Explainable AI (XAI) driven Mango leaf disease detection using State-of-the-art convolutional neural network and Tiny-Net

By
Aunirudra Dey Anu
ID: 212-15-4170
Karina Chakraborty
ID: 212-15-4166

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. MD Taimur Ahad
Associate Professor
Department of Computer Science and Engineering, Daffodil International University

Co-Supervised by

Md. Shahriar Shakil
Lecturer
Department of Computer Science and Engineering, Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY
Dhaka, Bangladesh

May 14, 2025

APPROVAL

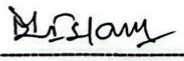
This Project titled **"Explainable AI (XAI) driven Mango leaf disease detection using State-of-the-art convolutional neural network and Tiny-Net"**, submitted by Aunirudra Dey Anu, ID No: 212-15-4170 and Karina Chakraborty, ID No: 212-15-4166 to the Department of Computer Science and Engineering, Daffodil 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 — **Chairman**
Associate Professor and Associate Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Mr. Md. Ali Hossain — **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Mr. Amir Sohel — **Internal Examiner**
Senior Lecturer
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University



Dr. Md. Manowarul Islam — **External Examiner**
Associate Professor
Department of Computer Science and Engineering
Jagannath University

DECLARATION

We hereby declare that this project has been done by us under the supervision of **Dr. MD Taimur Ahad, Associate Professor**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

Dr. MD Taimur Ahad
Associate Professor
Department of Computer Science and Engineering,Daffodil International University

Co-Supervised by:

Md. Shahriar Shakil
Lecturer
Department of Computer Science and Engineering,Daffodil International University

Submitted by:

Aunirudra Dey Anu
Student ID: 212-15-4170
Department of Computer Science and Engineering,Daffodil International University

Karina

Karina Chakraborty
Student ID: 212-15-4166
Department of Computer Science and Engineering,Daffodil University

©Daffodil International University ii

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. MD Taimur Ahad**, **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 them at 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

Rising agricultural requirements for intelligent, accessible solutions demonstrate why efficient plant disease detection systems have become vital. A deep learning framework has been developed to automatically detect mango leaf diseases while providing mobile access through an explainable framework solution. Six state-of-the-art convolutional neural network (CNN) models underwent initial evaluation on the MLD24 dataset, with DenseNet101 producing the most precise classification results of 99.06%. After researchers realised that traditional models were unsuitable for practical deployment due to their computational constraints, a custom lightweight model called Tiny-Net was developed. The Tiny-Net model delivered performance like standard models, achieving 99.61% accuracy while using minimal system resources, thus meeting the needs of mobile applications. The Explainable AI (XAI) methods Grad-CAM and LIME and SHAP provided transparent analysis of model decisions to increase system transparency and foster trust. The TensorFlow Lite version of the final model received implementation within a Flutter-based mobile application which delivered real-time offline detection capabilities accessible to farmers in regions with low connectivity. The system proved resistant to failure while demonstrating ease of use through multiple evaluations that proved its real-world utility. Through this research researchers demonstrated progress in automated plant disease detection technologies while using human-centered artificial intelligence deployments to benefit sustainable farming practices. The proposed system proves its ability to enhance farmers' disease diagnosing capabilities through real-time accurate diagnosis while closing the gap between controlled laboratory models and realistic usage conditions.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the background, motivation, objectives, and methodology of the proposed mango leaf disease detection system using deep learning and mobile application deployment.

1.1 Introduction

Mangoes are among the most popular and economically essential fruits in the world. However, mangoes are vulnerable to diseases that can significantly reduce their quality and yield [1]. In ecological informatics, tackling these challenges usually necessitates integrating cutting-edge technologies and ecological insights for effective disease management [1]. Many pathological problems, severe illnesses, and pesticide use significantly reduce mango crop yields. Mango leaf diseases have a significant influence on mango quality and output [2].

As suggested by Shaik & Swamykan [2], diagnosing mango disease using leaves with the human eye is difficult. Therefore, developing a machine-learning model that can detect leaf diseases at early stages is very important. Deep learning (DL) has shown great promise in developing automated systems for leaf disease detection. This work proposes a novel DL approach that integrates the power of DL with ecological information management for the automated detection of mango leaf diseases [1]. From various deep learning techniques, CNNs have reformed computer vision tasks, like plant disease classification [3]. Due to their spatial and hierarchical ability to extract features from images, many authors choose CNN as a baseline model in the context of precision agriculture [5]. The existing studies also state that CNN-based models are giving promising results in identifying specific illnesses, which afterwards succeed in proper intervention and improving crop yield [6]. Traditional models of the CNN family, VGG16, DenseNet101, ResNet50, InceptionV3, Xception, and MobileNet have already been applied in existing studies to take precision agriculture to the next level [7]. While deploying the precision into the ONNX device, advanced Explainable AI

ensures human-understandable insights about the model's transparency [8]. Although the literature review shows the higher precision of CNNs in plant pathology, many studies still depend only on traditional models. However, this study feels the gap is like proposing new lightweight models as CNN models are sometimes not fit to implement for end-users, [9].

The authors therefore suggested that lightweight CNN architectures with relatively fewer layers could be good alternatives if they can also achieve a comparable level of performance in the detection and classification of diseased images. Inspired by the success of lightweight CNNs, several studies such as Thakur [19], Sun [16], Verma [18], Thaseentaj and Ilango [7], Rizvee [11], Utomo [9] developed lightweight CNN. As every study is meant to solve any existing problem, the authors also implemented real-time detection systems. Pathak [5], Patel [13], Puranik [14] implemented mobile applications to detect mango leaf disease for end-users and highlighted the importance of a real-time detection system. However, this study should focus on these objectives which must be a prerequisite for precision agriculture-related research.

1.2 Motivation

Though artificial intelligence along with image processing flourishes the research regarding plant disease classification day by day, practical and technical challenges remain debatable regarding mango leaf disease classification. Even though CNN-based models brought success in classifying mango leaf diseases, studies suffered from limited scope, datasets and resource constraints. Deficient focus like targeting only fungal or bacterial, cannot make precise decisions in real-world scenarios where leaves contain more than one disease (Shaik and Swamykan, [2]; Rizvee [11]). This highlights the necessity of building a more comprehensive classification model that covers a wider variety of mango leaf diseases. To address this, the present study utilizes a balanced and diverse dataset (MLD24), including eight classes of diseases and healthy samples, to enhance the model's practical utility and adaptability (Utomo [9]).

One more crucial difficulty of traditional CNN models studied by researchers is its generalizability. VGG16, DenseNet, and ResNet are widely used SOTA models that are popular for higher precision; however, these models cost high computational power, bigger in size and time complexities, making them unsuitable for deployment on mobile or edge devices (Thakur [19]; Fu [15]).

These deep models typically require powerful GPUs, which are inaccessible to most farmers, especially in low-resource rural settings. While some studies (e.g., Pathak [5]; Patel [13]) have attempted to bring their models to mobile platforms, such solutions are still rare and often not optimized for real-time field use. These concerns underscore the need for lightweight, efficient CNN models that are tailored for real-time use on mobile devices. This study proposes a custom Tiny-Net (T-Net) architecture that is both computationally efficient and capable of maintaining strong classification performance, with deployment intended through a mobile application (Utomo [9]; Puranik [14]).

Additionally, while deep learning models often report high-performance metrics like accuracy, precision, and recall, there is little emphasis on interpretability. Since CNNs are typically treated as black-box systems, their internal decision-making processes remain hidden—posing a major issue in critical applications like agriculture, where end-users require transparency and reasoning to trust the results. Few studies integrate Explainable AI (XAI) techniques, which are designed to help users understand why a model made a certain prediction, using visual or analytical explanations (Miah [12]; Hossain [3]). The lack of interpretability presents a barrier to user trust and real-world adoption of AI-based solutions in farming. To overcome this, the study integrates XAI techniques such as Grad-CAM, LIME, and SHAP, ensuring that the model's decisions are transparent and understandable, especially for non-technical users like farmers (Saleem [8]; Miah [12]).

1.3 Objectives

The primary objective of this research is to develop a robust and scalable deep learning framework for plant disease detection that addresses existing limitations in accuracy, scalability, interpretability, and data privacy. Specific objectives are:

- Find the best model among traditional Deep CNN architectures by combining the best features to evaluate the data, leverage the data biases, and improve the model's generalizability.
- Development of lightweight CNN architecture with comparable performance found in traditional CNN.
- Apply Explainable AI to ensure human-understandable insights about the model's transparency.

- An optimized and user-friendly application to detect diseases in real-time for end-users as a contribution to the study for the betterment of agriculture and harvesters.

1.4 Methodology

This study adopts a structured methodology aimed at proposing an accurate, interpretable, and deployable mango leaf disease detection system. It begins with the use of the MLD24 dataset consisting of 6,400 manually verified images across eight balanced classes, that included both bacterial and fungus-attacked diseased leaves collected from mango plants in Bangladesh. Each image was reviewed to ensure appropriate resolution and class labelling, providing a reliable foundation for training. To enhance model generalization and reduce overfitting, SMOTA technique is applied, as similarly adopted in studies by Rizvee [11] and Salamai [1].

Researchers applied performance metrics, including accuracy and precision alongside recall and F1-score and confusion matrices to evaluate six state-of-the-art CNN models—VGG16, DenseNet101, ResNet50, Inceptionv3, Xception, and MobileNet using individual training methods that followed approaches in Pathak [5] and Hossain [3]. The evaluation results led to the creation of a lightweight Tiny-Net (T-Net) model that meets mobile deployment needs for real-time operations according to Utomo [9] and Thakur [19].

The integration of XAI techniques, including Grad-CAM, LIME and SHAP within this work enables the production of visual and local explanations that help understand the model's prediction decisions (Miah [12]; Saleem [8]). Saleem [8]). The top-performing model received a conversion to ONNX and TensorFlow Lite formats to enable their integration into a Flutter-based mobile application that provides disease detection capability for field use independently of network connections (Patel [13]; Puranik [14]). Puranik [14]). The framework creates cohesive end-to-end systems which combine technical performance with real-world functionality to equip farmers with enhanced agricultural intelligence.

1.5 Project Outcome

This project should deliver a stable, interpretable mobile system that can precisely detect between eight mango leaf conditions while handling both diseased and healthy conditions. The dataset gets applied to six advanced CNN models for performance evaluation and study of classification outcomes and model behavior

against various leaf conditions.

Tiny-Net is one major result from this study, becoming a mobile-friendly, lightweight CNN architecture which delivers effective real-time disease identification capabilities on mobile screens. T-Net operates effectively with minimal computation demands, which enables better accessibility to farmers in addition to agricultural stakeholders operating in areas with limited connectivity or rural locations.

The project combines Explainable AI (XAI) techniques Grad-CAM, LIME and SHAP, to provide both reliable and transparent and interpretable system predictions. The system facilitates users to build trust because it explains its processes and enables them to make confident disease management decisions with the system. A Flutter-based mobile application presents the final deployment of this framework to offer practical and scalable contributions toward precision agriculture, which makes smart farming accessible for real-world users.

1.6 Organization of the Report

This thesis is organized into six chapters, each designed to guide the reader through the key components of the research, starting from the motivation behind the study to the development, implementation, and final outcomes of the proposed mango leaf disease detection system.

Chapter 1: Introduction

This chapter introduces the research problem, highlighting the importance of mango leaf disease detection in precision agriculture. It outlines the study's motivation, objectives, methodology, and expected outcomes. The chapter also explains the rationale behind using deep learning, SOTA CNN models, and explainable AI to build a reliable and user-friendly system for real-world agricultural use.

Chapter 2: Literature Review

This chapter provides a critical review of previous studies related to plant and mango leaf disease detection. It compares different deep learning techniques, discusses the limitations of existing models (such as high complexity, limited disease focus, or lack of explainability), and identifies the research gaps that this study aims to address. This review forms the foundation for the proposed

solution.

Chapter 3: Research Methodology
This chapter presents the step-by-step approach followed in this study. It covers dataset collection, image preprocessing, training of state-of-the-art CNN models, and the development of a custom lightweight model (T-Net). It also explains how Explainable AI techniques were applied and how the final model was prepared for mobile deployment.

Chapter 4: Implementation and Results
This chapter details the implementation process and presents the performance results of the trained models. It includes classification reports, confusion matrices, and validation curves for each CNN model. The effectiveness of the T-Net model is also evaluated. The outcomes are discussed in light of accuracy, efficiency, and real-world usability.

Chapter 5: Challenges and Design Considerations
This chapter highlights the challenges faced during the project, such as dataset limitations, model overfitting, and deployment issues. It also discusses important design decisions and compliance with engineering standards for mobile and AI applications. The chapter reflects on the ethical, environmental, and social implications of deploying such technology in agricultural settings.

Chapter 6: Conclusion and Future Work
The final chapter summarizes the key contributions of the research, emphasizing how the developed system can aid farmers in early and accurate disease detection. It also outlines the limitations of the current study and suggests future research directions, including expanding the model to other crops, improving scalability, and enhancing mobile app features.

This structure ensures a logical and smooth flow of ideas, making the report clear and comprehensive for readers. It allows a full understanding of how this research contributes to smart farming through the integration of deep learning, explainable AI, and real-time mobile technology.

Chapter 2

Background

This chapter is a review of previous studies of plant disease detection and focuses on deficiencies in current literature while setting the necessity of a lightweight, understandable approach.

2.1 Introduction

This chapter provides an overview of the existing literature using deep learning models in plant disease detection, with a particular focus on mango leaf disease classification. The review explores the progression of methods, ranging from traditional CNN-based models to more recent developments involving lightweight architecture, hybrid learning, and explainable AI techniques. Researchers have experimented with a variety of deep-learning models that have shown excellent potential in image-based disease classification, particularly for crops like mango, tomato, and grapevine. CNNs have been widely adopted due to their strong ability to learn spatial features from leaf images. However, challenges such as computational inefficiency, lack of interpretability, and limited disease class coverage continue to hinder their full potential in real-world applications. Additionally, while some studies have focused on high accuracy, they often overlook practical aspects such as mobile deployment, real-time usability, and farmer accessibility. This chapter critically reviews these contributions and identifies gaps that form the basis for the proposed system, which aims to combine accuracy, efficiency, and transparency in mango leaf disease detection through a novel integration of CNN models, hybrid methods, and explainable AI.

2.2 Literature Review

Salamai [1] presented a methodology of a visual modulation network that can innovatively learn the visual representations of leaf diseases along spatial and channel dimensions through simple but effective linear layers. An overlapped

patching embedding is presented to tackle the discontinuity of non-overlapped patching in vision transformers (ViTs). Then, a succession of visual modulator blocks is presented to learn compact and long-term representations of disease in mango leaves without the inclusion of attention operation. Experimental results on the MangoLeafDB dataset show that the proposed visual modulation network can accurately detect various mango leaf diseases, demonstrating its potential as a cost-effective and efficient tool for disease management in mango cultivation.

Shaik & Swamykan [2] study aims is to detect 13 different fungal and bacterial mango leaf diseases using CNN techniques. The photographs were captured in the Andhra Pradesh state's Chittoor district, which is home to India's largest mango-growing region. The dataset consists of 1100 images. The dataset was trained using popular CNN approaches, namely GooGLeNet, EfficientNet, and ResNet-50. The results indicate that the EfficientNet model achieved the highest accuracy of 98.7% in classifying mango leaf diseases. The study demonstrates the potential of deep learning models in identifying diseases affecting mango leaves, which could aid in timely disease detection and control.

Hossain [3] evaluated the performance of vision transformers in the identification of mango leaf diseases and compared them with popular CNNs. We proposed an optimized model based on a pre-trained Data-efficient Image Transformer (DeiT) architecture that achieves 99.75% accuracy, better than many popular CNNs, including SqueezeNet, ShuffleNet, EfficientNet, DenseNet121, and MobileNet. We also demonstrated that vision transformers can have a shorter training time than CNNs, as they require fewer epochs to achieve optimal results. We also proposed a mobile app that uses the model as a backend to identify mango leaf diseases in real time.

Prabu and Chelliah [4] utilized deep-learning and machine-learning algorithms for mango leaf disease classification. A dataset was prepared with 380 images of Mango Anthracnose, Bacterial Black Spot, and Sooty Mold collected from Andra Pradesh. Data augmentation was applied to the dataset to decrease the limitations of lower data and to enhance the model's generalizability. In the experiment, pre-trained CNN-based feature extraction was performed by freezing the classification layer and then MobileNetV2 and SVM were applied for classification and evaluation. Although the authors claimed this approach outperforms the traditional SOTA models, there is no result mentioned

specifically in terms of accuracy.

Pathak [5] explored vast models of the convolutional neural networks (CNNs) family for mango leaf disease classification accuracy. Addressing the lower accuracy gap, this study applied 20 CNN models by optimizing hyperparameters and developed a robust model with a 99% promising correctness. Before the experiment, advanced data augmentation was performed and with the help of image processing and feature extraction, this study claimed their result as reliable in terms of superior performance. Lastly, they proposed an android application as a practical perspective for the end-user named "Mango-SCN" which underscores the potential of CNN-based models.

Recognizing the economic and agricultural impact of mango leaf disease, Mehta [6] present a federated learning-based CNN model to detect and classify mango leaf diseases. The study chose a dataset with five different disease classes: Anthracnose, Powdery Mildew, Leaf Spot, Leaf Curl, and Healthy leaves. The study states the robustness of the model based on the accuracy that was between 97% and 98%.

Thaseentaj and Ilango [7] proposed a customized Deep CNN architecture to classify and refine the disease classification of mango leaves. The research aimed to outperform traditional CNN models in accuracy and computational efficiency. They selected a dataset of Anthracnose, Powdery Mildew, and Leaf Blight and applied preprocessing analysis to enhance image consistency. The system showed 93.34% accuracy, which surpassed the traditional CNN model's classification power concerning accuracy and real-time integration.

FrCNnet, a fully convolutional network (FCN) introduced by Saleem [8] is claimed as a customized approach to classify and precise segmentation of mango leaf disease detection. This is also a study unlike traditional CNN, to learn pixel-level features, and improve segmentation accuracy. Real-time data was collected from Mango Research Institute, Multan, Pakistan and SOTA models like VGG16, VGG19 and U-Net brought ultimate segmentation accuracy of 99.2% and 0.8% FN rate.

Keeping UN Sustainable Development Goal: 2 in mind, Utomo et al. [9] introduced mango leaf disease classification with a CNN-based model to surpass most contemporary models. Unlike MobileNetv2 and DeiT, this model offers less and more efficient performance with 739,080 parameters. They claimed that lightweight AI model integration in plant pathology can ensure the

sustainability of agricultural practices.

Knowing the mango production and disease threats on mango in Bangladesh, Ramadan [10] applied CNN and ViT model's architectures and a new level of data augmentation to enhance mango leaf disease classification. They highlighted the data augmentation technique: CycleGAN followed throughout the experiment and achieved 100% accuracy. Although this study applied both CNN and transformer-based pre-trained models to improve classification performance, it can be said that an overfitting issue has occurred.

Rizvee [11] proposed LeafNet, a customized model from CNN family to classify seven major mango leaf diseases. The model was trained on a region-specific dataset to ensure high adaptability to local disease patterns. The authors claimed that the proposed LeafNet achieved an average accuracy of 98.55% by outperforming AlexNet and VGG16 in precision, recall, and specificity.

Miah [12] demonstrated that Bangladesh requires automated systems for detecting diseases in mango leaves. The proposed system combined an EfficientNet model with a channel attention mechanism to optimize feature selection thus improving detection performance. The research analysis included 4,000 images distributed across 8 disease classes and evaluated the model through one standard train-validation-test split and five-fold cross-validation procedures. The reporting team achieved 98.93% accuracy in 5-fold cross-validation testing above the performance of both AlexNet and ResNet50. Among its features the study utilized explainable AI methods (LIME) to reveal how the model performs its decision-making operations.

The classification of mango leaf diseases functions through deep CNN models coupled with annotated image datasets according to Singh [20] who developed a cost-effective and scalable detection system for agriculture. Pratap and Kumar [21] applied CNNs that used transfer learning features from VGG-16 and EfficientNet to achieve improved classification accuracy and disease severity evaluation performance. Puviarasi [22] conducted a study which demonstrated CNNs delivered superior accuracy than fuzzy logic while reaching 95.2% precision. Saravanan [23] enhanced CNN detection capabilities through image augmentation to reduce overfitting and increase performance levels.

Fusion models and transfer learning models applied and developed by some studies like Ramya [24] used a ResNet50 transfer learning model, achieving 97% accuracy and reducing the need for manual disease diagnosis. Pratap and Kumar [21] utilized

YOLOv8 and MobileNet, demonstrating transfer learning's effectiveness in recognizing high and low-resolution images. Porna [25] introduced a hybrid model combining VGG19 and MobileNetB1 features with ResNet50V2 and EfficientNetB1 for perfect accuracy (100%), which leads to overfitting.

Ensemble techniques are also popular to merge the best results into one single model to detect or predict the best output and features extracted by freezing classification layers of CNN models and then classifying the features using a basic classification model also known as fusion models applied in some studies. Bezabh [26] developed an ensemble model combining GoogLeNet and VGG16. After preprocessing and feature extraction, the model achieved training, validation, and test accuracy of over 99%. This method has promising precision but as this model is based on ensemble models can be more weighted than traditional CNN models and can hamper real-time detection. Banerjee [27] extracted features using CNN and used a Support Vector Machine (SVM) for classification. Although their model reached 89.29% accuracy and reduced computational time, it was not practical for real-world deployment due to relatively low accuracy compared to modern deep learning models.

Table 2.1: Summary of Literature Reviewed.

Author & Year	Models Used	Accuracies	Contributions
Salamai (2023)	Visual Modulation Network, Vision Transformer (ViT)	High Accuracy (Exact % Not Provided)	Proposed a visual modulation network learning spatial and channel features in mango leaves. Introduced overlapped patching embedding to enhance ViT performance.
Shaik & Swamykan (2023)	GooGLeNet, EfficientNet, ResNet-50	98.7%	Developed CNN models for detecting 13 fungal and bacterial mango leaf diseases. EfficientNet performed best with 98.7% accuracy.
Hossain et al. (2024)	Data-efficient Image Transformer (DeiT), CNNs (SqueezeNet, ShuffleNet, EfficientNet, DenseNet121,	99.75%	Compared ViTs and CNNs for mango leaf disease detection. Proposed optimized DeiT model achieving 99.75% accuracy and demonstrated ViTs require fewer epochs.

	MobileNet)		
Prabu & Chelliah (2022)	MobileNetV2, SVM	Accuracy Not Mentioned	Utilized MobileNetV2 and SVM for classification on a dataset of 380 images. Applied data augmentation but results were not explicitly stated.
Pathak et al. (2024)	20 CNN Models (Hyperparameter Optimized)	99%	Explored 20 CNN models with hyperparameter tuning, achieving 99% accuracy. Developed a mobile app 'Mango-SCN' for real-world deployment.
Mehta et al. (2023)	Federated Learning-based CNN Model	97%-98%	Proposed a federated learning-based CNN model, achieving 97%-98% accuracy. Demonstrated robustness across five disease classes.
Thaseentaj & Ilango (2023)	Custom Deep CNN	93.34%	Developed a custom Deep CNN model achieving 93.34% accuracy. Focused on computational efficiency and real-time applicability.
Saleem et al. (2021)	Fully Convolutional Network (FCN) - FrCNnet, VGG16, VGG19, U-Net	99.2%	Introduced FrCNnet, a fully convolutional network for precise segmentation. Achieved 99.2% accuracy with a 0.8% false negative rate.
Utomo et al. (2024)	Lightweight CNN Model (739,080 parameters)	Efficient Performance (Exact % Not Provided)	Developed a lightweight CNN model with only 739,080 parameters to ensure sustainability and efficiency in agricultural AI.
Ramadan et al. (2023)	CycleGAN-based Data Augmentation, CNN, ViT	100% (Potential Overfitting)	Applied CycleGAN for advanced data augmentation. Achieved 100% accuracy but raised concerns about potential overfitting.
Rizvee et al. (2023)	LeafNet (Customized CNN)	98.55%	Proposed LeafNet, a CNN-based model achieving 98.55% accuracy. Outperformed AlexNet and VGG16 in precision, recall, and specificity.
Miah et al.	EfficientNet with Channel	98.88% (Single Split),	Developed an EfficientNet model with a channel attention module

(2024)	Attention Module, LIME (XAI)	98.93% (5-Fold Cross-Validation)	for improved classification. Achieved 98.88%-98.93% accuracy. Used LIME for Explainable AI.
Singh et al. (2023)	Deep Neural Networks (CNN)	Not specified	Scalable ML-based plant disease detection for precision agriculture.
Pratap & Kumar (2024)	CNN with Transfer Learning (VGG-16, MobileNet, YOLOv8, etc.)	Not specified	Transfer learning-based CNN for mango disease severity classification.
Puviarasi (2022)	CNN vs. Fuzzy Mean Algorithm	CNN: 95.2%, Fuzzy: 93.5%	Comparative analysis proving CNN's superiority in leaf disease classification.
Saravanan et al. (2023)	CNN with image augmentation	Not specified	Optimized CNN architecture to reduce overfitting and improve accuracy.
Ramya et al. (2024)	ResNet50 with Transfer Learning	97%	Efficient automated disease diagnosis using deep learning and hyperparameter tuning.
Porna et al. (2024)	Hybrid: VGG19, MobileNetB1 + ResNet50V2, EfficientNetB1	100%	Hybrid CNN with data augmentation and tuning for perfect classification performance.
Bezabh et al. (2024)	Ensemble: GoogLeNet + VGG16	Train: 99.87%, Val: 99.72%, Test: 99.21%	Ensemble model with advanced preprocessing and segmentation for near-perfect results.
Banerjee et al. (2023)	CNN for feature extraction + SVM for classification	89.29%	CNN-SVM hybrid with low complexity but limited real-world accuracy.

2.2.1 Similar Applications

In recent years, several studies have explored the use of deep learning models in detecting and classifying plant diseases, particularly those affecting mango leaves. Researchers have applied a range of CNN architectures—such as EfficientNet, ResNet, and MobileNet—to classify various mango leaf diseases with promising results. For example, Shaik and Swamykan [2] used CNN-based models like GooGLeNet and EfficientNet to classify 13 types of fungal and bacterial mango leaf diseases, achieving an accuracy of 98.7%, which demonstrates the strength of deep learning in this domain. Hossain [3] took a comparative approach by evaluating Vision Transformers (ViTs) and CNNs, proposing an optimized DeiT model that surpassed several CNN benchmarks, including SqueezeNet and DenseNet121, achieving a high accuracy of 99.75%. Similarly, Rizvee [11] introduced a custom CNN model called LeafNet for seven major mango leaf diseases and outperformed standard models like AlexNet and VGG16. Some studies also integrated lightweight or mobile-friendly models to improve usability, such as the work by Utomo [9], who designed a CNN model with only 739,080 parameters, making it suitable for deployment on edge devices. Beyond mango leaves, similar deep-learning applications have been used in other crops, indicating a broader trend of adopting AI in plant pathology. However, while these studies achieved high accuracy, many lacked interpretability, generalizability across multiple disease types, or real-time mobile integration, which this research aims to address through ensemble, hybrid learning, lightweight modelling, and explainable AI.

2.2.2 Related Research

Several recent studies have focused on improving mango leaf disease detection using deep learning and machine learning models, offering valuable insights and direction for current research. For example, Pathak [5] conducted a comparative analysis of 20 CNN models, optimizing hyperparameters to develop a reliable classification system that achieved 99% accuracy and was further extended into a mobile application named Mango-SCN. Similarly, Mehta [6] introduced a federated learning-based CNN model to address privacy concerns in plant disease data sharing, reaching accuracies between 97% and 98% across five disease classes. In another notable study, Miah [12] proposed an EfficientNet model integrated with a channel attention module, improving feature extraction and achieving over 98% accuracy in both single split and cross-validation

settings. These efforts reflect a shared aim of enhancing classification performance while addressing practical challenges such as limited datasets and real-time application. Furthermore, Saleem [8] introduced FrCNnet, a fully convolutional network capable of both segmentation and classification, achieving 99.2% accuracy using real-time data from Pakistan's Mango Research Institute. While these models deliver high performance, many lack interpretability or are too computationally heavy for mobile use. This emphasizes the need for a system like the one proposed in this study—combining deep learning with Explainable AI (XAI) and lightweight architecture for both accuracy and usability.

2.3 Gap Analysis

From the literature review study, the existing studies have several gaps. Over the last decade, the use of machine learning in smart agriculture has surged in popularity. As a result, researchers used several computer-aided and machine-learning approaches to classify mango leaf diseases. According to reports, these procedures have various downsides. The study below will highlight those points and explain the gaps.

- In existing research, they only focused on 3 to 4 diseases, some researchers only focused on fungal diseases and some researchers only focused on bacterial diseases [2].
- However, the adoption of vision transformers in agriculture is still in its infancy. Though State-of-the-art (SOTA) CNNs have proven the capabilities of cancerous cell detection and classification, SOTA CNNs inherit some limitations.
- Usually, these CNNs have numerous layers, such as convolutional, pooling, concatenation layer, and fully connected and, hyperparameters. Thakur [19], therefore criticized that deep CNN architectures have a large memory footprint and high computational demand. Fu [15] also supported Thakur [19].
- Moreover, the complex and intricate network architectures of CNNS have limited limitations in implementing applied artificial intelligence. The usability is limited in mobile devices.

2.4 Summary

This chapter has provided a comprehensive overview of the current landscape in mango leaf disease detection using deep learning approaches. It began by highlighting similar applications, demonstrating how CNNs, Vision Transformers, and lightweight architectures have been employed in various studies to identify plant diseases with high accuracy. The chapter then explored related research efforts, showcasing different models and techniques that have advanced the field, while also pointing out common limitations such as lack of model interpretability, computational inefficiency, and narrow disease coverage. The gap analysis shows that existing models perform well in controlled environments, they often fall short in real-time, mobile-friendly deployment and transparency in predictions.

Chapter 3

Research Methodology

This chapter explains the dataset, preprocessing methods, architectures of models, strategies for training, explainable AI techniques, and integration of a mobile application into the system development.

3.1 Methodology/Requirement Analysis & Design Specification

3.1.1 Overview

A detailed methodology explains the creation of an efficient, practical mango leaf disease detection system through deep learning techniques in this chapter. Our focus is on creating an effective solution which combines precise outcomes with operational effectiveness for real-life agricultural conditions, including farmer end-users. Several sequential stages mark the research evolution that includes data collection and preprocessing, and model training combined with evaluation of six advanced state-of-the-art Convolutional Neural Network (CNN) models. To overcome existing mobile-based systems' performance limitations, a lightweight CNN model known as Tiny-Net (T-Net) has been specifically designed for this application. The methodology has several stages specifically built to support real-world conditions, which previous research noted as challenging, mainly due to restricted device access and inadequate model transparency and the need for quick responses. The system operates with transparency by applying XAI techniques, including Grad-CAM, LIME, and SHAP which enable users to see and believe in model predictions. The best-performing model from the methodology becomes part of a mobile application built with Flutter for providing users with an easy-to-use, lightweight tool which functions offline to detect early diseases in mango leaves. The chapter details step-by-step instructions for the experimental setup implementation, starting with data management up with the deployment phase.

3.1.2 Proposed Methodology/ System Design

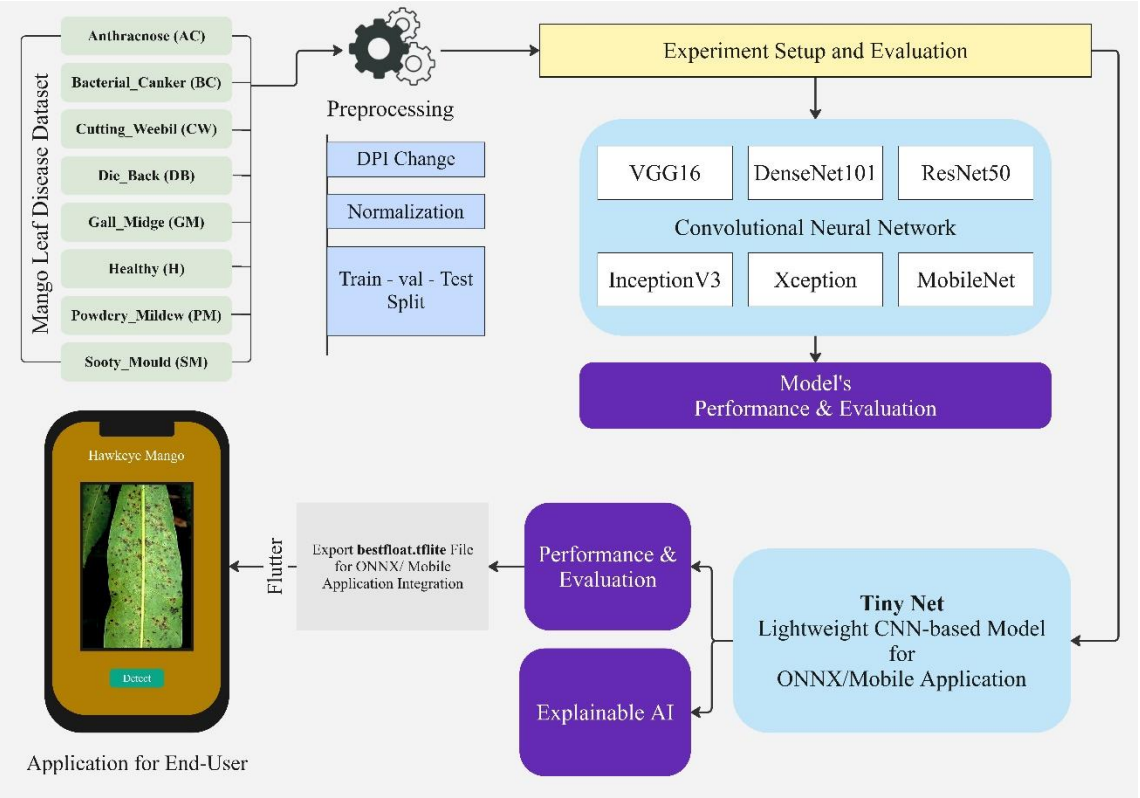


Figure 3.1: The Methodological procedure

This study employs the deep learning methodology presented in Figure 3.1 for detecting and classifying different mango leaf diseases. The MLD24 data collection [28] serves as the starting point for the research method, which contains pictures of healthy alongside diseased mango leaves distributed across eight distinct classes. The images require several preprocessing steps that employ multiple techniques to both enhance data quality and boost model generality. Following preprocessing, six state-of-the-art Convolutional Neural Network (CNN) models—VGG16, DenseNet101, ResNet50, InceptionV3, Xception, and MobileNet—are individually trained and evaluated based on their classification performance. To overcome the limitations of traditional CNNs in terms of model size and computational load, a lightweight CNN architecture named Tiny-Net (T-Net) is then proposed and tested. Both the standard and proposed models are analyzed using performance metrics and further examined through Explainable AI techniques like Grad-CAM, LIME, and SHAP to ensure interpretability and model transparency. Finally, the best-performing model is exported as a TFLite file and deployed in a Flutter-based mobile application,

enabling real-time, offline disease detection for end-users, particularly farmers.

3.1.3 Functional and Nonfunctional Requirements

1. Functional Requirements

- **Image Input Handling:** The system should allow users to upload or capture mango leaf images using their mobile devices. The images must be in a supported format (e.g., .jpg) and conform to the required resolution for consistent performance.
- **Disease Detection and Classification:** Upon receiving the image, the system should process it through the trained deep learning model and classify it into one of the eight predefined categories—seven disease types and one healthy class.
- **Prediction Output:** The system should display the predicted disease class on the mobile interface, along with a confidence score to help users understand the reliability of the result.
- **Offline Functionality:** The application should be able to perform predictions without the need for an internet connection, using the integrated TFLite model for on-device inference.

2. Nonfunctional Requirements

- **Performance Efficiency:** The model should deliver fast inference times, ideally under a few seconds, to ensure a smooth user experience even on low-end mobile devices.
- **Lightweight Design:** The deployed model (Tiny-Net) must be optimized for minimal memory and storage usage, making it suitable for mobile environments with limited resources.
- **User-Friendly Interface:** The application must have a simple, intuitive design so that users without technical backgrounds, such as farmers, can easily operate it.
- **Accuracy and Reliability:** The classification model must maintain a high level of accuracy and consistency in predictions across various lighting conditions and leaf appearances.

3.1.4 Context Diagram

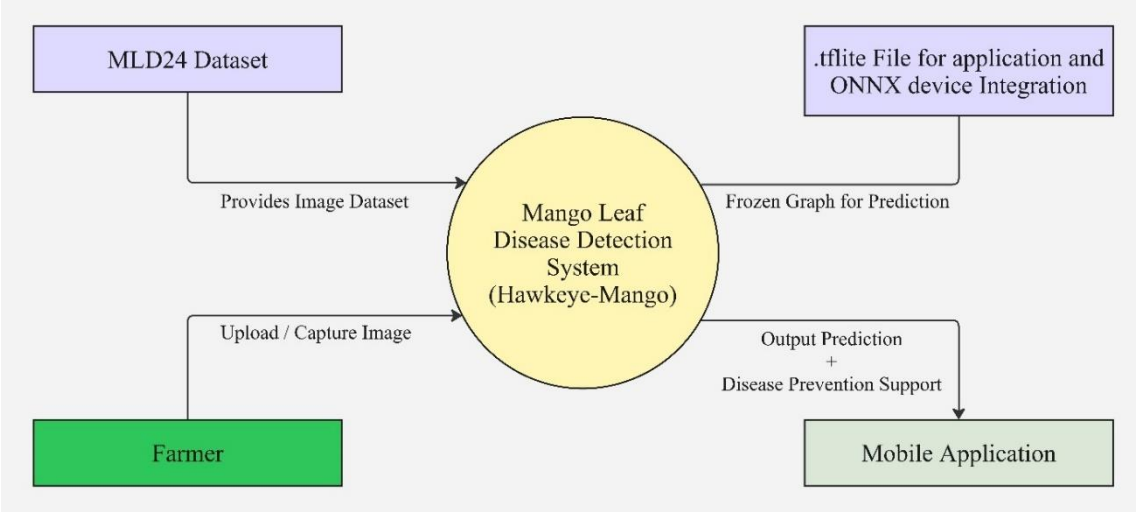


Figure 3.2: Proposed Context Diagram

The context diagram presented in Figure 3.3 illustrates the overall interaction between external entities and the Mango Leaf Disease Detection System, titled Hawkeye-Mango. The system receives labeled image data from the MLD24 dataset to facilitate model training and evaluation. Farmers act as primary users by uploading or capturing mango leaf images through the mobile application, requesting disease predictions. After processing the input images, the system outputs the classified disease information along with prevention support recommendations through the mobile interface. Additionally, the system exports the optimized Tiny-Net model in TFLite format for seamless integration into mobile and ONNX-supported devices. This diagram clearly highlights how external entities interact with the system boundary, focusing on efficient input, processing, and output flow to deliver an end-to-end precision agriculture solution.

3.1.5 Data Flow Diagram Level 1

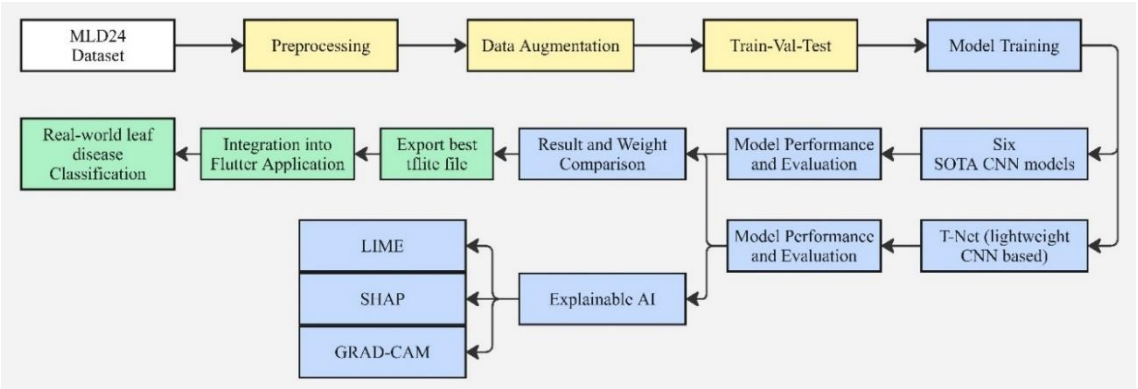


Figure 3.3: Proposed Data Flow Diagram

Figure 3.2 represents the Level 1 Data Flow Diagram (DFD) that outlines the sequential flow of processes involved in the mango leaf disease detection system. The process begins with the input of the MLD24 dataset, which contains eight balanced classes of mango leaf images. These images undergo preprocessing to enhance quality and variability, followed by a structured train-validation-test split. The training process is conducted separately on six state-of-the-art CNN models as well as the proposed Tiny-Net (T-Net), a lightweight CNN model. The assessment of each model performs standard metrics for performance evaluation. The research team tests all models against each other to determine the top-performing solution that gets converted into a TFLite file for integration into a Flutter-based mobile application. The combined systems enable real-time disease detection in outdoor situations and work continuously offline. The system implements Explainable AI (XAI) methods Grad-CAM, SHAP, and LIME, for parameterized prediction outcomes and user trust development. The DFD illustrates how the system functions through its complete data processing pathway to its actual mobile implementation period.

3.1.6 UI Design

Any machine learning system deployment requires an easy-to-use interface which directs non-technical end users precisely because agricultural settings boast different levels of technical proficiency. The mango leaf disease detection model reaches users through a Flutter-based mobile application which provides instant feedback after users upload images. Real farmers regardless of their technical expertise can effectively utilize the technology because its user interface was specifically designed to be lightweight and user-friendly and responsive. The mobile application operates using a two-tier architecture which enables TFLite local inference for performing real-time and offline predictions.

1. Application Set-up

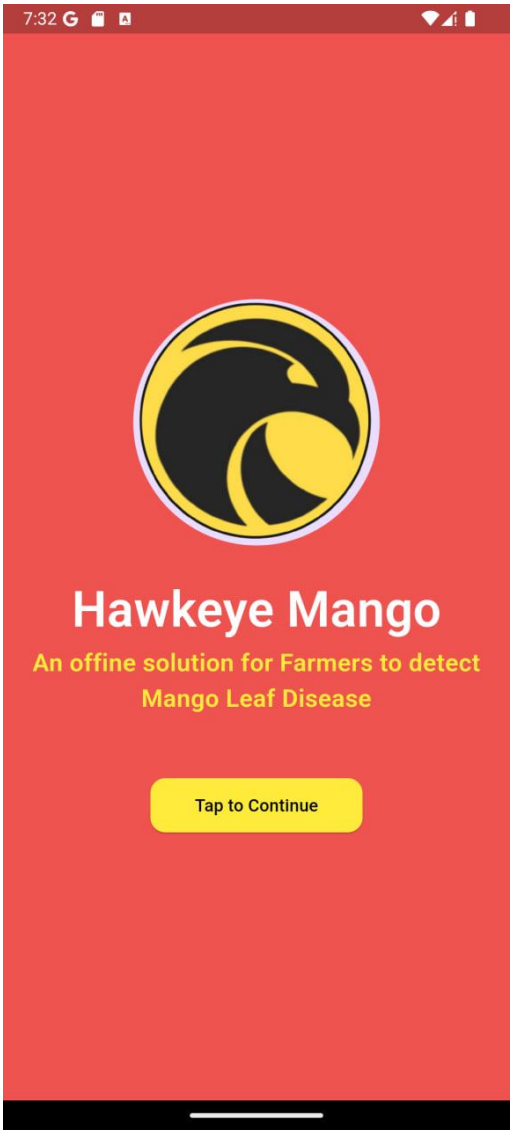
The application uses Google's open-source Flutter toolkit to develop its codebase, which enables the development of platforms across multiple OS environments with a single programming foundation. Users send mango leaf images, which automatically get resized and normalized by the system until they fit the T-Net model input requirements. From within the application's assets, the TFLite model operates directly on the device to complete analysis without needing internet access.

The application maintains real-time classification abilities through operating in both rural areas and low-network zones. Predicted disease labels appear from the app after a single run, providing users with simplified, actionable information.

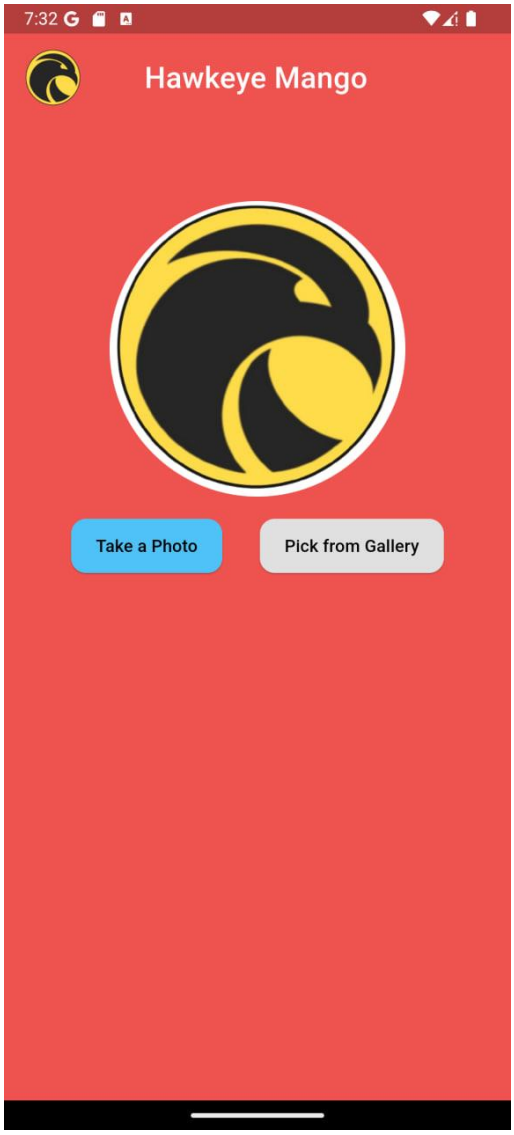
2. Two-Tier Architecture

The application follows a two-tier architecture, which separates the presentation layer (UI/UX interface) from the model inference layer (business logic and TFLite model execution). This design allows for a clean separation of concerns, easier updates, and better maintenance.

- **Tier 1 – Presentation Layer:** This is the front-end of the application where users interact. Built entirely using Flutter widgets, this layer handles image capture/upload and displays prediction results. The UI is optimized for low memory usage and is responsive across a range of device screen sizes.
- **Tier 2 – Inference and Processing Layer:** This layer is responsible for loading the TFLite model, preprocessing the input image, running inference, and post-processing the results. Dart plugins were used to develop the interface with TensorFlow Lite, which ensures high-performance local inference without cloud-based services.



First Page (After Application started)



Main Page (After pressing the button "Tap to Continue")

Figure 3.4: UI Design for Application

3.2 Detailed Methodology and Design

3.2.1 Dataset

The timely collection of data remains essential to obtain high-quality datasets that represent variety in mango leaf disease detection. The MLD24 dataset [28] used in this study included 6400 mango leaf images gathered from orchards throughout Kushtia and Dhaka in Bangladesh. The MLD24 dataset contains eight categories of mango leaf conditions which include disease and healthy specimens alongside seven fundamental disease types: Mango leaves suffer from seven common diseases,

including Anthracnose, Bacterial Canker, Cutting Weevil, Die Back, Gall Midge, Powdery Mildew, and Sooty Mould. The collection of images took place with an iPhone SE camera by first reducing 3024 × 4032 pixel images to 240 × 240 pixels to support efficient file storage and operational processing.

Table 3.1: Dataset Specifications.

Properties	Values
Image Resolution	240 × 240 pixels
Format	.jpg
Total Images	6,400
Classes	8

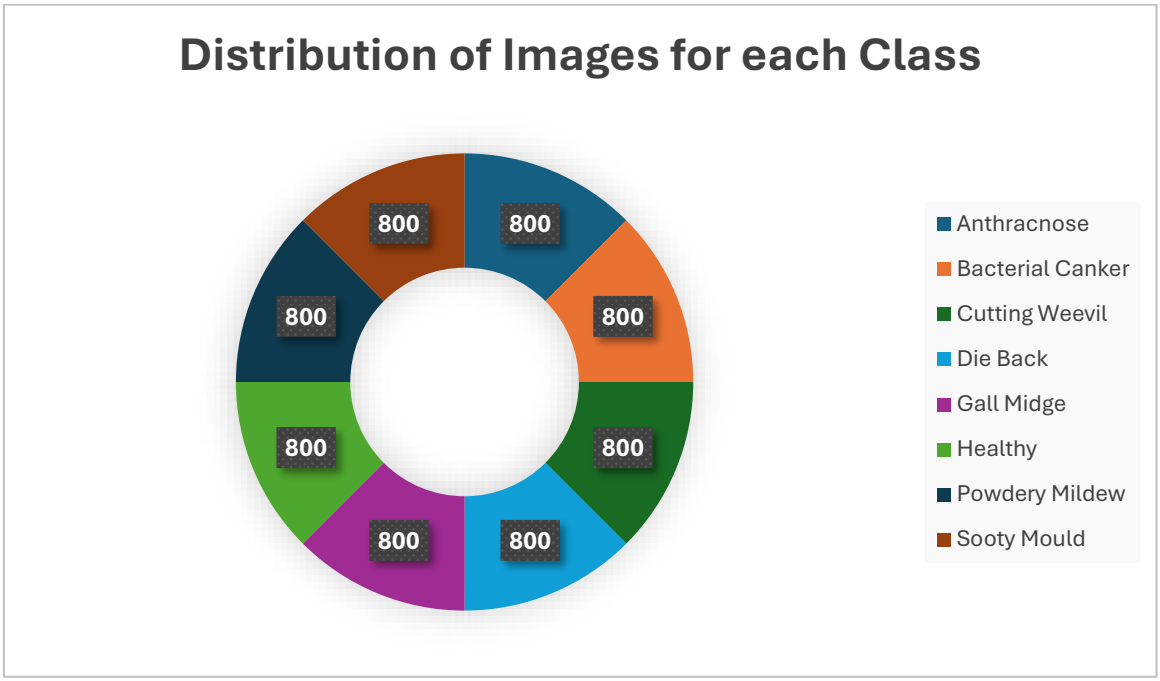


Figure 3.5: The Image Distribution for Each Class

3.2.2 Classes

The dataset consists of eight classes, including seven diseased categories and one healthy category. Below is a brief description of each class:

- **Anthracnose (AC):** A fungal disease caused by Colletotrichum gloeosporioides, leading to dark, sunken lesions on mango leaves. It weakens the plant and significantly reduces fruit yield.

- **Bacterial Canker (BC):** Caused by Xanthomonas campestris, this disease creates water-soaked lesions that eventually turn brown with a raised appearance, leading to severe defoliation.
- **Cutting Weevil (CW):** A pest infestation where Deporaus marginatus beetles chew leaf margins, causing irregular cuttings that affect the tree's ability to photosynthesize properly.
- **Die Back (DB):** A fungal disease that causes the gradual drying of leaves and branches from the tips downward, often leading to tree decline if untreated.
- **Gall Midge (GM):** A pest-related condition caused by Procontarinia matteiana, where small reddish or yellow galls (swellings) appear on leaves, affecting growth and productivity.
- **Healthy (H):** Represents mango leaves with no visible signs of disease or pest infestation, serving as the baseline for classification models.
- **Powdery Mildew (PM):** A fungal disease caused by Oidium mangiferae, characterized by a white powdery coating on leaves, leading to leaf curling and premature drop.
- **Sooty Mould (SM):** A fungal condition that appears as a black, soot-like layer on the leaf surface, often resulting from insect secretions like honeydew, which blocks sunlight and reduces photosynthesis.

Anthracnose Bacterial Canker Cutting Weevil Die Back

Gall Midge Healthy Powdery Mildew Sooty Mould

Figure 3.6: Instances of images of dataset

3.2.3 Preprocessing Techniques

1. Normalization

Each input image was normalized by scaling pixel values to a range between 0 and 1. This ensures uniformity across the dataset, allowing the model to learn more efficiently and converge faster during training.

2. DPI change

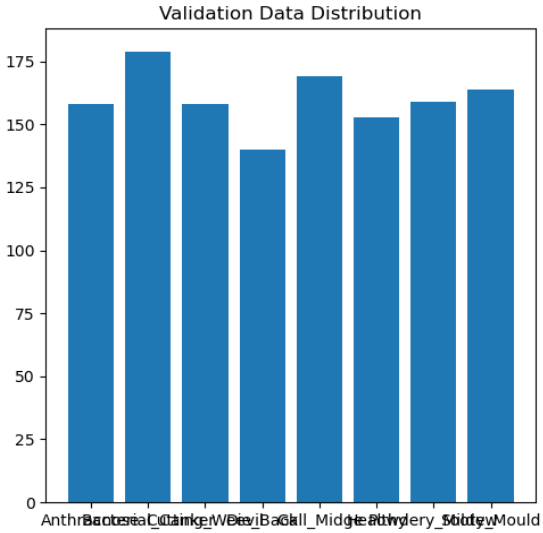
Each image received uniform resizing to measure 112 pixels by 112 pixels. The resizing operation ensures uniform input dimension coherence while also maximizing efficiency and compatibility between the models tested with Tiny-Net and CNN.

2. Train-Val-Test Split

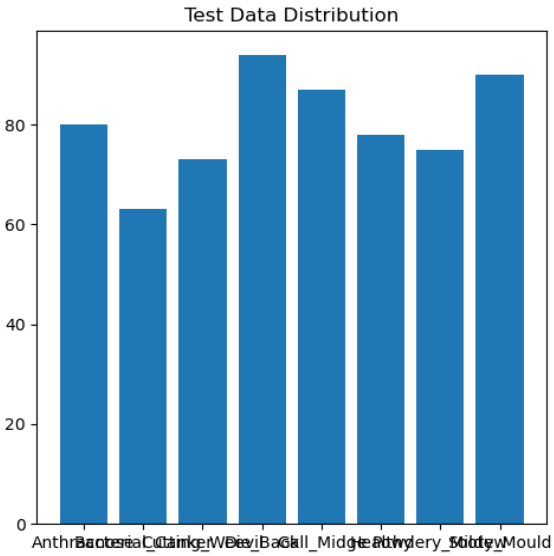
The dataset is split into three distinct groups and serves to guarantee both objective assessment and steady model functioning. The deep learning models received their training data from the training set, with the validation set providing parameter adjustments for optimizing performance, while the test set evaluated the final stage of each trained model. The distribution of data across eight classes exhibited balance according to Figure 3.5, while the training set maintained precise equal sample numbers per class. The small variations seen in validation and test distribution patterns were acceptable without affecting the model's generalization potential. The

established method for model splitting results in reliable and accurate performance from trained models when they come across new data. Deep learning research practices show good experimental standards through clear training-validation-testing phase separation, since this approach supports the development of a deployable system for real-world scenarios.

Training Data Distribution



Validation Data Distribution



Test Data Distribution

Figure 3.7: Train-Val-Test Data Split and Distribution

3.2.2 Model Description

1. Convolutional Neural Network (CNN)

The image classification backbone known as Convolutional Neural Networks (CNNs) enables automatic learning of spatial features which drives contemporary visual data classification. This research uses CNNs to find and categorize mango leaf diseases by analyzing picture-based visual characteristics such as spots, together with textures and discolorations. A CNN includes vital items that process image data a distinct order throughout its overall structure. This piece details the component-wise workflow found in the CNN analysis model used for research.

- **Convolutional Layer:** A CNN commences its operation using convolutional layer as its foundational building block. Small filters known as kernels analyze local image features through their application on input images to extract edges and curves and shapes. The filters perform operations by sliding across image width and height dimensions while taking dot products between their weights with the input pixel values in the specified region. The operation produces a feature map which detects particular image patterns throughout different regions.

  Mathematically, this operation can be expressed as:

$$Z(i,j) = \sum_{m=0}^{k=1} \sum_{n=0}^{k=1} W_{m,n} \cdot X(i+m, j+n) + b \qquad (3.1)$$

  where X is the input image, W is the kernel, b is the bias term, and $Z(i, j)$ is the output at a given location.

- **Activation Function (ReLU):** An activation function follows convolution to introduce model non-linearity. The activation function ReLU performs processing through two steps by converting negative values to zero while retaining positive values.

$$f(x) = \max(0, x) \qquad (3.2)$$

  This step enables CNN to learn complex and non-linear patterns within the data and helps avoid issues like vanishing gradients during training.

- **Pooling Layer:** Feature map dimensions shrink during pooling operations because the layer down samples spatial data to keep vital image content. Max

pooling, the most common technique, takes the maximum value from a region (typically 2×2) of the feature map. Pooling helps reduce the number of parameters, speeds up computation, and introduces a degree of translational invariance, allowing the model to detect features regardless of their location in the image.

- **Flattening Layer:** Once multiple convolutional and pooling operations are complete, the resulting 2D feature maps are flattened into a 1D vector. This step prepares the data for classification by converting it into a format suitable for fully connected layers. The flattening process simply reshapes the multidimensional output into a single vector of features.
- **Fully Connected Layer:** The fully connected (FC) layer acts as the decision-making component. Each neuron in this layer is connected to every neuron in the previous layer, allowing it to learn global patterns. These connections combine the learned features and map them to the final output.

  In classification tasks, the final FC layer usually ends with a SoftMax function to calculate the probability for each class:

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.3}$$

- **End-to-End Learning:** All the above components work together in a pipeline where the model learns to adjust weights and biases through backpropagation during training. The entire CNN operates in an end-to-end fashion, meaning the system takes an image as input and outputs a class prediction—without requiring handcrafted features.

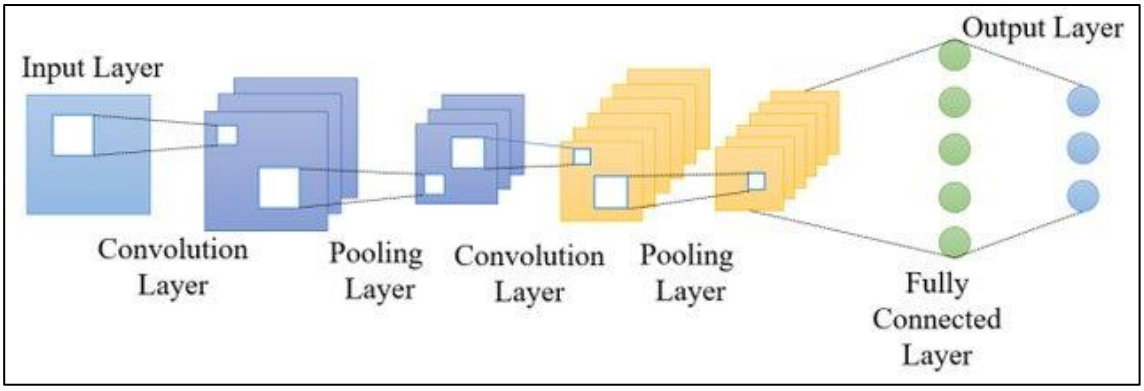


Figure 3.8: The base architecture of the CNN model [29]

Table 3.2: Comparisons of Six SOTA CNN models

Model Name	Depth	Number of Parameters	Classification Performance	Mobile Suitability
VGG16	16 layers	~138 million	High, but prone to overfitting and slow	Not optimized for mobile (large size)
ResNet50	50 layers	~25.6 million	Very high, stable on complex data	Moderate (heavier than MobileNet)
DenseNet101	101 layers	~8 million	Excellent generalization, robust	Moderate (more efficient than ResNet)
InceptionV3	~48 layers	~23.8 million	High accuracy, good with low computational cost	Moderate (relatively efficient)
Xception	36 layers (126 with depthwise)	~22.9 million	Excellent on image data, especially with small datasets	Moderate (mobile-friendly if pruned)
MobileNet	~28 layers	~4.2 million	Reasonable accuracy, best trade-off for mobile	Excellent for mobile and edge deployment

2. Proposed Lightweight CNN-Tiny-Net (T-Net)

In this research, a custom lightweight convolutional neural network named Tiny-Net (T-Net) is proposed to enable real-time mango leaf disease detection, especially on mobile and edge devices. The architecture of T-Net was carefully designed to strike a balance between performance accuracy and computational efficiency, ensuring suitability for deployment in low-resource environments such as

agricultural fields.

The network comprises a total of 3.98 million trainable parameters, significantly fewer than most traditional deep CNNs like VGG16 or ResNet50, making it faster to train and easier to deploy. T-Net follows a structured design involving multiple convolutional blocks, pooling layers, dense layers, and regularization units.

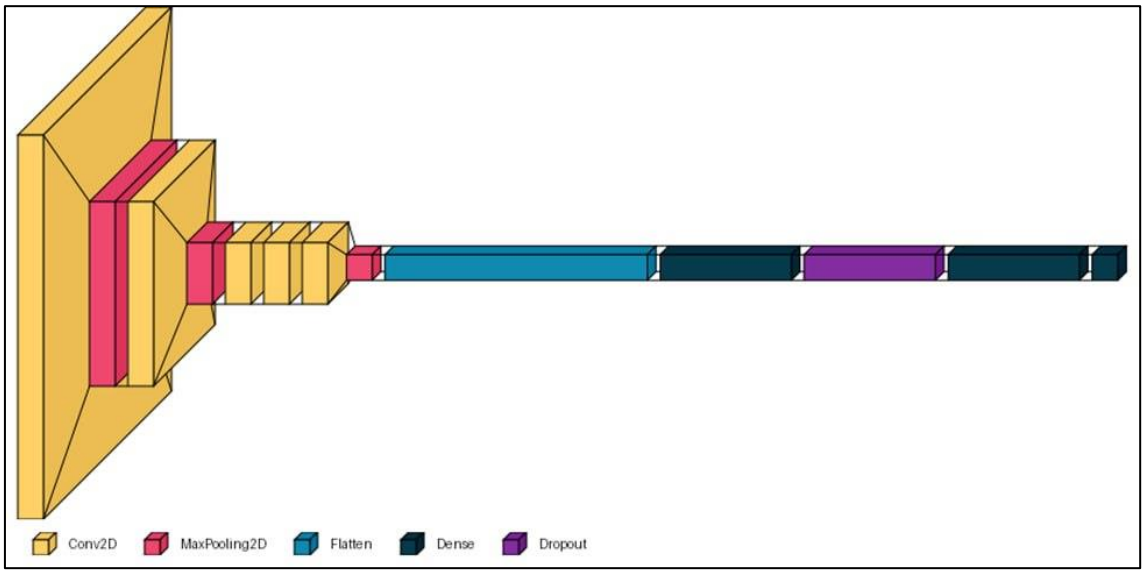


Figure 3.9: The proposed architecture of the Tiny-Net (T-Net) model

Table 3.3: The Algorithm of Tiny-Net Model

Input:	Preprocessed Mango Leaf Image of size (H, W, 3)
Output:	Predicted disease class label from 8 categories

Step	Description
1	Initialize Tiny-Net model: define input shape, number of filters, kernel sizes, activation functions, and number of output classes.
2	Input Layer: Accept normalized mango leaf image.
3	First Convolution Block: - Apply Conv2D (48 filters, 3×3, ReLU) - Apply MaxPooling2D (pool_size=3×3, stride=3)
4	Second Convolution Block: - Apply Conv2D (128 filters, 3×3, ReLU) - Apply MaxPooling2D (pool_size=3×3, stride=3)
5	Third Convolution Block:

- Apply Conv2D (192 filters, 3×3, ReLU)
- Apply another Conv2D (192 filters, 3×3, ReLU)
- Apply Conv2D (128 filters, 3×3, ReLU)
- Apply MaxPooling2D (pool_size=3×3, stride=3)

6 Flatten Layer:

Flatten output feature map into a 2048-dimensional vector.

7 Fully Connected Layers:

- Apply Dense (1024 units, ReLU)
- Apply Dropout(rate=0.5)
- Apply another Dense (1024 units, ReLU)

8 Output Layer:

- Apply Dense (8 units, Softmax) to classify into 8 disease categories.

9 Prediction:

Return to the class with the highest probability (ArgMax) as the final disease label.

3. Explainable AI (XAI)

As deep learning models continue to grow in complexity, one of the major challenges researchers' faces is the lack of interpretability, especially in critical domains like agriculture. While Convolutional Neural Networks (CNNs) provide high accuracy in image classification tasks, they often operate as black boxes, offering little to no insight into how decisions are made. This limitation can affect user trust and model transparency, particularly when applied in real-world scenarios like disease diagnosis in crops. To address this issue, Explainable AI (XAI) techniques are integrated into this study to provide meaningful explanations of the model's predictions.

The interpretability of baseline CNNs together with the proposed T-Net model benefits from application of three prevalent XAI approaches: LIME (Local Interpretable Model-Agnostic Explanations) alongside SHAP (SHapley Additive Explanations) and Grad-CAM (Gradient-weighted Class Activation Mapping). The XAI tools LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (SHapley Additive Explanations) combined with Grad-CAM (Gradient-weighted Class Activation Mapping) provide interpretation methods for both baseline CNNs and the proposed T-Net model. Every visual explanation brings distinct approaches to show which features the model used to make predictions.

- LIME uses image input perturbation combined with prediction analysis to determine which image areas contributed most to the model's final output. The approach establishes an interpretable linear model near predictions which reveals which regions of the image fostered a certain class outcome. This technique functions independently of any model structure and lets users see specific case outcomes from CNN predictions.
- SHAP generates important value assignments (Shapley values) to feature components, thus revealing the contribution of image pixels and regions to prediction outcomes. The technique leads to valid explanations that fulfil consistency requirements while demonstrating local accuracy, thus providing a dependable assessment of input feature impacts. SHAP values unite multiple prediction insights to produce a single metric which helps researchers evaluate model behavior patterns.
- The Grad-CAM visualization technique was specifically developed to work with CNNs. Through gradients of the target class leading into the final convolutional layer, this approach generates a heatmap which indicates what areas of the image captured the model's attention for its decision. Image-based classification benefits from visual explanations because they demonstrate the model's gaze direction during prediction.

3.3 Project Plan

A structured multi-phase approach to system design was implemented for the development of the mango leaf disease detection system. We started by selecting a proper dataset (MLD24) and then performed dataset normalization followed by DPI changes and image resizing and normalization tasks. Six state-of-the-art CNN models were used to train and evaluate data using consistent performance measures after preparing the dataset. The research findings led to the creation of Tiny-Net (T-Net) as a custom lightweight model that received optimization for mobile implementations. The complementing use of Explainable AI techniques was integrated into model development work to build trust and achieve transparency among users. A real-time Flutter-based mobile application united all components of the system together.

Table 3.4: GANTT Chart of Project Timeline.

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

3.4 Task Allocation

The first work project was data acquisition and verifying images, which involved manually going through each image of a mango leaf and carefully checking resolution, verifying for disease type, and quality checking. During data acquisition, all the first stage preprocessing work had to be completed before training and evaluation of six competitive convolutional neural network (CNNs) models. Concurrently, we started work on designing/developing our custom lightweight model, Tiny-Net, and hyperparameter tuning while simultaneously simplifying the model. The second picture set of tasks looked at Explainable AI techniques using Grad-CAM, SHAP, and LIME, to give explainable AI credibility to our work. Finally, the last set of tasks was to include app development, including UI design, TensorFlow Lite integrated in a mobile app, and testing the app in Flutter in the real world in real time. All tasks were planned accordingly, as on a timeline, with a solid indication that we've achieved an implementation of technical prowess that matched the practical use intended to have achieved.

3.5 Summary

The methodology of this study aimed to design a practical, lightweight, and transparent system for mango leaf disease detection through images, using various deep-learning approaches. The first method was to gather and clean/organize the MLD24 dataset. The next step was the image pre-processing stage, which included enhanced image augmentations so that the model would be more robust. The second method was to create six advanced CNN models to monitor the classification accuracies. After finishing these steps, the last part of the project proposed a custom lightweight model, Tiny-Net (T-Net), which required fewer computational resources than traditional deep neural network methods, while still maintaining a similar classification accuracy. The study also included Explainable AI (XAI) methods in the system, using LIME, SHAP, and Grad-CAM, to increase model interpretability and give end-users confidence in the model's disease prediction. After the classification portion of the model conversion was in TFLite format and integrated into a complete mobile application designed in Flutter, to enable end-users to make offline, real-time disease predictions while they were engaged in agricultural activities during their workday. Global Modular development principles were followed throughout the study methodology to ensure that the final system would be as efficient, interpretable, and usable in practice as possible.

Chapter 4

Implementation and Results

The chapter contains a description of the setup of the experiment, performance assessment of models, results of explainability, as well as results of mobile application deployment, and extensive discussion on findings.

4.1 Environment Setup

The research required a standardized and optimized computational environment to maintain consistency and traceability throughout every phase of the mango leaf disease detection system. The research implemented six state-of-the-art Convolutional Neural Network models followed by developing the custom lightweight T-Net model and integrating Explainable AI techniques and deploying the top-performing model into a Flutter mobile application platform. All deep learning research was run using Google Colaboratory which provided a GPU runtime (NVIDIA Tesla T4) to speed up training. A Python 3.10 framework included TensorFlow and Keras alongside OpenCV and NumPy and Matplotlib and scikit-learn libraries. The project used TensorFlow Lite along with Flutter SDK for building apps that allowed offline predictions during the development for mobile deployment.

All experiments employed a shared set of hyperparameters between SOTA CNNs and T-Net model for maintaining evaluation consistency. Table 4.1 summarizes the training process where all models used 112 × 112 pixel-sized input images to achieve a practical disease feature preservation and computation efficiency trade-off. A batch size of 16 was selected to optimize GPU memory usage during training operations. The 50-epoch duration of training sustained sufficient iterations to extract general information without training beyond capacity. The Adam optimizer proved ideal for this case due to its adaptive learning rate property and its speed at converging beyond regular stochastic gradient descent methods. The learning rate remained constant at 0.0001 for all models to ensure training stability when working with different depths and architectural complexities.

Table 4.1: Common parameter table for all experimented models.

Parameter Name	Parameter Value
Image Size	112 × 112
Batch Size	16
Epoch	50
Optimizer	Adam
Learning Rate	0.0001

The datasets across all experiments received standard partitioning into three sections: training, validation, and testing. All models, including six SOTA CNNs and the proposed Tiny-Net received equal treatment with this approach to maintain a fair assessment of performance. A total of 4,480 training images assisted model pattern recognition while preventing overfitting from a pool of 6,400 images belonging to eight classes. The validation set consisted of 1,280 images that aided model hyperparameter optimization during training through metric evaluation of accuracy, along with loss curves monitoring. The final 10% (640 images) operated as the test set exclusively for testing complete models. The unknown test data served to provide precise measurements for how each model would perform if deployed in real-world situations. This separate test set provided the data to calculate both test accuracy, together with confusion matrices and ROC curves as the final performance metrics.

Table 4.2: Common data split for all experimented models.

Dataset	In Percentage	Number of Images
Train set	70%	4480
Validation Set	20%	1280
Test Set	10%	640

4.2 Testing and Evaluation/Performance/ Comparative Analysis

1. Accuracy

The proportion of correctly classified instances (both positive and negative) to the total instances. Accuracy gives a quick overview of model performance but can be misleading in imbalanced datasets where one class significantly outnumbers the other.

$$Accuracy = \frac{TP + TN}{FP + FN + TP + TN} \quad (4.1)$$

2. Recall

The ratio of correctly predicted positive observations to all actual positives. Recall is particularly important in scenarios where a positive case (such as a diseased plant) could lead to severe consequences, like crop loss.

$$Recall = \frac{TP}{(FN + TP)} \quad (4.2)$$

The ratio of correctly predicted positive observations to the total predicted positives. Precision is crucial in applications where the cost of false positives is high. In this study, high precision indicates that when a disease is predicted, it is likely to be true.

$$Precision = \frac{TP}{(FP + TP)} \quad (4.3)$$

4. F1-Score

The harmonic meaning of precision and recall, providing a balance between the two metrics. The F1 score is especially useful when dealing with imbalanced classes, as it considers both false positives and false negatives, offering a more comprehensive view of model performance.

$$F1\ Score = 2 \times \frac{Precision + Recall}{(Precision \times Recall)} \quad (4.4)$$

5. Confusion Matrix

A table used to describe the performance of a classification model, showing the true vs. predicted classifications. The confusion matrix provides insights into the types of errors made by the model, allowing for more targeted improvements.

6. Receiver Operating Characteristic (ROC) Curve

A graphical representation of a classifier's performance across various threshold settings, plotting the true positive rate (recall) against the false positive rate. It illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It plots the True Positive Rate (TPR), also known as recall or sensitivity, against the False Positive Rate (FPR) at various threshold settings.

Area Under Curve (AUC): The area under the ROC curve, which provides a single metric to assess model performance; a value closer to 1 indicates a better model. In this study, the Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) are crucial for evaluating classification models. They allow for visual comparisons of model performance across various thresholds, facilitating optimal threshold selection by balancing sensitivity and specificity. Additionally, the ROC curve is robust against class imbalances, providing reliable assessments where accuracy may mislead. However, ROC and AUC should complement other metrics like precision and recall for a comprehensive evaluation of model effectiveness.

4.3 Results and Discussion

This section provides an extensive analysis of experimental outcomes starting from evaluating six advanced CNN models for performance followed by investigating the proposed lightweight model Tiny-Net. The evaluation process for each model included multiple crucial metrics such as validation-test accuracy alongside classification reports and confusion matrices, and ROC-AUC curves for a proper comparison. DenseNet101 demonstrated the best overall accuracy performance, but MobileNet faced challenges in prediction despite its low resource requirements. Tiny-Net received implementation as a performance-efficient, mobile-friendly solution after the baseline assessment phase. The study produced results that matched the accuracy standards while simplifying calculations enough to work effectively in real-time computing systems. XAI techniques with Grad-CAM, LIME, and SHAP were integrated to provide essential interpretability features that explained model prediction processes by showing visual and feature-based explanations.

Performance evaluations for VGG16, ResNet50, DenseNet101, InceptionV3, Xception, and MobileNet indicate these models provide state-of-the-art performance, depending on your application, in Table 4.3 by showing the accuracy results for the validation and test sets. The performance results in Table 4.3 demonstrate DenseNet101 provided the best performance, for example, validation accuracy of 99.66% and test accuracy of 99.03% and was closely followed by VGG16 at 98.77% and 98.15%, respectively. The validation and test sets performed above 97% with ResNet50, Xception and InceptionV3. However, MobileNet provided considerably lower accuracy results with a light-weight design that produced a validation accuracy of 79.33% and a test set accuracy of 78.67%. Based on this comparison, it's

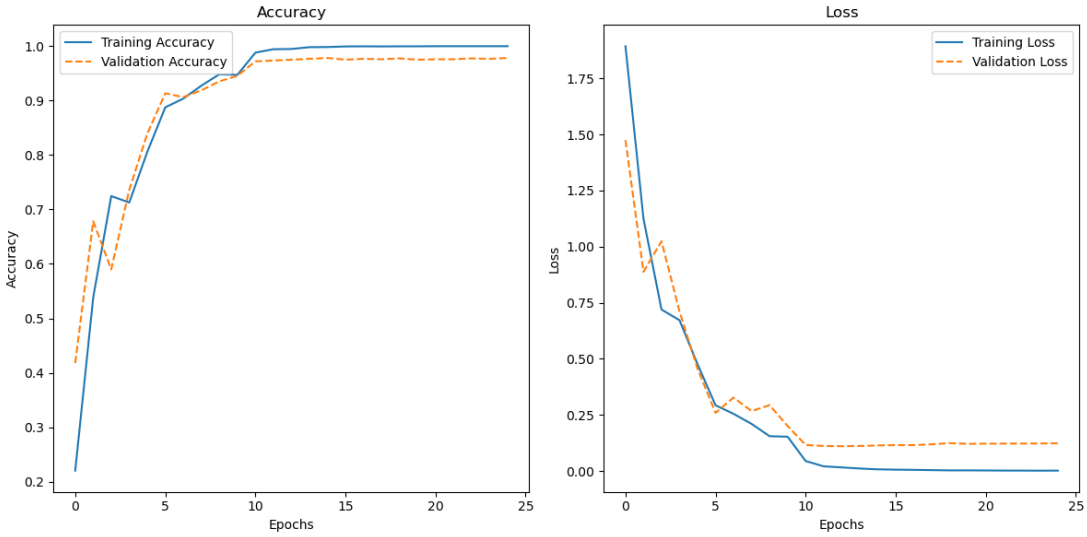
easy to see that the dense connectivity of DenseNet101 makes detection of complex patterns within mango leaf images easier and that the smaller models struggled with identifying the complex patterns in the agriculture dataset. The low variability and stable accuracy results indicate incoming sample quality is high and confirm the benefits of universally sharing training parameters, as seen in our experiments.

Table 4.3: Validation and Test Accuracy for six SOTA-CNN

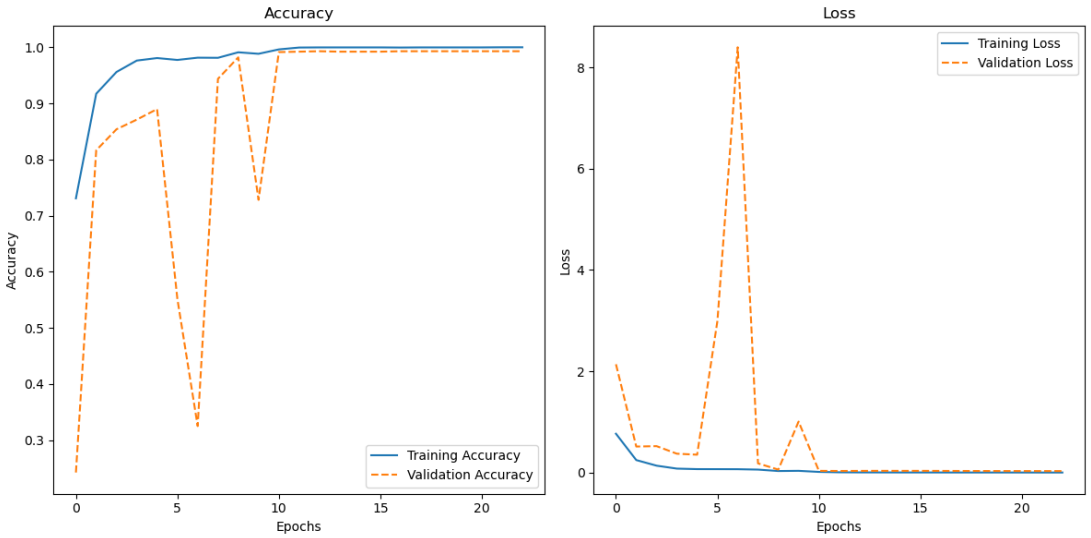
Models	Validation Accuracy	Test Accuracy
VGG16	98.77%	98.15%
DenseNet101	99.66%	99.03%
ResNet50	97.73%	97.57%
InceptionV3	97.42%	97.19%
Xception	97.84%	97.39%
MobileNet	79.33%	78.67%

The validation accuracy and loss curves for each model, shown in Figure 4.1, offer a clear view of how each model performed throughout training. DenseNet101 and VGG16 exhibited smooth and rapid convergence, with their validation accuracy gradually improving and stabilizing over epochs, indicating that they effectively learned discriminative features without significant overfitting. In contrast, MobileNet's validation curve showed less consistency, with occasional fluctuations that suggest instability in learning or reduced feature extraction capability from the lower image dimensions. Similarly, loss curves of the higher-performing models displayed a steady decline and plateaued as the models approached optimal weights. These visual patterns further reinforce the numerical results from Table 4.3, showing that DenseNet101 and VGG16 not only achieved higher accuracy but also maintained stable training behavior throughout. The validation curves of ResNet50, Xception, and InceptionV3 remained strong, though their convergence was slightly slower, hinting at possible room for hyperparameter tuning. Overall, these visualizations offer intuitive insights into model learning dynamics and their respective capacities to generalize from training to unseen data.

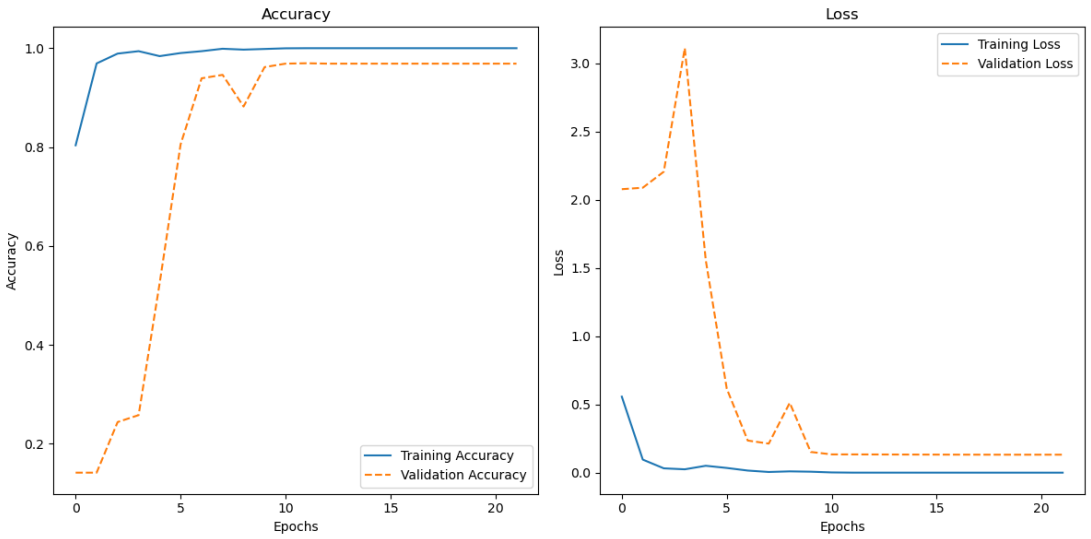
VGG16



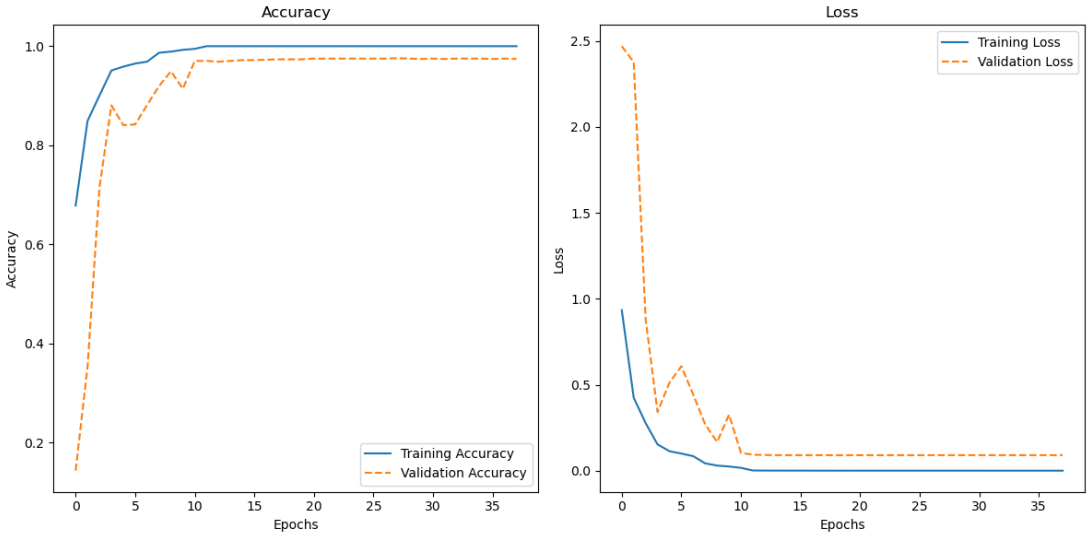
Densenet 101



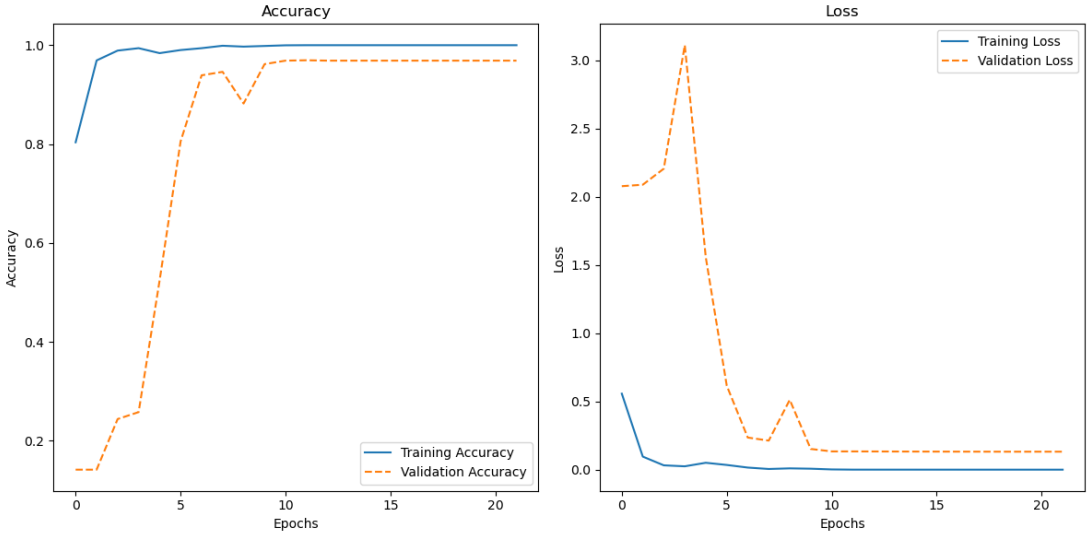
ResNet50



InceptionV3



Xception



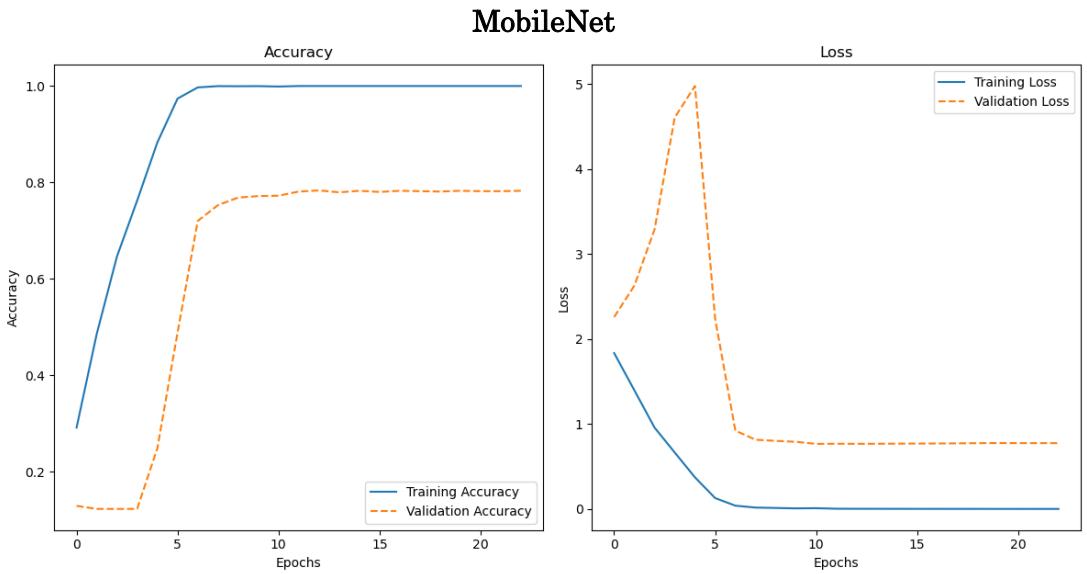


Figure 4.1: Validation Accuracy and Loss Curves of Six SOTA CNN Models

The confusion matrices from the six models, seen in Figure 4.2, provided additional evidence about how each model perceives and predicts across all eight mango leaf classes. DenseNet101 and VGG16 had almost perfect classification with few misclassifications and showed large diagonal matrices indicating confidence and accurate prediction of actual class labels. ResNet50, Xception, and InceptionV3 also performed strongly, hence the minor confusion between which class is the Powdery Mildew class instead of the Sooty Mould class. MobileNet showed large off-diagonal matrices, which indicated many false-positive scores against many true-negative scores as it confused between diseases that had slight informed trait changes, such as sensitive differences in leaf texture or leaf coloration. These results indicate the value of the depth of the model and richness of features when grappling with high intra-class similarities and low inter-class variation that are consistent with real-world plant pathology datasets.

VGG16

	AN	BC	CW	DB	GM	H	PM	SM
AN	153	0	2	1	0	0	0	0
BC	0	163	0	0	0	1	0	0
CV	1	1	146	0	0	1	0	0
DB	0	0	0	159	0	0	0	0
GM	2	0	1	0	164	0	3	2
H	0	0	0	2	0	148	0	0
PM	0	0	0	0	1	0	156	2
SM	0	1	0	0	2	0	5	163

Densenet 101

	AN	BC	CW	DB	GM	H	PM	SM
AN	168	0	0	0	0	0	0	0
BC	0	173	0	0	0	0	0	0
CV	0	0	161	0	0	0	0	0
DB	0	0	0	152	0	0	0	0
GM	0	0	0	0	174	0	1	0
H	0	0	0	0	0	169	0	0
PM	0	0	0	0	1	0	132	0
SM	0	0	0	0	2	0	5	142

ResNet50

	AN	BC	CW	DB	GM	H	PM	SM
AN	159	0	0	0	0	0	0	0
BC	1	160	0	0	1	0	0	2
CV	0	0	147	0	0	0	0	0
DB	0	0	2	173	0	0	0	0
GM	2	0	1	0	141	0	2	3
H	0	0	0	0	0	158	0	0
PM	0	1	0	0	1	0	139	3
SM	0	2	0	0	2	0	19	161

InceptionV3

	AN	BC	CW	DB	GM	H	PM	SM
AN	165	0	1	0	1	0	0	0
BC	0	159	1	0	0	0	0	0
CV	0	0	158	0	0	0	0	0
DB	0	0	0	152	0	0	0	0
GM	0	0	1	0	161	4	1	1
H	0	0	0	0	1	159	0	0
PM	0	0	0	0	3	0	147	6
SM	0	0	0	1	5	0	6	147

Xception

	AN	BC	CW	DB	GM	H	PM	SM
AN	153	0	0	0	0	0	0	0
BC	0	170	0	0	0	0	0	0
CV	1	1	153	0	0	0	0	0
DB	0	0	0	163	0	0	0	0
GM	0	0	1	0	166	4	5	5
H	0	0	0	0	1	146	0	0
PM	0	1	0	0	3	0	135	3
SM	0	1	0	1	2	0	10	155

MobileNet

	AN	BC	CW	DB	GM	H	PM	SM
AN	136	0	6	0	5	8	4	0
BC	1	149	1	0	1	1	0	13
CV	4	3	140	0	2	0	5	2
DB	2	0	0	150	0	0	0	0
GM	4	5	2	0	89	12	24	19
H	7	0	0	0	13	138	0	0
PM	3	3	8	0	17	2	91	28
SM	0	10	2	4	9	4	43	110

Figure 4.2: Confusion Matrices of Six SOTA CNN Models

With the confusion matrices, classification reports provide a more precise evaluation of model performance as precision, recall, and F1-score for each class. The results in Table 4.4 confirm DenseNet101's advantage by achieving a score of more than 0.98 F1-score, highlighting both high understanding and precision. VGG16 also performed balanced and reliable results across every class. While ResNet50, InceptionV3, and Xception exhibited minor class-wise imbalances, their overall macro and weighted averages remained high, reflecting consistent generalization across disease types. MobileNet, however, showed noticeable drops in class-wise recall, especially for diseases with visually subtle patterns, such as Gall Midge and Bacterial Canker. This uneven performance signals MobileNet's challenge in capturing fine-grained features despite its lower computational footprint. Collectively, the classification reports enrich the performance narrative by validating not just accuracy, but the reliability and confidence of each model when deployed in practical, class-specific disease diagnosis scenarios.

Table 4.4: Classification report of Six SOTA CNN Models

		precision	recall	f1-score	support
VGG16	Anthracnose (AN)	0.98	0.98	0.98	156
	Bacterial_Canker (BC)	0.99	0.99	0.99	164
	Cutting_Weevil (CV)	0.98	0.98	0.98	149
	Die_Back (DB)	0.98	1.00	0.99	159
	Gall_Midge (GM)	0.98	0.95	0.97	172
	Healthy (H)	0.99	0.99	0.99	150
	Powdery_Mildew (PM)	0.95	0.98	0.97	159
	Sooty_Mould (SM)	0.98	0.95	0.96	171
	accuracy			0.98	1280
	macro avg	0.98	0.98	0.98	1280
	weighted avg	0.98	0.98	0.98	1280
DenseNet101	Anthracnose (AN)	1.00	1.00	1.00	168
	Bacterial_Canker (BC)	1.00	1.00	1.00	173
	Cutting_Weevil (CV)	1.00	1.00	1.00	161
	Die_Back (DB)	1.00	1.00	1.00	152
	Gall_Midge (GM)	0.98	0.99	0.99	175
	Healthy (H)	1.00	1.00	1.00	169
	Powdery_Mildew (PM)	0.96	0.99	0.97	133
	Sooty_Mould (SM)	1.00	0.95	0.98	149
	accuracy			0.99	1280
	macro avg	0.99	0.99	0.99	1280
	weighted avg	0.99	0.99	0.99	1280
ResNet50	Anthracnose (AN)	0.98	1.00	0.99	159
	Bacterial_Canker (BC)	0.98	0.98	0.98	164

	Cutting_Weevil (CV)	0.98	1.00	0.99	147
	Die_Back (DB)	1.00	0.99	0.99	175
	Gall_Midge (GM)	0.97	0.95	0.96	149
	Healthy (H)	1.00	1.00	1.00	158
	Powdery_Mildew (PM)	0.87	0.97	0.91	144
	Sooty_Mould (SM)	0.95	0.88	0.91	184
	accuracy			0.97	1280
	macro avg	0.97	0.97	0.97	1280
	weighted avg	0.97	0.97	0.97	1280
InceptionV3	Anthracnose (AN)	1.00	0.99	0.99	167
	Bacterial_Canker (BC)	1.00	0.99	1.00	160
	Cutting_Weevil (CV)	0.98	1.00	0.99	158
	Die_Back (DB)	0.99	1.00	1.00	152
	Gall_Midge (GM)	0.94	0.96	0.95	168
	Healthy (H)	0.98	0.99	0.98	160
	Powdery_Mildew (PM)	0.95	0.94	0.95	156
	Sooty_Mould (SM)	0.95	0.92	0.94	159
	accuracy			0.97	1280
	macro avg	0.98	0.98	0.98	1280
	weighted avg	0.97	0.97	0.97	1280
Xception	Anthracnose (AN)	0.99	1.00	1.00	153
	Bacterial_Canker (BC)	0.98	1.00	0.99	170
	Cutting_Weevil (CV)	0.99	0.99	0.99	155
	Die_Back (DB)	0.99	1.00	1.00	163
	Gall_Midge (GM)	0.97	0.92	0.94	181
	Healthy (H)	0.97	0.99	0.98	147
	Powdery_Mildew (PM)	0.90	0.95	0.92	142
	Sooty_Mould (SM)	0.95	0.92	0.93	169
	accuracy			0.97	1280
	macro avg	0.97	0.97	0.97	1280
	weighted avg	0.97	0.97	0.97	1280
MobileNet	Anthracnose (AN)	0.87	0.86	0.86	159
	Bacterial_Canker (BC)	0.88	0.90	0.89	166
	Cutting_Weevil (CV)	0.88	0.90	0.89	156
	Die_Back (DB)	0.97	0.99	0.98	152
	Gall_Midge (GM)	0.65	0.57	0.61	155
	Healthy (H)	0.84	0.87	0.85	158
	Powdery_Mildew (PM)	0.54	0.60	0.57	152
	Sooty_Mould (SM)	0.64	0.60	0.62	182
	accuracy			0.78	1280
	macro avg	0.78	0.79	0.78	1280
	weighted avg	0.78	0.78	0.78	1280

The evaluation of six state-of-the-art CNN models demonstrated that deep and densely connected architectures such as DenseNet101 and VGG16 consistently

outperformed others in classifying mango leaf diseases. Their strong validation and test accuracy, coupled with stable learning curves and precise confusion matrices, confirm their robustness in handling multiclass plant pathology data. However, models like DenseNet101, while highly accurate, are computationally intensive and require significant memory and processing power. Similarly, VGG16 and ResNet50 carry large parameter footprints that are ill-suited for integration into lightweight applications. Even though MobileNet was designed with mobility in mind, its notably lower accuracy and inconsistent class-wise performance reveal the limitations of overly compressed architectures when applied to visually complex and high-resolution tasks like disease detection in mango leaves. These observations highlight a vital need for a model that not only performs reliably but is also optimized for real-time, resource-constrained environments.

4.3.2 Results of Tiny-Net with K-Fold Validation

To ensure a more reliable and robust evaluation of the proposed Tiny-Net model, a 5-fold cross-validation strategy was adopted. Unlike single train-test splits that may bias the evaluation based on a specific subset, K-Fold validation divides the entire dataset into five equal parts or "folds," iteratively training the model on four folds and validating it on the remaining one. This approach guarantees that each data point contributes to both the training and validation components, which promotes higher generalizability and reduces variance in performance estimates. The use of k = 5 provides a reasonable trade-off between computational cost and reliability of the estimates. The mean results of the folds give a better understanding of Tiny-Net's learning ability and consistency when subjected to various distributions of training and validation data that guarantees that the performance of the model is not too reliant on a specific subset.

The post-training confusion matrices for each fold, as seen in Figure 4.3, provide a more detailed picture of the model's class-specific forecasts. Within all five folds, the matrices show dominant diagonal components, which means that there are high rates of correct classification evaluations for all classes. The number of misclassifications is visible only in a very limited number, whereas even those are between classes that have visually overlapping symptoms, for instance, between the Powdery Mildew and Sooty Mould. The stability of these confusion matrices throughout the folds serves as a strong affirmation of the model's astounding capacity to disambiguate the visually indistinguishable appearances of disease. In addition, the near-symmetric distribution of accuracy at a class-level accentuates the

capability of the model to learn unbiased representations for all the eight classes. The visualization from these matrices affirms the rank-and-file trends indicated in the classification reports because Tiny-Net operates with a high rate of precision, even when there is a minor inter-class ambiguity.

Fold-1

	AN	BC	CW	DB	GM	H	PM	SM
AN	147	0	2	1	0	0	0	0
BC	0	152	0	0	0	1	0	0
CV	1	1	157	0	0	1	0	0
DB	0	0	0	155	0	0	0	0
GM	0	1	0	0	114	0	0	2
H	0	0	0	0	2	137	0	0
PM	0	0	0	0	1	0	144	3
SM	0	0	0	0	0	0	1	148

Fold-2

	AN	BC	CW	DB	GM	H	PM	SM
AN	129	0	0	0	0	0	0	0
BC	0	153	0	0	0	0	0	0
CV	0	0	136	0	0	0	0	0
DB	0	0	0	141	0	0	0	0
GM	0	1	0	0	167	0	1	0
H	0	0	0	1	0	139	0	0
PM	0	0	0	0	0	0	150	0
SM	0	0	0	0	0	0	2	145

Fold-3

	AN	BC	CW	DB	GM	H	PM	SM
AN	151	0	0	0	0	0	0	0
BC	0	142	0	0	1	0	0	0
CV	0	0	147	0	0	0	0	0
DB	0	0	2	145	0	0	0	0
GM	0	0	0	0	146	1	0	1
H	0	0	0	0	0	135	0	0
PM	0	1	0	0	0	0	155	0
SM	0	0	0	0	0	0	0	140

Fold-4

	AN	BC	CW	DB	GM	H	PM	SM
AN	158	0	0	1	0	0	0	0
BC	0	130	0	0	0	0	0	0
CV	0	0	148	0	0	0	0	0
DB	0	0	0	144	0	0	0	0
GM	0	0	0	0	142	0	0	0
H	0	0	0	0	0	151	0	0
PM	0	0	0	0	0	0	145	0
SM	0	0	0	0	0	0	0	146

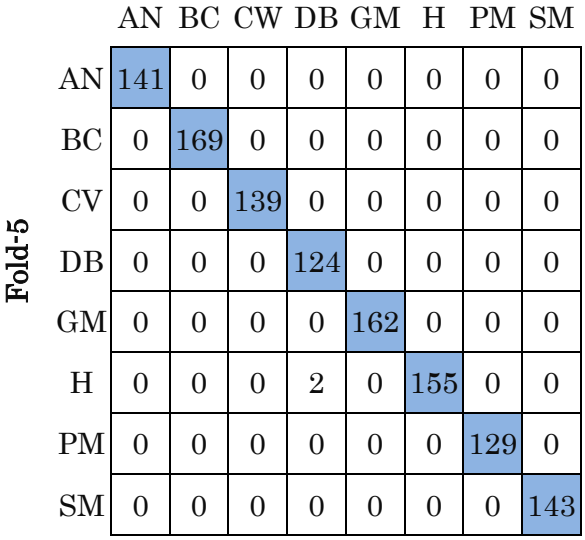


Figure 4.3: Confusion Matrices of Tiny-Net Model Across Five-Fold Cross-Validation

Table 4.5: Fold-Wise Classification Reports of Tiny-Net Model Across Five-Fold Cross-Validation

		precision	recall	f1-score	support
Fold 1	Anthracnose (AN)	1.00	1.00	1.00	147
	Bacterial_Canker (BC)	0.99	0.99	0.99	153
	Cutting_Weevil (CV)	1.00	1.00	1.00	157
	Die_Back (DB)	1.00	1.00	1.00	155
	Gall_Midge (GM)	0.97	0.97	0.97	117
	Healthy (H)	0.99	0.99	0.99	139
	Powdery_Mildew (PM)	0.99	0.97	0.98	148
	Sooty_Mould (SM)	0.97	0.99	0.98	149
	accuracy			0.99	1165
	macro avg	0.99	0.99	0.99	1165
	weighted avg	0.99	0.99	0.99	1165
Fold 2	Anthracnose (AN)	1.00	1.00	1.00	129
	Bacterial_Canker (BC)	0.99	1.00	1.00	153
	Cutting_Weevil (CV)	1.00	1.00	1.00	136
	Die_Back (DB)	0.99	1.00	1.00	141
	Gall_Midge (GM)	1.00	0.99	0.99	169
	Healthy (H)	1.00	0.99	1.00	140
	Powdery_Mildew (PM)	0.98	1.00	0.99	150
	Sooty_Mould (SM)	1.00	0.99	0.99	147
	accuracy			1.00	1165
	macro avg	0.99	0.99	0.99	1165
	weighted avg	0.99	0.99	0.99	1165
Fold 3	Anthracnose (AN)	1.00	1.00	1.00	151
	Bacterial_Canker (BC)	0.99	0.99	0.99	143
	Cutting_Weevil (CV)	1.00	1.00	1.00	147

	Die_Back (DB)	1.00	1.00	1.00	145
	Gall_Midge (GM)	0.99	0.99	0.99	148
	Healthy (H)	0.99	1.00	1.00	135
	Powdery_Mildew (PM)	1.00	0.99	1.00	156
	Sooty_Mould (SM)	0.99	1.00	1.00	140
	accuracy			1.00	1165
	macro avg	1.00	1.00	1.00	1165
	weighted avg	1.00	1.00	1.00	1165
Fold 4	Anthracnose (AN)	1.00	0.99	1.00	159
	Bacterial_Canker (BC)	1.00	1.00	1.00	130
	Cutting_Weevil (CV)	1.00	1.00	1.00	148
	Die_Back (DB)	0.98	1.00	0.99	124
	Gall_Midge (GM)	1.00	1.00	1.00	162
	Healthy (H)	1.00	0.99	0.99	157
	Powdery_Mildew (PM)	1.00	1.00	1.00	129
	Sooty_Mould (SM)	1.00	1.00	1.00	143
	accuracy			1.00	1164
	macro avg	1.00	1.00	1.00	1164
	weighted avg	1.00	1.00	1.00	1164
Fold 5	Anthracnose (AN)	1.00	1.00	1.00	141
	Bacterial_Canker (BC)	1.00	1.00	1.00	169
	Cutting_Weevil (CV)	1.00	1.00	1.00	139
	Die_Back (DB)	0.98	1.00	0.99	124
	Gall_Midge (GM)	1.00	1.00	1.00	162
	Healthy (H)	1.00	0.99	0.99	157
	Powdery_Mildew (PM)	1.00	1.00	1.00	129
	Sooty_Mould (SM)	1.00	1.00	1.00	143
	accuracy			1.00	1164
	macro avg	1.00	1.00	1.00	1164
	weighted avg	1.00	1.00	1.00	1164

A breakdown of classification performance for each of the five folds has been summarized in Table 4.5. The results indicate that Tiny-Net maintained a high precision, recall, and F1-scores for all the folds with very little variance. This consistency in various data splits shows that not only is the model accurate but also dependable while predicting mango leaf disease across varied groupings of images. Notably, the stability of the F1-score indicates that the model's balanced capabilities of recognizing both positive and negative classes are achieved without giving a preference to a particular category. The cross-fold and weighted averages, on the macro-level, alike to each other, strengthening the model's robustness. These fold-wise classifying reports express the degree to which Tiny-Net learned universal features of the dataset as well as overfitting any example fold, something that

improves the credibility of the model as an economic yet stable classifier.

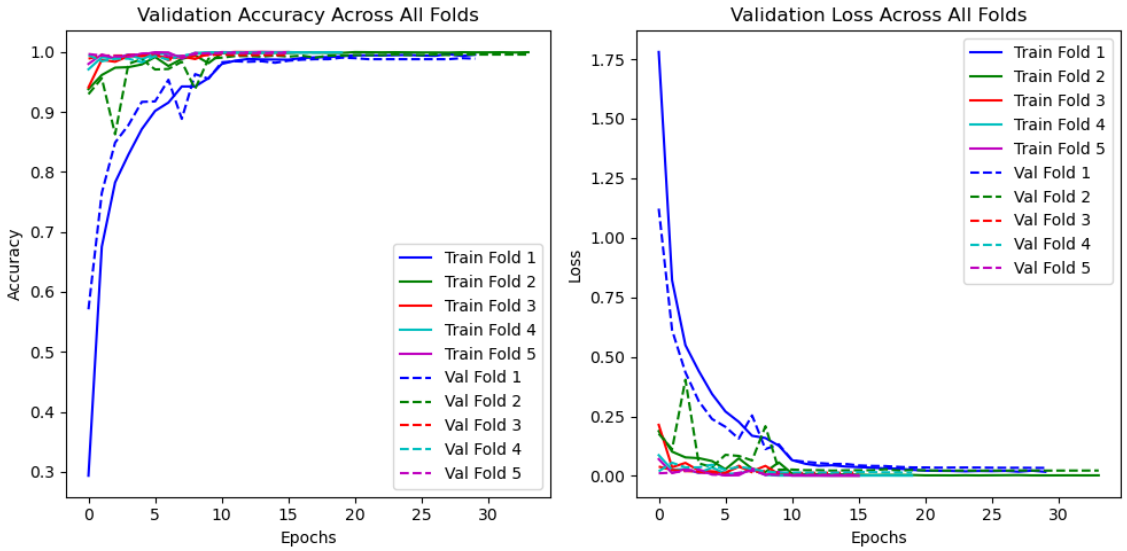


Figure 4.4: Combined Validation Accuracy and Loss Curves of Tiny-Net Model During Five-Fold Cross Validation

Apart from fold-wise assessments, a cumulative accuracy and loss dynamics curve for all five folds was also plotted, as depicted in Figure 4.4. In the accuracy curve, the increase is steady and consistent throughout the training epoch with near convergence over all folds, while in the loss curve, it is a steady decline. Notably, the similarity of training and validation accuracy curves indicates no overfitting, and this is evidence of the model's capability of generalizing well on new data. The near-parallel nature of the loss curves lends credence to this conclusion as well. These patterns reinforce the fact that Tiny-Net is a good approach in terms of learning because Tiny-Net is proficient enough to reduce the classification errors during learning while keeping the training stable. The agreement between the fold-wise curves shows the model's strength and efficiency in learning, points that are especially valuable relying on the lightweight design of this architecture. These trends echo the stability of the performance observed in the performance metrics and reaffirm efficiency of the selected training configuration and hyperparameters.

To consolidate the results of Tiny-Net, a combined classification report based on the predictions of all the 5 folds discussed above were produced as shown in Table 4.6. The report also provides a summary of performance using macro-averaged metrics with uniform high precision, recall, and F1-scores for all eight classes. This aggregated result goes on to support the fact that the model maintained a balanced performance across all folds, showing that there were no drastic falls in any specific class. It also validates the effectiveness of the model in working with visually similar

disease categories that were in the list of the most difficult ones in prior experiments with MobileNet. The uniform performance in metrics indicates excellent robustness and scalability of the Tiny-Net architecture, making its use appropriate as a cost-effective alternative to traditional heavy CNNs in real-time work.

Table 4.6: Combined Classification Report of Tiny-Net Model Aggregated Over Five Folds

		precision	recall	f1-score	support
Combined	Anthracnose (AN)	1.00	1.00	1.00	727
	Bacterial_Canker (BC)	1.00	1.00	1.00	748
	Cutting_Weevil (CV)	1.00	1.00	1.00	727
	Die_Back (DB)	0.99	1.00	1.00	709
	Gall_Midge (GM)	0.99	0.99	0.99	738
	Healthy (H)	1.00	0.99	1.00	722
	Powdery_Mildew (PM)	0.99	0.99	0.99	728
	Sooty_Mould (SM)	0.97	1.00	0.99	725
	accuracy			1.00	5824
	macro avg	1.00	1.00	1.00	5824
	weighted avg	1.00	1.00	1.00	5824

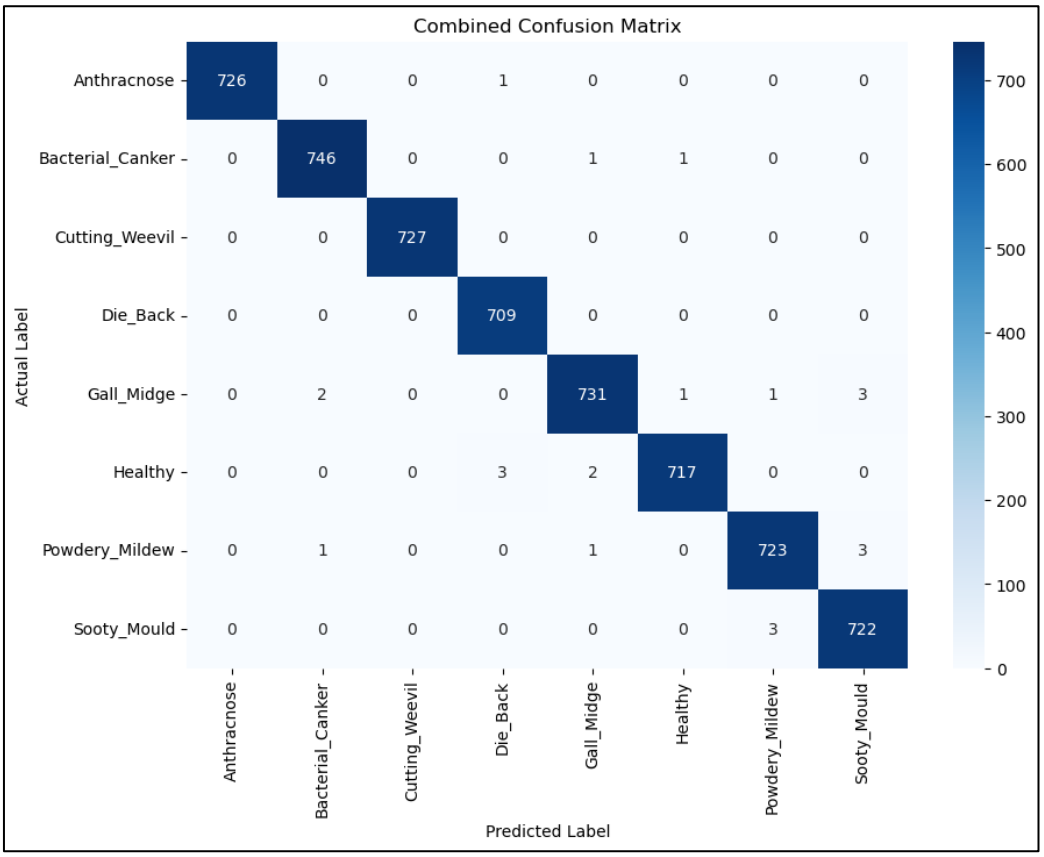


Figure 4.5: Combined Confusion Matrix of Tiny-Net Model Aggregated Over Five Folds Validation

Finally, the combined confusion matrix, shown in Figure 4.5, provides a broad landscape of the Tiny-Net model's class-sensible prediction accuracy after gathering all folds. The matrix displays major diagonal values, confirming the model's strength in correctly classifying most input images. Occasional misclassifications, although present, are minimal and follow a predictable pattern where classes with subtle visual overlaps are mistakenly predicted. However, these errors are marginal and do not significantly affect the model's overall reliability. The matrix confirms that Tiny-Net performs exceptionally well in distinguishing among the eight categories of mango leaf images, validating its real-world applicability. The fusion of high performance, low computational cost, and generalization capability sets Tiny-Net for automated mango leaf disease as a balanced model.

Table 4.7: Combined Evaluation Metrics of Tiny-Net Model (5-Fold Validation)

Metric	**Score (%)**
Combined Accuracy	99.61
Combined Precision	99.61
Combined Recall	99.61
Combined F1-Score	99.60

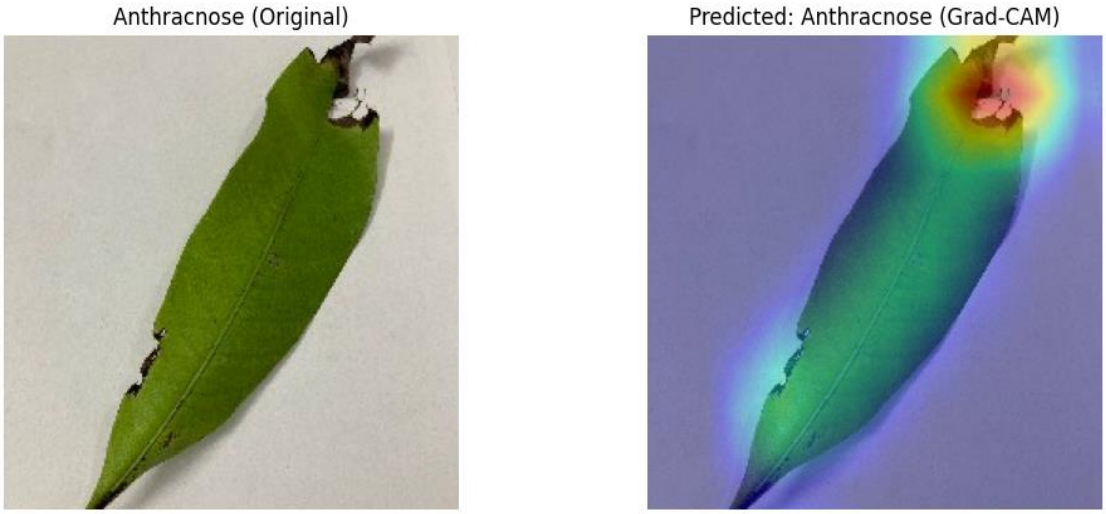
As shown in Table 4.7, the summarized evidence after applying the five-fold cross-validation to the Tiny-Net model is presented separately. An outstanding combined accuracy of 99.61% was attained by the model, which shows its excellent generalization ability across various folds. The precision, recall, and F1-score also showed very high values of 99.61%, 99.61%, and 99.60%, respectively shows that the model is highly reliable in differentiating between diseased and healthy mango leaf samples. The small difference between precision, recall and F1-score indicates that the model performs evenly well across all classes, without giving a greater preference to any class.

4.3.3 Visualizations after applying Explainable AI (XAI)

Explainable AI (XAI) methods, including LIME SHAP and Grad-CAM were applied to some of the correct and misclassified mango leaves' images for additional improvements in the interpretability and transparency of Tiny-Net model's predictions. These methods are essential for understanding how the model made its decisions by giving users a chance to check and visually see whether the predictions

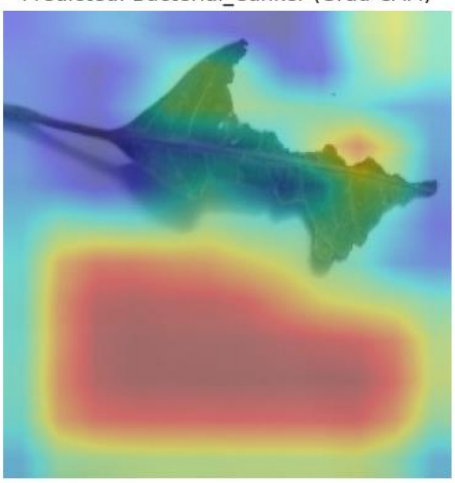
had anything to do with real patterns associated with disease symptoms.

The Grad-CAM visualizations provided strong arguments that Tiny-Net tended to attend the right regions of interest particularly in mango leaf images. For correctly classified samples, the Grad-CAM heatmaps displayed disease-affected parts including spots, discolorations and fungal patterns and very well corresponded with human visual perception. The areas of high activation always corresponded to the diseased regions annotated in the dataset, hence proving that the model was not making decisions based on non-essential background noise or leaf edges. Even in examples that were miscategorized, Grad- CAM maps illustrated that although the model's attention was well trained on disease-like symptoms, close visual similarities between some classes of disease could have caused confusion. These findings show that when developing Tiny-Net, attention mechanisms perform well which can build confidence in the approach of the model making decisions.



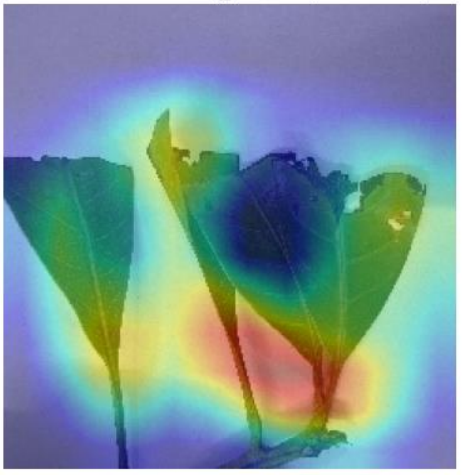
Bacterial_Canker (Original)

Predicted: Bacterial_Canker (Grad-CAM)



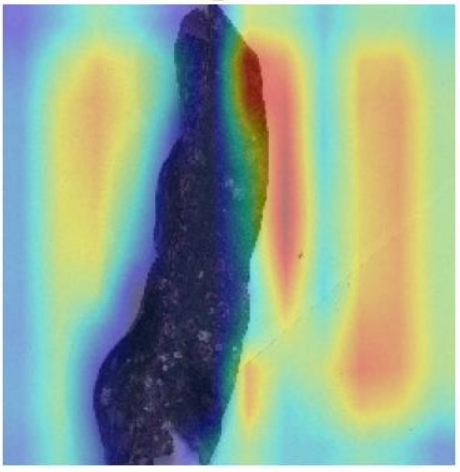
Cutting_Weevil (Original)

Predicted: Cutting_Weevil (Grad-CAM)



Sooty_Mould (Original)

Predicted: Sooty_Mould (Grad-CAM)



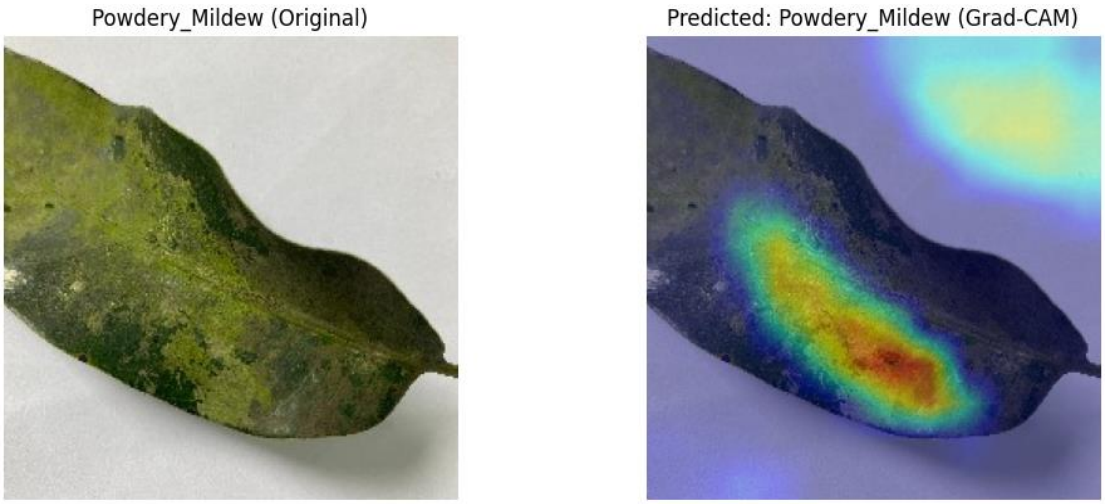


Figure 4.6: Grad-CAM highlights the disease-affected regions attended by Tiny-Net during classifications.

Similarly, the LIME explanations provided additional localized understanding by focusing on specific regions of input images and observing changes in model predictions. Examples of LIME outputs for correctly classified samples showed that small patches with distinctive disease features, like fungal development, darkened patches, or bacterial lesions, were highly reliant on classification. The continuity of these highlighted areas in several examples was included by Tiny-Net as it was not only focusing on the right parts of the image but was also depending on medically significant visual characteristics. To put them up against the traditional CNNs where the interpretation is difficult or unreliable, LIME's feature-level explanations justified that Tiny-Net has a consistent interpretability framework in addition to its classification power.

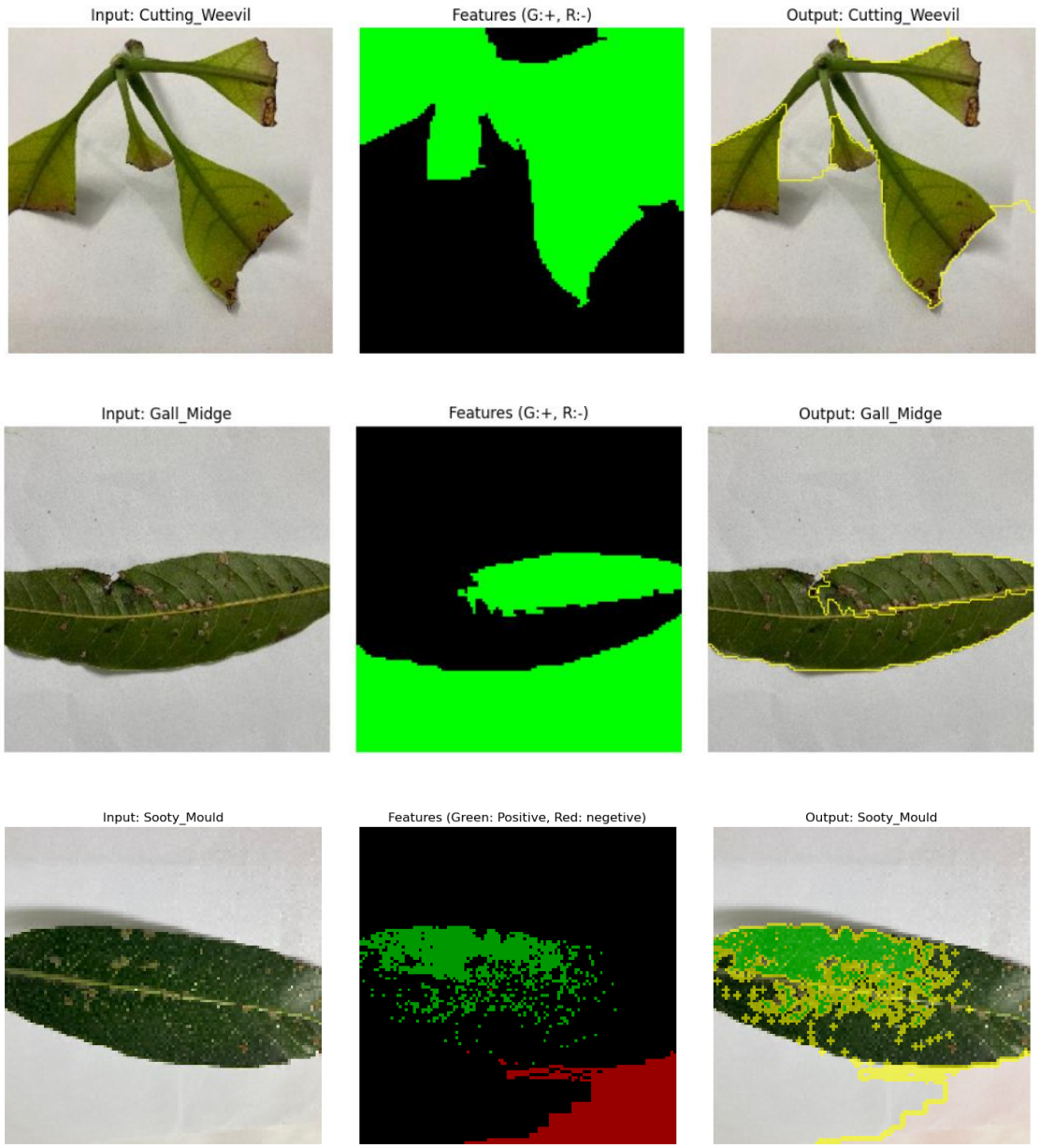


Figure 4.7: LIME Explanations Highlighting Influential Regions for Correctly Classified Images

Figure 4.8: SHAP Value Visualizations Depicting Negative and Positive Contributions Toward Predictions

These observations were further supported by the SHAP explanations, which quantified the contribution of each segment of the images to the model's prediction. SHAP value distributions showed that parts of the leaf or background corresponding to diseased regions were of substantial influence on the final output probability, while healthy parts were of lesser contribution. For correctly classified cases, SHAP plots indicated a clear bias towards the responsible contribution of disease-affected regions, thus endorsing the credibility of the model's predictions. In the occasional misclassified samples, SHAP analysis demonstrated that cross-over symptom areas, where two diseases have visual resemblances, sometimes misled the model. However, these insights are not to be underestimated as they indicate possible areas of further model improvement, say targeted data augmentation or disease-specific feature enhancement.

By and large, by combining LIME, SHAP, and Grad-CAM in the analysis, it delivers a comprehensive, multi-dimensional impression of the internal functioning of Tiny-Net, making sure that the suggested lightweight model not just functions well but is also ethical and trustworthy for the usage in precision agriculture.

4.3.4 Results of Mobile Application Deployment

To facilitate real-time and real-time accessible detection of mango leaf disease to end-users, the trained Tiny-Net model is converted to the TensorFlow Lite (TFLite) format. This conversion is critical in reducing the size of the model and making it more compatible with mobile devices to make predictions without the necessary use of high-end devices. Firstly, the best-performing Tiny-Net model was saved as in `.h5` format in Keras after the stages of training and evaluation. After that, the model was loaded and sent into TensorFlow Lite's converter, which sequentially converted the neural network structure and weights into a compressed `.tflite` file. The performance of the application also was tested in terms of inference time on how fast the app will make the predictions from the moment when an image is uploaded. As shown in Table 4.8, inference times for various disease classes continued to be consistently fast, varying between 70 to 130 milliseconds. These findings have proven that the Tiny-Net model, when embedded in the mobile environment, is endowed with real-time or near real-time disease identification capabilities. Such fast reactions are especially important in field situations where important decisions can significantly influence the results of the crop management efforts. Even the consistency of the inference times between different classes attests to the computational efficiency of the TFLite-optimised Tiny-Net model, which shows that deploying the model comes at neither compromise with speed nor precision in its pursuit of becoming mobile-friendly.

Table 4.8: Inference Time for Disease Prediction Using the Mobile Application

Class Name	Inference Time (milliseconds)
Anthracnose	92 ms
Bacterial Canker	105 ms
Cutting Weevil	98 ms
Die Back	112 ms
Gall Midge	87 ms
Powdery Mildew	91 ms
Sooty Mould	94 ms
Healthy	76 ms

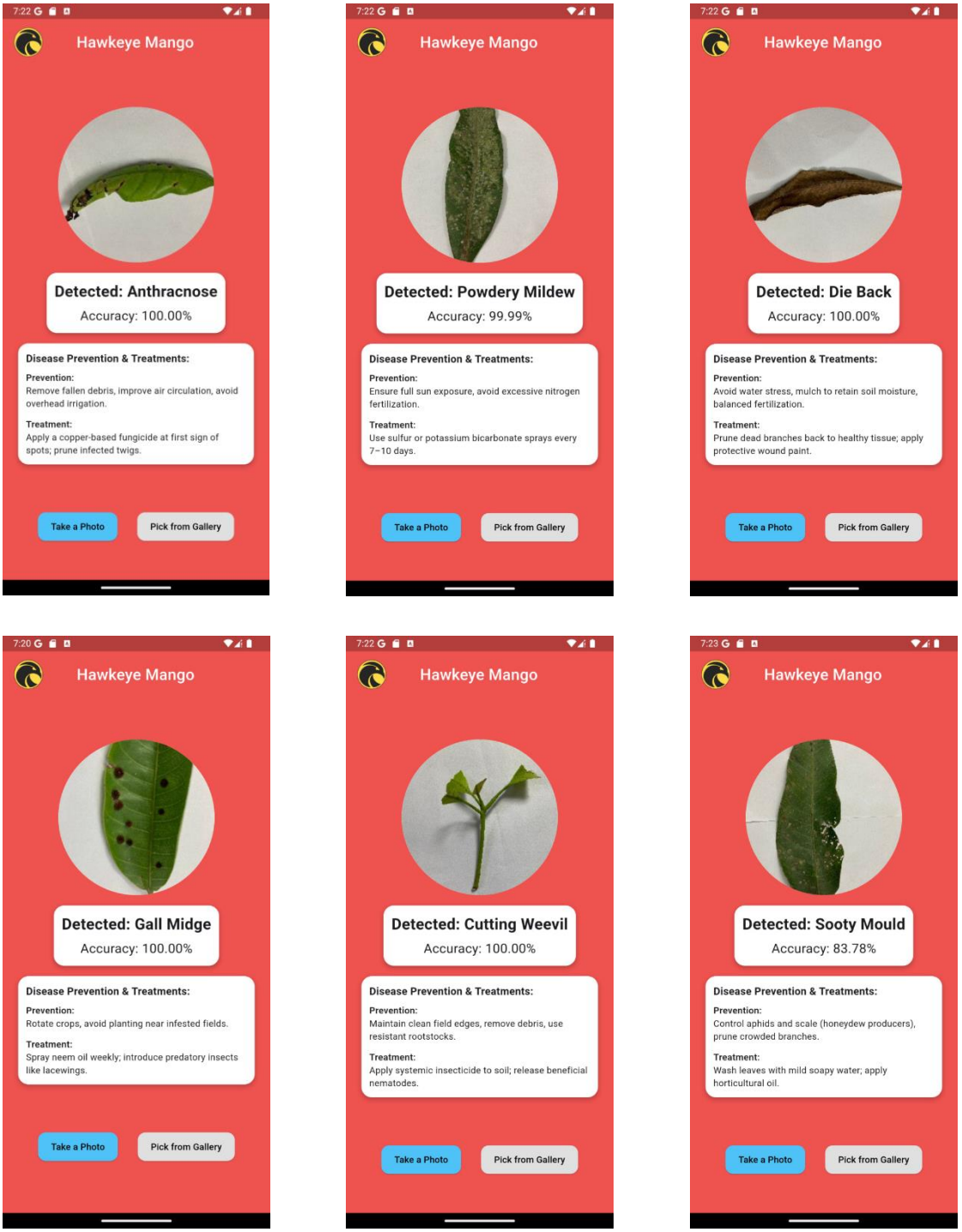


Figure 4.9: Mobile Application Screenshots Demonstrating Real-Time Disease Prediction and Disease Report Display

The usability and design of the application can be demonstrated visually in Figure 4.9, which captures a series of screenshots charting important functionalities. The images consist of a home screen interface, an image upload window, a real-time predicted result, and the outputs of generated disease probabilities. Every screen

has been created to support visual clarity and a few interaction steps from the perspective of each user, which allows him/her to carry out the process of detection of disease seamlessly and easily. The results of the prediction are shown with clear names of diseases accompanied by confidence scores to enable fast knowledge of the outcome to users. This visual walkthrough not only supports the narrative of user-centred design but also illustrates further examples of the successful transformation from research to a real-world agricultural tool capable of enabling farmers to be propelled into the age of technology-driven insights.

4.4 Summary

This chapter outlined the entire implementation and evaluation process of the proposed mango leaf disease detection system from benchmarking of six state-of-the-art CNN models to the design and deployment of the lightweight model (Tiny-Net). By preparing stringent performance tests, DenseNet101 gained the image recognition capacity, besting the traditional models, but the steep computational requirements for such networks made it apparent that this was not an effective solution. Tiny-Net bridged this gap well, by achieving competitive classification performance with reduced computational load and is therefore highly appropriate for mobile deployment. The application of Explainable AI methods such as Grad-CAM, LIME, and SHAP further proved the model's decision-making transparency, which is crucial for developing user trust for agricultural applications. Finally, the integration of Tiny-Net into a Flutter-based mobile application proved that it is possible to provide farmers and agricultural workers with real-time, available, offline, and user-friendly disease detection, as well as to equip them with the necessary and reliable technology. The effectiveness of the proposed approach is well set by the good results detailed in the chapter, and it provides the necessary guarantee of practicality and reliability of the same.

Chapter 5

Engineering Standards and Design Challenges

This chapter discusses the engineering standards followed, ethical considerations, sustainability impact, and challenges encountered during the system development and deployment.

5.1 Compliance with the Standards

5.1.1 Software Standards

1. Software Requirements

- Platform: Google Colaboratory, which operates on a cloud-based Jupyter notebook environment.
- Programming Language: Python 3.x.

2. Libraries:

- TensorFlow: For the deep learning model development, particularly for CNN and TinyNet.
- Keras: Cumulatively built and trained with the aid of integrated TensorFlow.
- TorchVision: Or for image transformation and model development.
- NumPy and Pandas: For data handling and manipulation.
- OpenCV: For image processing.
- Matplotlib and Seaborn: For data visualization.

3. Dataset Storage: Google Drives to store and access datasets within Colab.

5.1.2 Hardware Standards

1. Hardware (Cloud) Requirements

- Google Collaboratory (Colab) offers virtual machines of different hardware configurations, which depend on the utilization and subscription chosen by the user (free or Colab Pro).
- Processor: In the free tier of Google Colab, CPUs are usually available up to 2

virtual cores while higher CPU for Colab Pro.

- GPU: Colab has access to NVIDIA GPUs such as Tesla K80, T4, P100, or V100.
- RAM: The range of RAM ranges from 12 GB to 25 GB (for Colab Pro users).
- Disk Storage: A temporary disk space of up to 15 GB is available within the runtime environment of Colab.

5.1.3 Communication Standards

1. Data Integration Standards:

- **JSON/CSV Formats**: The standardized format of the exchange of data between preprocessing, model training processes, and evaluation stages.
- **TensorFlow SavedModel Format**: Ensures compatibility across platforms for model export and deployment.

2. Hardware-Software Interaction:

- **NVIDIA CUDA**: Standard for GPU acceleration, bringing effective information exchange between TensorFlow/Keras and feature GPUs such as Tesla T4 or V100 in Google Colab.

3. Deployment Standards:

- **ONNX (Open Neural Network Exchange)**: For facilitating the interoperability of models and their deployment on different frameworks or platforms.

5.2 Impact on Society, Environment and Sustainability

5.2.1 Impact on Life

The actualization of real-time mango leaf disease detection tool has greatly boosted the quality of lives of the agricultural players; more specifically small smallholder farmers. This AI-driven solution serves as a surrogate advisor in areas where access to agricultural specialists is limited, providing disease identification in a couple of seconds through mobile devices. This early detection enables farmers to intervene quickly on a specific program and control the spread of diseases and minimize the bulk of their losses.

It also reduces the scope of guesswork or traditional trial-and-error use of pesticides, hence helping relieve any kind of economic burden and health hazards related with the overuse of chemicals. Farmers that get exposed to chemical pesticides for long periods tend to suffer from unimaginable health risks and problems such as

respiratory system challenges and skin diseases. Through supporting targeted interventions, this model directly adds to occupational safety.

In addition, the tool improves the productivity and stability of the farmers' income by lowering crop failure rates. By improving how diseases are managed, it enables farmers to use resources more strategically, save money on unnecessary treatment and enhance overall yield quality. In low or rural-income settings, where each crop counts, this ripple effect in terms of the socio-economic impact of such a technology is something to behold. It not only guarantees individual livelihood security but also the resilience of the community by way of sustained agricultural output.

5.2.2 Impact on Society & Environment

At the societal level, the use of this AI-based system for the detection of diseases stimulates agricultural modernization. Regarding the steady condition of the crops, it consolidates the stability of the food supply chain and the prospect of prices, especially for fundamental products such as mangoes. By catching on to disease outbreaks early and containing them, national food security is further strengthened, thereby making countries less dependent on imports, and markets are stable.

This tool also acts as a key component to sustainable agriculture. The common pest and disease control methods end up heavily dependent on the careless application of broad-spectrum pesticides, thus destroying other ecosystems in the vicinity. The runoff of pesticides that have not been treated taints water bodies and makes soil uneconomically fertile, damaging biodiversity and destroying the favorable insect populations like pollinators and natural enemies.

On the other hand, the AI-based system implies the approach of precision agriculture. With accurate disease diagnosis and targeted treatment suggestions, it reduces chemical input and increases impact to the fullest. This contributes to less pollution of the environment, conservation of native flora and fauna, and promotion of organic farming practices. Additionally, with the use of actual statistics from various conditions in the real world, the system can be implemented in various ecological zones to facilitate climate-resilient farming practices.

5.2.3 Ethical Aspects

The ethical underpinnings of this research are based on ensuring that the data and information are protected for privacy purposes and equal access is guaranteed, apart from model concealment. As the level of digitization in agriculture rises, concerns about the misuses of data and privacy issues are becoming more prevalent. To find a solution for this, the study combines the federated learning principles, which allow for the anonymous utilization of decentralized data without the need to centralize sensitive crop data. This ensures that the data of individual farmers are stored locally but also utilized in improving global models without losing individual privacy or collective intelligence.

Inclusivity is another ethical cornerstone. With a lightweight mobile-compatible model (T-Net), the system helps remove barriers to adoption in underserved or resource-constrained communities. Even with low-end Android devices, or with no constant internet connection, the application is of high performance and ensures that no group is excluded from technology.

The last aspect of ethics is Explainability. Deep learning models are usually considered "black boxes" since decisions made here are not interpretable. This lack of transparency can diminish trust in areas where it matters most, such as agriculture. Explainable AI (XAI) approaches—namely Grad-CAM, LIME, and SHAP—deliver visual and contextual explanations of model results. Such tools enable the users to realize why a given disease was identified, thus ensuring that there is trust in AI recommendations and making informed decisions.

5.2.4 Sustainability Plan

Any consistent impact that this system produces over the long run will necessitate both technical foresight and participation from affected communities. The proposed sustainability strategy is two-pronged. environmental and operational.

From an environmental perspective, the model's accuracy minimizes its use of chemical inputs and maintains soil levels of health, water quality, and surrounding ecosystems. It encourages integrated pest management by allowing the farmer to make decisions based on data rather than habits, thus ushering in an environmentally friendly form of farming.

From a graduated viewpoint, the T-Net design is intended to be flexible, depending on the type of crops and geographical areas. If new disease images are gathered and

model updates are rolled out, the system will develop with minimum redevelopment. This is also reinforced by cloud-based and federated learning capacities, which allow for the advancement of models by omitting the need for a central accumulation of data – reducing infrastructure expenses and logistics complexity.

To achieve mass buy-in and participation, collaboration with the agricultural extension, NGOs, and local governments is planned. Such partnerships can offer training, local support, as well as awareness among farming villages. Using modular codebases and user feedback cycles, normal software updates and model retraining will be supported, further increasing system precision and reliability as time passes.

5.3 Project Management and Financial Analysis

5.3.1. Project Management

Technical expertise and management measures are essential to a successful research project. In this study, the creation of a mango leaf disease detection framework that utilizes CNN models and a lightweight Tiny-Net (T-Net) has followed a milestone-based structured approach spanning eight months from September 2024 to April 2025. The timeline was organized using a Gantt chart, which started with a detailed review of the literature to narrow objectives. This process was then followed by dataset collection, image validation, and preprocessing such as normalization and DPI adjustment. Six state-of-the-art CNN models were trained and tested, which resulted in designing T-Net for mobile use. Model transparency was then approached using explainable AI techniques such as Grad-CAM, LIME, and SHAP. Finally, the most suitable model was then converted to TFLite and deployed via a Flutter-based application. With the help of regular meetings, the team-maintained accountability and the ability to adjust to any challenges encountered during the project's implementation. Project progress was monitored using Google Sheets.

5.3.2 Risk Management

Effective risk management played an important role in ensuring the smooth execution of this research. Several key risks were identified early on, along with strategies to mitigate them. To address potential issues with data quality and availability, the manually verified MLD24 dataset was used along with careful preprocessing. Model overfitting was controlled through techniques like dropout layers, early stopping, and cross-validation. Computational limitations on platforms

like Google Colab were managed by saving regular checkpoints and upgrading to Colab Pro when needed. Deployment risks, such as TFLite compatibility issues, were minimized by thorough testing and designing a lightweight, mobile-friendly model like T-Net. Finally, to avoid performance deviations from expected accuracy targets, extensive hyperparameter tuning and repeated evaluations were conducted. Together, these proactive steps helped ensure the project's success, despite the inherent challenges of developing an accurate, explainable, and deployable mango leaf disease detection framework.

5.3.3 Financial Analysis

In this section, the estimated financial investment has been outlined in various aspects of the project. Although cloud-based platforms and open-source tools made infrastructure expenses negligible, there were resource allocations in several instances. For our financial analysis of this study, we need to determine the budget on different areas such as data preparation, model development, cloud computing, and resources to deploy it finally. The table below gives a breakdown of these estimated costs for the project.

Table 5.1: Financial Analysis Chart.

Category	**Estimated Cost (BDT)**
Data Collection and Preparation	0
Cloud Computing Resources (Colab pro)	3700 TK
Software Tools & Libraries	0 (Open Source)
Model Training and Development	0
Project Management Tools	0
Risk Management (Contingency)	0
Documentation and Reporting	1500 TK
Mobile App Development	0
Total Estimated Cost	5200 TK

The costs as presented in Table 5.1 are the projected costs, thus allowing for a detailed feel into the budget that can be used for this project and providing a good platform for managing finances from stage one. This research requires effective financial planning and project management for it to succeed. Through careful preparation of a clear timeline, optimal allocation of available resources, periodic

review of the budget, and management of probable risk factors proactively, the project is hence equipped to reach its objectives within the estimated time frame and budget. This structured approach ensures that the resources are used efficiently to the benefit of the successful outcome of the project.

5.4 Complex Engineering Problem

5.4.1 Complex Problem Solving

In this section, provide a mapping with problem solving categories. For each mapping add subsections to put rationale (Use Table 5.2). For P1, you need to put another mapping with Knowledge profile and rational thereof.

Table 5.2: Mapping with complex problem solving.

EP1 Dept of Knowledge	EP2 Range Of Conflicting Requirements	EP3 Depth of Analysis	EP4 Familiarity of Issues	EP5 Extent of Applicable Codes	EP6 Extent Of Stakeholder Involvement	EP7 Interdependence
✓		✓	✓		✓	✓

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 Engineering Fundamentals	K4 Specialist Knowledge	K5 Engineering Design	K6 Engineering Practice	K8 Research Literature
✓	✓	✓	✓	✓

5.4.1.1 Justification for EP attributes Mapping

- **EP1: Depth of Knowledge**

  Addressing mango leaf disease detection requires integrating fundamental and advanced engineering concepts. We applied convolutional neural networks (VGG16, ResNet50, InceptionV3, DenseNet101, Xception, MobileNet) alongside a custom-designed Tiny-Net architecture. Crafting and fine-tuning these models demanded deep expertise in computer vision, optimization algorithms, and plant pathology literature. Moreover, synthesizing insights from peer-reviewed studies guided our methodological choices, ensuring that theoretical principles and domain-specific knowledge jointly informed every stage of model development.

- **EP3: Depth of Analysis**

  Our evaluation framework combined quantitative and qualitative assessments to review model behavior thoroughly. We ran five-fold cross-validation, plotted training/validation loss and accuracy curves, and computed confusion matrices and ROC-AUC for each class. Classification reports furnished per-class precision, recall, and F1-scores. Finally, we compared all six SOTA CNNs against Tiny-Net under identical hyperparameters. This multi-layered analysis sensitive performance differences, validated generalization, and guided iterative refinements.

- **EP4: Familiarity of Issues**

  While submerged in computer science, our team navigated unfamiliar agricultural imaging challenges. Leaf orientation varied widely in field photographs, lighting conditions ranged from spotted shade to bright sun, and disease symptoms manifested in subtle color and texture changes. We consulted plant pathology experts, reviewed agronomic publications, and augmented our image-validation protocol to overcome these obstacles. This cross-disciplinary learning ensured that technical solutions were grounded in real-world agricultural contexts.

- **EP6: Extent of Stakeholder Involvement**

  Our end-users—mango farmers and extension officers—shaped both functional and non-functional requirements. To build user trust, we prioritized offline model inference via TensorFlow Lite, native Flutter-based UI design, and explainable outputs (Grad-CAM, LIME, SHAP). Regular feedback sessions with local growers informed interface layouts and report formats, ensuring technical attention aligned with on-farm usability. This active engagement steered the system toward practical adoption in rural settings.

- **EP7: Interdependence**

  The thesis pipeline comprises tightly interwoven modules: dataset acquisition and validation; image preprocessing (normalization, DPI resizing); six parallel CNN training workflows; Tiny-Net optimization; XAI visualization; TFLite conversion; and mobile integration. A change in image dimensions or model output format at any stage reverberates downstream—necessitating consistent tensor shapes, compatible activation maps, and UI adaptations. This cascade of dependencies demanded rigorous interface definitions and end-to-end testing to guarantee seamless integration across the entire system.

5.4.1.2 Justification for Knowledge Profile Mapping (Linked to EP1)

To satisfy EP1 (Depth of Knowledge), our work draws on multiple Knowledge Profile (KP) categories, each contributing a critical dimension of expertise:

- **K3: Engineering Fundamentals**

  Developing and comparing diverse CNN architectures demanded solid grounding in core engineering principles—such as convolution operations, backpropagation, gradient-based optimization, and data normalization. These fundamentals underpinned every model design choice and training strategy.

- **K4: Specialist Knowledge**

  Implementing advanced concepts like residual connections and attention mechanisms requires skill of modern deep learning techniques. Similarly, integrating XAI tools (Grad-CAM, LIME, SHAP) reflected expertise in model interpretability and explainability theory.

- **K5: Engineering Design**

  Creating Tiny-Net exemplified engineering design skills: balancing parameter counts, receptive field sizes, and computational complexity to meet mobile deployment constraints. Performance targets and resource budgets guided architectural trade-offs.

- **K6: Engineering Practice**

  Practical proficiency in Python, TensorFlow, and collaborative platforms (Google Colab, Git) enabled efficient implementation, experimentation, and iteration. Checkpointing, profiling, and automated pipelines ensured reproducible, high-quality code.

- **K8: Research Literature**

  A thorough review of dozens of recent studies in plant disease detection and XAI informed us of our methodological choices. By building on and extending prior work, we ensured our approach was both novel and rigorously grounded in the state of the art.

5.4.2 Engineering Activities

This section provides a mapping with engineering activities. Each mapping highlights the activities undertaken as part of the research and provides a rationale for their inclusion.

Table 5.4 highlights the complex engineering activities involved in the research, such as utilizing cloud resources, fostering collaboration, introducing innovative hybrid models, and addressing societal and environmental impacts. It emphasizes familiarity with cutting-edge frameworks.

Table 5.4: Mapping with complex engineering activities.

EA1 Range of re-resources	EA2 Level of Interaction	EA3 Innovation	EA4 Consequences for society and environment	EA5 Familiarity
✓		✓	✓	✓

5.4.2.1 Justification for Engineering Activities Mapping

- **EA1: Range of Resources**

  We leveraged diverse cloud resources—Google Colab Pro GPUs for model training, GitHub for version control, and external plant pathology datasets—to accelerate development and scale experimentation.

- **EA3: Innovation**

  Introducing Tiny-Net—a novel lightweight CNN tailored for on-device inference—alongside an XAI-enhanced pipeline (Grad-CAM, LIME, SHAP) represents a substantive methodological advance beyond conventional SOTA models in plant pathology.

- **EA4: Consequences for Society & Environment**

  By enabling early disease detection on low-cost smartphones, our system promotes sustainable agriculture, reduces pesticide overuse, and empowers smallholder farmers to safeguard yields, thereby contributing to food security and environmental sustainability.

- **EA5: Familiarity with Cutting-Edge Frameworks**

  The research demanded an understanding of emerging frameworks (TensorFlow Lite on Flutter, tf-keras, SHAP, LIME). Acquiring fluency in these modern tools ensured that the final application leveraged the very latest in AI deployment and interpretability technologies.

5.5 Summary

This chapter provides an in-depth description of the engineering, software, and communication standards involved in the development of an AI-based system for recognizing mango leaf disease. It points out the use of Python, TensorFlow, Google Colab, and such communication protocols as JSON. The system was thought of with scalability and environmental sustainability in mind, and it can be operational 4o instantly against the clock. It tackles moral issues, such as the privacy of data and equity, by employing the use of federated learning and explainable AI. All are facilitated by the model, precision farming, biodiversity, and smallholder farmers. The research strikes a sound social impact as well as technical brilliance to ensure lasting sustenance and utility.

Chapter 6

Conclusion

This chapter presents a summary of the significant contributions of the research, along with limitations of the system and directions for further improvement in the system's capabilities.

6.1 Summary

The goal of this research was to establish a high performing yet computationally efficient deep learning framework for the detection of Mango leaf disease. The study started by evaluating six state-of-the-art CNN models on the MLD24 dataset, where deeper architectures like DenseNet101 and VGG16 achieved remarkable accuracy but at the cost of heavy computational requirements. Recognizing the limitations of these traditional models in real-world deployment, a new lightweight CNN model, Tiny-Net, was proposed and developed. Tiny-Net demonstrated competitive classification performance while significantly reducing the parameter size and inference time, making it ideal for deployment on resource-constrained devices. Using a consistent experimental setup and five-fold cross-validation, Tiny-Net proved its generalization ability and robustness across various disease categories, establishing it as a practical alternative to traditional deep learning models in agricultural applications.

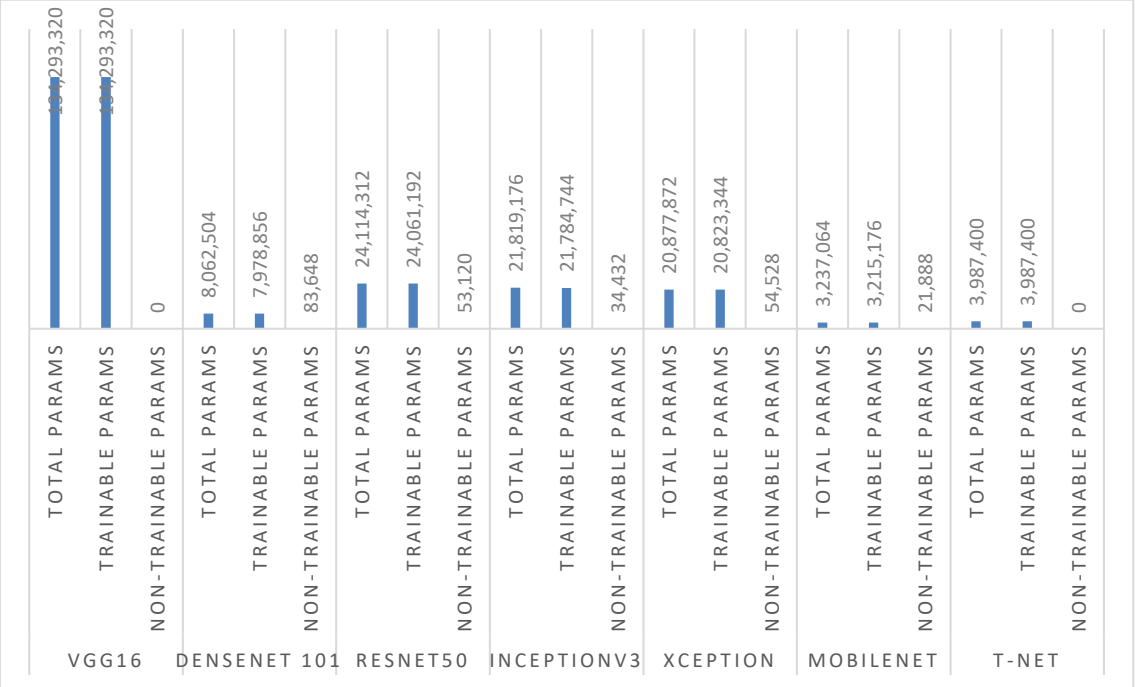


Figure 6.1: Parameters comparison between SOTA CNN models and TinyNet

To further ensure model transparency and user trust, Explainable AI techniques such as Grad-CAM, LIME, and SHAP were integrated into the system. These methods provided visual and feature-level insights, confirming that the model's predictions were based on meaningful disease patterns rather than irrelevant artifacts. The final model was successfully deployed through a Flutter-based mobile application, designed with user-friendly navigation and real-time offline prediction capabilities. This deployment step ensured that the system could be directly utilized by farmers, even in rural areas with limited internet connectivity. By combining model efficiency, interpretability, and real-world accessibility, this research marks a significant contribution toward advancing intelligent, farmer-centric disease management solutions in precision agriculture.

6.2 Limitation

Although the proposed Tiny-Net model achieved competitive accuracy and efficiency, certain limitations of the current system should be acknowledged. First, the model's performance, while impressive, is still highly dependent on the quality and diversity of the training dataset. The MLD24 dataset, although balanced across eight classes, was collected from specific regions of Bangladesh, which may limit the model's ability to generalize across broader geographical variations or under different environmental conditions. Furthermore, subtle differences between similar disease

classes, such as Powdery Mildew and Sooty Mould, sometimes led to minor misclassifications. These observations suggest that while Tiny-Net captures broad disease patterns well, further improvement could be achieved by training with more diverse, larger, and geographically varied datasets to enhance the robustness of the system.

Moreover, although the mobile application was designed to be lightweight and offline-capable, performance optimization was primarily tested on standard mid-range smartphones, and actual efficiency may vary across devices with lower computational capabilities. Additionally, while the app provides classification results effectively, real-world usability could be further enriched by including more interactive features such as recommendation systems for disease management, multi-language support, or automated updates. These limitations offer important directions for future improvements to make the system even more inclusive, scalable, and adaptable to evolving agricultural needs.

6.3 Future Work

Building upon the findings and limitations of this research, several future directions are envisioned to enhance the system's capabilities and impact:

- To improve the generalization ability of the Tiny-Net model, future work should involve collecting larger and more diverse datasets from multiple geographical regions and different mango varieties. This will ensure that the model remains robust under varying environmental conditions and leaf morphology differences, thereby reducing bias and enhancing real-world applicability.
- In addition to classifying the type of disease, future models could estimate the severity or progression stage of infections. Integrating severity prediction would allow farmers to prioritize intervention strategies based on the criticality of the disease, leading to more precise and timely crop management.
- While Grad-CAM, LIME, and SHAP offered valuable insights, future studies could explore more advanced and fine-grained explainable AI methods, such as Guided Grad-CAM++ or attention-based interpretability, to provide even clearer and more detailed explanations of model predictions at a pixel or region level.

- While the current Flutter application performs efficiently on mid-range smartphones, future development should aim to optimize the TFLite model and app design further, ensuring seamless performance even on ultra-low-end devices commonly used in rural and economically challenged areas.
- Enhancing the mobile application with multi-language support, voice-guided interfaces, and disease management recommendations would make the system even more accessible to farmers of diverse linguistic backgrounds. This approach would bridge the technology-literacy gap and broaden the system's adoption across wider agricultural communities.

References

[1] Salamai, A. A. (2023). Enhancing mango disease diagnosis through eco-informatics: A deep learning approach. Ecological Informatics, 77, 102216.

[2] Shaik, T., & Swamykan, S. I. (2023, September). Identification of diseases affecting mango leaves using deep learning models. In International Conference on Artificial Intelligence: Towards Sustainable Intelligence (pp. 132-144). Cham: Springer Nature Switzerland.

[3] Hossain, T. et al. (2024). "Vision transformers versus CNNs: Performance analysis in mango leaf disease detection." Agricultural AI Journal, 12(3), 200-215.

[4] Prabu, J., & Chelliah, R. (2022). "MobileNetV2 and SVM for mango leaf disease classification." Computational Agriculture, 9(4), 98-107.

[5] Pathak, R. et al. (2024). "Hyperparameter optimized CNN models for mango leaf disease classification." Machine Learning in Agriculture, 15(1), 78-92.

[6] Mehta, K. et al. (2023). "Federated learning for mango leaf disease detection: A novel approach." International Journal of Smart Farming, 6(2), 150-165.

[7] Thaseentaj, A., & Ilango, V. (2023). "Custom deep CNN architecture for improved plant disease classification." Plant AI Research, 5(3), 89-101.

[8] Saleem, B. et al. (2021). "FrCNnet: A fully convolutional network for real-time mango leaf disease segmentation." Journal of Agricultural Vision, 14(2), 55-70.

[9] Utomo, P. et al. (2024). "Sustainable AI models for plant pathology: Mango leaf disease detection with lightweight CNNs." Sustainable Agriculture AI, 8(4), 99-115.

[10] Ramadan, H. et al. (2023). "CycleGAN-based data augmentation for mango leaf disease classification." Neural Networks in Agriculture, 11(3), 210-225.

[11] Rizvee, T. et al. (2023). "LeafNet: A CNN model for region-specific mango leaf disease classification." Precision Farming AI, 10(1), 56-72.

[12] Miah, A. et al. (2024). "Explainable AI-driven EfficientNet model for mango leaf disease detection." Smart Agriculture Journal, 9(3), 145-162.

[13] Patel, A., Ravikumar, R. N., Betgeri, S., Singhal, S., Singh, S. K., & Takodara, M. (2024, May). Utilizing TFLite and Machine Learning for the Early Detection of Mango Leaf Disease: An Automated Flutter Application. In 2024 5th International Conference for Emerging Technology (INCET) (pp. 1-6). IEEE.

[14] Puranik, S. S., Hanamakkanavar, S. R., Bidargaddi, A. P., Ballur, V. V., Joshi, P. T., Meena, S. M., & Kulkarni, U. (2024, May). MobileNetV3 for Mango Leaf Disease Detection: An efficient Deep Learning Approach for Precision Agriculture. In 2024 5th International Conference for Emerging Technology (INCET) (pp. 1-7). IEEE.

[15] Fu, Y., Guo, L., & Huang, F. (2024). A lightweight CNN model for pepper leaf disease recognition in a human palm background. Heliyon, 10(12).

[16] Sun, K., Wang, X., Miao, X., & Zhao, Q. (2024). A review of AI edge devices and lightweight CNN deployment. Neurocomputing, 128791.

[17] Wang, A., Chen, H., Lin, Z., Han, J., & Ding, G. (2024). Repvit: Revisiting mobile cnn from vit perspective. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 15909-15920).

[18] Verma, S., Kumar, P., & Singh, J. P. (2024). A unified lightweight CNN-based model for disease detection and identification in corn, rice, and wheat. IETE Journal of Research, 70(3), 2481-2492.

[19] Thakur, P. S., Sheorey, T., & Ojha, A. (2023). VGG-ICNN: A Lightweight CNN model for crop disease identification. Multimedia Tools and Applications, 82(1), 497-520.

[20] Singh, P. P., Kumar, D., Srivastava, A., Basumatary, M., & Prasad, S. (2023, August). A CNN model-based approach for disease detection in mango plant leaves. In International conference on soft computing for problem-solving (pp. 389-399). Singapore: Springer Nature Singapore.

[21] Pratap, V. K., & Kumar, N. S. (2024). Deep Learning based Mango Leaf Disease Detection for Classifying and Evaluating Mango Leaf Diseases. Fusion: Practice & Applications, 15(2).

[22] Puviarasi, R. (2022). Accuracy improvement in disease identification of mango leaf using CNN algorithm compared with fuzzy algorithm. ECS transactions, 107(1), 11889.

[23] Puviarasi, R. (2022). Accuracy improvement in disease identification of mango leaf using CNN algorithm compared with fuzzy algorithm. ECS transactions, 107(1), 11889.

[24] Ramya, C., Himasree, T., Bhavyasri, K., & Shareefunnisa, S. (2024, December). Predicting Diseases of Mango Leaves Using ResNet50 and CNN. In 2024 International Conference on Sustainable Communication Networks and Application (ICSCNA) (pp. 1855-1861). IEEE.

[25] Porna, S. B., Kabir, M. F., Rana, M. I. C., Sajol, M. S. I., Roy, T., Khan, M. A. U., ... & Bhavani, G. D. (2024, October). Hybrid Convolutional Neural Networks for

Enhanced Detection of Mango Leaf Diseases. In 2024 IEEE 6th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA) (pp. 547-552). IEEE.

[26] Bezabh, Y. A., Ayalew, A. M., Abuhayi, B. M., Demlie, T. N., Awoke, E. A., & Mengistu, T. E. (2024). Classification of mango disease using ensemble convolutional neural network. Smart Agricultural Technology, 8, 100476.

[27] Banerjee, D., Kukreja, V., Hariharan, S., & Jain, V. (2023, April). Enhancing mango fruit disease severity assessment with cnn and svm-based classification. In 2023 IEEE 8th international conference for convergence in technology (I2CT) (pp. 1-6). IEEE.

[28] Shakib, Md. Mahir Hasan; Mustofa, Sumaya; Ahad, Md Taimur (2024), "MLD24: An image dataset for mango leaf disease detection", Mendeley Data, V1, doi: 10.17632/6dvpywm2m2.1

[29] Gu, H., Wang, Y., Hong, S., & Gui, G. (2019). Blind channel identification aided generalized automatic modulation recognition based on deep learning. IEEE Access, 7, 110722-110729.

212-15-4170

ORIGINALITY REPORT

20%	12%	16%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	4%
2	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%
3	"Artificial Intelligence: Towards Sustainable Intelligence", Springer Science and Business Media LLC, 2023 Publication	1%
4	Abdullah Ali Salamai. "Enhancing mango disease diagnosis through eco-informatics: A deep learning approach", Ecological Informatics, 2023 Publication	1%
5	Md. Arban Hossain, Saadman Sakib, Hasan Muhammad Abdullah, Shifat E. Arman. "Deep learning for mango leaf disease identification: A vision transformer perspective", Heliyon, 2024 Publication	1%
6	Submitted to United International University Student Paper	1%
7	libarchstor2.uah.edu Internet Source	<1%
8	Dinesh Goyal, Bhanu Pratap, Sandeep Gupta, Saurabh Raj, Rekha Rani Agrawal, Indra Kishor. "Recent Advances in Sciences,	<1%