range thresholds, indicating challenges in maintaining balance between precision

and recall. Among the three, YOLOv10 is clearly the most balanced model, offering superior performance under varying threshold conditions and making it highly suitable for precision-critical and recall-sensitive applications alike.

YOLOv8 (small)



YOLOv9 (small)



YOLOv10 (small)

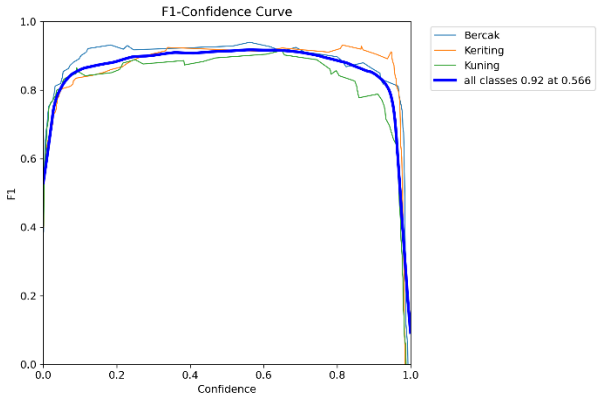
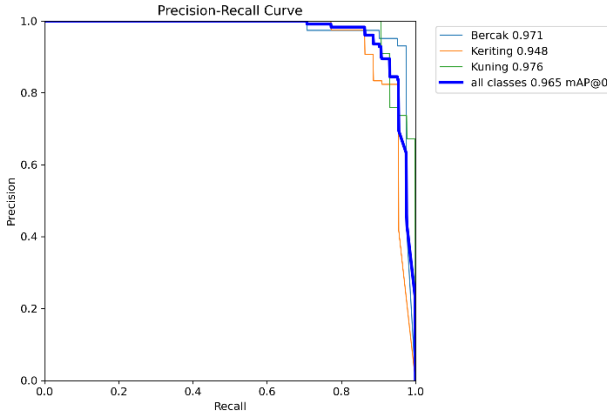


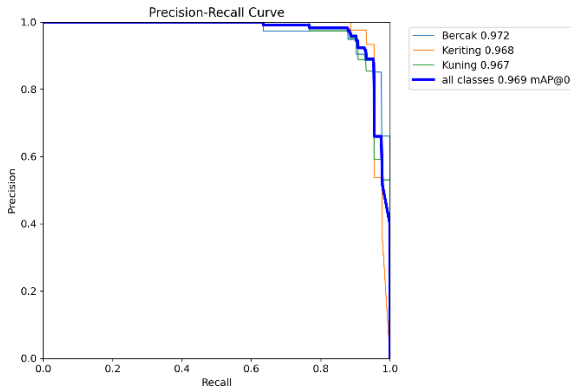
Figure 4.6: F1 curve for different YOLO models.

Figure 4.7 shows the Precision-Recall (PR) curve for all models. YOLOv10's PR curve encloses the largest area, showcasing its superiority in managing the trade-off between false positives and false negatives. Its curve is smooth and consistently high, indicating reliable performance across all thresholds. YOLOv9's curve closely follows, demonstrating its competitiveness, particularly for the Keriting class. YOLOv8's PR curve, however, dips at various points, especially at higher thresholds, reflecting weaker performance under stricter conditions. This visualization consolidates the earlier findings, where YOLOv10 consistently excels in both detection, precision and sensitivity.

YOLOv8 (small)



YOLOv9 (small)



YOLOv10 (small)

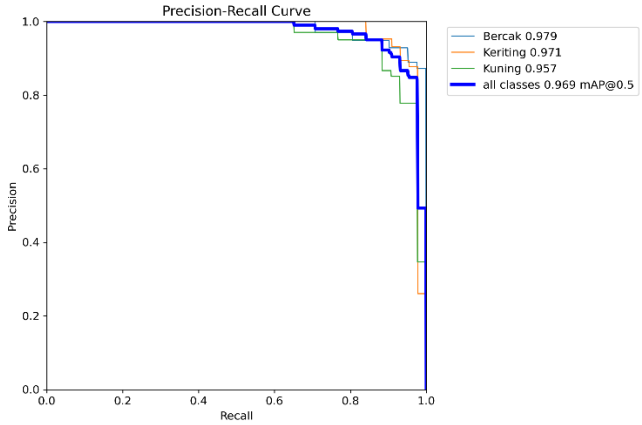


Figure 4.7: Precision-Recall curve for different YOLO models.

Table 4.3 provides a comprehensive classification report including precision, recall, mAP50, and mAP50-95 scores for each class and overall. YOLOv10 achieves the highest recall (0.925) and mAP50-95 (0.912), demonstrating both strong class coverage and excellent localization accuracy. Per-class performance shows that YOLOv10 maintains high values across all three categories, with notably strong results in detecting Bercak (recall 0.935). YOLOv9 closely follows, with outstanding recall for Keriting (0.995) and overall solid metrics. YOLOv8, while accurate (precision 0.93), falls behind slightly in recall and mAP50-95, confirming the model's relatively limited sensitivity compared to its successors.

Table 4.3: Classification report for different YOLO models.

Class	Images	Instances	Precision	Recall	mAP50	mAP50-95
YOLOv8 (small)						
all	117	128	0.93	0.899	0.965	0.909
Bercak	39	41	0.951	0.902	0.971	0.899
Keriting	39	44	0.875	0.886	0.948	0.91
Kuning	39	43	0.965	0.907	0.976	0.919
YOLOv9 (small)						
all	117	128	0.927	0.918	0.969	0.911
Bercak	39	41	0.948	0.891	0.972	0.907
Keriting	39	44	0.912	0.995	0.968	0.901
Kuning	39	43	0.922	0.907	0.977	0.924
YOLOv10 (small)						
all	117	128	0.909	0.925	0.969	0.912
Bercak	39	41	0.927	0.935	0.979	0.905
Keriting	39	44	0.875	0.955	0.971	0.925
Kuning	39	43	0.925	0.884	0.957	0.906

Table 4.4: Performance Comparison of YOLO Models on mAP, GFLOPs, and Inference Speed.

Model Name	mAP50	mAP50-95	GFLOPs	Inference Speed
YOLOv8 (small)	96.5 %	90.9 %	28.4	11.3 ms
YOLOv9 (small)	96.9 %	91.1 %	26.7	9.0 ms
YOLOv10 (small)	96.9%	91.2%	24.5	7.3 ms

Table 4.4 summarizes the computational and performance metrics—mAP, GFLOPs, and inference speed. YOLOv10 not only matches YOLOv9 in mAP50 (96.9%) but surpasses it in mAP50-95 (91.2%) while maintaining the lowest computational

footprint at 24.5 GFLOPs. Additionally, it achieves the fastest inference speed (7.3 ms), outperforming YOLOv9 (9.0 ms) and YOLOv8 (11.3 ms). This positions YOLOv10 as the most efficient and accurate model, offering the best trade-off between performance and resource demand, making it ideal for real-time deployment on edge devices or mobile platforms

4.4 Summary

This chapter presented a comparative evaluation of YOLOv8, YOLOv9, and YOLOv10 (small versions) for multi-class disease detection. Among the models, YOLOv10 consistently outperformed the others in accuracy, localization, and computational efficiency. Training and validation curves showed faster convergence and lower loss values for YOLOv10, indicating more stable learning. Bounding box analysis confirmed its superior localization, especially in handling complex disease regions. Confusion matrices highlighted fewer misclassifications for YOLOv10, particularly in distinguishing similar classes. Metric-wise, YOLOv10 maintained the highest precision, recall, and F1-scores across confidence thresholds, suggesting a strong balance between sensitivity and specificity. Precision-Recall curves further reinforced its robustness, showing the largest area under the curve. The classification report revealed the highest recall (0.925) and mAP50-95 (91.2%) for YOLOv10, while performance for YOLOv9 remained close but slightly lower. YOLOv8 lagged behind with occasional inconsistencies. Finally, the model comparison table showed YOLOv10 as the most efficient, achieving the lowest GFLOPs (24.5) and fastest inference speed (7.3 ms), making it ideal for real-time deployment. Overall, YOLOv10 emerged as the most reliable and efficient model, offering a strong balance between accuracy and speed for practical applications.

Chapter 5

Engineering Standards and Design Challenges

Chapter 5 discusses the engineering standards, compliance requirements, and design challenges encountered during the development of the chilli leaf disease detection system. It highlights adherence to software standards through Flutter development, Google Colab training, and Android Studio deployment. Hardware standards for mobile device compatibility and efficient GPU use during model training are also addressed. The chapter outlines communication standards, focusing on TensorFlow Lite integration for offline inference. It further analyzes the societal, environmental, and ethical impacts of the system, presents the project management and financial analysis, and maps the work to complex engineering problem-solving and activities, demonstrating the interdisciplinary nature and real-world relevance of the project.

5.1 Compliance with the Standards

5.1.1 Software Standards

The software development process complied with several widely accepted practices and standards:

- **Flutter Development Standards:** The mobile application was developed using Flutter, which follows the Material Design guidelines recommended by Google. It ensures consistent UI behavior, cross-platform compatibility, and accessibility for users across various devices and screen sizes.
- **Google Colab GPU Environment:** Model training was conducted using Python and TensorFlow in Google Colaboratory, leveraging cloud-based GPU acceleration. This environment adheres to open-source software development standards and ensures reproducibility, scalability, and efficient computation.

- **Android Studio Deployment:** The final mobile application was tested and deployed using Android Studio, which complies with Android SDK

development guidelines. Proper software versioning, compatibility checks, and debugging tools were used to ensure stability and performance.

- **Coding Practices:** Modular coding, clean architecture, version control (using Git), and proper documentation were maintained throughout the development cycle to align with modern software engineering standards.

5.1.2 Hardware Standards

Although the core of this system is software-driven, it interacts with hardware components during deployment:

- **Mobile Device Compatibility:** The mobile application is designed to run on Android smartphones with standard ARM-based processors, sufficient memory (minimum 2 GB RAM), and a camera for image capture. The app is optimized for low-resource devices to support farmers in rural areas.
- **GPU Utilization on Google Colab:** Model training was performed using NVIDIA Tesla K80 or T4 GPUs in Google Colab, which comply with high-performance computing standards and are well-suited for deep learning applications.
- **Device Resource Optimization:** The trained model was converted to TensorFlow Lite (TFLite) format to meet hardware constraints of mobile devices, reducing memory usage and ensuring real-time inference without needing external servers or high-end hardware.

5.1.3 Communication Standards

Communication between the mobile application and the detection model follows the following key practices:

- **TensorFlow Lite Integration:** The TFLite model is embedded within the Android app, enabling offline, on-device inference. This eliminates the need for external network calls, ensuring fast and secure communication between the app interface and the model.
- **Platform-Specific APIs:** Standard Flutter and Android APIs are used to access camera input, storage, and system resources. This guarantees compatibility with a wide range of devices while maintaining privacy and data handling standards.
- **Secure Model Access:** Store and run the model on the device to avoid unnecessary data transfer to the unsecured networks. Applying this solution,

we meet requirements for privacy-driven communication standards that are applicable to field agriculture.

5.2 Impact on Society, Environment and Sustainability

Improvements in AI-powered solutions for agriculture can potentially revolutionize how farmers over the years have planted their soil. This part explores the implications of the proposed YOLO based chilli leaf disease detection system for the users themselves, farming collectives, environmental conservation, and responsible technology application. By adopting a lightweight mobile app, this system increases accessibility, scalability, and preservation of a digital empowerment in the rural areas for the long-term.

5.2.1 Impact on Life

In order to overcome this challenge, a mobile application powered by YOLO object detection models is incorporated into the system allowing for quick and easy real-time detection of chilli leaf problems. With one picture of a leaf taken by a smartphone, the farmer can immediately obtain disease identification and take action upon it with either preventive or corrective measures.

Such methods enhance farmer autonomy, reducing the need for outside advice and assistance in the case of residents of outlying districts. Decisions within this framework need to be made in a hurry in countering the transmission of disease and saving larger portions of the yield, respectively. These are the households whose stability is to be increased, the crop yield and, consequently, the improvement of the standards of living.

5.2.2 Impact on Society & Environment

Provided that diseases are detected early, the demand for pesticides will be minimal, thereby reducing the environmental impacts of the crop care. This approach contributes to sustaining the potential of agricultural regions for future use, in sustainable ways.

By being adapted to various crops and environments the technology can create environmental sustainability by increasing its scope, thus providing standardized minimal invasive disease management across farming systems. When implementing privacy-guarding solutions like federated learning, we observe data security principles, which is completely essential in the digital farming era. This approach

establishes confidence in the application of AI in agriculture and facilitates a scenario for systematic, evidentiary-based growth in sustainable agriculture.

5.2.3 Ethical Aspects

It is essential if we are to achieve believability, impartiality and easier communication in agriculture by developing AI solutions in a responsible and transparent way. Ethical considerations are observed by the system in the following ways:

- Data Privacy: The system performs all processing independently on the user's device using the TFLite version of the YOLO model for training that has been used for image analysis. The design makes any image analysis performed completely within the device so that no personal data or farm imagery can be sent to outside groups or servers.
- Transparency and Accessibility: By making the interface intuitive, multilingual and easily to understand, the application assists farmers with few digital skills to use it without having to hire expert help. Through the application, farmers from different backgrounds and regions are given equal opportunities to use technology.
- Bias Mitigation: The model is trained on diverse disease samples recorded at different growth phases and in varying visual settings ensuring that it performs well in diverse situations and also has no bias. If the system can be incorporated with XAI tools such as Grad-CAM and LIME, this would increase users' confidence in AI-generated disease detection predictions.

5.2.4 Sustainability Plan

The flexible and compact structure of the system is important for sustainable operation.

Using TensorFlow Lite, the app works offline, and so the app remains useful to rural users who have limited access to the internet for dependable performance. This technique results in consistent offline results with almost no extra cost/power demand irrespective of remote server availability. The app is tailored to fit different Android phones, including those outdated in their software. Due to the optimization of the design, the need for minimal processing power, it consumes minimal resources, proving its durability and decreasing the need for hardware upgrades. The modular

setup of the system provides flexibility for changes needed for changing needs, e.g., adding other crops, identifying new diseases, and reproduction of evaluation tools for both soil and weather. As such, the flexibility of the application is consistent with evolving changes in agricultural practices. The purpose of our work is to promote the precision farming, allowing the farmers to reduce costs, operate more efficiently, rely on valid data when making decisions. In addition, the use of the application strengthens food security by enhancing resilience to environmental shift as well as practicing sustainable farming techniques.

5.3 Project Management and Financial Analysis

Effective project management and resource planning are critical in ensuring the successful development and deployment of any engineering solution. In this research, a structured and time-bound approach was followed to manage the development of the chilli leaf disease detection system—from dataset preparation to mobile deployment. This section discusses the project's planning, task allocation, and financial considerations to evaluate its overall feasibility and sustainability.

5.3.1. Project Management

The project was managed using standard software development lifecycle (SDLC) principles, which ensured systematic progression through requirement gathering, design, implementation, testing, and deployment. A timeline was established at the beginning of the project with clearly defined milestones. Agile-inspired task-tracking tools such as Google Sheets and Trello were used to monitor progress and adjust timelines where necessary.

The development process was divided into several phases:

- **Requirement Analysis and Planning:** Initial phase involved understanding user needs (farmers), defining technical requirements, and selecting suitable tools (YOLOv8/9/10, Flutter, Android Studio, Google Colab).
- **Dataset Collection and Annotation:** Images of chilli leaves were collected and annotated with bounding boxes using tools like LabelImg. This phase required coordination with local data sources and agricultural experts for ground truth verification.
- **Model Training and Optimization:** YOLO models were trained using Google Colab with GPU support. Training hyperparameters were adjusted to maximize performance while keeping model size optimized for mobile

deployment.

- **TFLite Conversion and App Integration:** The best-performing model was converted into TensorFlow Lite format and integrated into a Flutter-based mobile application developed and tested using Android Studio.
- **Evaluation and Deployment:** The system was evaluated on various metrics (mAP, precision, recall, inference time) and tested on Android devices for real-time performance.

5.3.2 Risk Management

To ensure project success, several risk factors were identified and addressed early on:

- **Model Accuracy Risk:** Mitigated through extensive training on diverse datasets and performance benchmarking.
- **Mobile Compatibility Issues:** Addressed by testing on multiple Android devices with varying specifications.
- **Data Availability:** Overcome by combining existing open-source image sets with manually collected samples.
- **Integration Challenges:** Handled modular development, ensuring smooth integration of the model with the mobile app interface.

5.3.3 Financial Analysis

Although this was an academic research project with limited funding, key financial aspects were considered to ensure cost-efficiency and sustainability. Below is a breakdown of estimated project costs:

Table 5.1: Financial Analysis Chart.

Category	Resources / Tools	Estimated Cost
Computing Resources	Google Colab Pro (optional)	Free / 1200 taka per month (optional)
Software Tools	Flutter SDK, Android Studio	Free (Open Source)
Dataset Collection & Annotation	Manual image collection, LabelImg	Manual effort (in-house)

Device Testing	Android Smartphones (2–3 devices)	Existing
Total Estimated Cost		Minimal – Within student budget

5.4 Complex Engineering Problem

This section highlights the complex nature of the engineering problem addressed in this project. Designing a lightweight, highly accurate, and real-time chilli leaf disease detection system that runs efficiently on mobile devices required solving multifaceted challenges across deep learning, image processing, and mobile software development. The integration of object detection (YOLOv8/9/10), model optimization (TFLite), and user-friendly deployment (Flutter) required in-depth technical expertise, resolution of conflicting constraints (accuracy vs. speed), and consideration for real-world usage in rural farming environments. The following subsections provide detailed mappings to problem-solving categories, knowledge profiles, and engineering activities, demonstrating the depth and interdisciplinary nature of this research.

5.4.1 Complex Problem Solving

To successfully complete this project, multiple aspects of complex problem-solving were involved—from selecting the right model architecture to ensuring optimized performance on mobile devices under practical conditions. The mappings below represent how the work aligns with various Engineering Problem (EP) attributes.

Table 5.2: Mapping with complex problem solving.

EP1	EP2	EP3	EP4	EP5	EP6	EP7
✓		✓	✓			✓

Justifications:

- EP1 – Depth of Knowledge: This project required in-depth technical knowledge across multiple domains, including deep learning-based object detection (YOLOv8, YOLOv9, YOLOv10), image annotation, model evaluation using mAP metrics, and mobile application development using Flutter. Additionally, optimization through TensorFlow Lite for real-time mobile deployment demanded solid understanding of both machine learning

and software engineering fundamentals.

- EP3 – Depth of Analysis: A rigorous comparative analysis of the YOLO models was conducted using detailed performance metrics such as mAP50, mAP50–95, GFLOPs, and inference speed. These evaluations were critical in identifying the most efficient model for mobile deployment and required the ability to interpret and apply advanced performance benchmarking techniques.
- EP4 – Familiarity of Issues: The project addressed several well-known engineering problems, such as handling imbalanced datasets, optimizing large models for mobile devices, and ensuring application responsiveness across various Android smartphones. These challenges are commonly encountered in real-world ML deployment tasks and were managed using skills developed through academic training and previous research experience.
- EP7 – Interdependence: The system was designed with modularity and scalability in mind, enabling future integration with additional crop datasets or IoT-based smart farming solutions. This interdependency across AI model development, mobile software, and agricultural application demonstrates a broader system-level perspective beyond isolated problem-solving.

Mapping with Knowledge Profile for EP1

Table 5.3 maps the Depth of Knowledge (EP1) to the Knowledge Profile categories. It illustrates the application of engineering fundamentals, advanced techniques, and research literature in the project.

Table 5.3: Mapping with knowledge Profile.

K3	K4	K5	K6	K8
✓		✓		✓

Justifications:

- **K3 – Engineering Fundamentals:** Core engineering concepts in machine learning, object detection, data preprocessing, and algorithmic optimization were fundamental to the development of the system. This included applying mathematical principles in evaluating model performance, resizing and normalizing images, and preparing annotated data suitable for training object detection models like YOLO.
- **K5 – Engineering Design:** The project involved the complete design of a

practical and user-oriented system—from selecting the optimal YOLO model to converting it into TensorFlow Lite for real-time deployment. Careful design considerations were also made for the mobile application interface using Flutter to ensure usability, performance, and cross-device compatibility.

- **K8 – Research Literature:** A thorough literature review guided architectural decisions and validated the research gap. Existing works on plant disease detection using earlier YOLO versions and CNNs helped inform model selection, while performance benchmarks from prior studies were used to set evaluation criteria. This ensured that the system design was grounded in current, evidence-based practices.

5.4.2 Engineering Activities

The development and deployment of this system involved a wide range of engineering activities across domains such as AI model training, software engineering, optimization for edge computing, and mobile UI/UX development. The following mapping shows how the project aligns with Complex Engineering Activities (EA) standards.

Table 5.4: Mapping with complex engineering activities.

EA1	EA2	EA3	EA4	EA5
✓		✓	✓	✓

Justifications:

- EA1 – Range of Resources: The project utilized a wide array of tools and platforms, including Google Colab for cloud-based model training, TensorFlow and TensorFlow Lite for model development and optimization, Flutter and Dart for mobile app development, and Android Studio for testing and deployment.
- EA3 – Innovation: The integration of advanced YOLO models (YOLOv8, YOLOv9, YOLOv10) into a mobile-optimized, offline-capable disease detection system represents a novel application in the field of precision agriculture.
- EA4 – Societal & Environmental Consequences: The mobile application supports early disease detection, enabling farmers to apply pesticides only when necessary. This promotes environmentally responsible farming and helps reduce chemical overuse.

- EA5 – Familiarity: The technical tasks undertaken in this project—model training, dataset annotation, TFLite conversion, and mobile app development—were grounded in knowledge gained from academic coursework and prior research activities. This familiarity with relevant technologies enabled the team to efficiently navigate technical challenges and ensure successful project delivery.

5.5 Summary

This chapter presented the engineering considerations, defined standards, and its challenges of designing an YOLO-enabled chilli leaf disease detector and deploying it as a mobile application. The technique incorporates deep learning, software engineering, and mobile development all of which require thought on planning, optimization, and multidisciplinary skillset.

Implementation of ensured software compliance was conducted through the use of recommended Flutter practices and open-source tools (including Google Colab in the model-training) and following structured Android Studio practices for deployment. In order to fulfill hardware requirements, the team conducted in-depth Android device device testing and used TensorFlow Lite to enable robust, lightweight deployment in the real world. To guarantee efficient and secure communication, an offline inference process was used between the mobile app and the model to protect privacy and enhance performance.

This system promotes improved farmers' livelihoods, sustainable agriculture and conforms to ethical criteria for technology deployment based on the impact analysis. It facilitates rural regions with instant, AI- driven solutions for determining early diseases and maximized care for crops.

The project management and financials of the research were undertaken efficiently with minimum resources. Through tight task distribution management, systematic tracking progress, and using easily available means, we managed to implement cost-effectively.

Chapter 6

Conclusion

Chapter 6 concludes the thesis by presenting a comprehensive summary of the research outcomes, acknowledging current limitations, and outlining directions for future improvement. It highlights the successful implementation of a lightweight, real-time chilli leaf disease detection system using YOLOv8s, YOLOv9s, and YOLOv10s models, with YOLOv10s selected for deployment based on its superior accuracy and efficiency. The chosen model was optimized using TensorFlow Lite and integrated into a Flutter-based Android application, enabling offline, on-device disease detection for farmers. The chapter also addresses key limitations such as limited dataset diversity, device compatibility issues, and the lack of multilingual support. To overcome these, several future directions are proposed, including expanding the system for multi-crop disease detection, improving data collection across different environmental conditions, and enhancing user accessibility through language localization and interface improvements. Overall, the chapter reinforces the practical significance of the research by bridging deep learning and real-world agriculture through a scalable, mobile-driven solution.

6.1 Summary

The primary objective of this research was to design, develop, and evaluate a real-time, mobile-compatible chilli leaf disease detection system using state-of-the-art object detection models from the YOLO (You Only Look Once) family. Recognizing the growing threat of chilli leaf diseases to crop yield and farmer income—especially in rural and underserved communities, this study aimed to bridge the gap between high-performance machine learning models and practical deployment in real-world agricultural environments.

The research began with a comprehensive literature review, which revealed that while many studies explored chilli disease detection using CNNs and early YOLO models (e.g., YOLOv4–v7), there was limited focus on the latest YOLO versions (YOLOv8, YOLOv9, and YOLOv10), lightweight optimization, and actual mobile deployment. These gaps formed the foundation of research motivation.

The methodology involved collecting and annotating a dataset of chilli leaf images, training multiple YOLO models (YOLOv8s, YOLOv9s, YOLOv10s), evaluating their performance using standard metrics (mAP, precision, recall, inference time), and selecting the most suitable model for mobile use. The best-performing model was then converted to TensorFlow Lite (TFLite) to reduce computational overhead and integrated into a Flutter-based Android mobile application. This mobile app was developed and tested using Android Studio to ensure compatibility, responsiveness, and offline performance.

Farmers can remotely use smartphones to collect chilli leaf images and in return get real-time disease diagnosis feedback in the app without the need for internet or expensive hardware. The developed mobile app is easy to use and efficient and offers accurate predictions allowing farmers to respond quickly, reducing pesticide overuse. Research indicates that deep learning models such as YOLO can be easily implemented in a practical, real time and efficient system for smart agriculture. It feeds into the aim of precision farming by offering a low cost, sustainable and scalable solution to one of agriculture's recurring problems.

6.2 Limitation

Although achieving its key objectives, the study faced a number of constraints that may affect its system's ability to generalize and adapt to new environments:

- Dataset Constraints: It is beneficial that the dataset is diverse and annotated but still it may not represent all disease forms, lighting circumstances, or natural leaf orients. The performance of the model would probably benefit if the dataset was extended to include regional as well as seasonal dissimilarities.
- Model Scalability: Performance on devices which have limited memory or old hardware cannot be guaranteed despite the model's success in Android smartphones. Turning to techniques such as pruning combined with training quantization-awarely may substantially enhance the system's scalability.
- Disease Overlap and Similarity: The confusion of the model is compounded by the fact that in some instances, there are chilli leaf diseases which have overlapping visual indicators, at times when the lighting is poor or when only a part of the leaf is available for viewing. Such overlaps may undermine the

model accuracy and completeness in certain environments.

- Lack of Interpretability: Though the model provides rapid predictions, its current implementation does not include Explainable AI (XAI), such as Grad-CAM, thus making it difficult for users to comprehend the prediction method.
- Language and Accessibility Features: Farmers using languages other than English, or those with limited ability to read, may have difficulty using the app as it is currently only available in English. Perhaps the inclusion of language-specific features and speech help could drive adoption among non english speaking farmers.

6.3 Future Work

To further enhance the effectiveness, usability, and reach of the system, several avenues for future development are identified:

- **Dataset Expansion and Collaboration:** Future versions of the system should include a larger, more diverse dataset collected across different regions and seasons. Collaborating with agricultural institutes and extension programs can accelerate this process.
- **Multi-Crop and Multi-Disease Support:** The application can be extended to support other crops such as tomatoes, eggplants, and cucumbers, each with their unique disease profiles. A multi-crop model would increase the tool's usefulness across farming communities.
- **Integration of Explainable AI (XAI):** Incorporating visual explanation methods such as Grad-CAM, LIME, or SHAP can help users understand what part of the leaf image contributed to the diagnosis, thus improving user trust and transparency.
- **IoT Integration and Smart Farming:** The app can be enhanced to support integration with IoT sensors (soil moisture, temperature, humidity) to provide more context-aware recommendations, aligning with the future vision of intelligent farming systems.
- **User Interface Localization and Accessibility:** Adding support for local languages (e.g., Bangla, Hindi) and offline voice assistance will make the system more accessible to users in remote areas, regardless of their literacy.

References

[1] Manoj, R., Girish, S., Pushpa, B. R., Rani, N. S., & Sangamesha, D. (2024, July). Early-Stage Disease Prediction in Chilli Plant Using YOLO Models. In 2024 Second International Conference on Advances in Information Technology (ICAIT) (Vol. 1, pp. 1-9). IEEE.

[2] [hapiee, M. N. A., Abdul Manan, A. A., Mohd Razman, M. A., Mohd Khairuddin, I., & PP Abdul Majeed, A. (2022). Chili plant classification using transfer learning models through object detection. In Enabling Industry 4.0 through Advances in Mechatronics: Selected Articles from iM3F 2021, Malaysia (pp. 541-551). Singapore: Springer Nature Singapore.

[3] Kantharaju, K. (2023, April). Chili Leaves Disease Identification Using Artificial Neural Network Algorithms. In XVIII International Conference on Data Science and Intelligent Analysis of Information (pp. 865-875). Cham: Springer Nature Switzerland.

[4] Li, D., Zhang, C., Li, J., Li, M., Huang, M., & Tang, Y. (2024). MCCM: multi-scale feature extraction network for disease classification and recognition of chili leaves. Frontiers in Plant Science, 15, 1367738.

[5] Naik, B. N., Ramanathan, M., & Ponnusamy, P. (2023). Refined single-stage object detection deep-learning technique for chilli leaf disease detection. Journal of Electronic Imaging, 32(3), 033039-033039.

[6] Chen, W., Gao, H., Ding, D., Dong, X., & Luo, X. (2023, May). Chili pepper pests recognition based on hsv color space and convolutional neural networks. In 2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI) (pp. 241-245). IEEE.

[7] Bademiyya, S. I., & Ashtaputre, S. (2019). Estimation of yield loss due to powdery mildew of chilli caused by Leveillula taurica (Lev.) Arn. Int. J. Pure App. Biosci, 7(1), 323-326.

[8] Zehra, S. B., Ahmad, A., Sharma, A., Sofi, S., Lateef, A., Bashir, Z., ... & Rathore, J. P. (2017). Chilli leaf curl virus an emerging threat to chilli in India. Int. J. Pure Appl. Biosci, 5, 404-414.

[9] Solahudin, M., Pramudya, B., & Manaf, R. (2015). Gemini virus attack analysis in field of chili (Capsicum annuum L.) using aerial photography and Bayesian segmentation method. Procedia Environmental Sciences, 24, 254-257.

[10] Kanaparthi, K. R., & Ilango, S. S. (2023). A survey on training issues in chili leaf diseases identification using deep learning techniques. Procedia Computer Science, 218, 2123-2132.

[11] Sambrani, Y., & Bhairannawar, S. (2023, June). Chili disease detection and classification using various machine learning techniques. In 2023 International Conference on Applied Intelligence and Sustainable Computing (ICAISC) (pp. 1-6). IEEE.

[12] Karna, N., Kelikualiq, R., Aditya, B., Putra, M. A. P., Rahyuni, D., & Hertiana, S. N. (2024, November). Health Monitoring System in Smart Greenbox for Chili Plant Using Convolutional Neural Network. In 2024 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS) (pp. 38-43). IEEE.

[13] Sulistyaningrum, D. R., Mubarok, M. K. N., & Setiyono, B. (2024, October). Image Captioning on Diseased Chili Plants Using ResNeXt50 and Transformer. In 2024 IEEE International Conference on Data and Software Engineering (ICoDSE) (pp. 149-154). IEEE.

[14] Bandopadhyay, S., Gaula, A. K., Haider, I., & Kumar, G. (2023, December). Chili-Net: An Approach for Classifying Chili Leaf Diseases Using Deep Neural Networks. In Machine Intelligence and Digital Interaction Conference (pp. 45-55). Cham: Springer Nature Switzerland.

[15] Husin, Z. B., Shakaff, A. Y. B. M., Aziz, A. H. B. A., & Farook, R. B. S. M. (2012, February). Feasibility study on plant chili disease detection using image processing techniques. In 2012 Third International Conference on Intelligent Systems Modelling and Simulation (pp. 291-296). IEEE.

[16] Das Chagas Silva Araujo, S., Malemath, V. S., & Sundaram, K. M. (2021). Symptom-based identification of G-4 chili leaf diseases based on rotation invariant. Frontiers in Robotics and AI, 8, 650134.

[17] Sahuri, G. (2020). Implementation of Deep Learning Methods in Detecting Disease on Chili Leaf. International Journal of Advanced Studies in Computers, Science and Engineering, 9(6), 10-15.

[18] Araujo, S. D. C. S. E., Malemathh, V. S., & Sundaram, K. M. (2021, December). Affine Invariant approach for disease detection on chili plant. In 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET) (pp. 1-6). IEEE.

[19] Sari, Y., Baskara, A. R., & Wahyuni, R. (2021, November). Classification of Chili Leaf Disease Using the Gray Level Co-occurrence Matrix (GLCM) and the Support Vector Machine (SVM) Methods. In 2021 Sixth International Conference on Informatics and Computing (ICIC) (pp. 1-4). IEEE.

[20] Pratap, V. K., & Kumar, N. S. (2023). High-precision multiclass classification of chili leaf disease through customized EffecientNetB4 from chili leaf images. Smart Agricultural Technology, 5, 100295.

203-15-14537

ORIGINALITY REPORT

15%	10%	8%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	5%
2	"Digital Interaction and Machine Intelligence", Springer Science and Business Media LLC, 2024 Publication	1%
3	B. Nageswararao Naik, R. Malmathanraj, P. Palanisamy. "Detection and classification of chilli leaf disease using a squeeze-and-excitation-based CNN model", Ecological Informatics, 2022 Publication	1%
4	Submitted to United International University Student Paper	<1%
5	"Advanced Computing", Springer Science and Business Media LLC, 2024 Publication	<1%
6	researchoutput.csu.edu.au Internet Source	<1%
7	dspace.daffodilvarsity.edu.bd:8080 Internet Source	<1%
8	www.coursehero.com Internet Source	<1%
9	Dan Li, Chao Zhang, Jinguang Li, Mingliang Li, Michael Huang, You Tang. "MCCM: multi-scale feature extraction network for disease classification and recognition of chili leaves", Frontiers in Plant Science, 2024 Publication	<1%
10	Aminuddin, Nuramin Fitri Bin. "Development of Chilli Leaf Disease Identification Using Convolutional Neural Network", Universiti Tun Hussein Onn (Malaysia), 2024 Publication	<1%
11	Toni Kusnandar, Judhi Santoso, Kridanto Surendro. "Enhancing Color Selection in HSV Color Space", Ingénierie des systèmes d information, 2024 Publication	<1%
12	www.preprints.org	<1%