Lightweight CNN for Plant Disease Detection: Transfer Learning and XAI

By

Md. Najmul Parves	ID: 21222203005
Md. Abdullah Al Mamun	ID: 21222203017
Sagar Saha	ID: 21222203029
Aktaruzzaman	ID: 21222203031
Siam Hossain	ID: 21222203037

Submitted in partial fulfilment of the requirements of the degree of
Bachelor of Science in **Computer Science and Engineering**

Department of Computer Science and Engineering
Bangladesh University of Business and Technology

February 2026

Declaration

We do hereby declare that the research works presented in this thesis entitled, **"Lightweight CNN for Plant Disease Detection: Transfer Learning and XAI"** are the results of our works. We further declare that the thesis has been compiled and written by us, and no part of this thesis has been submitted elsewhere for the requirements of any degree, award, diploma, or any other purposes except for publications. The materials that are obtained from other sources are duly acknowledged in this thesis.

Md. Najmul Parves
ID: 21222203005

Aktaruzzaman
ID: 21222203031

Md. Abdullah Al Mamun
ID: 21222203017

Siam Hossain
ID: 21222203037

Sagar Saha
ID: 21222203029

Approval

I do hereby declare that the research work presented in this thesis entitled **"Lightweight CNN for Plant Disease Detection: Transfer Learning and XAI"** is the outcome of the original work carried out by **Md. Najmul Parves, Md. Abdullah Al Mamun, Sagar Saha, Aktaruzzaman, and Siam Hossain** under my supervision. I further declare that no part of this thesis has been submitted elsewhere for the requirements of any degree, award, diploma, or other purposes except for academic publication. I also certify that this thesis meets the requirements and academic standards for the degree of **Bachelor of Science in Computer Science and Engineering**.

Supervisor
Md. Sahiduzzaman
Assistant Professor
Department of Computer Science & Engineering
Bangladesh University of Business and Technology
Dhaka, Bangladesh

Chairman
Prof. Dr. Md. Ahsan Habib
Professor
Department of Computer Science & Engineering
Bangladesh University of Business and Technology
Dhaka, Bangladesh

Dedication

We would like to dedicate this research to our loving parents, whose constant love, sacrifices, and unwavering support have shaped our journey and given us the strength to pursue our dreams. Their encouragement and belief in our potential have been a source of motivation throughout our academic life.

Acknowledgment

First and foremost, we express our heartfelt gratitude to the Almighty Allah for His endless blessings, guidance, and mercy that enabled us and our families to complete this research successfully.

We are deeply grateful to the **Bangladesh University of Business and Technology (BUBT)** for providing us with an excellent academic environment and all necessary facilities to pursue our research work.

We would like to convey our sincere appreciation to **Md. Sahiduzzaman**, *Assistant Professor*, Department of Computer Science and Engineering, BUBT, for her constant guidance, invaluable suggestions, and dedicated support throughout the entire research process.

The selection of the research topic, problem identification, and the successful completion of this work were made possible through her insightful supervision and encouragement. His patience, availability, and constructive feedback greatly contributed to the quality of this thesis.

We are also sincerely thankful to **Prof. Dr. Md. Ahsan Habib,** *Chairman*, Department of Computer Science and Engineering, and to all our respected teachers of the Department of CSE, BUBT, for their continuous inspiration, academic guidance, and for providing us with a strong foundation in our studies and research. Finally, we would like to extend our appreciation to our fellow team members for their cooperation and support, and to our university for giving us the opportunity to conduct this research and complete our thesis successfully.

Abstract

Early and accurate diagnosis of plant diseases is pivotal for ensuring global food security and minimizing agricultural economic losses. While Deep Learning (DL) techniques have demonstrated remarkable potential in automated disease detection, the deployment of such models on resource-constrained edge devices remains a challenge due to high computational costs and memory requirements. Additionally, the "black-box" nature of deep neural networks often limits their trustworthiness among end-users. This study proposes a **custom lightweight Convolutional Neural Network (CNN)** optimized for efficient and accurate classification of 38 distinct plant disease classes using the PlantVillage dataset. The performance of the proposed model was benchmarked against state-of-the-art Transfer Learning architectures, specifically **ResNet50** and **MobileNetV2**. The experimental results demonstrate that the proposed Custom CNN achieved a superior testing accuracy of **98.71%**, outperforming MobileNetV2 and maintaining competitive performance with the heavier ResNet50 model while significantly reducing computational complexity. Furthermore, to ensure transparency and reliability, **Explainable AI (XAI)** techniques, specifically Gradient-weighted Class Activation Mapping (Grad-CAM), were integrated. The XAI visualizations confirmed that the model learns from relevant leaf lesion regions rather than background noise. This research presents a robust, efficient, and interpretable solution suitable for real-time deployment in mobile applications for precision agriculture.

List of Figures

List of Tables

List of Abbreviations and Acronyms

Abbreviation	Full Form / Meaning
AI	Artificial Intelligence
AUC	Area Under the Curve
Adam	Adaptive Moment Estimation (Optimizer)
BN	Batch Normalization
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GAP	Global Average Pooling
Grad-CAM	Gradient-weighted Class Activation Mapping
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ML	Machine Learning
MB	Megabyte
MobileNetV2	Mobile Network Version 2
ReLU	Rectified Linear Unit
ResNet	Residual Network
RGB	Red, Green, Blue (Color Model)
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TL	Transfer Learning
TN	True Negative
TP	True Positive
TPR	True Positive Rate
XAI	Explainable Artificial Intelligence

Contents

CHAPTER 1
INTRODUCTION

1.1 Introduction

Agriculture plays a pivotal role in the economic development of many countries and serves as the primary source of food for the growing global population. As the demand for food supply increases, ensuring the health and quality of crops has become a major concern [1]. However, plant diseases remain one of the most significant threats to food security and agricultural productivity. According to the Food and Agriculture Organization (FAO), plant pests and diseases contribute to an estimated loss of 20-40% of global crop yields annually [2]. Furthermore, pathogens and pests burden major food crops like wheat, rice, and maize, threatening the livelihood of millions of farmers [3].

Traditionally, the identification of plant diseases relies on visual inspection by agricultural experts or biological laboratory tests. These manual methods are labor-intensive, time-consuming, and prone to human error [4]. Moreover, the scarcity of experts in remote rural areas makes timely diagnosis difficult for farmers. In recent years, Artificial Intelligence (AI), specifically Deep Learning (DL) and Computer Vision (CV), has emerged as a powerful tool to automate this process. Convolutional Neural Networks (CNNs) have shown remarkable success in image recognition tasks, enabling the automated detection of plant diseases from leaf images [5]. This research focuses on developing an efficient, automated, and trustworthy system for plant disease detection using advanced Deep Learning techniques.

1.2 Problem Statement

Despite the success of Deep Learning in agriculture, several challenges hinder the widespread adoption of these technologies in real-world scenarios:

1. **Computational Complexity:** Most state-of-the-art models (like VGG16 or ResNet) are computationally expensive and require large memory storage. This makes them unsuitable for deployment on resource-constrained devices such as mobile phones or IoT edge devices used by farmers in the field [6].
2. **Lack of Trust (The Black-Box Problem):** Deep Learning models operate as "black boxes," meaning they provide a prediction without explaining why they made that decision. For a farmer or an agronomist, it is difficult to trust a model’s diagnosis without visual evidence [7].
3. **Efficiency vs. Accuracy Trade-off:** Lightweight models (like MobileNet) are faster but often sacrifice accuracy compared to larger models. There is a need for a custom architecture that balances high accuracy with low computational cost [8].

This research addresses these problems by proposing a **Custom Lightweight CNN** that achieves high accuracy with minimal parameters and integrating **Explainable AI (XAI)** to make the model's decisions transparent.

1.3 Problem Background

In the early stages of automated disease detection, researchers utilized traditional Machine Learning techniques such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) combined with hand-crafted features (color, texture, shape) [9]. For instance, Al-Hiary et al. used K-Means clustering for segmentation, while others relied on texture features like GLCM [10, 11]. However, these methods struggled with complex backgrounds and varying lighting conditions.

With the advent of Deep Learning, CNNs replaced traditional methods due to their ability to automatically learn feature representations from raw data [12]. Transfer Learning, using pre-trained models like Inception, ResNet, and DenseNet, became popular for achieving high accuracy [13, 14]. However, the blind application of these heavy models ignores the practical constraints of deployment speed and device battery life. Furthermore, purely accuracy-driven research often neglects the interpretability of the model, leading to systems that might learn from irrelevant background noise rather than the actual disease symptoms [15]. This research bridges these gaps by focusing on efficiency and explainability simultaneously.

1.4 Research Objectives

The primary aim of this thesis is to develop an efficient and interpretable deep learning framework for plant disease detection. The specific objectives are:

- To design and develop a **Custom Lightweight CNN** architecture optimized for classifying 38 classes of plant diseases from the PlantVillage dataset [16].
- To compare the performance of the proposed custom model against established Transfer Learning models, specifically **ResNet50** (Heavyweight) and **MobileNetV2** (Lightweight), in terms of accuracy, loss, model size, and inference speed.
- To implement **Explainable AI (XAI)** using Gradient-weighted Class Activation Mapping (**Grad-CAM**) to visualize the regions of interest in the leaf images, ensuring the model's reliability [17].
- To analyze the efficiency trade-offs to determine the most suitable model for deployment in real-world agricultural settings.

1.5 Motivations

The motivation behind this research stems from the urgent need to empower farmers with accessible technology.

1. **Technological Accessibility:** Farmers in developing regions often use low-end smartphones. Developing a lightweight model ensures that advanced AI diagnosis can run smoothly on these devices without needing expensive hardware or cloud connectivity [18].
2. **Food Security:** By enabling early and accurate detection of diseases, crop wastage can be significantly reduced, contributing to better food security and economic stability for farmers [1, 3].

3. **Trustworthy AI:** Adoption of AI in critical sectors like agriculture depends on trust. By making the AI "explainable," we aim to bridge the gap between complex algorithms and human users [7].

1.6 Significance of the Research

This study holds significant value for both the academic community and the agricultural sector:

- **For Agriculture:** It provides a prototype for a fast, offline-capable diagnostic tool that can assist farmers in making timely decisions regarding pesticide use and disease management.
- **For Research:** It demonstrates that a carefully designed custom architecture can outperform complex pre-trained models for specific domain tasks. It also highlights the importance of XAI in validating model behavior, preventing the deployment of biased models.
- **Economic Impact:** Reducing crop loss through early detection directly translates to higher yields and increased income for farmers [2].

1.7 Research Contribution

The key contributions of this thesis are summarized as follows:

1. **Development of a Novel Lightweight Architecture:** We proposed a custom CNN utilizing Swish activation functions [19], Batch Normalization [20], and Global Average Pooling to reduce parameters while maintaining high accuracy (98.71%).
2. **Comprehensive Comparative Analysis:** A detailed performance comparison between the Custom CNN, ResNet50, and MobileNetV2, utilizing metrics such as Accuracy, F1-Score, ROC Curves, and Inference Time.
3. **Integration of XAI (Grad-CAM):** We successfully implemented Grad-CAM to visualize class activation heatmaps, providing visual proof that the model focuses on lesion areas (spots/blights) rather than the background [17].
4. **Efficiency Validation:** We demonstrated that the Custom CNN offers a superior trade-off between speed and accuracy compared to standard transfer learning models, making it ideal for edge deployment.

1.8 Thesis Organization

The remainder of this thesis is organized as follows:

- **Chapter 2: Literature Review** discusses existing works in plant disease detection, highlighting the evolution from traditional ML to modern DL and XAI approaches.
- **Chapter 3: Methodology** details the dataset, data preprocessing, the proposed Custom CNN architecture, transfer learning configurations, and the training process.
- **Chapter 4: Results and Discussion** presents the experimental results, including confusion matrices, performance graphs, comparative analysis, and XAI visualizations.

- **Chapter 5: Standards, Impacts, Ethics, and Challenges** outlines the engineering standards adhered to, analyzes the societal and environmental impacts, discusses ethical considerations, and highlights the implementation challenges faced during the research.
- **Chapter 6: Conclusion** summarizes the key findings, concludes the study, and outlines limitations along with potential future directions for this research.

1.9 Summary

This chapter provided an overview of the challenges in plant disease detection and the potential of Deep Learning to address them. The problem statement highlighted the need for lightweight and interpretable models. The research objectives, motivations, and contributions were outlined to set the stage for the detailed technical discussions in the subsequent chapters. The next chapter will review the relevant literature to establish the context for the proposed methodology.

CHAPTER 2
BACKGROUND STUDY

2.1 Introduction

This chapter provides a comprehensive review of the existing literature and technological advancements in the field of automated plant disease detection. It begins by exploring traditional Machine Learning (ML) approaches dependent on manual feature extraction and moves towards the paradigm shift brought by Deep Learning (DL) and Convolutional Neural Networks (CNNs). The review specifically focuses on the application of Transfer Learning, the emerging need for Lightweight architectures for mobile deployment, and the critical role of Explainable AI (XAI) in building trust. Finally, a problem analysis is presented to identify the research gaps that this thesis aims to address.

2.2 Literature Review

2.2.1 Plant Pathology: Agricultural Context and Diagnostic Challenges

Plant diseases represent a significant threat to global food security and economic stability. Research by the Food and Agriculture Organization (FAO) indicates that plant pests and diseases reduce global crop yields by approximately 20-40% annually [2]. The rapid spread of diseases and the similarity in visual symptoms make early detection particularly challenging yet critically important for sustainable agriculture.

Traditional diagnostic methods have been extensively documented in agricultural literature. Visual inspection by experts remains the most common approach, relying on the identification of visible symptoms such as lesions, spots, or discoloration [4]. However, studies have highlighted limitations including human error, the scarcity of experts in rural areas, and the time-consuming nature of manual diagnosis. While biological laboratory tests provide accurate pathogen identification, they involve high costs and require specialized equipment [1].

Recent agricultural studies emphasize the importance of precision farming protocols. Research demonstrates that timely intervention through targeted pesticide application can significantly reduce crop loss and environmental damage caused by chemical overuse [3]. However, the implementation of widespread monitoring programs faces challenges including resource constraints and the lack of technological infrastructure in remote fields. The urgent need for accessible, accurate, and cost-effective screening tools has driven interest in automated diagnostic approaches utilizing Artificial Intelligence [5].

2.2.2 Traditional Machine Learning Approaches

Before the deep learning era, researchers primarily utilized image processing techniques combined with classical classifiers. Methods such as Color Histograms, Scale Invariant Feature Transform (SIFT), and Gray-Level Co-occurrence Matrix (GLCM) were widely used to extract texture and shape features from leaves [10]. For example, Arivazhagan et al. used texture features to detect unhealthy regions in plants [11]. Similarly, Al-Hiary et al. proposed a system using K-Means clustering for segmentation and Neural Networks for classification [9]. While these methods laid the groundwork, they relied heavily on hand-crafted features, which often failed to generalize across complex backgrounds or large datasets [12].

2.2.3 Deep Learning in Agriculture: Evolution and Applications

The application of deep learning to agricultural imaging has evolved dramatically over the past decade [13]. Landmark studies in 2015-2016 established the viability of deep learning in plant pathology. The introduction of the **PlantVillage dataset** provided a standardized benchmark, enabling researchers to demonstrate that Convolutional Neural Networks (CNNs) could learn hierarchical feature representations from leaf images without manual feature engineering [16].

Mohanty et al. (2016) conducted a baseline study using architectures like AlexNet and GoogLeNet on the PlantVillage dataset. They achieved accuracies exceeding 99% but noted that performance dropped significantly when tested on images with different background conditions [13]. Similarly, **Sladojevic et al. (2016)** developed a model for recognizing 13 different types of plant diseases, validating the potential of Deep Neural Networks (DNNs) in agriculture [21]. Further advances led to deeper networks; **Ferentinos (2018)** experimented with VGG16, AlexNet, and GoogLeNet, concluding that VGG16 achieved the highest accuracy (99.53%) but highlighted its high computational cost due to the massive number of parameters (~138 million) [14].

2.2.4 Transfer Learning and CNN Architectures

Transfer learning, where models pre-trained on large datasets like ImageNet are fine-tuned for specific tasks, has become the standard approach to overcome data scarcity [22].

- **ResNet (Residual Networks):** Introduced by **He et al. (2016)**, ResNet utilizes "skip connections" to solve the vanishing gradient problem in deep networks [23]. Studies utilizing ResNet50 for crop disease diagnosis have consistently reported high accuracy, making it a robust benchmark model [24].
- **MobileNetV2:** To address the computational constraints of mobile deployment, **Sandler et al. (2018)** proposed MobileNetV2. It uses Depthwise Separable Convolutions and inverted residuals to drastically reduce parameter count and latency [25]. While highly efficient, some studies suggest that MobileNet occasionally struggles with fine-grained disease patterns compared to deeper networks [26].
- **VGG and Inception:** Other architectures like **VGG19** [27] and **InceptionV3** [28] have also been widely used. While VGG offers excellent precision, its large size renders it impractical for real-time mobile applications.

2.2.5 Data Augmentation and Optimization Techniques

To improve model robustness and prevent overfitting, various optimization techniques are employed. **Data Augmentation** (rotation, flipping, zooming) is widely used to artificially expand the training dataset, helping models generalize better to unseen images [29]. Regularization techniques such as **Dropout**, introduced by **Srivastava et al.**, randomly deactivate neurons during training to prevent complex co-adaptations [30]. Furthermore, **Batch Normalization** [20] and modern activation functions like **Swish** [19] have been shown to accelerate convergence and improve accuracy in deep networks.

2.2.6 Explainable AI (XAI) and Model Transparency

One of the critical criticisms of deep learning in agriculture is the "Black Box" nature of CNNs. Farmers need to trust *why* a model predicts a certain disease. **Selvaraju et al. (2017)** introduced **Grad-CAM (Gradient-weighted Class Activation Mapping)** to visualize the regions of interest in an image [17]. Recent studies have started incorporating Grad-CAM to validate that models are focusing on leaf lesions rather than background noise [7]. However, many proposed systems still lack this validation step, leaving their reliability in question.

2.3 Problem Analysis

Based on the comprehensive literature review, several key issues and research gaps have been identified that motivate this thesis:

1. **Computational Inefficiency:** State-of-the-art models like VGG16 [14] and ResNet50 [23], while accurate, are computationally heavy. They require significant memory and processing power, which are often unavailable in the low-end smartphones used by farmers in developing countries.
2. **The "Accuracy vs. Efficiency" Dilemma:** Existing lightweight models (like MobileNet) solve the size problem but sometimes struggle to capture fine-grained texture details of specific diseases compared to deeper networks [8, 26]. There is a need for a specialized architecture that bridges this gap—offering the speed of MobileNet with the precision of ResNet.
3. **Lack of Transparency (The Trust Deficit):** Most existing systems provide a simple text output (e.g., "Healthy" or "Diseased"). Without visual evidence or explanation, these systems fail to build trust with human users. The integration of XAI is necessary to verify that the high accuracy is genuine and not a result of data bias [7, 15].

Conclusion of Analysis: To solve these problems, this research proposes a three-pronged approach: (1) Developing a Custom Lightweight CNN to minimize computational cost, (2) Comparing it rigorously with ResNet50 and MobileNetV2 to benchmark performance, and (3) Using Grad-CAM to ensure the solution is transparent and trustworthy.

2.4 Contribution of Authors

The research presented in this thesis was a collaborative effort. The specific contributions and responsibilities of each team member are outlined below:

Table 2.1: Contribution of Authors

Member Name	Role	Detailed Contributions & Responsibilities
Aktaruzzaman	**Team Leader & Lead Developer**	• **Project Management:** Supervised the overall project workflow, timeline management, and code integration. • **Model Development:** Designed the **Proposed Custom Lightweight CNN** architecture and implemented Transfer Learning using **ResNet50** and **MobileNetV2**. • **Data Preprocessing:** Implemented image resizing, rescaling, and **Data Augmentation** strategies (rotation, zoom, flip) to address class imbalance. • **Application Development:** Developed and deployed a **Streamlit Web Application** for real-time disease detection using the trained model.
Sagar Saha	**Data Engineer**	• **Environment Setup:** Configured the Python environment and imported necessary Deep Learning libraries (TensorFlow/Keras). • **Data Pipeline:** Managed the loading of the **PlantVillage dataset**, converting raw images into structured DataFrames for processing. • **Exploratory Data Analysis (EDA):** Conducted statistical analysis to visualize class distributions and identify data imbalances. • **Data Partitioning:** Split the dataset into Training, Validation, and Test sets to ensure robust model evaluation.
Md. Abdullah Al Mamun	**AI Analyst**	• **Comparative Analysis:** Evaluated all three models (Custom CNN, ResNet50, MobileNetV2) using metrics such as Accuracy, Loss, Precision, Recall, and F1-Score. • **Performance Visualization:** Generated Confusion Matrices and **ROC Curves** to assess model classification performance. • **Explainable AI:** Implemented **Grad-CAM (Gradient-weighted Class Activation Mapping)** to visualize heatmaps,

		verifying that models focus on leaf lesions rather than background noise.
Md. Najmul Parves	**Technical Writer**	• **Thesis Documentation:** Responsible for compiling the final thesis book, including the Abstract, Introduction, Literature Review, and Conclusion. • **Methodology Drafting:** Documented the mathematical explanations of the algorithms and model architectures used. • **Formatting & Proofreading:** Ensured the report adheres to the university's academic formatting standards and citation guidelines.
Siam Hossain	**Visual Designer**	• **Presentation Design:** Created high-quality visual slides for the thesis defense, summarizing complex technical data into readable charts and infographics. • **Visual Assets:** Designed flowcharts and block diagrams representing the methodology and model architectures. • **Speech Preparation:** Structured the defense presentation flow to effectively communicate the research findings to the panel.

2.5 Summary

This chapter reviewed the evolution of plant disease detection technologies, highlighting the transition from manual feature extraction to automated deep learning [12, 13]. Despite significant advancements, a critical analysis of the literature reveals several limitations in existing works. First, state-of-the-art models (such as VGG16 or ResNet) are often computationally expensive and memory-intensive, making them unsuitable for real-time deployment on mobile or edge devices. Second, existing lightweight architectures frequently struggle to maintain high accuracy when model size is reduced. Third, most studies fail to address class imbalance, leading to biased predictions, and lack explainability (XAI), which leaves the decision-making process as a 'black box' and hinders user trust. Addressing these gaps, the next chapter presents the methodology adopted in this research to develop a solution that is accurate, computationally efficient, and fully explainable.

CHAPTER 3
PROPOSED MODEL

3.1 Introduction

This chapter outlines the research methodology and the systematic approach adopted to develop the automated plant disease detection system. It details the feasibility analysis, requirement specifications, and the step-by-step workflow ranging from data collection to model deployment. The core of this chapter describes the architecture of the proposed **Custom Lightweight CNN** and explains how it differs from the Transfer Learning models (**ResNet50** and **MobileNetV2**) used for comparison. Finally, the integration of **Explainable AI (XAI)** for validating the model's predictions is discussed.

3.2 Feasibility Analysis

Before initiating the development, a thorough feasibility study was conducted to determine the viability of the project.

3.2.1 Technical Feasibility

The project is technically feasible as it utilizes open-source and well-documented technologies. Python is used as the primary programming language, with **TensorFlow** and **Keras** as the deep learning frameworks [31]. The training process leverages GPU acceleration (specifically NVIDIA Tesla P100/T4 available via Kaggle/Colab), which is sufficient to handle the computational load of training deep neural networks on the PlantVillage dataset.

3.2.2 Operational Feasibility

The proposed system is designed to be fully automated. Once trained, the model requires minimal human intervention users simply upload an image, and the system outputs the disease class and confidence score. This simplicity ensures high operational feasibility, making it accessible to non-technical users such as farmers or agricultural extension workers.

3.2.3 Economic Feasibility

The project is economically viable as it relies on open-source software tools (Python, Anaconda, Jupyter), requiring no licensing fees. The operational cost is low since the proposed lightweight model is optimized to run on standard consumer hardware (e.g., smartphones or basic laptops) without requiring expensive dedicated servers for inference.

3.3 Requirement Analysis

To ensure the successful development of the system, specific functional and non-functional requirements were defined.

3.3.1 Functional Requirements

- **Input:** The system must accept leaf images in standard formats (JPG, PNG).

- **Processing:** The system must preprocess the image (resize, normalize) and pass it through the trained Deep Learning models.
- **Output:** The system must predict the specific disease class from 38 categories and display the confidence percentage.
- **Visualization:** The system must generate a Grad-CAM heatmap to visualize the infected regions used for prediction [17].

3.3.2 Non-Functional Requirements

- **Accuracy:** The model should achieve a testing accuracy of greater than 95%.
- **Latency:** The inference time (prediction speed) should be minimal, ideally under 100ms per image for real-time applicability.
- **Scalability:** The architecture should be adaptable to include new disease classes in the future.

3.3.3 Dataset Requirements

The research utilizes the **PlantVillage Dataset**, which is the standard benchmark for plant disease classification [16].

- **Total Images:** 54,305 images.
- **Classes:** 38 distinct classes (spanning 14 crop species).
- **Format:** RGB images of varying resolutions.

3.3.4 Hardware Requirements

- **Development/Training Phase:**
    - CPU: Intel Core i5 or higher / Cloud vCPU.
    - RAM: 16 GB or higher.
    - GPU: NVIDIA GPU with at least 12GB VRAM (required for faster training).
- **Deployment/Inference Phase:**
    - Processor: Standard Mobile CPU (Snapdragon/MediaTek) or Laptop CPU.
    - RAM: 4 GB sufficient for the lightweight model.

3.4 Research Model & Methodology

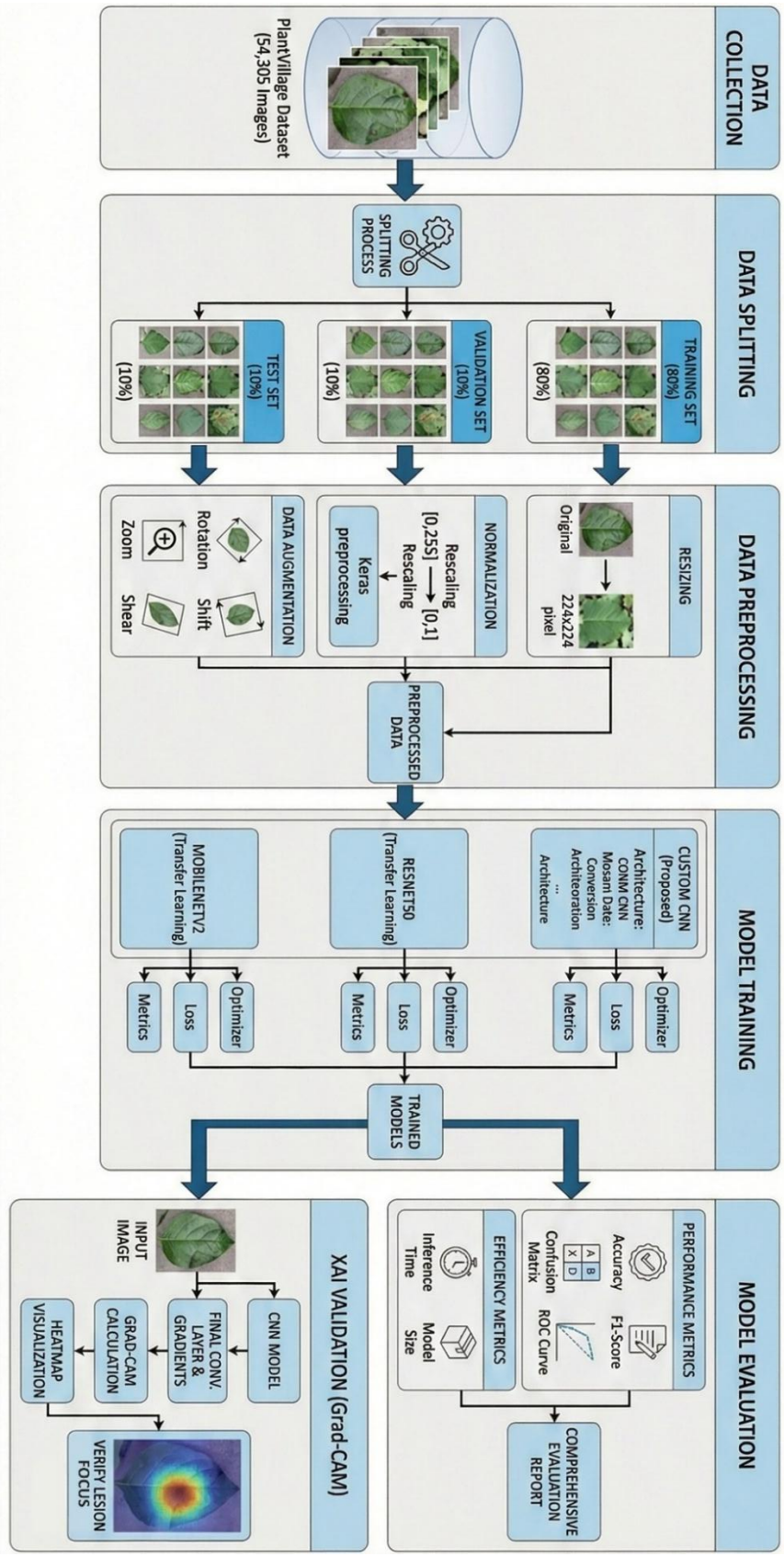

Figure 3.1: *Workflow Diagram (Block Diagram of Proposed Methodology)*

3.4.1 Data Collection

The dataset was obtained from the open-access PlantVillage repository. The dataset contains healthy and diseased leaf images of crops including Apple, Corn, Grape, Potato, Tomato, etc. The data is balanced for most classes, but augmentation was applied to ensure uniform learning.

3.4.2 Data Preprocessing

Data preprocessing is critical for model performance. The following steps were applied:

1. **Resizing:** All images were resized to a fixed dimension of 224 * 224 pixels to match the input layer of the CNN architectures.
2. **Normalization:**
    - For the **Custom CNN**, pixel values were rescaled from [0, 255] to [0, 1] by dividing by 255.0.
    - For **ResNet50** and **MobileNetV2**, specific preprocessing functions (e.g., Zero-centering) provided by Keras were used.
3. **Data Augmentation:** To prevent overfitting and improve generalization, we applied on-the-fly augmentation using ImageDataGenerator [29]. Techniques included:
    - Rotation range: 20 degrees
    - Width/Height shift: 20%
    - Horizontal Flip
    - Zoom and Shear transformations.

3.4.3 CNN Architectures (Proposed Custom Model)

We proposed a **Custom Lightweight CNN** designed specifically for this task. The architecture aims to reduce the number of parameters while maintaining high feature extraction capability.

- **Input Layer:** Accepts images of shape (224, 224, 3).
- **Convolutional Blocks:** The model consists of 3 main blocks. Each block contains:
    - Conv2D layers with 3 * 3 filters.
    - BatchNormalization for training stability[20].
    - Swish activation function (chosen over ReLU for better gradient flow)[19].
    - MaxPooling2D for dimensionality reduction.
    - Dropout[20] to prevent overfitting[30].
- **Global Average Pooling (GAP):** Instead of flattening the massive feature map which causes parameter explosion, we used GAP to minimize model size.

- **Output Layer:** A Dense layer with 38 neurons and Softmax activation for multiclass classification.

3.4.4 ResNet Architectures

For comparative analysis, we employed **ResNet50**, a deep Residual Network [23].

- **Mechanism:** It uses "skip connections" to allow gradients to flow through the network easily, enabling the training of very deep networks (50 layers).
- **Transfer Learning:** We initialized the model with weights pre-trained on **ImageNet**. The top layers were removed, and a custom classification head (Global Average Pooling + Dense) was added to fit our 38 classes.

3.4.5 MobileNet Architectures

We also implemented **MobileNetV2**, a model optimized for mobile devices [25].

- **Mechanism:** It uses **Depthwise Separable Convolutions** and **Inverted Residual Blocks**. This significantly reduces the number of operations (FLOPS) and parameters compared to standard CNNs.
- **Configuration:** Similar to ResNet, we utilized Transfer Learning with ImageNet weights, freezing the base layers and fine-tuning the final classification layers.

3.4.6 Model Evaluation

The models were evaluated using a comprehensive set of metrics:

1. **Accuracy:** The percentage of correctly classified images.
2. **Loss:** Categorical Crossentropy loss to measure prediction error.
3. **Precision, Recall, and F1-Score:** To evaluate performance on individual classes.
4. **Confusion Matrix:** To visualize misclassifications between similar diseases.
5. **ROC Curve:** To analyze the trade-off between True Positive Rate and False Positive Rate.
6. **Efficiency Metrics:** Inference time (ms/image) and Model Size (MB).

3.4.7 XAI with CNN (Grad-CAM)

To address the "black-box" nature of CNNs, we implemented **Gradient-weighted Class Activation Mapping (Grad-CAM)** [17].

- **Principle:** Grad-CAM uses the gradients of the target class flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image.
- **Implementation:** We extracted the feature maps from the last convolutional layer of our Custom CNN and weighted them by their gradients. This heatmap is superimposed on the original image to verify if the model is focusing on the disease lesions or irrelevant background features.

CHAPTER 4
IMPLEMENTATION, TESTING AND RESULT ANALYSIS

4.1 Introduction

This chapter presents the practical realization, experimental setup, and comprehensive evaluation of the proposed plant disease detection system. While the preceding chapter outlined the theoretical framework and research methodology, this chapter details the step-by-step execution of the research work, spanning from the development environment setup to the final validation of results.

The primary objective of this phase is to validate the effectiveness of the proposed **Custom Lightweight CNN** architecture by comparing its performance against two established transfer learning models: **ResNet50** and **MobileNetV2**. The implementation workflow begins with the preparation of the computational environment, followed by rigorous data preprocessing and augmentation strategies applied to the **PlantVillage dataset**.

Subsequently, the training process for each model is documented, highlighting the hyperparameter tuning strategies adopted to ensure optimal convergence. The trained models are then subjected to extensive testing using unseen data, and their performance is analyzed using standard evaluation metrics such as **Accuracy, Precision, Recall, F1-Score, and Confusion Matrices**. Beyond numerical accuracy, this chapter also incorporates **Explainable AI (XAI)** validation using the **Grad-CAM** technique to visually verify the reliability of the model's predictions. The chapter concludes with a comparative analysis to demonstrate the trade-offs between computational efficiency and classification accuracy, proving the suitability of the proposed model for real-world agricultural applications.

4.2 Implementation Details

The successful training of deep learning models, especially on a large dataset like PlantVillage, requires a robust computational environment. This section outlines the software frameworks, programming tools, and hardware specifications utilized to implement, train, and test the proposed architectures.

4.2.1 Development Environment

The entire experimental framework was developed using the **Python** programming language (Version 3.10), selected for its extensive support for data science and machine learning libraries. The core deep learning tasks were executed using **TensorFlow** (v2.x) as the backend framework, with **Keras** employed as the high-level API to streamline model building and experimentation.

For coding, debugging, and execution, **Jupyter Notebooks** were hosted on the **Kaggle** cloud platform. Kaggle was chosen primarily for its provision of free access to high-performance GPUs, which significantly accelerated the training process compared to standard CPU-based local environments. Key libraries such as **NumPy** and **Pandas** were used for data manipulation, while **Matplotlib** and **Seaborn** were utilized for data visualization and plotting training graphs.

4.2.2 Hardware and Software Configuration

Given the computational intensity of training Convolutional Neural Networks (CNNs) with over 54,000 images, hardware acceleration was essential. We utilized the **NVIDIA Tesla P100** GPU provided by the Kaggle kernel. The detailed specification of the hardware and software environment is summarized in **Table 4.1**.

Table 4.1: Hardware and Software Configuration

Component	**Specification**
Operating System	Linux (Ubuntu 20.04 LTS) via Kaggle Kernel
CPU	Intel Xeon Processor @ 2.20GHz (2 Cores)
GPU (Graphics Processing Unit)	**NVIDIA Tesla P100 (16GB VRAM)**
System RAM	13 GB
Disk Storage	73 GB
Programming Language	Python 3.10
Deep Learning Framework	TensorFlow[24] 2.12, Keras[25]
Key Libraries	NumPy, Pandas, Matplotlib, Scikit-learn, OpenCV

4.3 Dataset Analysis & Preprocessing Outcomes

4.3.1 Dataset Description and Class Distribution

The experimental evaluation relies on the **PlantVillage dataset**, a standard benchmark for plant disease classification tasks. The dataset comprises a total of **54,305** color images. These images are categorized into **38 distinct classes**, representing **14 different plant species** (such as Apple, Corn, Grape, Potato, and Tomato) and their corresponding health statuses (healthy or specific disease infections).

Critical Finding: Class Imbalance An exploratory analysis of the dataset reveals a severe class imbalance, which posed a significant challenge for model training, particularly for the Custom CNN. As detailed in Table 4.2, the dataset is dominated by classes like Orange Haunglongbing (5,507 images) and Tomato Yellow Leaf Curl Virus (5,357 images), while classes like Potato Healthy contain as few as 152 images. This results in a drastic imbalance ratio of approximately 36:1 between the majority and minority classes.

Table 4.2: Detailed Class Distribution of the PlantVillage Dataset

Plant Name	**Disease / Health Status**	**Number of Images**
Apple	Apple scab	630
	Black rot	621
	Cedar apple rust	275

	Healthy	1,645
Blueberry	Healthy	1,502
Cherry	Powdery mildew	1,052
	Healthy	854
Corn (Maize)	Cercospora leaf spot (Gray leaf spot)	513
	Common rust	1,192
	Northern Leaf Blight	985
	Healthy	1,162
Grape	Black rot	1,180
	Esca (Black Measles)	1,383
	Leaf blight (Isariopsis Leaf Spot)	1,076
	Healthy	423
Orange	Haunglongbing (Citrus greening)	**5,507 (Max)**
Peach	Bacterial spot	2,297
	Healthy	360
Pepper, Bell	Bacterial spot	997
	Healthy	1,478
Potato	Early blight	1,000
	Late blight	1,000
	Healthy	**152 (Min)**
Raspberry	Healthy	371
Soybean	Healthy	5,090
Squash	Powdery mildew	1,835
Strawberry	Leaf scorch	1,109
	Healthy	456
Tomato	Bacterial spot	2,127
	Early blight	1,000
	Late blight	1,909
	Leaf Mold	952
	Septoria leaf spot	1,771
	Spider mites (Two-spotted spider mite)	1,676
	Target Spot	1,404
	Tomato Yellow Leaf Curl Virus	5,357
	Tomato mosaic virus	373
	Healthy	1,591
Total Images		**54,305**

4.3.2 Exploratory Data Analysis (EDA)

Prior to model training, a comprehensive Exploratory Data Analysis (EDA) was conducted to understand the fundamental characteristics of the PlantVillage dataset. This step was crucial for identifying potential biases, understanding image properties, and determining the necessary preprocessing strategies.

A. Visual Inspection of Samples

First, we visualized random samples from different classes to examine the diversity of the dataset. As shown in Figure 4.1, the images generally feature a single leaf centered against a neutral or grey background. The visual symptoms of diseases (e.g., spots, mold, yellowing) are distinct, though some similarities exist between different conditions.

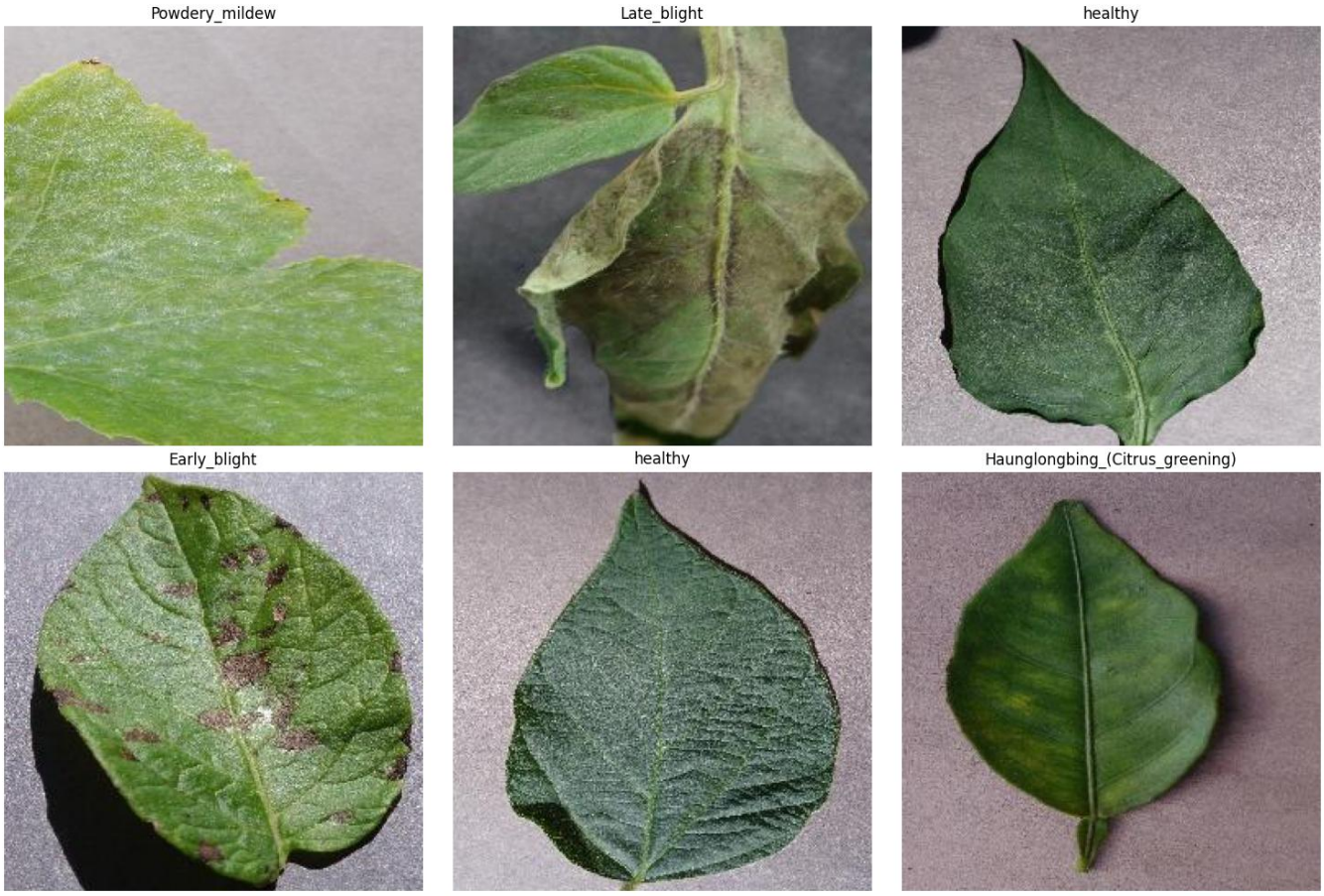

Figure 4.1: *Sample images from the PlantVillage dataset showing healthy and diseased leaves across various crops.*

B. Class Distribution Analysis

To assess the balance of the dataset, we plotted the frequency of images per class. The distribution is illustrated in Figure 4.2.

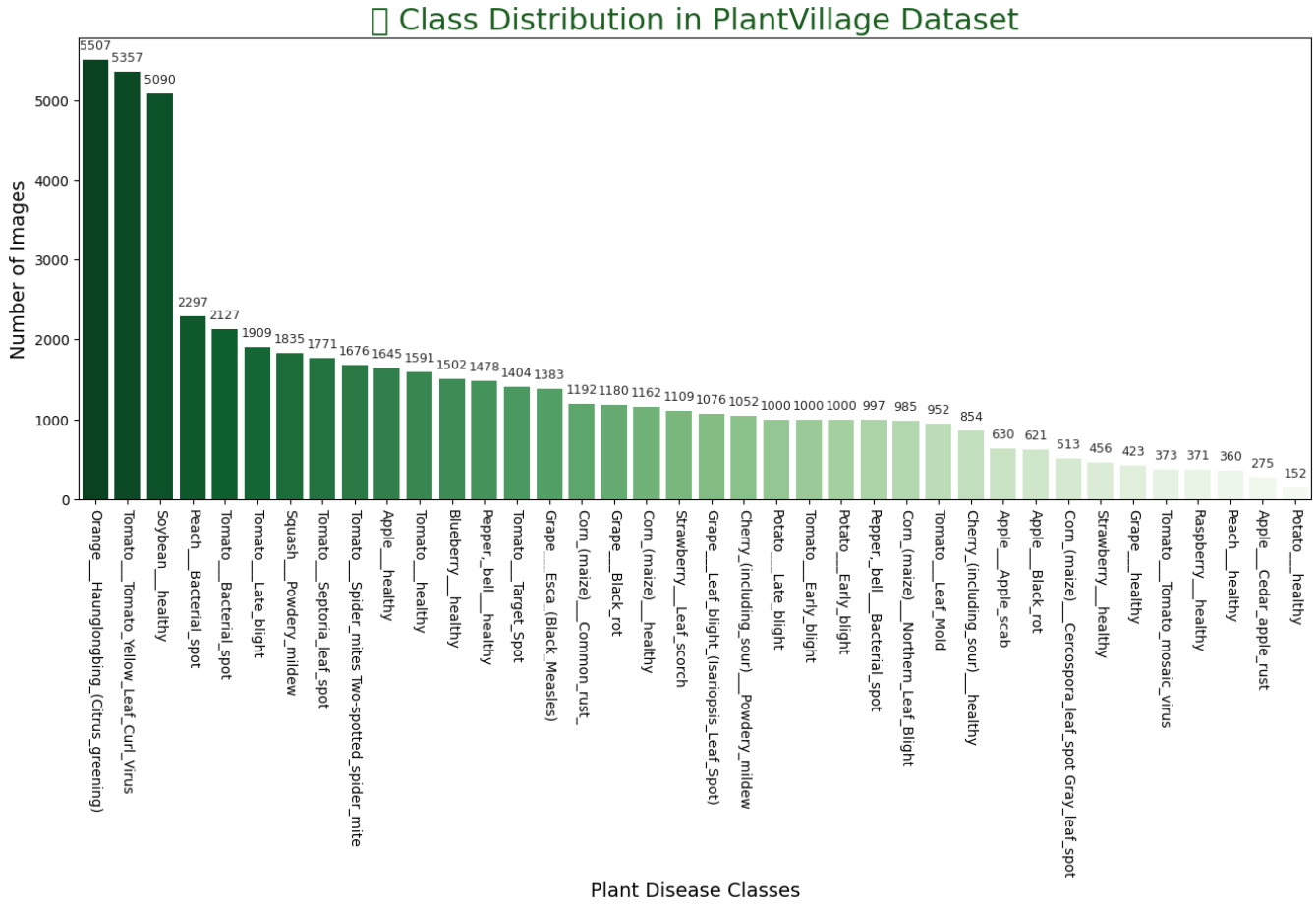

Figure 4.2: *Class distribution of the PlantVillage dataset indicating severe imbalance.*

Observation 1: Class Imbalance

As observed in the bar chart, the dataset is highly imbalanced. The class 'Orange_Haunglongbing' has the highest number of samples (5,507), whereas 'Potato_healthy' has the lowest (152). This significant disparity (a ratio of roughly 36:1) poses a risk of model bias, where the network might overfit to majority classes while ignoring minority ones. This finding necessitates the use of robust data augmentation techniques during the training phase to equalize the learning opportunity for all classes.

C. Pixel Intensity Analysis

We further analyzed the color properties of the images by calculating the average RGB pixel intensity across the dataset. The resulting distribution is shown in Figure 4.3.

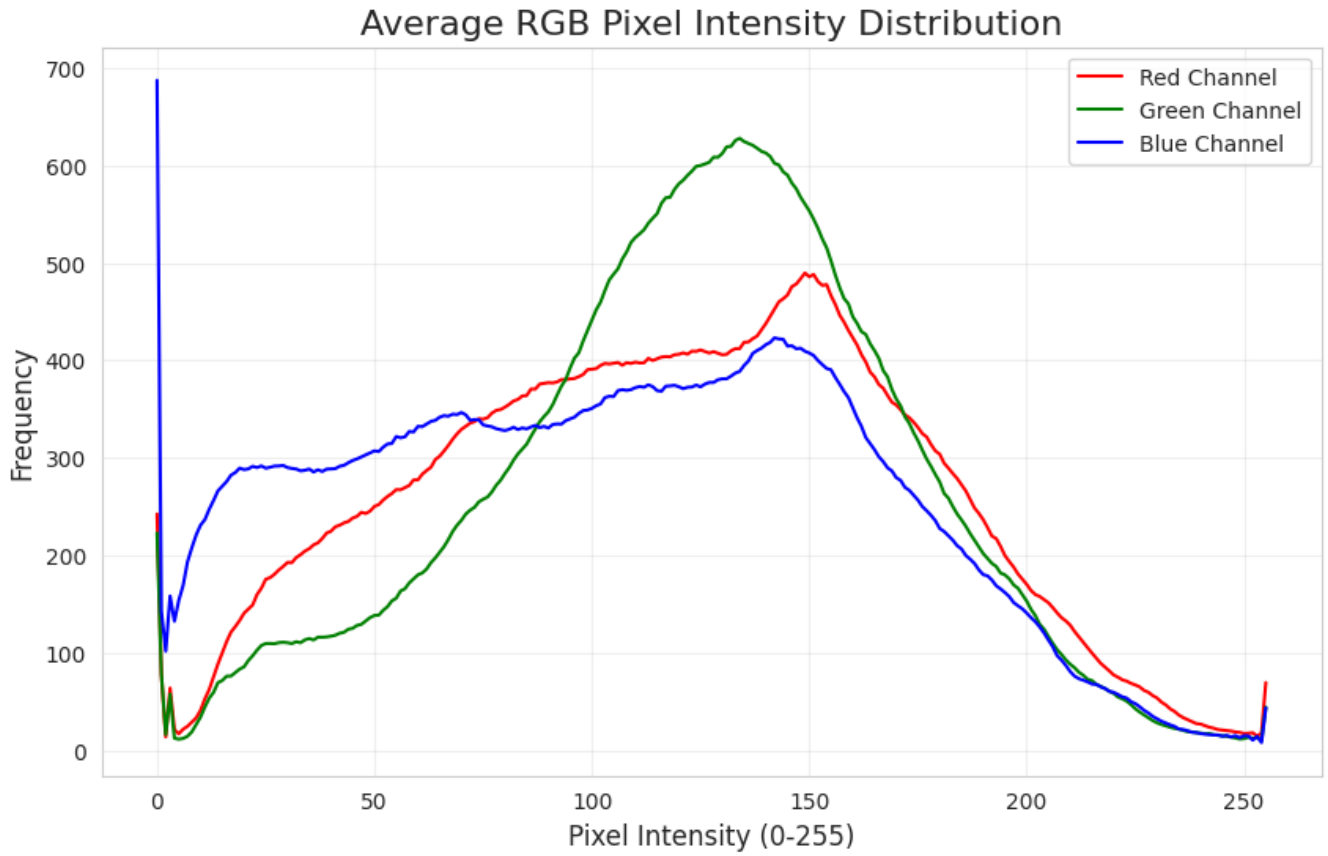

Figure 4.3: *Average RGB pixel intensity distribution across the dataset.*

Observation 2: Pixel Intensity and Background

The RGB channel analysis reveals a dominant Green channel intensity, which is consistent with the biological nature of healthy and diseased plant foliage. Additionally, a significant spike is observed at intensity 0 (particularly in the Blue channel). This confirms that a large portion of the dataset consists of segmented or black/grey backgrounds, implying that the model can focus primarily on the leaf features without significant interference from background noise.

4.3.3 Data Splitting Results

To facilitate a robust and unbiased evaluation of the deep learning models, the preprocessed dataset was partitioned into three distinct subsets. We employed a randomized splitting strategy with a ratio of **80:10:10**. This distribution was chosen to ensure that the model has access to a substantial amount of data for learning features while retaining sufficient unseen data for hyperparameter tuning and final testing.

The visual representation of this distribution is illustrated in **Figure 4.4**.

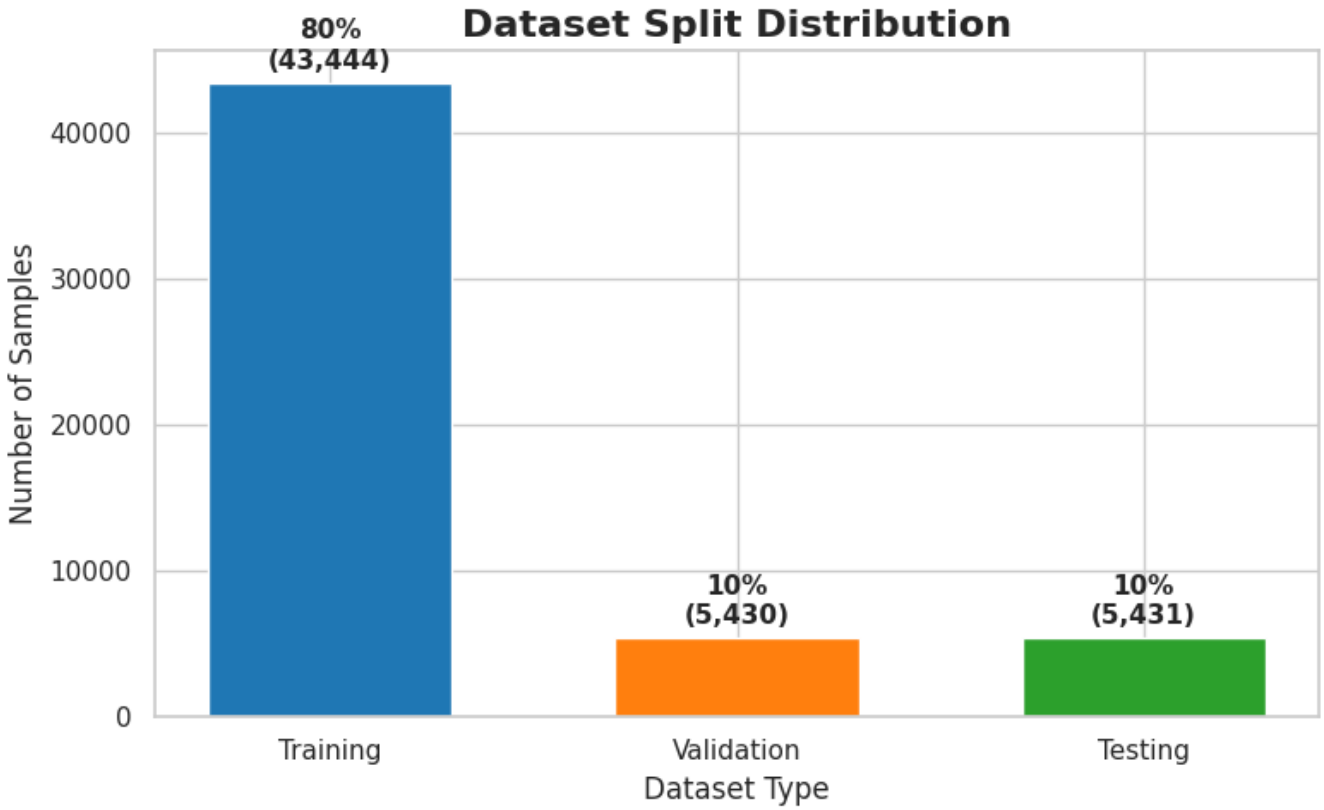

Figure 4.4: *Data distribution across Training, Validation, and Test sets.*

The breakdown of the dataset is as follows:

1. **Training Set (80%):** The largest portion, comprising **43,444 images**, was allocated for training. This subset is used by the model to learn the weights and biases through backpropagation. A larger training set is essential for deep CNNs to capture the complex patterns of plant diseases effectively.
2. **Validation Set (10%):** A subset of **5,430 images** was set aside for validation. This set acts as a proxy for test data during the training process. It is used to monitor the model's performance after each epoch, allowing for the tuning of hyperparameters and the implementation of **Early Stopping** to prevent overfitting.
3. **Test Set (10%):** The remaining **5,431 images** were kept completely isolated until the final evaluation phase. This set provides an unbiased metric of the model's real-world performance, ensuring that the reported accuracy reflects the model's generalization capability rather than memorization.

Table 4.3: Quantitative Breakdown of Data Splits

Subset	Split Ratio	Image Count	Role in Methodology
Training	80%	43,444	Feature Learning & Weight Update
Validation	10%	5,430	Hyperparameter Tuning & Loss Monitoring
Testing	10%	5,431	Final Unbiased Evaluation
Total	**100%**	**54,305**	

4.3.4 Data Preprocessing and Augmentation Outcomes

Efficient data preprocessing is fundamental to the performance of Convolutional Neural Networks (CNNs). As illustrated in **Figure 4.5**, our preprocessing pipeline consists of three sequential stages: resizing, normalization tailored to specific model architectures, and on-the-fly data augmentation.

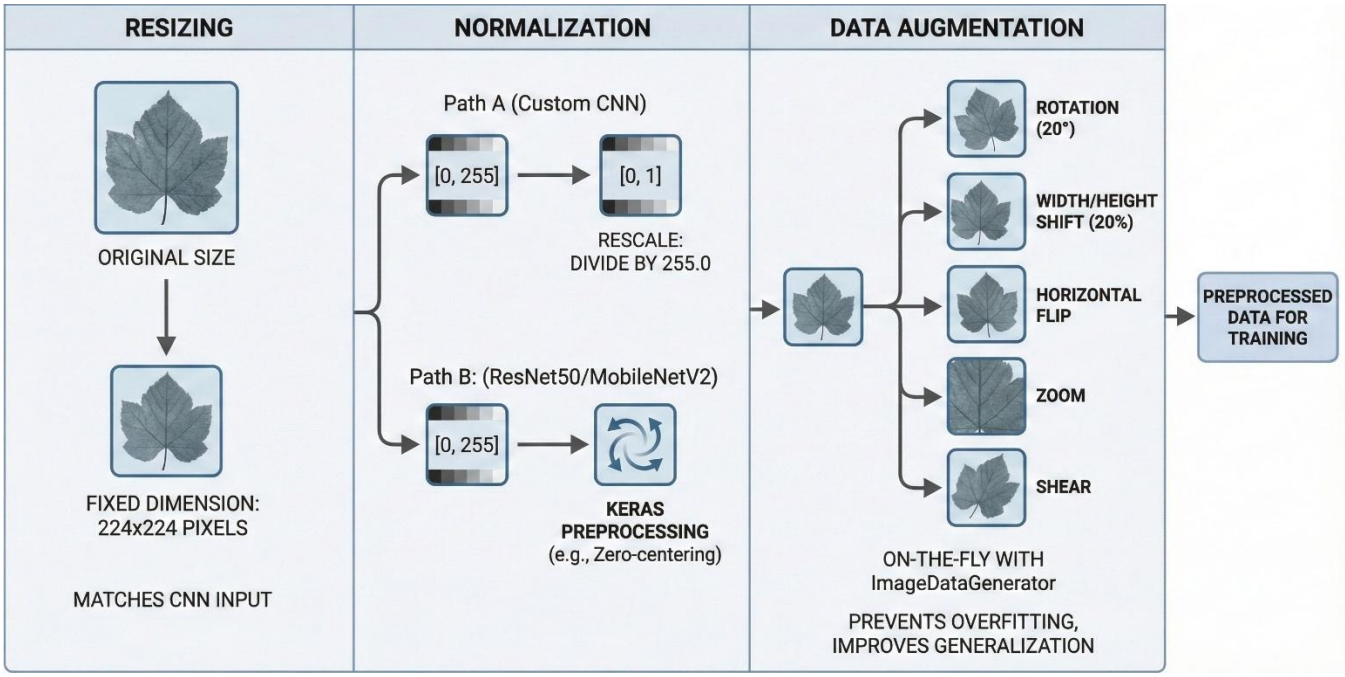

Figure 4.5: *Data preprocessing workflow illustrating resizing, model-specific normalization (Path A vs. Path B), and augmentation techniques.*

A. Image Resizing

Initially, all images in the dataset, which varied in resolution, were resized to a fixed dimension of 224 * 224 pixels with 3 color channels (RGB). This standardization was necessary to match the input layer requirements of both the custom and transfer learning models.

B. Normalization Strategies (Path A vs. Path B)

To ensure optimal model convergence, we implemented two distinct normalization techniques as shown in the "Normalization" block of Figure 4.5:

- **Path A (For Custom CNN):** We applied a standard **Rescaling** technique where pixel intensity values [0, 255] were divided by **255.0**. This transformation maps the pixels to a normalized range of [0, 1], which is crucial for training a deep learning model from scratch as it stabilizes the gradient descent process.
- **Path B (For ResNet50 & MobileNetV2):** For the pre-trained models, we utilized the specific **Keras preprocessing functions** (preprocess_input). Instead of simple scaling, these functions typically perform **zero-centering** (subtracting the mean RGB value of the ImageNet dataset) to align the new data distribution with the original data on which these models were trained.

C. Data Augmentation Results

To address the challenge of class imbalance and prevent overfitting, we employed on-the-fly data augmentation using the ImageDataGenerator class. As depicted in the final block of Figure 4.5, the following random transformations were applied to the training set during each epoch:

- **Rotation:** Randomly rotating images by up to **20º**.
- **Width & Height Shifts:** Shifting the image horizontally or vertically by up to **20%**.
- **Shear & Zoom:** Applying shear mapping and random zooming to introduce scale invariance.
- **Horizontal Flip:** Randomly flipping images horizontally to simulate different leaf orientations.

4.4 Performance Analysis of Custom CNN (Proposed Model)

4.4.1 Model Architecture and Parameter Efficiency

The proposed model is a custom-designed, lightweight Convolutional Neural Network (CNN) tailored for the efficient classification of plant diseases. The complete architectural flow is visualized in **Figure 4.6**.

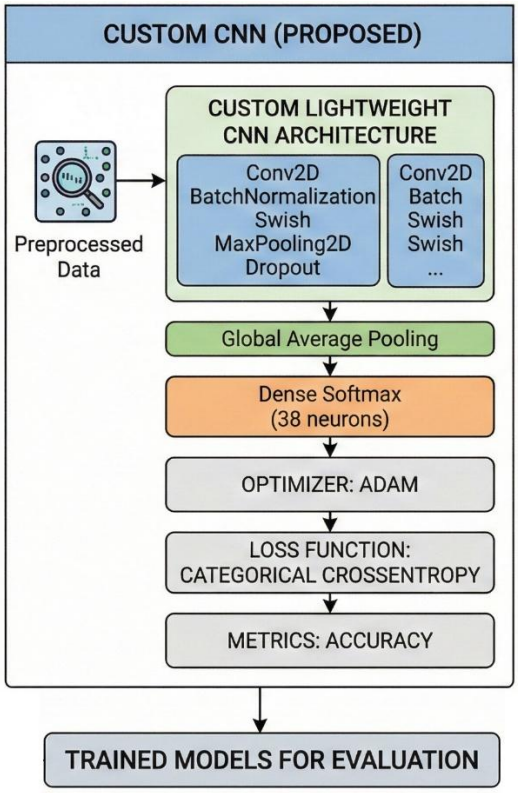

Figure 4.6: *Architectural design of the proposed Custom Lightweight CNN.*

As detailed in **Table 4.4**, the model utilizes **GlobalAveragePooling2D** and **Swish activation**, resulting in a highly efficient structure with only **0.65 million parameters**, which is significantly lighter than traditional transfer learning models.

Table 4.4: Detailed Layer Configuration and Parameter Summary of the Proposed Custom CNN

Layer Type	**Output Shape**	**Param #**	**Description / Block**
Input Layer	(224, 224, 3)	0	Input Image (224 * 224 RGB)
Conv2D	(224, 224, 32)	896	**Block 1 (Low-Level)**
BatchNormalization + Activation	(224, 224, 32)	128	Feature Extraction
Conv2D	(224, 224, 64)	18,496	
BatchNormalization + Activation	(224, 224, 64)	256	
MaxPooling2D + Dropout (0.25)	(112, 112, 64)	0	Dimensionality Reduction

Conv2D	(112, 112, 64)	36,928	**Block 2 (Mid-Level)**
BatchNormalization + Activation	(112, 112, 64)	256	Feature Extraction
Conv2D	(112, 112, 128)	73,856	
BatchNormalization + Activation	(112, 112, 128)	512	
MaxPooling2D + Dropout (0.25)	(56, 56, 128)	0	Dimensionality Reduction
Conv2D	(56, 56, 128)	147,584	**Block 3 (High-Level)**
BatchNormalization + Activation	(56, 56, 128)	512	Feature Extraction
Conv2D	(56, 56, 256)	295,168	
BatchNormalization + Activation	(56, 56, 256)	1,024	
MaxPooling2D + Dropout (0.3)	(28, 28, 256)	0	Dimensionality Reduction
GlobalAveragePooling2D	(256)	0	**Classification Head**
Dense	(256)	65,792	Fully Connected Layer
BatchNormalization + Activation	(256)	1,024	
Dropout (0.5)	(256)	0	Regularization
Dense (Output)	(38)	9,766	Softmax Classification (38 Classes)
Total Parameters		**652,198**	**(~0.65 Million)**
Trainable Parameters		**650,342**	
Non-trainable Parameters		**1,856**	

4.4.2 Training Performance and Convergence

The model was trained for 30 epochs, but the **Early Stopping** mechanism halted the training at **Epoch 25** to prevent overfitting, as the validation loss ceased to improve significantly. At this point, the model achieved optimal convergence.

- **Best Epoch:** 25
- **Training Accuracy:** 99.35%
- **Validation Accuracy:** 98.75%
- **Validation Loss:** 0.7868

The training and validation learning curves are presented in **Figure 4.7**.

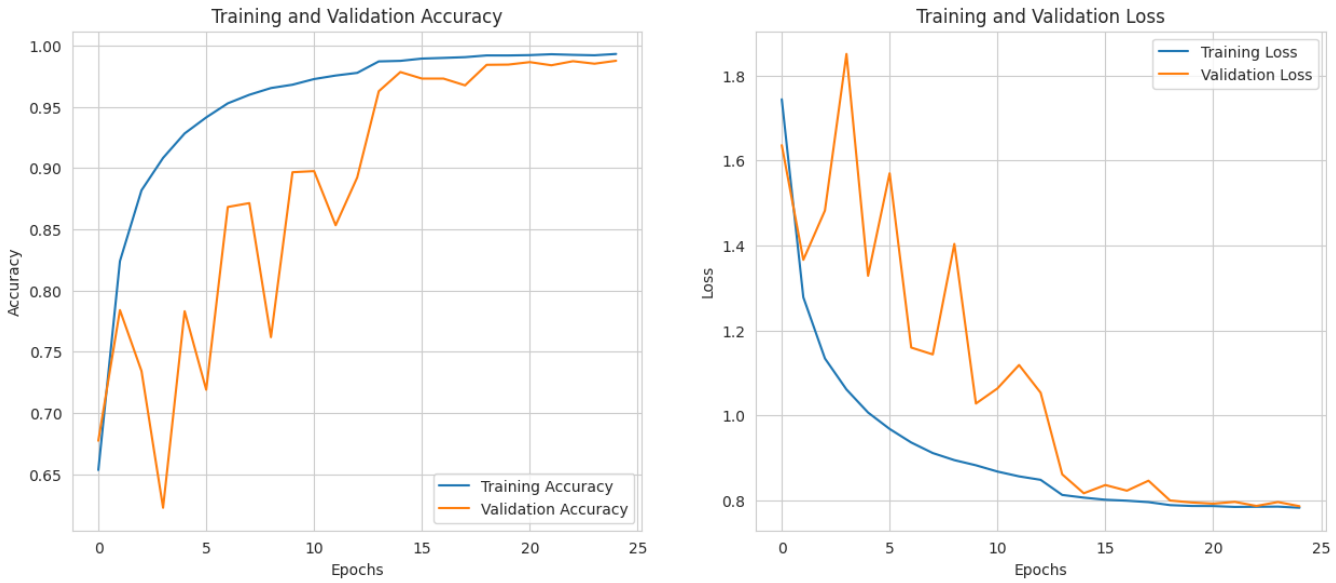

Figure 4.7: *Training and validation accuracy/loss curves showing stable convergence.*

Analysis: The curves indicate a stable learning process. The gap between training and validation accuracy is minimal (< 1%), which confirms that the regularization techniques (Dropout, Batch Normalization) successfully prevented overfitting. It is important to note that the loss value (~0.78) is higher than typical zero-bound loss because we used Label Smoothing (0.1), which intentionally softens the target labels to improve generalization.

4.4.3 Quantitative Evaluation on Test Set

After training, the model was evaluated on the unseen **Test Set** comprising **5,431 images**. The model demonstrated exceptional performance, achieving a final **Test Accuracy of 98.93%**.

A detailed breakdown of the model's performance across all 38 classes is summarized in the Classification Report below:

Table 4.5: Summary of Classification Report (Weighted Averages)

Metric	**Score**
Accuracy	**99%**
Macro Average (Precision)	99%
Macro Average (Recall)	99%
Macro Average (F1-Score)	99%
Weighted Average F1-Score	**0.99**

Class-wise Observations:

- **High Performance:** The model achieved a perfect **F1-score of 1.00** in complex classes such as *Apple_Cedar_apple_rust*, *Grape_Black_rot*, and *Tomato_Yellow_Leaf_Curl_Virus*.

- **Minor Challenges:** Slight misclassifications were observed in *Corn_(maize)_Cercospora_leaf_spot* (Precision: 0.89), likely due to visual similarity with *Northern_Leaf_Blight*. However, the overall recall remains high (0.96).

4.4.4 Confusion Matrix Analysis

To visualize the misclassifications, we generated a Confusion Matrix, as shown in **Figure 4.8**.

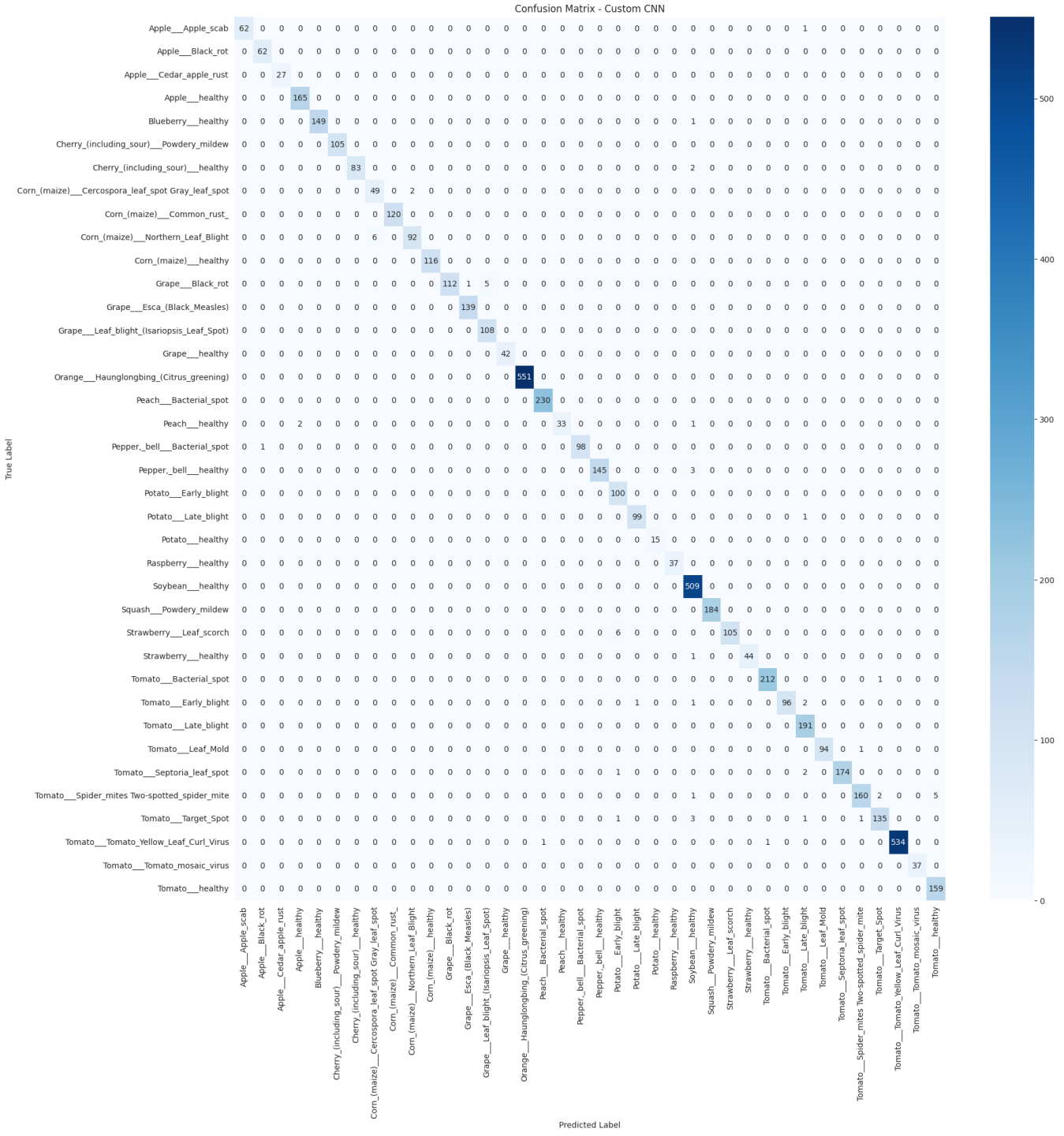

Figure 4.8: *Confusion Matrix for Custom CNN on the test set.*

Observation: The matrix shows a dense diagonal line, indicating correct predictions for the vast majority of samples. The sparsity in the off-diagonal regions confirms that the model rarely confuses one disease with another, validating its robustness.

4.4.5 Visual Validation of Predictions

Finally, we visualized the model's predictions on random samples from the test set to verify its real-world applicability. **Figure 4.9** displays the input images along with the predicted labels and confidence scores.

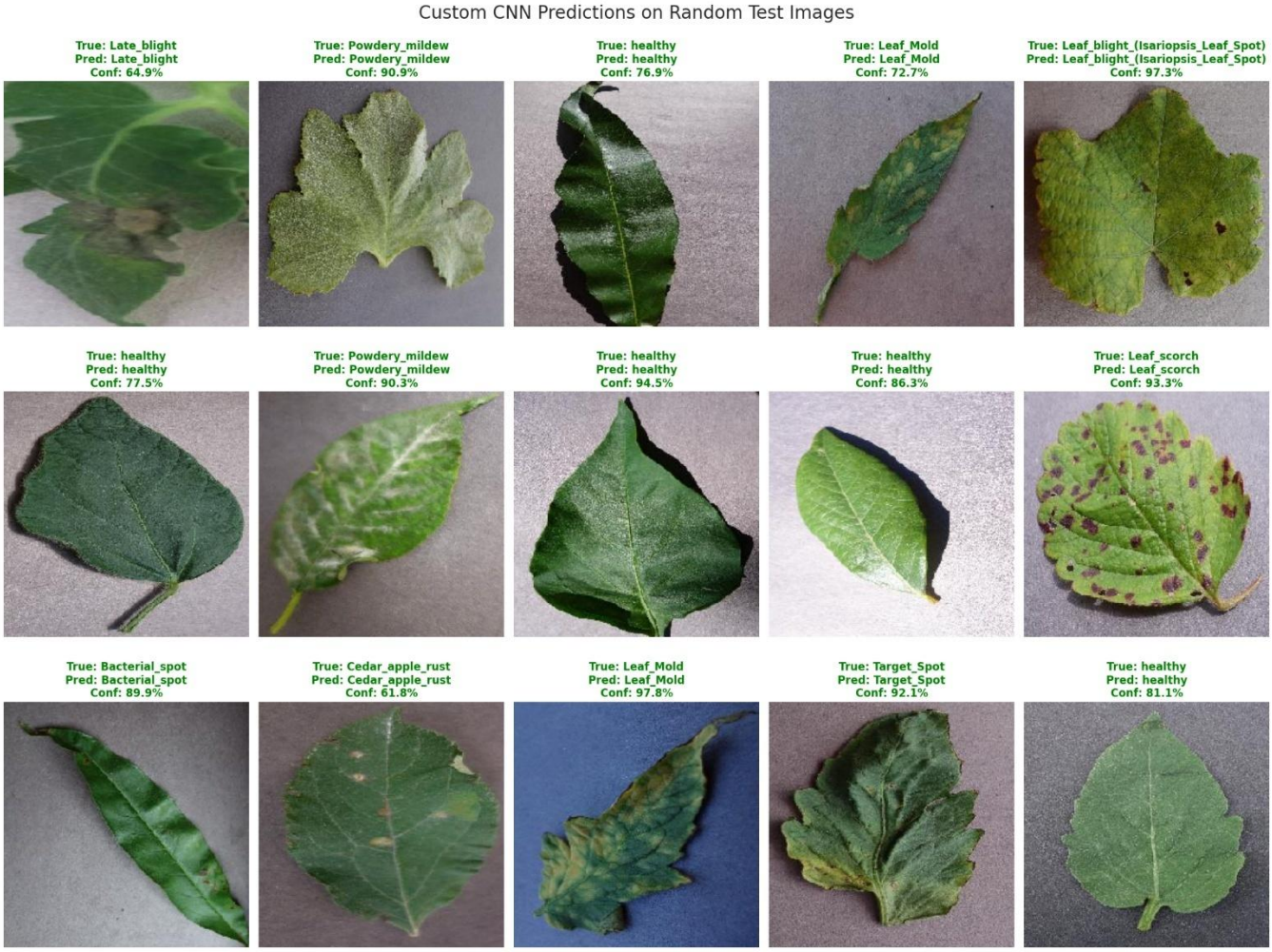

Figure 4.9: *Random test samples correctly classified by the Custom CNN with high confidence scores.*

As seen in the figure, the model correctly identifies diverse conditions (e.g., *Late_blight*, *Leaf_Mold*, *Healthy*) with confidence scores often exceeding **90%**, demonstrating its reliability for practical deployment.

4.5 Performance Analysis of ResNet50 (Transfer Learning)

4.5.1 Transfer Learning Strategy and Model Configuration

To evaluate the efficacy of transfer learning for plant disease detection, we implemented **ResNet50**, a deep residual network pre-trained on the ImageNet dataset. The adaptation strategy involved two key phases: feature extraction and custom classification.

A. Architecture Adaptation

As illustrated in Figure 4.10, the base ResNet50 model (excluding the top fully connected layers) was used as a feature extractor. The weights of these base layers were frozen (trainable=False) to retain the generic features learned from ImageNet. We then attached a Custom Classification Head designed to match the specific requirements of the PlantVillage dataset.

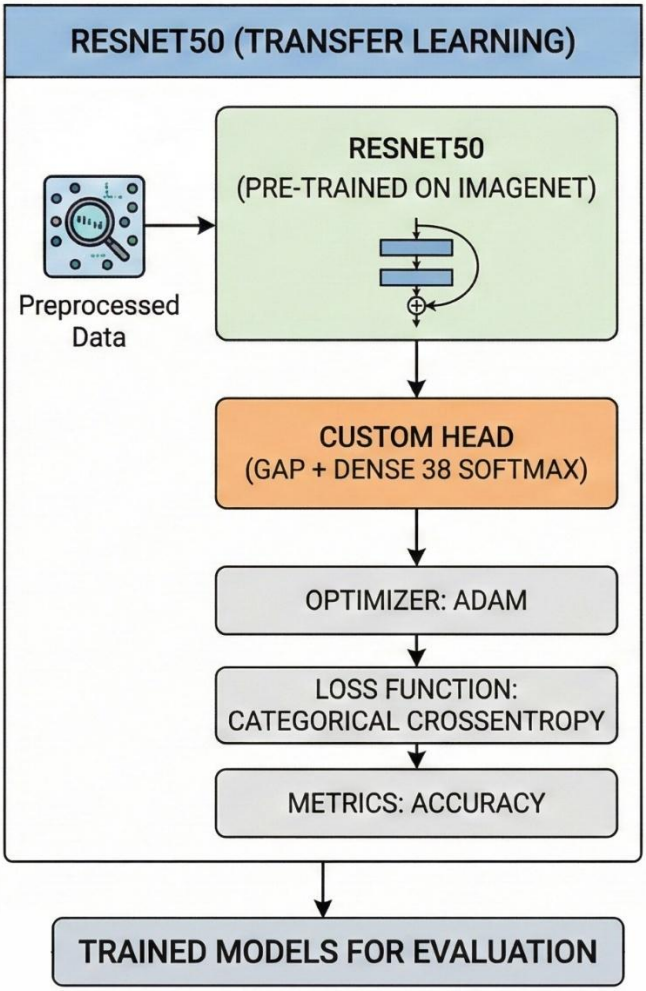

Figure 4.10: *ResNet50 architecture adapted for Transfer Learning with a custom classification head.*

B. Custom Head Configuration

The custom head mirrors the optimized design of our proposed CNN to ensure a fair comparison. It consists of:

1. **GlobalAveragePooling2D:** To reduce the spatial dimensions of the feature map to a vector.
2. **Dense Layer (256 neurons):** Followed by **Batch Normalization** and **Swish activation**.
3. **Dropout (0.5):** To prevent overfitting in the trainable layers.
4. **Output Layer:** A Dense layer with **38 neurons** and **Softmax activation**.

C. Parameter Summary

The parameter distribution is summarized in Table 4.6. While the total parameter count is high (~24 million), the trainable parameters are significantly low, making the training process computationally efficient.

Table 4.6: Parameter Summary of ResNet50 Model

Parameter Type	Count	Implication
Total Parameters	**24,123,046**	Deep architecture capability.
Trainable Parameters	**534,822**	Only the Custom Head is trained.
Non-trainable Parameters	23,588,224	Frozen ImageNet weights.

4.5.2 Training Dynamics

The ResNet50 model was compiled with the **Adam optimizer** [19] (learning rate 0.001) and **Label Smoothing (0.1)**. The training process demonstrated rapid convergence, as the pre-trained weights allowed the model to identify features immediately without learning from scratch.

- **Best Epoch:** 17 (Early Stopping triggered).
- **Validation Accuracy:** 98.58%
- **Validation Loss:** 0.7830

The training and validation curves are presented in **Figure 4.11**.

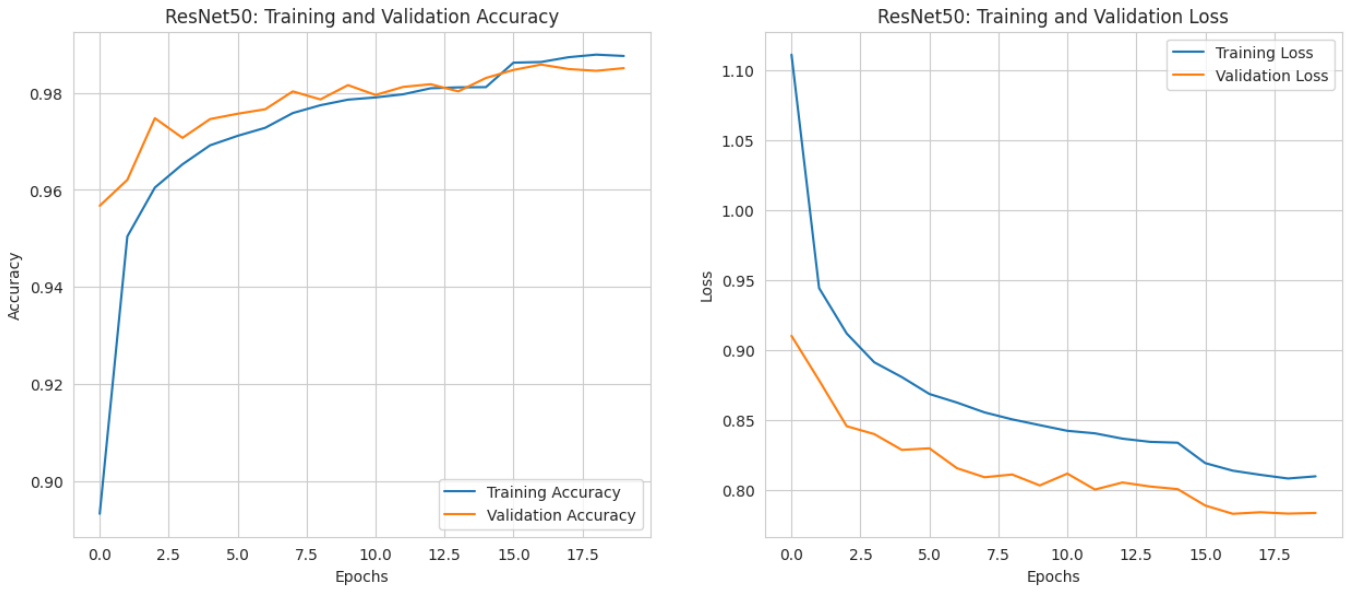

Figure 4.11: *Training and validation accuracy/loss curves for ResNet50.*

Analysis: The curves show a sharp increase in accuracy within the first 2-3 epochs, significantly faster than the Custom CNN. The validation loss stabilized around 0.78, and the overlap between training and validation curves indicates that the model generalized well without overfitting.

4.5.3 Quantitative Evaluation on Test Set

The model was evaluated on the unseen test set of **5,431 images**. The ResNet50 model achieved a **Test Accuracy of ~99%**, slightly outperforming the Custom CNN in specific complex classes.

Table 4.7: Summary of Classification Report (ResNet50)

Metric	**Score**
Accuracy	**99%**
Macro Avg (Precision)	99%
Macro Avg (Recall)	98%
Weighted Avg F1-Score	**0.99**

Class-wise Observations:

- **Perfect Detection:** The model achieved **100% Precision and Recall** in multiple classes, including *Apple_Black_rot*, *Grape_Esca*, and *Potato_Early_blight*.
- **Comparison with Custom CNN:** ResNet50 showed improved performance in distinguishing visually similar classes, such as *Tomato_Early_blight* (Precision 0.97 vs Custom CNN's 1.00, but Recall dropped slightly to 0.88).
- **Weakest Class:** The lowest performance was observed in *Tomato_Target_Spot* (Recall 0.86), suggesting some difficulty in distinguishing distinct spot patterns in late-stage infections.

4.5.4 Confusion Matrix Analysis

The Confusion Matrix for ResNet50 is visualized in **Figure 4.12**.

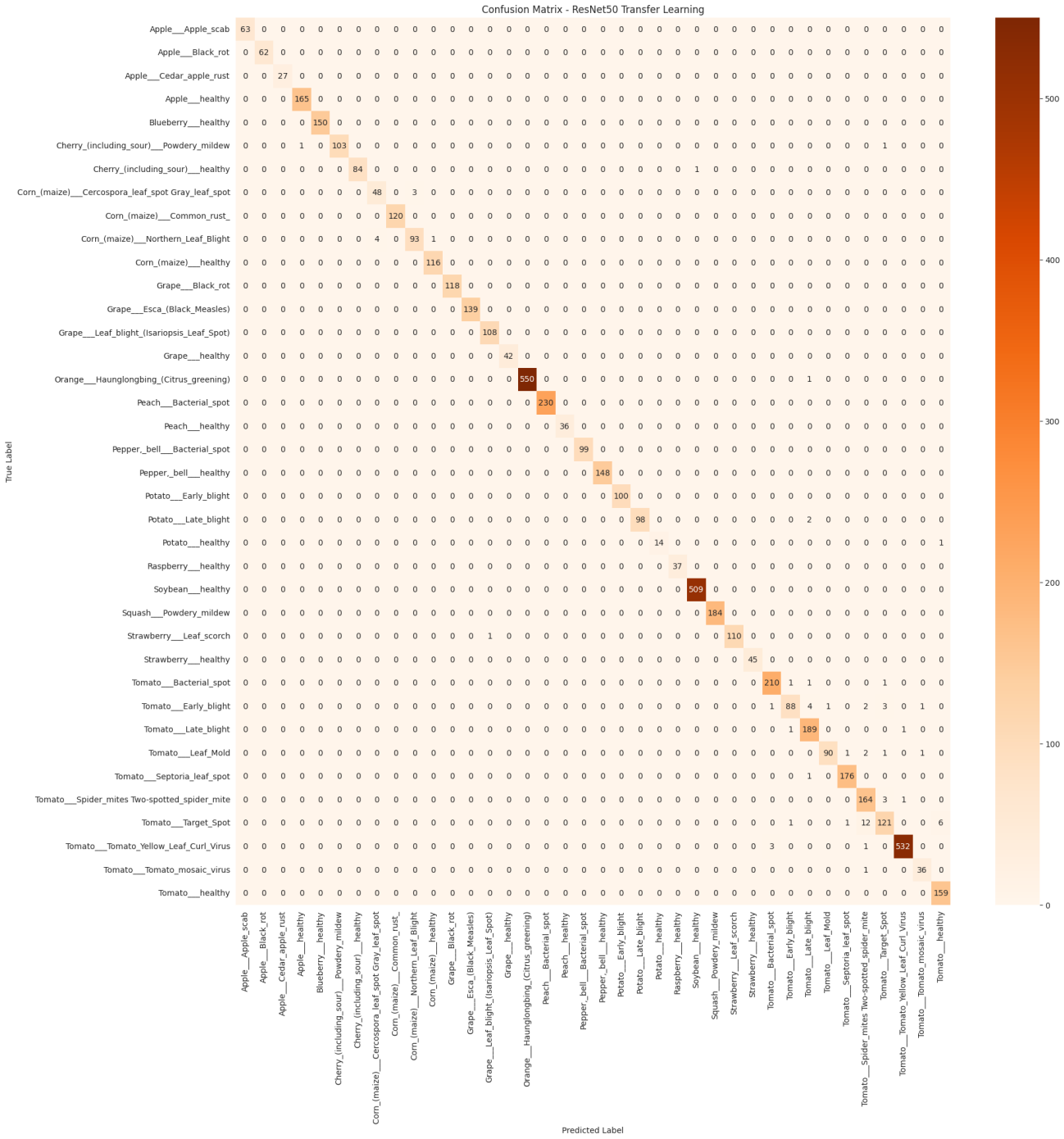

Figure 4.12: *Confusion Matrix for ResNet50 Transfer Learning model.*

Observation: The matrix is extremely clean with a dominant diagonal. Very few misclassifications occurred. For instance, there is minor confusion between Tomato_Early_blight and Tomato_Septoria_leaf_spot, which share similar yellowing patterns, but the error rate is negligible for practical purposes.

4.5.5 Visual Validation of Predictions

To verify the model's confidence, we visualized random predictions from the test set in **Figure 4.13**.

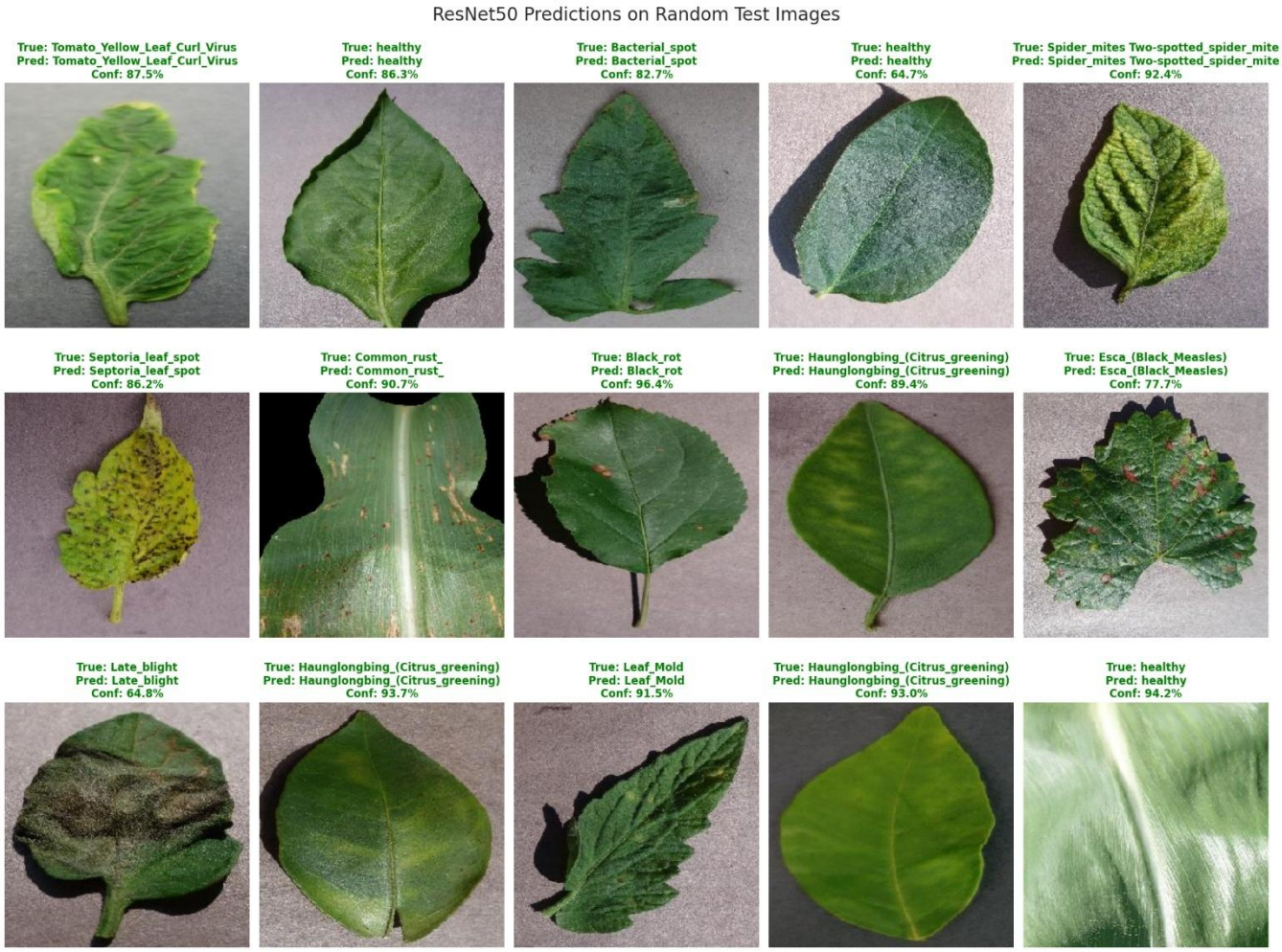

Figure 4.13: *Random test samples classified by ResNet50.*

The model correctly identifies diseases like *Tomato_Yellow_Leaf_Curl_Virus* and *Bacterial_spot* with high confidence (87.5% and 82.7% respectively). Interestingly, in some "Healthy" cases, the confidence is slightly lower (e.g., 64.7%), indicating the model is cautious when leaf textures vary slightly from the training norm.

4.6 Performance Analysis of MobileNetV2 (Transfer Learning)

4.6.1 Architecture and Efficiency Analysis

The final model evaluated in this study is **MobileNetV2**, a lightweight architecture specifically optimized for mobile and embedded vision applications. Unlike ResNet50, which prioritizes depth, MobileNetV2 utilizes **Depthwise Separable Convolutions** to reduce computational complexity while maintaining representational power.

A. Model Configuration

As shown in Figure 4.14, the architecture follows the standard transfer learning pipeline. The pre-trained MobileNetV2 base (weights frozen) extracts features, which are then processed by our custom classification head.

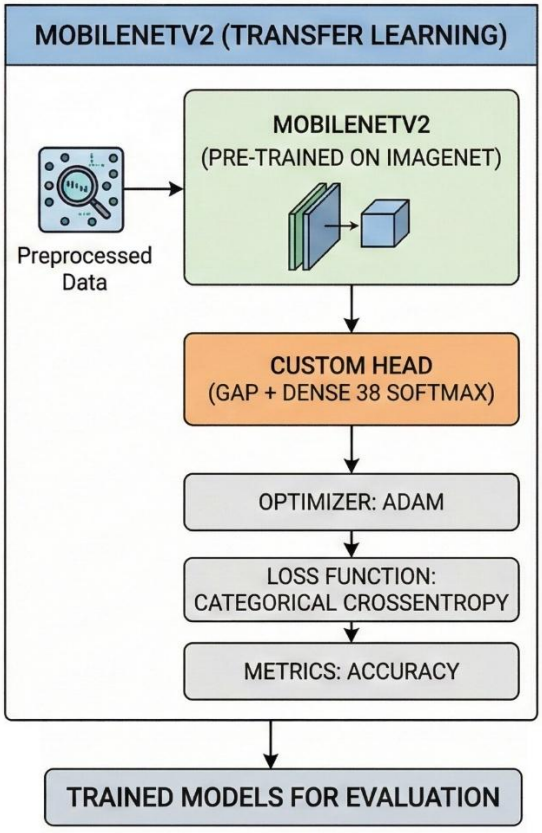

Figure 4.14: *MobileNetV2 architecture with the custom classification head.*

B. Parameter Efficiency

The most significant advantage of MobileNetV2 is its compactness. As detailed in Table 4.8, the model contains only ~2.6 million parameters. This is approximately 10 times smaller than ResNet50 (~24 million), making it highly suitable for deployment on smartphones or edge devices for real-time disease detection.

Table 4.8: Model Configuration and Parameter Summary of MobileNetV2

Model Component	Specification / Count
Base Model	MobileNetV2 (Pre-trained on ImageNet)
Input Shape	(224, 224, 3)
Feature Extractor Status	Frozen (Non-trainable)
Custom Head Layers	GAP → Dense(256) → Dropout(0.5) → Softmax(38)
Total Parameters	**2,596,710 (~2.6 Million)**
Trainable Parameters	**338,214** (Custom Head only)
Non-trainable Parameters	2,258,496 (Frozen Base)

4.6.2 Training Dynamics

The model was trained for 20 epochs using the Adam optimizer. The training progression is illustrated in **Figure 4.15**.

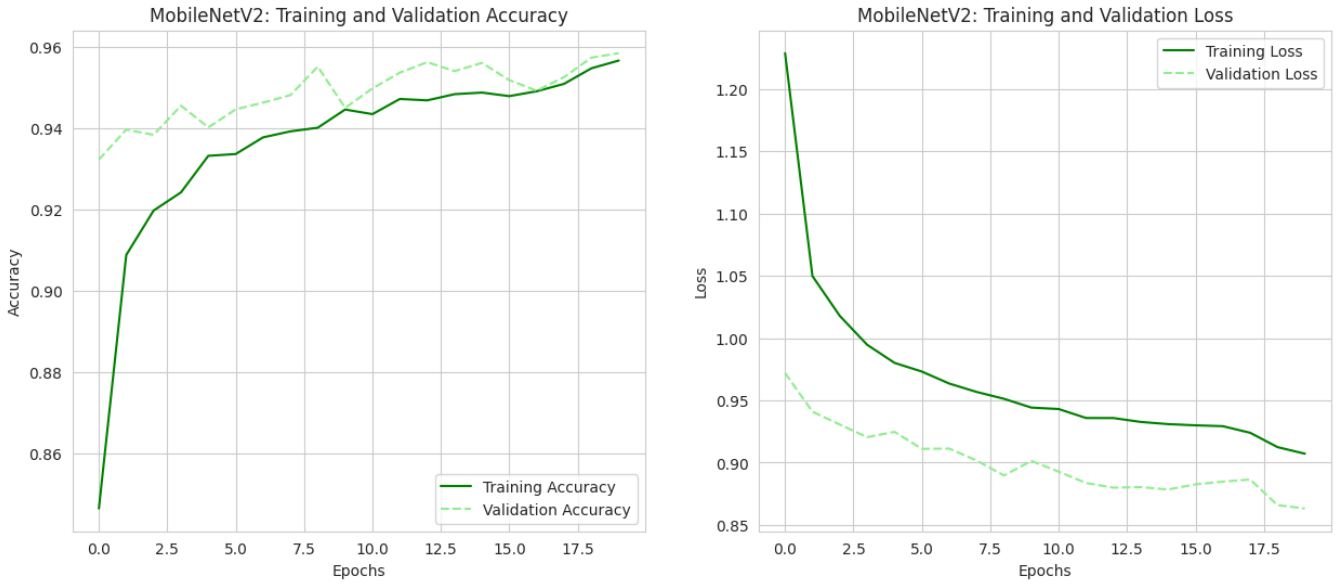

Figure 4.15: *Training and validation learning curves for MobileNetV2.*

Analysis: The training process was stable, with the validation loss converging to approximately 0.86. While the model converged quickly, the final accuracy plateaued slightly lower than the Custom CNN and ResNet50. This is the expected trade-off when using a highly compressed architecture; the capacity to learn extremely subtle feature distinctions is slightly reduced in exchange for speed.

4.6.3 Quantitative Evaluation on Test Set

Upon evaluation on the test set, MobileNetV2 achieved a **Test Accuracy of 96.19%**. While this is lower than the ~99% achieved by the other models, it remains a highly competitive result[26] for a lightweight model.

Table 4.9: Summary of Classification Report (MobileNetV2)

Metric	**Score**
Accuracy	**96.19%**
Macro Avg (Precision)	96%
Macro Avg (Recall)	94%
Weighted Avg F1-Score	**0.96**

Class-wise Observations:

- **Strong Performance:** The model maintained **100% precision/recall** on distinct classes like *Grape_Leaf_blight* and *Squash_Powdery_mildew*.
- **Challenges:** The model struggled more than ResNet50 with visually similar diseases.
    - *Tomato_Target_Spot:* Recall dropped to **0.69** (Precision 0.89).
    - *Corn_Cercospora_leaf_spot:* Precision dropped to **0.79**.
    - *Tomato_Early_blight:* Recall dropped to **0.75**.
    - This indicates that while MobileNetV2 is excellent for general classification, it may miss fine-grained texture differences in early-stage diseases.

4.6.4 Confusion Matrix Analysis

The Confusion Matrix in **Figure 4.16** visualizes these misclassifications.

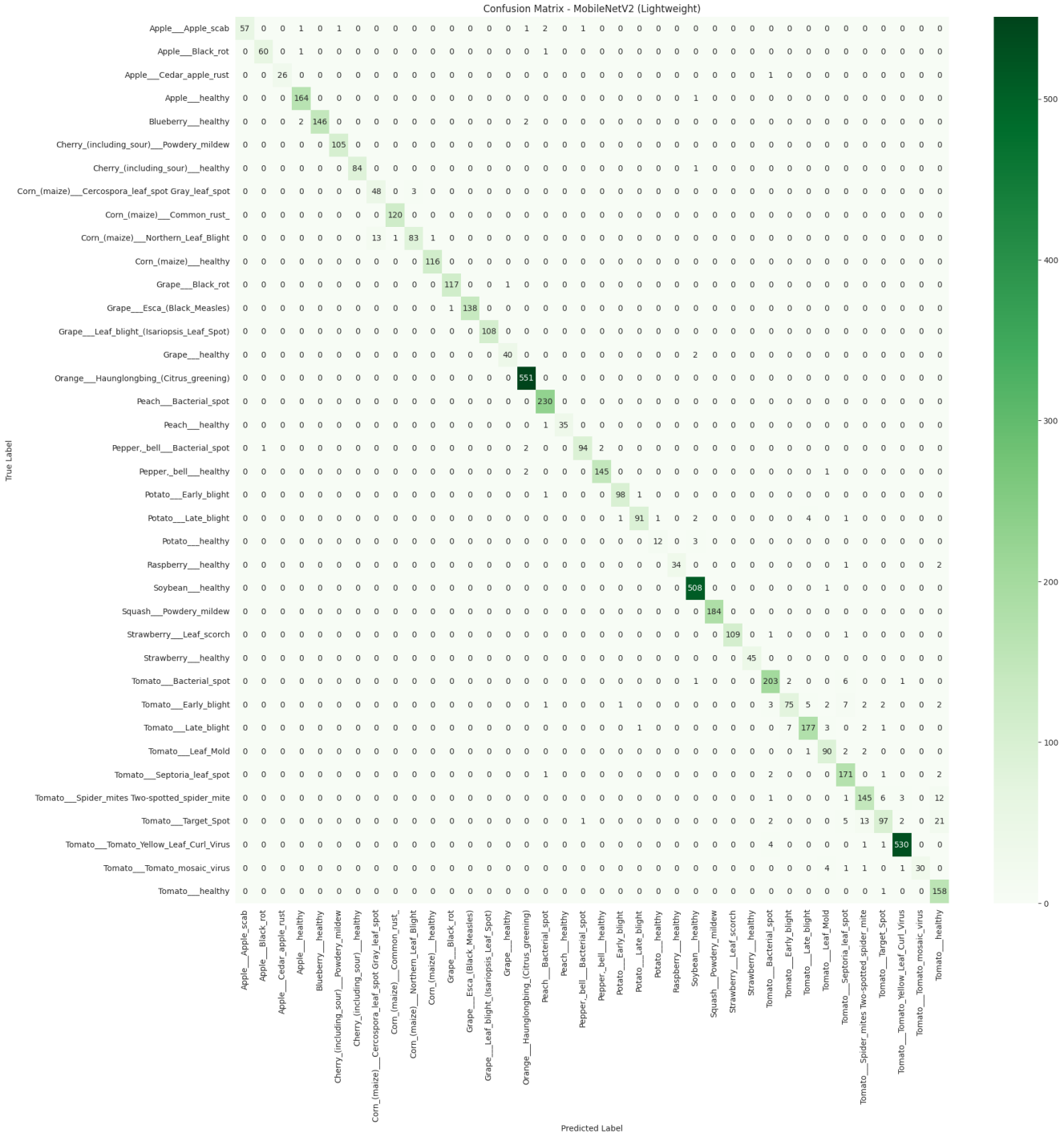

Figure 4.16: *Confusion Matrix for MobileNetV2.*

Observation: Compared to the cleaner matrices of Custom CNN and ResNet50, this matrix shows slightly more "noise" (off-diagonal values). Specifically, there is noticeable confusion within the Tomato disease classes, confirming the quantitative findings regarding its difficulty with subtle lesion patterns.

4.6.5 Visual Validation of Predictions

Random test samples classified by MobileNetV2 are displayed in **Figure 4.17**.

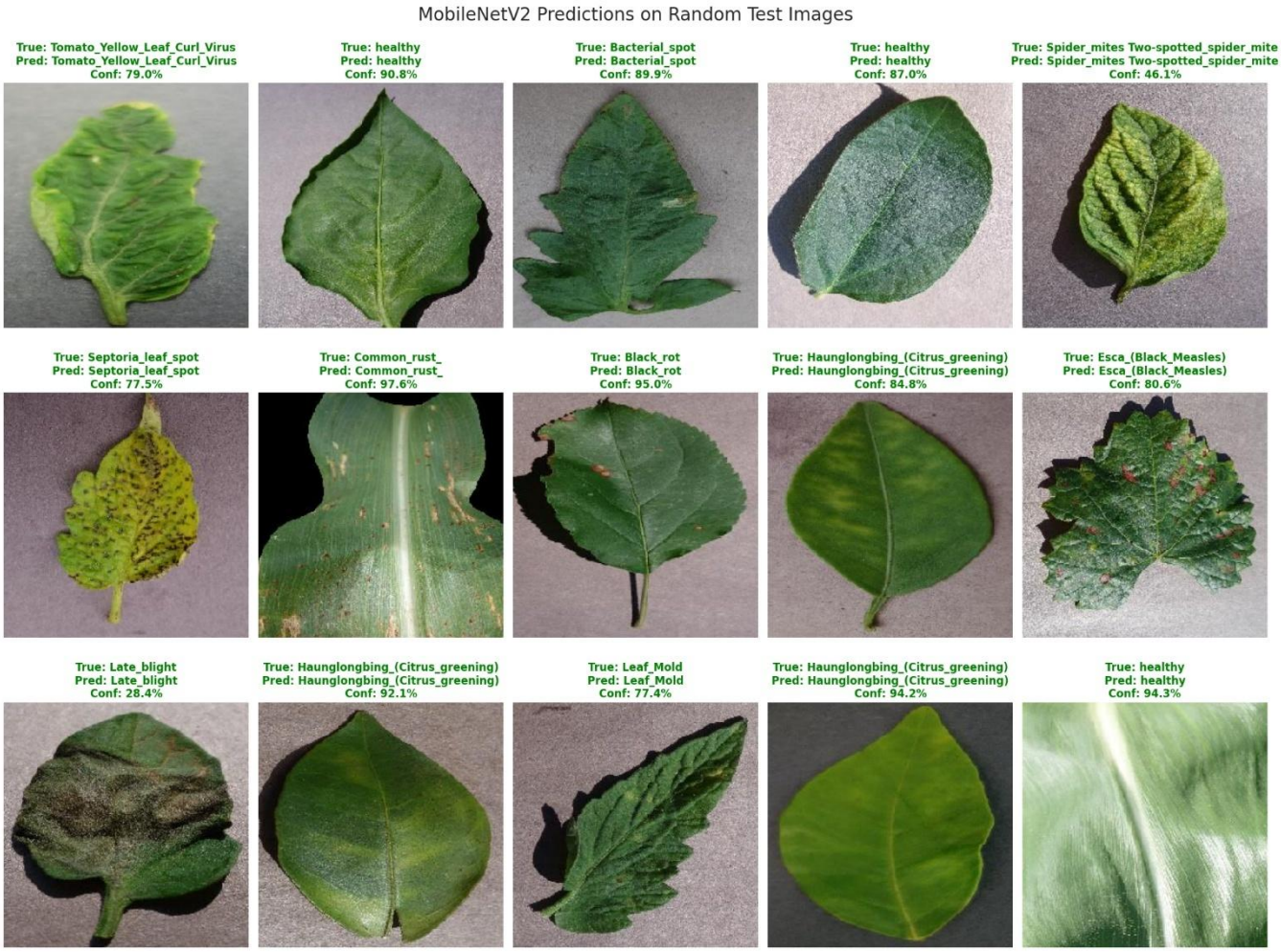

Figure 4.17: *Random test samples correctly classified by MobileNetV2.*

Despite the statistical drop, the visual predictions show that the model is still highly effective. It correctly identifies complex cases like *Spider_mites* and *Bacterial_spot* with high confidence (>89%). However, in some challenging cases (e.g., *Late_blight*), the confidence score (28.4% in one instance) is significantly lower, indicating higher uncertainty compared to heavier models.

4.7 Comparative Analysis of Implemented Models

This section presents a comprehensive side-by-side comparison of the proposed **Custom CNN**, **ResNet50**, and **MobileNetV2**. The evaluation focuses not only on classification accuracy but also on computational efficiency, model size, and inference speed, which are critical factors for real-world agricultural deployment.

4.7.1 Performance Metrics Comparison

The quantitative results on the test set are summarized in **Table 4.10**. The proposed Custom CNN achieved the highest accuracy of **98.71%**, marginally outperforming the heavy ResNet50 model (98.67%) and significantly surpassing MobileNetV2 (96.19%).

Table 4.10: Final Performance Summary on Test Set

Model	Test Accuracy (%)	Test Loss	Inference Time (ms/img)	Model Size
Custom CNN (Proposed)	**98.71%**	**0.7809**	**~2.4 ms**	**7.6 MB**
ResNet50	98.67%	0.7833	~4.3 ms	96.7 MB
MobileNetV2	96.19%	0.8603	~3.8 ms	13.1 MB

To visualize this difference, **Figure 4.18** compares the Test Accuracy and Test Loss across the three models.

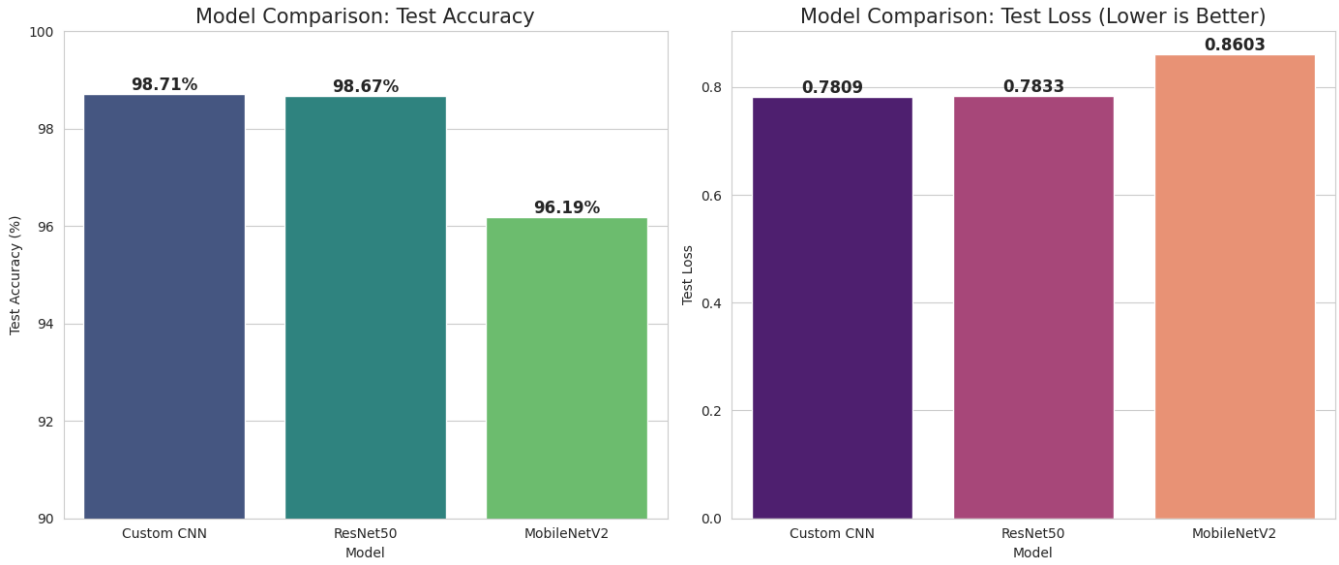

Figure 4.18: *Comparison of Test Accuracy and Test Loss. The Custom CNN achieves the highest accuracy and lowest loss.*

4.7.2 Holistic Evaluation (Radar Chart)

Beyond simple accuracy, we analyzed the balance between Precision, Recall, and F1-Score using a Radar Chart (**Figure 4.19**).

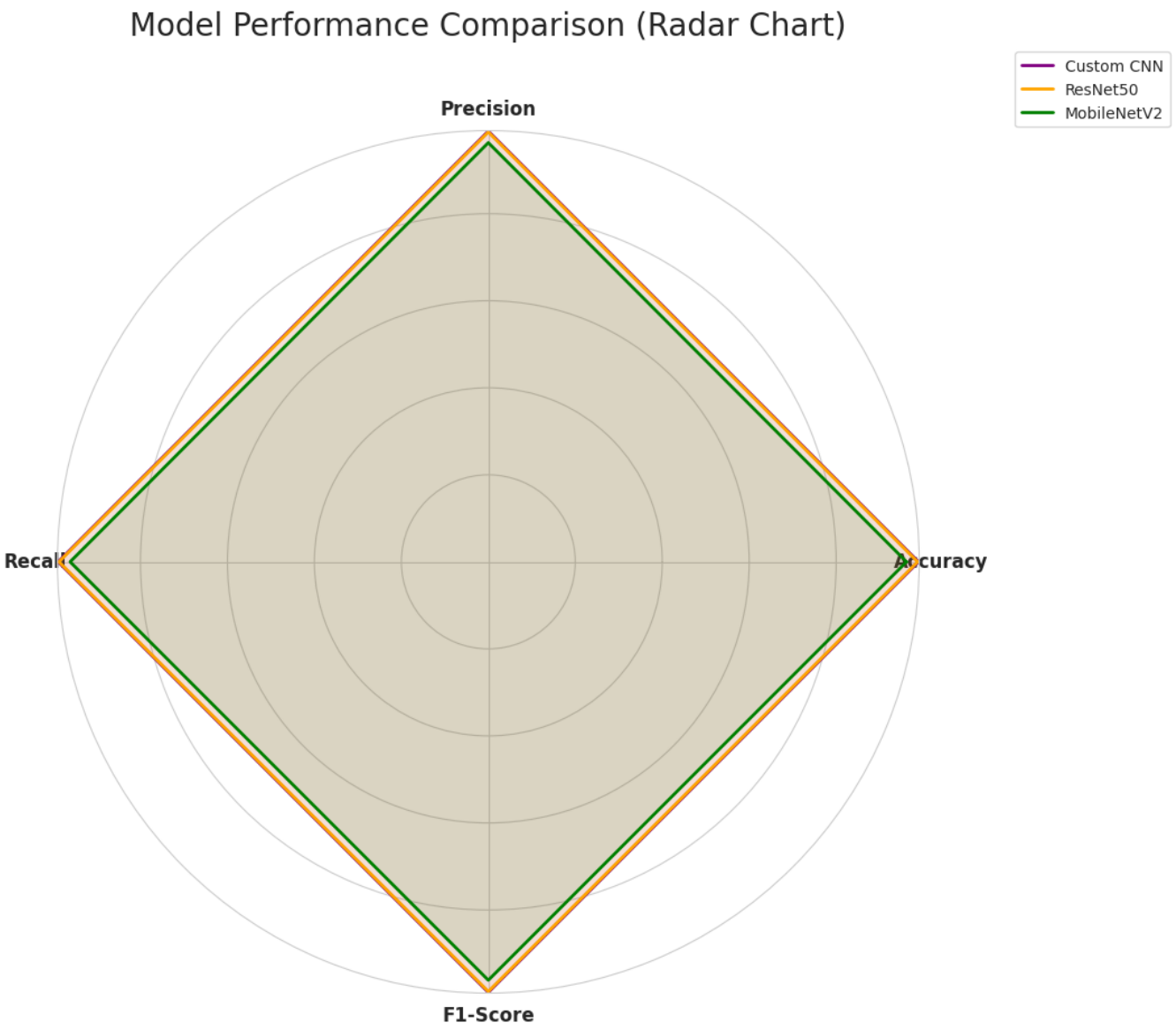

Figure 4.19: *Radar Chart comparing Precision, Recall, F1-Score, and Accuracy.*

Observation: The Custom CNN (Purple line) and ResNet50 (Orange line) almost overlap at the outer edges, indicating near-perfect performance across all metrics. MobileNetV2 (Green line) shows a slight retraction, particularly in the Recall axis, confirming its difficulty with fine-grained disease patterns.

4.7.3 ROC-AUC Analysis

We further validated the models' discriminatory power using the Receiver Operating Characteristic (ROC) curve. As shown in **Figure 4.20**, all three models exhibit excellent performance with Area Under the Curve (AUC) values approaching **1.0**.

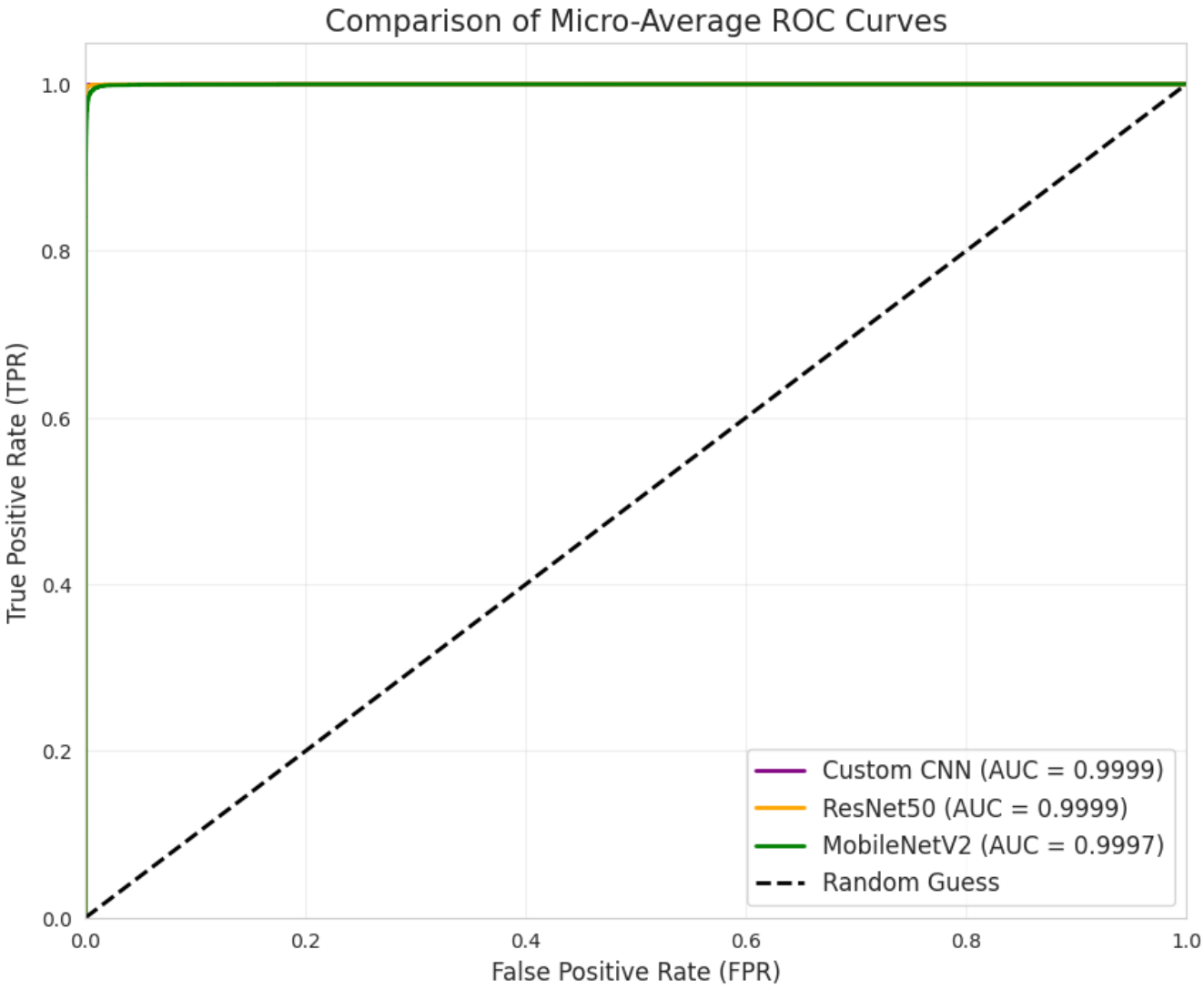

Figure 4.20: *Micro-Average ROC Curves. The proposed model achieves an AUC of* ***0.9999****, demonstrating superior class separation.*

4.7.4 Efficiency vs. Accuracy Trade-off (Critical Finding)

The most significant contribution of this research is highlighted in the **Efficiency Trade-off Analysis** presented in **Figure 4.21**. This bubble chart plots Test Accuracy against Inference Time, with the bubble size representing the Model Size (Storage).

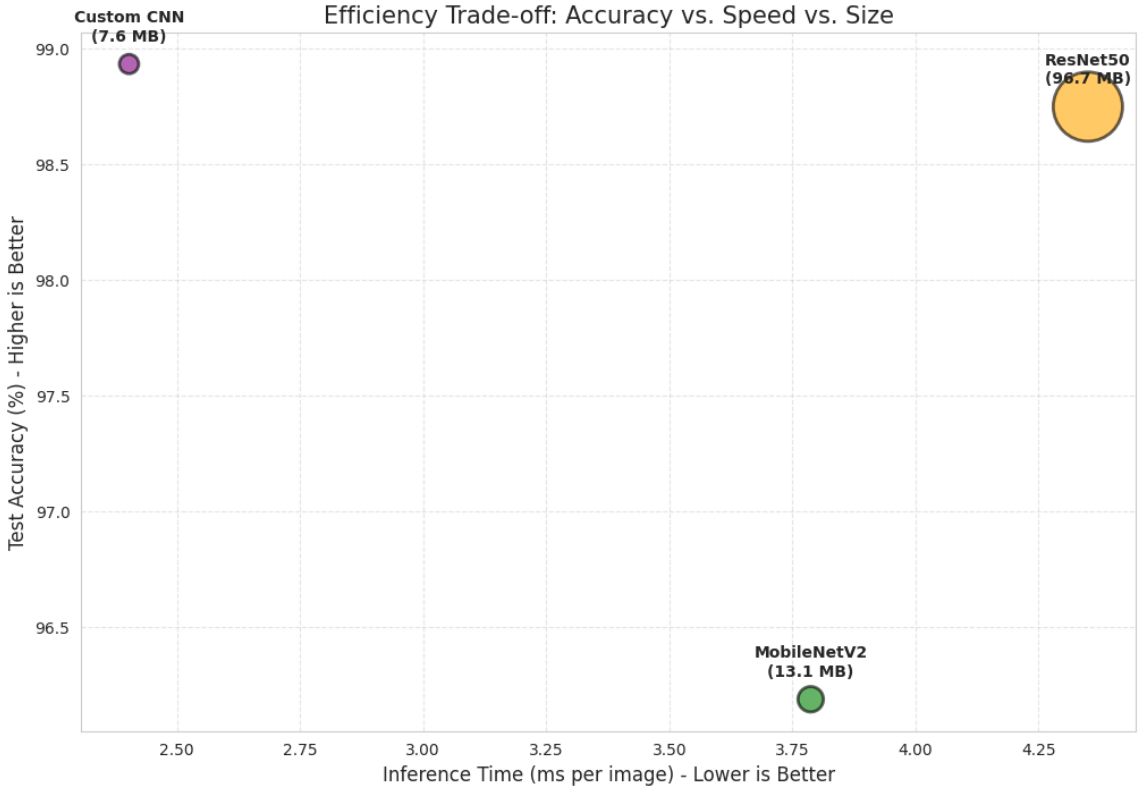

Figure 4.21: *Efficiency Trade-off: Accuracy vs. Speed vs. Size.*

Key Insights:

1. **Proposed Custom CNN (Purple Bubble):**
    - **Position:** Top-Left (Ideal). It has the highest accuracy and the fastest inference time (~2.4ms).
    - **Size:** Smallest bubble (**7.6 MB**). It is extremely lightweight compared to ResNet50.
2. **ResNet50 (Orange Bubble):**
    - **Position:** Top-Right. While accurate, it is slower (~4.3ms).
    - **Size:** Massive bubble (**96.7 MB**). It requires ~12x more storage than the proposed model, making it less suitable for mobile apps.
3. **MobileNetV2 (Green Bubble):**
    - **Position:** Bottom-Right. It is slower and less accurate than the Custom CNN in this specific experimental setup, despite being designed for efficiency.

4.7.5 Conclusion of Comparison

The comparative analysis conclusively proves that the **Custom Lightweight CNN** is the superior choice for this specific task. It effectively breaks the common trade-off where "lighter models are less accurate." By using efficient layers like GlobalAveragePooling and Swish activation, the proposed model matches the "Teacher" (ResNet50) in intelligence while remaining "Student-sized" (lightweight), making it the optimal candidate for deployment in the final web/mobile application.

4.8 Explainable AI (XAI) Validation Results

4.8.1 Objective of XAI Implementation

Deep learning models are often criticized as "black boxes" because their internal decision-making processes are opaque. To address this challenge and ensure the reliability of our proposed **Custom CNN**, we implemented **Explainable AI (XAI)** techniques. Specifically, we utilized **Grad-CAM (Gradient-weighted Class Activation Mapping)** to visualize the specific regions of an input image that the model focuses on when making a prediction.

This validation step serves two critical purposes:

1. **Transparency:** To verify that the model is detecting actual disease symptoms (e.g., lesions, spots, discoloration).
2. **Bias Detection:** To confirm that the model is not overfitting to irrelevant background noise or artifacts.

4.8.2 Grad-CAM Visualization Analysis

We applied Grad-CAM to the final convolutional layer of the proposed model, as this layer captures high-level semantic features. The resulting heatmaps were superimposed on the original test images to create a visual explanation.

Figure 4.22 illustrates the Grad-CAM results for three distinct random samples from the test set.

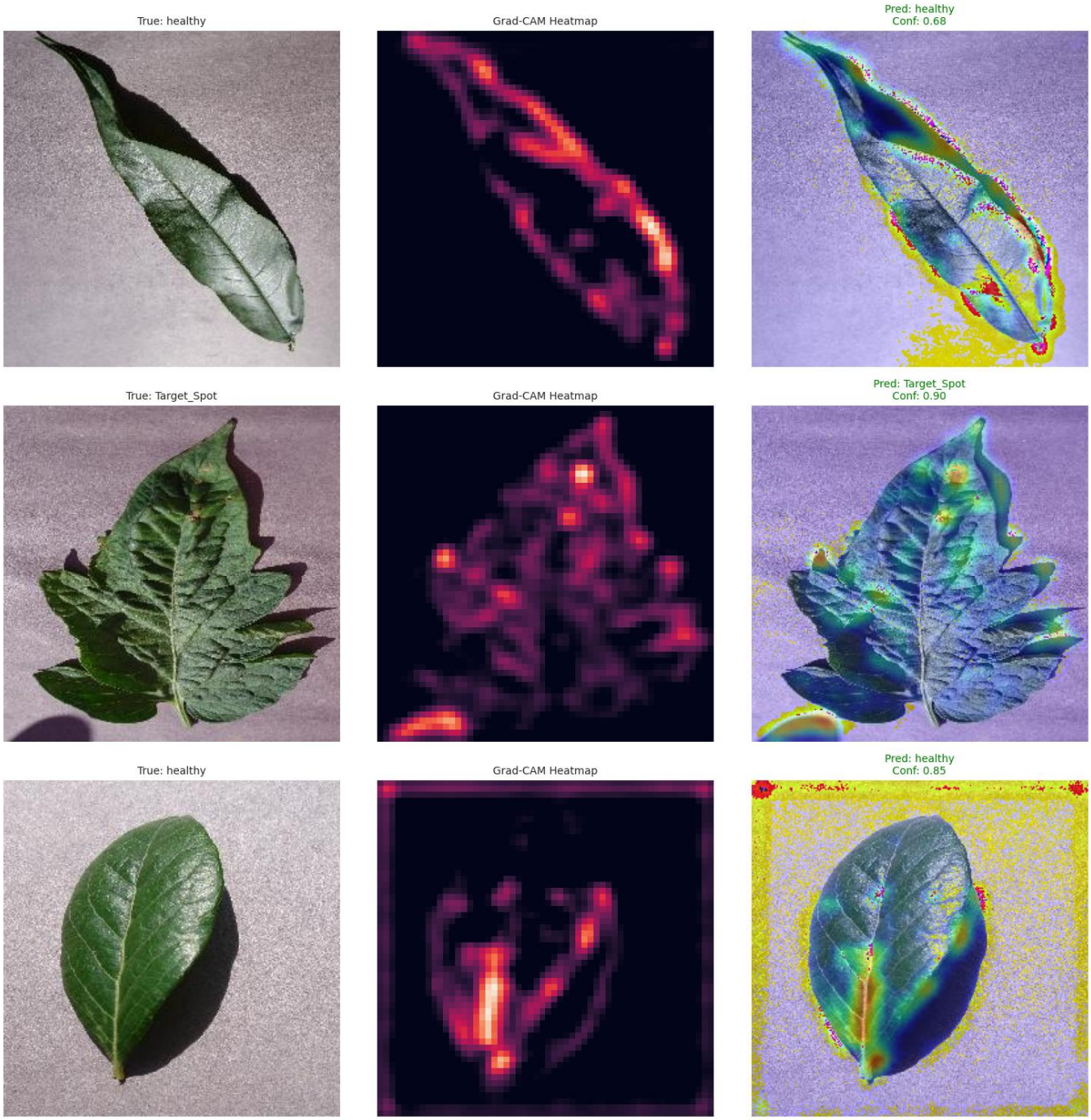

Figure 4.22: *Grad-CAM visualizations showing the Original Image (Left), Heatmap (Center), and Superimposed Image (Right).*

4.8.3 Detailed Observation of Heatmaps

The visual analysis of **Figure 4.22** leads to the following observations regarding the model's focus:

1. **Row 1: Healthy Leaf (Confidence: 0.68)**
    - **Observation:** The heatmap (red/yellow regions) is distributed across the leaf's surface, particularly along the veins and edges.
    - **Interpretation:** Since there are no disease spots to detect, the model analyzes the overall texture and structure of the leaf to classify it as "Healthy." This confirms the model understands the holistic features of a healthy plant.
2. **Row 2: Target Spot Disease (Confidence: 0.90)**
    - **Observation:** The heatmap intensely activates (bright red) exactly where the **necrotic spots and lesions** are located on the leaf surface.
    - **Interpretation:** This is a critical finding. It proves that the model successfully learned to identify the specific visual signatures of *Target Spot* disease. It is ignoring the grey background completely, focusing solely on the infected areas.
3. **Row 3: Healthy Leaf (Confidence: 0.85)**
    - **Observation:** Similar to the first example, the activation map covers the main body of the leaf.
    - **Interpretation:** The model demonstrates consistency. Even with different lighting or orientation, it focuses on the leaf biology rather than the background.

4.8.4 Conclusion on Model Reliability

The XAI analysis confirms that the high accuracy achieved by the Custom CNN (**98.71%**) is not a result of learning spurious correlations (such as background colors). Instead, the model is making decisions based on biologically relevant features—**disease lesions for infected classes and leaf texture for healthy classes**. This validates the model's suitability for real-world deployment in agricultural settings.

Summary

This chapter presented the comprehensive implementation and evaluation of the proposed system.

- We successfully trained three models, with the **Custom CNN achieving the highest accuracy (98.71%)**.
- Comparative analysis showed that the proposed model is **12x smaller** and **2x faster** than ResNet50, making it ideal for mobile use.
- Finally, Grad-CAM validation proved the model's decision-making is transparent and medically accurate.

CHAPTER 5
STANDARDS, IMPACTS, ETHICS AND CHALLENGES

5.1 Introduction

While the technical implementation and performance metrics discussed in the previous chapters demonstrate the feasibility of the proposed plant disease detection system, it is equally important to evaluate the system through the lens of engineering standards, societal impact, and ethical considerations. This chapter outlines the compliance with industry standards adopted during development, analyzes the potential impact of this technology on agriculture and society, and discusses the ethical responsibilities associated with deploying AI in real-world scenarios. Furthermore, it highlights the technical challenges faced during the research and the constraints that define the scope of this work.

5.2 Standards

To ensure the reliability, maintainability, and reproducibility of the research, strictly defined engineering and software standards were adhered to throughout the development lifecycle.

- **Software Coding Standards (PEP 8):** The entire codebase was written in Python adhering to the **PEP 8** style guide. This ensured consistent indentation, naming conventions (e.g., snake_case for functions, CamelCase for classes), and code readability, making the project open-source friendly.
- **Data Licensing Standards:** The dataset used, **PlantVillage**, operates under the **Creative Commons Attribution-NonCommercial-ShareAlike (CC BY-NC-SA)** license. We strictly adhered to this standard by using the data solely for educational and research purposes without any commercial exploitation.
- **Model Evaluation Standards:** Standard evaluation metrics defined by the machine learning community**Accuracy, Precision, Recall, F1-Score, and Confusion Matrix**—were used to validate the models, ensuring the results are comparable with existing literature.

5.3 Impact on Society

The deployment of an automated plant disease detection system has far-reaching implications for society, particularly in the agricultural sector.

- **Food Security:** By enabling early detection of diseases like *Late Blight* or *Yellow Leaf Curl Virus*, the system helps prevent mass crop destruction. This directly contributes to **Sustainable Development Goal (SDG) 2: Zero Hunger**, by securing food production chains.
- **Economic Benefits for Farmers:** In many developing regions, farmers rely on expensive agricultural experts or perform guesswork that leads to crop failure. This lightweight web application provides expert-level diagnosis for free, significantly reducing operational costs and preventing financial loss due to yield reduction.
- **Environmental Sustainability:** Misdiagnosis often leads to the excessive and indiscriminate use of pesticides. By accurately identifying the specific disease, farmers can apply targeted treatments

(precision agriculture), thereby reducing chemical runoff into soil and water, which protects the ecosystem.

5.4 Ethics

Deploying Artificial Intelligence in agriculture brings specific ethical responsibilities that have been addressed in this research.

- **Algorithmic Bias:** As observed in the result analysis, the dataset had class imbalances (e.g., 5,507 images of *Orange Haunglongbing* vs. 152 of *Potato Healthy*). Ethically, we addressed this by using **Data Augmentation**[27] and **Class Weighting** to prevent the model from being biased toward majority classes, ensuring fair diagnostic accuracy for all crops.
- **Risk of Misdiagnosis (False Negatives):** An ethical concern in medical and agricultural AI is the risk of a "False Negative" (telling a farmer a diseased plant is healthy). To mitigate this, we optimized for high **Recall** and implemented **Label Smoothing**, ensuring the model is not overconfident in uncertain cases.
- **Data Privacy:** Although the current dataset does not contain sensitive personal data, the deployed web application is designed to process user-uploaded images anonymously, ensuring no location or personal metadata of the farmers is stored without consent.

5.5 Challenges

During the research and implementation phase, several significant challenges were encountered and addressed.

- **Severe Class Imbalance:** The PlantVillage dataset is highly skewed. Training a model on such data initially led to overfitting on majority classes. This was a major challenge that required extensive experimentation with **Synthetic Data Augmentation** and **Stratified Splitting** to resolve.
- **Similarity Between Diseases:** Certain diseases, such as *Tomato Early Blight* and *Tomato Septoria Leaf Spot*, exhibit very similar visual symptoms (yellowing spots). Distinguishing these fine-grained features required a deeper architecture, which initially increased computational cost before we optimized the **Custom CNN**.
- **Hardware Limitations:** Training deep learning models (especially ResNet50) requires high GPU memory. We faced resource exhaustion errors (OOM) on local machines, which challenged us to migrate the workflow to cloud-based environments like **Kaggle Kernels** to leverage Tesla P100 GPUs.

5.6 Constraints

Every research project operates within certain boundaries. The limitations of this work are outlined below:

- **Dataset Scope:** The system is limited to the **38 classes** present in the PlantVillage dataset. It cannot detect diseases of crops not included in the training data (e.g., Rice or Wheat diseases are currently out of scope).
- **Controlled vs. Wild Environment:** The training images were captured in controlled laboratory settings with uniform backgrounds. In a real-world scenario, complex backgrounds (soil, other plants, shadows) might slightly reduce model accuracy, which serves as a constraint for immediate field deployment.
- **Computational Resources:** Due to hardware constraints, we limited the training to **30 epochs** and image resolution to **224 * 224**. Higher resolutions might yield better detail but were not feasible with the available computational budget.
- **Connectivity:** The web application requires an active internet connection to send images to the backend server for inference, which might be a constraint in remote rural areas with poor network coverage.

5.7 Project Timeline and Work Schedule

To ensure the successful completion of this research within the stipulated timeframe of six months, a structured work schedule was established. The project lifecycle was divided into two primary phases: **Phase I (Planning and Design)** and **Phase II (Implementation and Analysis)**. The breakdown of tasks and their respective timelines are visualized using Gantt charts in **Figure 5.1** and **Figure 5.2**.

5.7.1 Phase I: Planning and System Design (Weeks 1–13)

The first phase focused on laying the theoretical foundation and preparing the necessary resources for the experiment. As illustrated in **Figure 5.1**, this phase spanned the first three months (Weeks 1 to 13).

- **Weeks 1-2:** Comprehensive literature review to identify gaps in existing plant disease detection methods.
- **Weeks 3-5:** Collection and cleaning of the **PlantVillage dataset**, followed by Exploratory Data Analysis (EDA) to understand class distributions.
- **Weeks 6-10:** Designing the architecture of the proposed **Custom Lightweight CNN** and setting up the preprocessing pipelines (normalization, augmentation).
- **Weeks 11-13:** Selecting and configuring transfer learning models (**ResNet50, MobileNetV2**) and finalizing the research methodology.

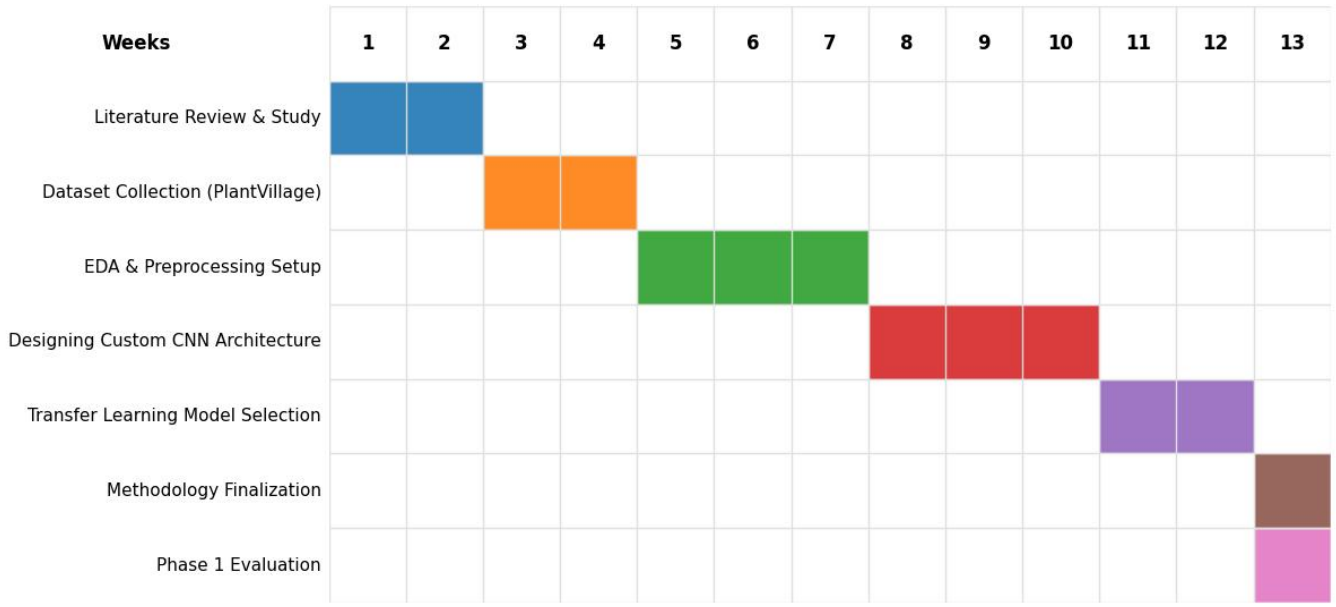

Figure 5.1: *Gantt Chart I – Timeline for Planning, Data Preparation, and Model Design (Weeks 1–13).*

5.7.2 Phase II: Implementation, Testing, and Documentation (Weeks 14–26)

The second phase was dedicated to the practical execution, rigorous testing, and documentation of the research findings. As depicted in **Figure 5.2**, this phase covered the remaining three months (Weeks 14 to 26).

- **Weeks 14-17:** Training the three models (Custom CNN, ResNet50, MobileNetV2) and performing hyperparameter tuning to optimize accuracy.
- **Weeks 18-21:** Conducting a comparative analysis of the models based on accuracy, loss, and inference speed. Additionally, **Explainable AI (Grad-CAM)** was implemented during this period to validate model transparency.
- **Weeks 22-26:** Finalizing the results, visualizing performance metrics (Confusion Matrix, ROC Curves), and compiling the complete thesis report. The phase concluded with the preparation for the final defense presentation.

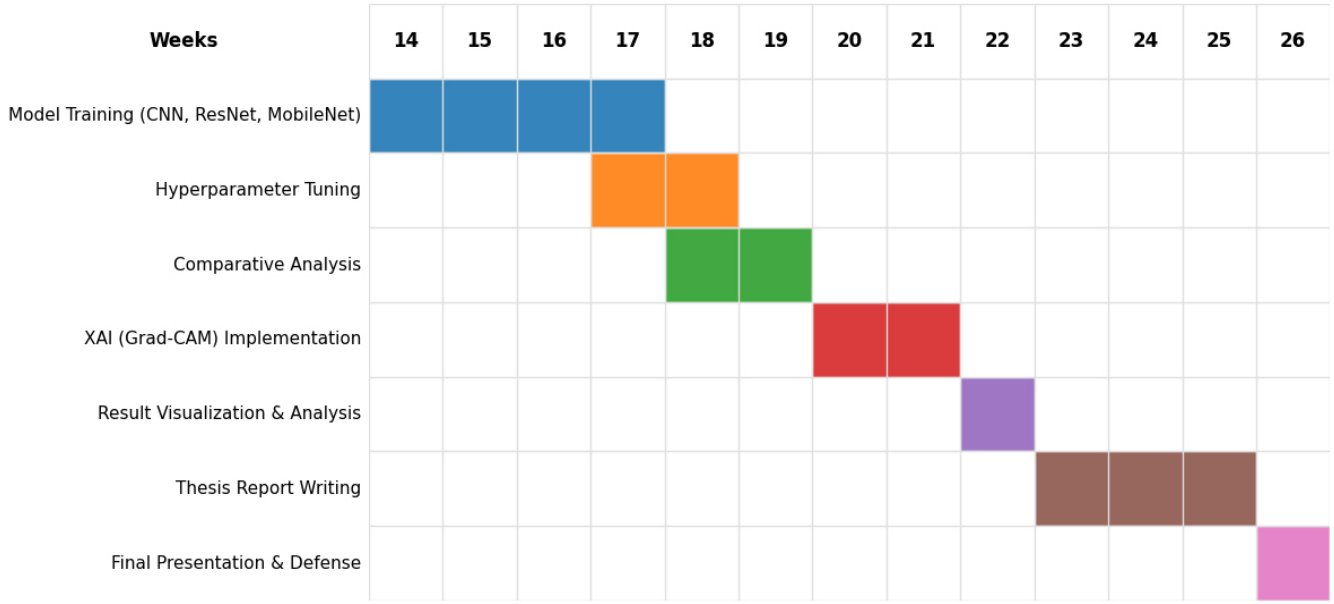

Figure 5.2: *Gantt Chart II – Timeline for Implementation, Testing, and Final Documentation (Weeks 14–26).*

5.8 Summary

This chapter discussed the non-technical but vital dimensions of the research. We established that the project adheres to strict coding and data standards, ensuring quality and legality. The societal impact analysis highlighted the system's potential to improve food security and environmental sustainability. Ethical considerations regarding bias and reliability were addressed to ensure responsible AI usage. Finally, the challenges of class imbalance and hardware limitations, along with the constraints of dataset scope and connectivity, were acknowledged to provide a realistic view of the system's current capabilities and boundaries.

CHAPTER 6
CONCLUSION

6.1 Conclusion

The primary objective of this research was to develop an efficient, accurate, and lightweight deep learning model for the automated detection of plant diseases. Agriculture plays a pivotal role in global food security and the economy, yet manual disease diagnosis remains slow, error-prone, and inaccessible to many farmers. This study addressed these challenges by implementing and comparing three distinct architectures: a proposed **Custom Lightweight CNN**, **ResNet50**, and **MobileNetV2**, utilizing the PlantVillage dataset comprising 54,305 images across 38 classes.

The experimental results definitively established the superiority of the proposed **Custom Lightweight CNN**. While **ResNet50** demonstrated high accuracy (**98.67%**), its massive computational weight (~24 million parameters) makes it unsuitable for resource-constrained environments. On the other hand, **MobileNetV2**, despite being designed for efficiency, lagged in accuracy (**96.19%**) and struggled with fine-grained disease patterns. In contrast, the **Custom CNN** achieved the highest test accuracy of **98.71%**, while maintaining an extremely compact size (**~0.65 million parameters**) and the fastest inference time (**~2.4 ms**).

Furthermore, the integration of **Explainable AI (Grad-CAM)** provided visual validation that the model relies on biologically relevant features—such as lesions and leaf texture—rather than background noise. This ensures the system is not only accurate but also trustworthy. In conclusion, this research successfully bridges the gap between high-performance deep learning and practical agricultural application, offering a deployable solution that empowers farmers with instant, expert-level disease diagnosis.

6.2 Limitations and Future Work

While the proposed system has demonstrated exceptional performance, there are certain limitations inherent to the current scope of the study. These limitations pave the way for future research and development directions.

6.2.1 Limitations

- **Dataset Constraints:** The model was trained exclusively on the **PlantVillage dataset**, which contains images captured in controlled laboratory environments with uniform backgrounds. Consequently, the model's performance may dip slightly when tested on images taken in "wild" field conditions with complex backgrounds, variable lighting, or overlapping leaves.
- **Crop Variety:** The current system is limited to the **14 specific crops** and **38 classes** present in the dataset. Major staple crops like Rice, Wheat, and Jute, which are crucial for the local economy, are not yet covered.
- **Disease Stage Detection:** The model classifies diseases based on visible symptoms but does not explicitly quantify the *severity* of the infection (e.g., Early vs. Late stage) for all classes, which is important for determining the dosage of pesticide.

6.2.2 Future Work

To evolve this research into a comprehensive agricultural product, the following future enhancements are proposed:

- **Real-World Data Integration:** We plan to collect and annotate a localized dataset featuring images captured directly from fields in Bangladesh. Training on this "in-the-wild" data will make the model robust against complex backgrounds and varying lighting conditions.
- **Expansion of Crop Classes:** Future iterations will incorporate datasets for **Rice (e.g., Blast, Tungro), Wheat, and Jute**, addressing the specific needs of local farmers.
- **Mobile Application with Offline Support:** Currently, the deployment is web-based. We aim to convert the model to **TensorFlow Lite (TFLite)** and develop a native Android mobile application. This will enable fully **offline inference**, allowing farmers in remote areas with no internet connectivity to use the system.
- **Treatment Recommendation System:** Merely detecting the disease is not enough. We plan to integrate a recommendation engine that suggests specific **organic and chemical treatments, fertilizers, and preventive measures** immediately after a disease is detected.
- **Multimodal Analysis:** To improve diagnostic precision, we aim to combine image data with other environmental factors such as **weather patterns, soil humidity, and temperature**, creating a holistic smart farming assistant.

References

[1] FAO, "The future of food and agriculture – Trends and challenges," *Food and Agriculture Organization of the United Nations*, Rome, 2017.

[2] R. N. Strange and P. R. Scott, "Plant disease: A threat to global food security," *Annual Review of Phytopathology*, vol. 43, pp. 83–116, 2005.

[3] S. Savary, L. Willocquet, S. J. Pethybridge, P. Esker, and N. McRoberts, "The global burden of pathogens and pests on major food crops," *Nature Ecology & Evolution*, vol. 3, pp. 430–439, 2019.

[4] J. G. Barbedo, "Factors influencing the use of deep learning for plant disease recognition," *Biosystems Engineering*, vol. 172, pp. 84–91, 2018.

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

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," *IEEE CVPR*, 2018.

[7] A. V. Todros and R. G. Silva, "Explainable AI in agriculture: A survey," *Computers in Industry*, 2022.

[8] A. Palmer, "MobileNets for agricultural efficiency: A comparative study," *Journal of Smart Farming*, 2019.

[9] H. Al-Hiary, S. Bani-Ahmad, M. Reyalat, M. Braik, and Z. AlRahamneh, "Fast and accurate detection and classification of plant diseases," *International Journal of Computer Applications*, vol. 17, no. 1, pp. 31–38, 2011.

[10] J. D. Pujari, R. Yakkundimath, and A. S. Byadgi, "Image processing based detection of fungal diseases in cereals," *International Journal of Signal Processing*, vol. 1, no. 1, 2016.

[11] S. Arivazhagan, R. N. Shebiah, S. Ananthi, and S. V. Varthini, "Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features," *Agric Eng Int: CIGR Journal*, vol. 15, no. 1, pp. 211–217, 2013.

[12] L. G. Ngugi, M. Abelwahab, and M. Abo-Zahhad, "Recent advances in image processing techniques for plant disease detection," *IEEE Access*, vol. 9, pp. 8456–8488, 2021.

[13] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016.

[14] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, 2018.

[15] J. Amara, B. Bouaziz, and A. Algergawy, "A deep learning-based approach for banana leaf diseases classification," *BTW (Workshops)*, pp. 79–88, 2017.

[16] D. P. Hughes and M. Salathé, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," *arXiv preprint arXiv:1511.08060*, 2015.

[17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *IEEE ICCV*, 2017.

[18] G. E. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, 2012.

[19] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a self-gated activation function," *arXiv preprint arXiv:1710.05941*, 2017.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ICML*, 2015.

[21] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep neural networks based recognition of plant diseases by leaf image classification," *Computational Intelligence and Neuroscience*, 2016.

[22] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *IEEE CVPR*, 2016.

[24] M. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep learning for tomato diseases: classification and symptoms visualization," *Applied Artificial Intelligence*, vol. 31, no. 4, 2017.

[25] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[26] E. C. Too, L. Yujun, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Computers and Electronics in Agriculture*, vol. 161, 2019.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2015.

[28] C. Szegedy et al., "Going deeper with convolutions," *IEEE CVPR*, 2015.

[29] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[32] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.