A Comparative Study of Supervised and Semi-Supervised Learning for Mango Leaf Disease Classification

by

Shahriar Hasan(2010776115)

Fahim Al Islam(2010976119)

Course Code: CSE4202

Bachelor of Science in Computer Science and Engineering

Under the supervision of

Professor Dr. A. K. M. Akhtar Hossain

Department of Computer Science and Engineering

University of Rajshahi, Rajshahi-6205, Bangladesh

CONTRIBUTION

Student ID	Contribution List
2010776115	1. Data Preprocessing (50%) 2. Model trained on (VGG16, MobileNetV2, ResNet101, DenseNet121 & Proposed Model) 3. Thesis Report Writing: • Introduction • Literature Review (50%) • Methodology (50%) • Result & Discussion • Conclusion, Limitation & Future Work
2010976119	1. Data Preprocessing (50%) 2. Model trained on (VGG16, MobileNetV2, ResNet101 & DenseNet121 for Semi-supervised Learning) 3. Thesis Report Writing: • Dataset • Literature Review (50%) • Methodology (50%) • Model Visualization and Interpretation • System Interface and Availability

Table 1: Overview of Individual Student Contributions

CERTIFICATE FROM SUPERVISOR

I am glad to certify that the thesis report titled **A Comparative Study of Supervised and Semi-Supervised Learning for Mango Leaf Disease Classification** submitted for the degree of Bachelor of Science in Computer Science and Engineering by **Shahriar Hasan (ID: 2010776115)** & **Fahim Al Islam (ID: 2010976119)**. This thesis work has been carried out by them under my guidance and supervision. This thesis work has not been submitted else where for any Degree.

Professor Dr. A. K. M. Akhtar Hossain
Department of Computer Science and Engineering
University of Rajshahi, Rajshahi-6205, Bangladesh
Date: October 28, 2025

ACKNOWLEDGEMENTS

We would like to express our sincere thanks to everyone who supported us throughout our deep learning thesis. This work would not have been possible without their help and encouragement. First, we are very grateful to **Professor Dr. A. K. M. Akhtar Hossain**, our thesis supervisor, for his continuous guidance, helpful advice, and strong support during this research. His expertise and patience motivated us to complete this work successfully.
We would also like to thank **Dr. Sangeeta Biswas, Associate Professor**, for her valuable suggestions and for giving her time whenever we needed support. We are thankful to everyone who shared ideas and helped us to improve our research. Their contributions made this work stronger.
Finally, we thank the Almighty Allah for giving us the strength, knowledge, and determination to complete this thesis.

ID: 2010776115

Department of Computer Science
and Engineering
University of Rajshahi,
Rajshahi-6205, Bangladesh

ID: 2010976119

Department of Computer Science
and Engineering
University of Rajshahi,
Rajshahi-6205, Bangladesh

Date: October 28, 2025

DECLARATION

Thesis Title	A Comparative Study of Supervised and Semi-Supervised Learning for Mango Leaf Disease Classification.
Student IDs	2010776115 & 2010976119
Supervisor	Professor Dr. A. K. M. Akhtar Hossain

We declare that this thesis, titled *"A Comparative Study of Supervised and Semi-Supervised Learning for Mango Leaf Disease Classification"*, is our own work, except where we have cited other sources. This thesis has not been submitted for any other degree, either in part or in full. While similar research has been carried out by other scholars, our study is unique because we designed a custom deep learning model. This model was trained and tested on publicly available mango leaf disease datasets, providing new insights and results in the field.

ID: 2010776115

Department of Computer Science
and Engineering
University of Rajshahi,
Rajshahi-6205, Bangladesh

ID: 2010976119

Department of Computer Science
and Engineering
University of Rajshahi,
Rajshahi-6205, Bangladesh

Date: October 28, 2025

Abstract

Mango leaf diseases are a major threat to crop production, and detecting them early is important to prevent large losses. Manual diagnosis is slow, subjective, and often not available in rural areas. This study explores a deep learning-based system that uses both supervised and semi-supervised learning to classify eight types of mango leaf diseases. Four pretrained models (VGG16, MobileNetV2, DenseNet121, and ResNet101) and a proposed lightweight model were tested on publicly available datasets. In supervised experiments, ResNet101 achieved the highest 99.75% test accuracy, followed closely by the proposed model with 99.00%. Semi-supervised learning improved lightweight models like MobileNetV2, increasing accuracy from 91.00% to 97.50%, showing the benefit of using unlabeled data when labelled data are limited. Deeper models such as ResNet101 showed lower performance in semi-supervised learning due to errors in pseudo-labels. A mobile application called *PlantDoc Advisor* was built using the proposed model to detect diseases in real time, making it useful for farmers in the field. Overall, the proposed model gave the best balance of accuracy, stability, and efficiency, making it suitable for real agricultural use. These results show that deep learning can help provide fast, reliable, and low-cost plant disease detection to support precision agriculture.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mango (*Mangifera indica L.*) is a vital fruit crop in tropical and subtropical regions, particularly in South Asian countries such as Bangladesh and India, where it contributes significantly to agricultural income and rural livelihoods. However, mango cultivation is frequently threatened by various foliar diseases, including Anthracnose, Powdery Mildew, Bacterial Canker, Die Back, and Gall Midge. These diseases can cause substantial yield losses if not detected and managed promptly. Therefore, accurate and early disease detection is critical for sustainable mango production.

Recent advancements in deep learning, especially Convolutional Neural Networks (CNNs), have enabled the development of automated plant disease classification systems using image-based data. Supervised learning approaches, in which models are trained on large sets of labeled images, have demonstrated high accuracy in identifying plant diseases. However, these models require extensive labeled datasets, and generating such data often demands expert knowledge, making the process labor-intensive and costly.

Semi-supervised learning (SSL) techniques provide a potential solution by leveraging both labeled and unlabeled data. SSL has gained attention for reducing the dependence on labeled data while maintaining competitive performance. Despite its success in general image classification tasks, SSL has been relatively underexplored in the context of agricultural disease classification, particularly for mango leaf disease detection.

This thesis presents a comparative study of supervised and semi-supervised learning methods for mango leaf disease classification. Established pre-trained CNN models, including VGG16, ResNet101, MobileNetV2, and DenseNet121, are trained under both supervised and semi-supervised settings to serve as benchmarks. ResNet101, being a very deep architecture, is evaluated to understand how depth and complexity affect performance under semi-supervised learning, where noisy pseudo-labels may impact learning stability. This comparison helps identify the strengths and limitations of each learning approach when applied to real-world agricultural data.

In addition to pre-trained models, custom CNN architectures are designed and evaluated specifically for the mango leaf disease dataset. The aim is to assess whether lightweight or task-specific models can achieve comparable or superior performance to complex pre-trained architectures, particularly when labeled data is limited or semi-supervised learning is applied.

To demonstrate the practical applicability of the proposed model, a mobile application named *PlantDoc Advisor* was developed. This app utilizes the proposed CNN model to perform real-time disease detection on mango leaves, enabling farmers and agricultural practitioners to quickly identify diseases in the field. By integrating the model into a mo-

bile platform, *PlantDoc Advisor* provides a convenient and scalable tool for early disease diagnosis, helping to reduce yield losses and support timely intervention.
Through these experiments and the development of *PlantDoc Advisor*, this study seeks to identify effective approaches that minimize the need for labeled data without compromising classification accuracy, while also offering a practical solution for real-world agricultural applications.

Chapter 2

Literature Review

Rizvee et al.[1] developed LeafNet, a CNN model tailored for detecting seven mango leaf diseases using region-specific images from Bangladesh. The model achieved high performance with 98.55% accuracy and outperformed AlexNet and VGG16. Its lightweight structure and robustness make it suitable for early disease detection in real-world agricultural settings.
Rao et al.[2] implemented a deep learning approach using a modified AlexNet for detecting mango and grape leaf diseases. With a dataset of over 8,000 images, their system achieved 89% accuracy for mango and 99% for grape leaves. Integrated into an Android app (JIT CROPFIX), the model supports real-time disease detection on smartphones.
Zhang et al.[3] proposed improved GoogLeNet and CIFAR10 models for automatic recognition of maize leaf diseases. By optimizing network parameters and architecture, their system achieved up to 98.9% accuracy while reducing training iterations and computational cost, proving effective for real-time agricultural diagnostics.
Bhujel et al.[4] introduced a lightweight CNN enhanced with various attention modules (e.g., CBAM) for tomato leaf disease classification. Their CBAM-based model achieved 99.69% accuracy with drastically fewer parameters than ResNet50, enabling efficient deployment on low-resource devices without compromising detection accuracy.
Li and Chao et al.[5] presented a semi-supervised few-shot learning framework that leverages both labeled and high-confidence pseudo-labeled data for plant disease recognition. Their method, tested on PlantVillage, achieved up to 4.6% performance improvement, showing strong generalisation even with minimal labeled samples.
Amorim et al.[6] addressed the challenge of limited labeled data in agricultural image classification by proposing a semi-supervised learning approach. Their method leverages a small set of labeled samples alongside a large pool of unlabeled images to enhance the training of Convolutional Neural Networks (CNNs). By employing label propagation techniques, the model automatically assigns labels to the unlabeled dataset and improves classification accuracy. The approach was validated on UAV-captured images of soybean leaves and herbivorous pests, demonstrating its effectiveness in supporting pest identification and control in real-world agricultural settings, especially when labeled data is scarce.
Arivazhagan and Ligi et al.[7] proposed a CNN-based deep learning approach for automating the identification of mango leaf diseases. Their study focused on five common diseases—Anthracnose, Alternaria leaf spots, Leaf Gall, Leaf Webber, and Leaf Burn—using a dataset of 1200 images of both healthy and diseased leaves. The model achieved an impressive classification accuracy of 96.67%, demonstrating its effectiveness for real-time

disease diagnosis in agricultural settings and its potential to assist in early-stage disease control and monitoring across large farms.

Rajbongshi et al.[8] implemented a transfer learning-based classification framework using several advanced CNN models, including DenseNet201, ResNet50, InceptionV3, InceptionResNetV2, and Xception, to identify mango leaf diseases. Their methodology involved image acquisition, segmentation, and feature extraction to classify various disease classes such as Anthracnose, Gall Machi, Powdery Mildew, and Red Rust. Using a dataset of 1500 mango leaf images, the study concluded that DenseNet201 yielded the best performance, achieving an accuracy of 98.00%, thereby confirming the reliability of deep learning in precise and scalable disease classification.

Ahmed et al.[9] addressed a critical gap in the availability of agricultural datasets by introducing MangoLeafBD, the first publicly available, standardized dataset specifically curated for mango leaf disease classification. The dataset comprises 4000 images of approximately 1800 unique leaves, collected from four mango orchards in Bangladesh. Covering seven major disease classes, this dataset provides a diverse and rich foundation for developing and benchmarking machine learning models in agricultural disease detection. The authors emphasized that although the dataset was sourced in Bangladesh, the diseases represented are prevalent in many mango-growing regions, making the dataset broadly applicable for global research.

Mia et al.[10] proposed a Neural Network Ensemble (NNE) for Mango Leaf Disease Recognition (MLDR) to address the challenges of detecting mango leaf diseases with the naked eye. The system uses machine learning to monitor different leaf symptoms and automatically classify diseases by comparing new leaf images with trained data. The model achieved an average accuracy of 80% and provides an automated alternative to traditional manual inspection. Its implementation allows timely disease detection without the presence of an agriculturist, helping to properly treat affected leaves, increase mango production, and support the global market demand.

Chapter 3

Dataset

The success of any deep learning-based classification system heavily depends on the quality, diversity, and balance of the dataset used. In this research, we have developed a comprehensive mango leaf image dataset by merging two well-known public sources. The dataset represents a wide range of visual symptoms across eight different classes—seven disease categories and one healthy category. The collected data includes variations in lighting, background, camera angles, and disease stages, which helps to generalise the models to real-world conditions.

3.0.1 Data Collection

To develop a robust and generalizable model for mango leaf disease classification, we have utilized a high-quality datasets obtained from the Mendeley Data repository. This datasets collectively offer a diverse and balanced collection of mango leaf images representing both healthy and diseased categories. The images were captured under natural orchard conditions from multiple regions in Bangladesh, ensuring variability in lighting, angles, and background. This variation helps improve the model's ability to generalize to unseen real-world scenarios.
The dataset comprises **4,000 mango leaf images** captured in natural orchard conditions across four different locations in Bangladesh[11]:

- Sher-e-Bangla Agricultural University orchard
- Jahangirnagar University orchard
- Udaypur village mango orchard
- Itakhola village mango orchard

Specifications:

- **Image Size:** 240 × 320 pixels
- **Format:** JPG
- **Classes:** 8 (7 diseased, 1 healthy)
- **Images per Class:** 500
- **Data Augmentation:** Around 2,200 images via rotation, zooming, and flipping
- **Acquisition Device:** Mobile phone cameras

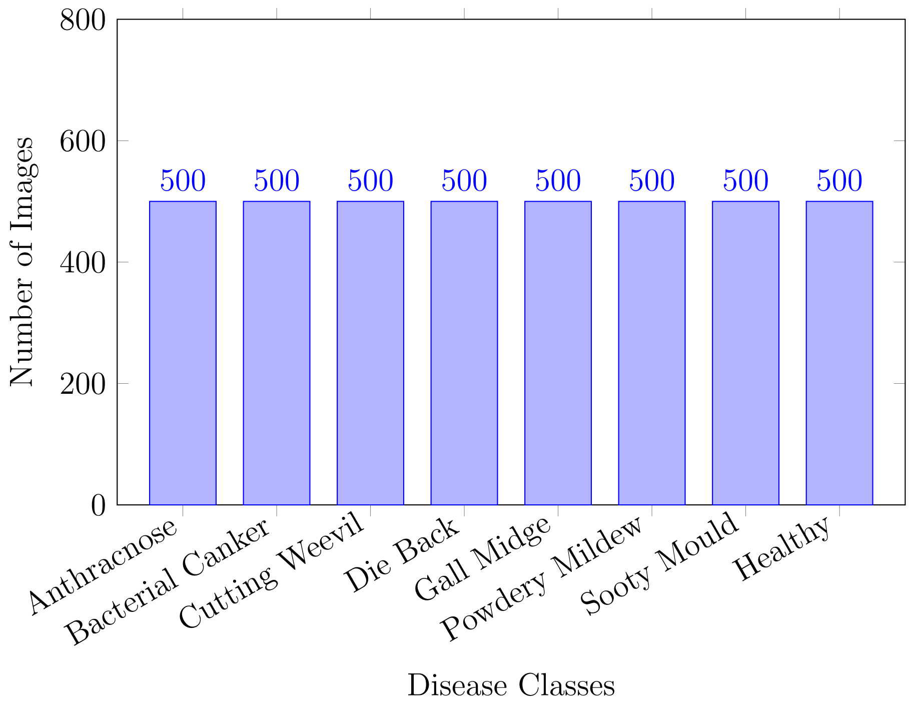
3.0.2 Data Statistics and Distribution

After merging the two datasets, we ensured an equal number of samples across each class. Each of the eight classes now contains **1,300 images**, resulting in a final dataset of **10,400 images**.

Table 3.1: Number of Images per Class

Class	**Dataset**
Anthracnose	500
Bacterial Canker	500
Cutting Weevil	500
Die Back	500
Gall Midge	500
Powdery Mildew	500
Sooty Mould	500
Healthy	500
Total	**4000**

Figure 3.1: Image Count per Class



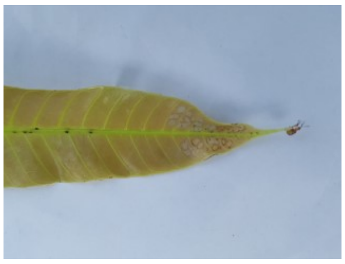
3.0.3 Disease Classes and Their Visual Features

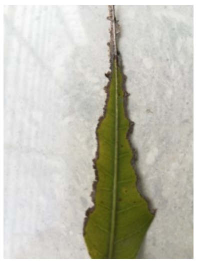
Each class in the dataset represents a specific mango leaf condition. Understanding the visual characteristics of each disease is important for both human inspection and model interpretability.

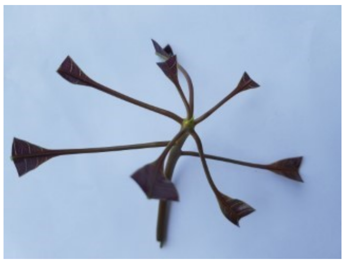
Table 3.2: Visual Characteristics of Each Class

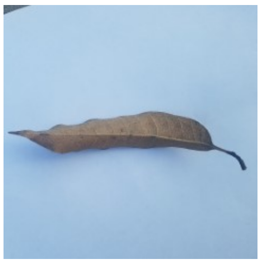
Disease Class	**Key Visual Characteristics**
Anthracnose	Small, dark brown to black lesions with irregular shapes. Commonly starts at the leaf tip and may coalesce to form large necrotic areas. Lesions often have a concentric ring pattern and may lead to leaf curling or drying.
Bacterial Canker	Water-soaked lesions that enlarge and turn dark brown or black. Lesions are often angular and surrounded by yellow halos. Severe cases show cracking along veins and deformation of the leaf surface.
Cutting Weevil	Leaves show notched, chewed, or semi-circular cuts along the margins, caused by adult weevils feeding. Damage appears as clean cuts and may affect multiple edges of a single leaf.
Die Back	Yellowing and browning begins at the tip and progresses downward toward the base. May cause full leaf drying and extend into twigs. Often associated with poor plant vigor or fungal infection.
Gall Midge	Distinct swollen blisters, wart-like growths, or raised galls appear on leaf surfaces, particularly near veins. Caused by insect larvae feeding inside the tissue.
Powdery Mildew	A whitish or gray powder-like fungal coating appears on both surfaces, but mainly on the upper side. Advanced stages cover large areas and may distort leaves.
Sooty Mould	A black, soot-like fungal layer covers the leaf surface. Although the mold itself does not infect the plant, it blocks sunlight and photosynthesis. Often linked to honeydew secretions from insects.
Healthy	Uniform bright green color, smooth texture, no visible signs of necrosis, deformation, or discoloration. Veins and edges are clearly defined and free of lesions or foreign growth.

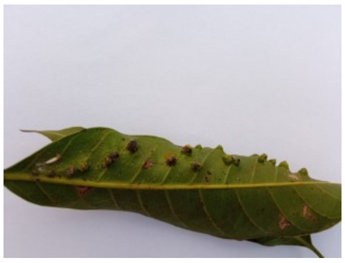
These images and visual descriptions help in understanding the symptom patterns for each disease and serve as a visual reference for model training and evaluation.

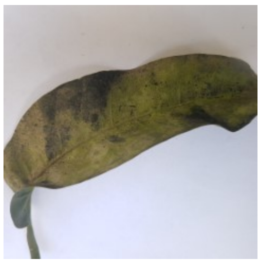

(a) Anthracnose

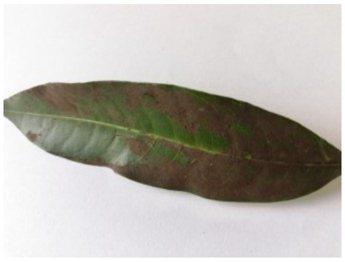

(b) Bacterial Canker


(c) Cutting Weevil


(d) Die Back


(e) Gall Midge


(f) Powdery Mildew


(g) Sooty Mould

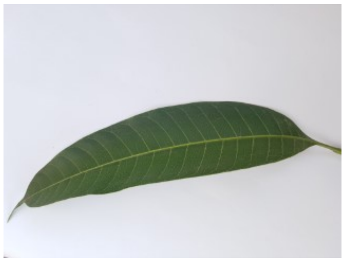

(h) Healthy

Figure 3.2: Representative Images of Each Class

Chapter 4

Methodology

4.1 Dataset Preparation

This study used a publicly available Mango Leaf Disease Dataset from Kaggle, which contains 4,000 images of healthy and diseased mango leaves captured under natural lighting conditions with a uniform background. The dataset includes eight categories: Anthracnose, Bacterial Canker, Cutting Weevil, Die Back, Gall Midge, Powdery Mildew, Sooty Mould and Healthy.
All images were visually inspected for quality and resized to a uniform resolution of 224×224 pixels to ensure compatibility across all convolutional neural network (CNN) models. The dataset was then divided into training, validation, and testing subsets using stratified sampling to maintain balanced class representation.

4.1.1 Supervised Dataset Preparation

In the supervised learning setup, all samples were treated as labeled. The dataset was divided into **80% for training**, **10% for validation**, and **10% for testing**, resulting in 3,200, 400, and 400 images, respectively.

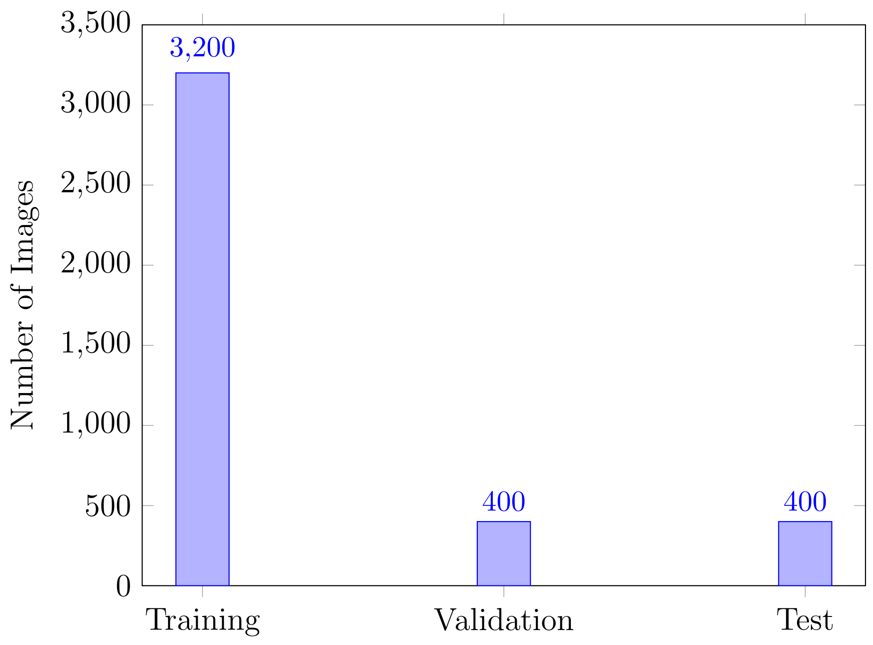


Figure 4.1: Supervised dataset split across training, validation, and test sets.

All images were normalized to pixel values between 0 and 1. The validation set was used for model tuning and early stopping, while the test set was used exclusively for the final evaluation. This preprocessing ensured the model was trained on a diverse yet balanced dataset, minimizing overfitting and improving robustness.

4.1.2 Semi-Supervised Dataset Preparation

For the semi-supervised setup, the same dataset was used, but with a reduced number of labeled samples to simulate limited annotation scenarios. From the total dataset, 80% was used for training, 10% for validation, and 10% for testing. Within the training subset, only **25% (800 images)** were labeled, while the remaining **75% (2,400 images)** were treated as unlabeled. Class balance was maintained within the labeled portion.

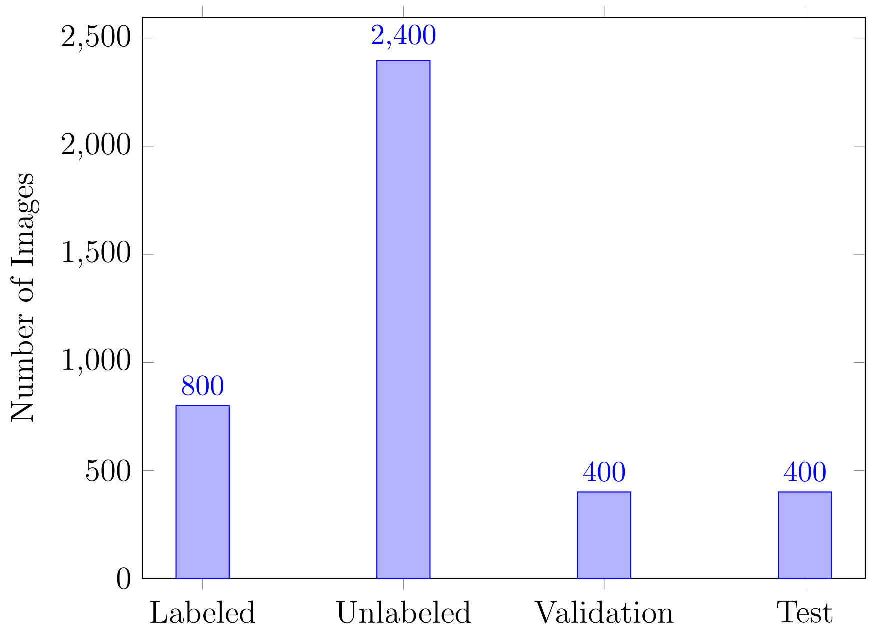


Figure 4.2: Semi-supervised dataset split showing labeled, unlabeled, validation, and test samples.

The validation and test sets were kept completely separate and were not used during any training phase.

4.2 VGG16 Model Architecture with Adapted Dense Layers

4.2.1 Initial Model Configuration

The VGG16 model is initialized with ImageNet pre-trained weights while excluding the top classification layers. The base model receives input images of size $224 \times 224 \times 3$. To adapt VGG16 for our mango leaf disease classification task, we added custom dense layers on top of the frozen convolutional base:

- Global Average Pooling 2D layer to reduce spatial dimensions.
- Dense layer with 256 neurons, ReLU activation, and L2 regularization (0.016).
- Batch Normalization layer for training stability.
- Dropout layer (0.5) to prevent overfitting.
- Dense layer with 128 neurons, ReLU activation, and L2 regularization (0.016).
- Dropout layer (0.3) for further regularization.
- Output Dense layer with softmax activation for multi-class classification.

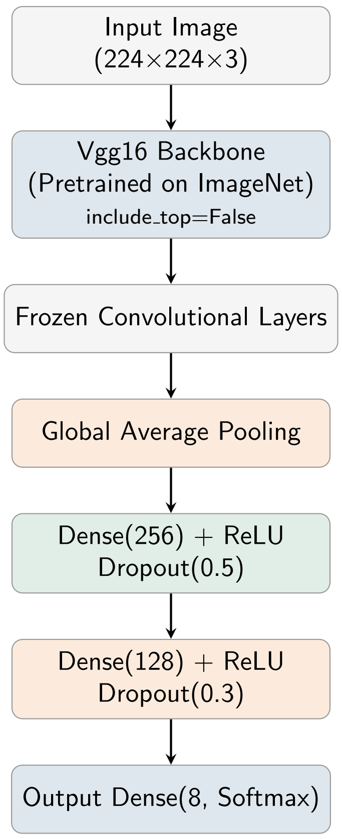


Figure 4.3: Transfer learning model using Vgg16 as the feature extractor.

4.2.2 Supervised Training Strategy

The supervised training was performed using a pre-trained VGG16 backbone with custom dense layers for classification. Let the input images be denoted as $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, where H and W are the height and width, and C is the number of channels. The target labels are one-hot encoded vectors $\mathbf{Y} \in \{0,1\}^{n_{\text{classes}}}$. The model outputs predictions $\hat{\mathbf{Y}}$ through the softmax function [12]:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{n_{\text{classes}}} e^{z_j}}, \quad i = 1, 2, \ldots, n_{\text{classes}},$$

where z_i are the logits from the final dense layer. The training minimizes the **categorical cross-entropy loss** [13]:

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = -\sum_{i=1}^{n_{\text{classes}}} y_i \log(\hat{y}_i).$$

Initially, the VGG16 base was **frozen**, and only the added dense layers were trained using the **Adamax optimizer** with learning rate $\eta = 10^{-3}$. Early stopping with a patience of 10 epochs and model checkpointing were employed to prevent overfitting. To improve generalization, **dropout** layers and L2 regularization ($\lambda \|\mathbf{W}\|_2^2$) were applied in the dense layers, where $\mathbf{W}$ denotes the layer weights. The training process was monitored using the accuracy metric:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

Finally, the model was evaluated on the validation and test sets, and training progress was visualized through plots of loss $\mathcal{L}$ and accuracy across epochs.

4.2.3 Semi-supervised Fine-Tuning the Model

The semi-supervised approach enhances the model by leveraging unlabeled data:

- **Phase 1:** Initial training on only 25% of the labeled training data, following the same approach as supervised learning.
- **Pseudo-labeling:** The model trained on limited labeled data is used to generate predictions on the remaining unlabeled data. Samples with prediction confidence above 95% are added to the training set with their predicted labels.
- **Phase 2:** The model is fine-tuned on the combined dataset (original labeled data + high-confidence pseudo-labeled data). The top 4 layers of VGG16 are unfrozen, and the model is trained with a very low learning rate (10^{-5}) for up to 100 epochs with early stopping and learning rate reduction [14].

4.3 MobileNetV2 Model Architecture with Adapted Dense Layers

Initial Model Configuration

MobileNetV2 is initialized with ImageNet pre-trained weights without its top classification layers. The model accepts $224 \times 224 \times 3$ input images. For our specific task, we added custom dense layers:

- Global Average Pooling 2D layer.
- Dense layer with 256 neurons, ReLU activation, and L2 regularization (0.016).
- Batch Normalization layer.
- Dropout layer (0.5).
- Dense layer with 128 neurons, ReLU activation, and L2 regularization (0.016).
- Dropout layer (0.3).
- Output Dense layer with softmax activation.

4.3.1 Supervised Training Strategy

The supervised training was performed using a pre-trained MobileNetV2 backbone with custom dense layers for classification. Let the input images be denoted as $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, where H and W are the height and width, and C is the number of channels. The target labels are one-hot encoded vectors $\mathbf{Y} \in \{0, 1\}^{n_{\text{classes}}}$. The model outputs predictions $\hat{\mathbf{Y}}$ through the softmax function:

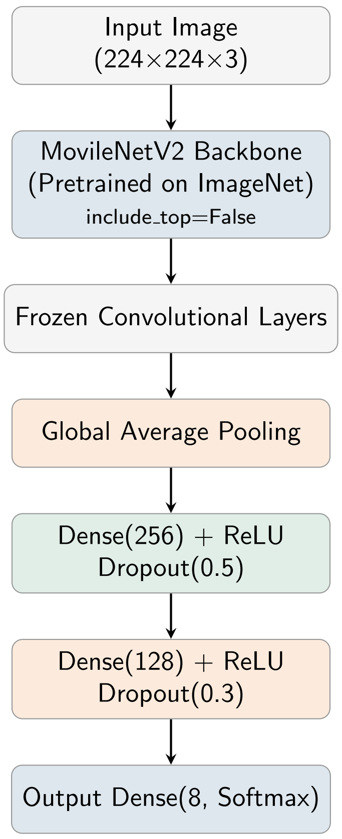


Figure 4.4: Transfer learning model using MobileNetV2 as the feature extractor.

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{n_{\text{classes}}} e^{z_j}}, \quad i = 1, 2, \ldots, n_{\text{classes}},$$

where z_i are the logits from the final dense layer. The training minimizes the **categorical cross-entropy loss**:

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = -\sum_{i=1}^{n_{\text{classes}}} y_i \log(\hat{y}_i).$$

Initially, the MobileNetV2 base was **frozen**, and only the added dense layers were trained using the **Adamax optimizer** with learning rate $\eta = 10^{-3}$. Early stopping with a patience of 10 epochs and model checkpointing were employed to prevent overfitting. To improve generalization, **dropout** layers and L2 regularization ($\lambda\|\mathbf{W}\|_2^2$) were applied in the dense layers, where $\mathbf{W}$ denotes the layer weights. The training process was monitored using the accuracy metric:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

Finally, the model was evaluated on the validation and test sets, and training progress was visualized through plots of loss $\mathcal{L}$ and accuracy across epochs.

4.3.2 Semi-supervised Fine-Tuning the Model

The semi-supervised approach for MobileNetV2 follows the same pattern as VGG16:

- **Phase 1:** Initial training on only 25% of the labeled data with the two-stage approach.
- **Pseudo-labeling:** The model generates predictions on unlabeled data, with high-confidence samples ($\geq$95%) added to the training set.
- **Phase 2:** Fine-tuning on the combined dataset with the top 20 layers of MobileNetV2 unfrozen, using a very low learning rate (10^{-5}) for up to 100 epochs with early stopping and learning rate reduction [14].

4.4 ResNet101 Model Architecture with Adapted Dense Layers

4.4.1 Initial Model Configuration

ResNet101 is initialized with ImageNet pre-trained weights, excluding the top classification layers. The model processes $224 \times 224 \times 3$ input images. The custom dense layers added for our task include:

- Global Average Pooling 2D layer.
- Dense layer with 256 neurons, ReLU activation, and L2 regularization (0.016).
- Batch Normalization layer.
- Dropout layer (0.5).
- Dense layer with 128 neurons, ReLU activation, and L2 regularization (0.016).
- Dropout layer (0.3).
- Output Dense layer with softmax activation.

4.4.2 Supervised Training Strategy

The supervised training was performed using a pre-trained ResNet101 backbone with custom dense layers for classification. Let the input images be denoted as $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, where H and W are the height and width, and C is the number of channels. The target labels are one-hot encoded vectors $\mathbf{Y} \in \{0, 1\}^{n_{\text{classes}}}$. The model outputs predictions $\hat{\mathbf{Y}}$ through the softmax function:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{n_{\text{classes}}} e^{z_j}}, \quad i = 1, 2, \ldots, n_{\text{classes}},$$

where z_i are the logits from the final dense layer. The training minimizes the **categorical cross-entropy loss**:

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = -\sum_{i=1}^{n_{\text{classes}}} y_i \log(\hat{y}_i).$$

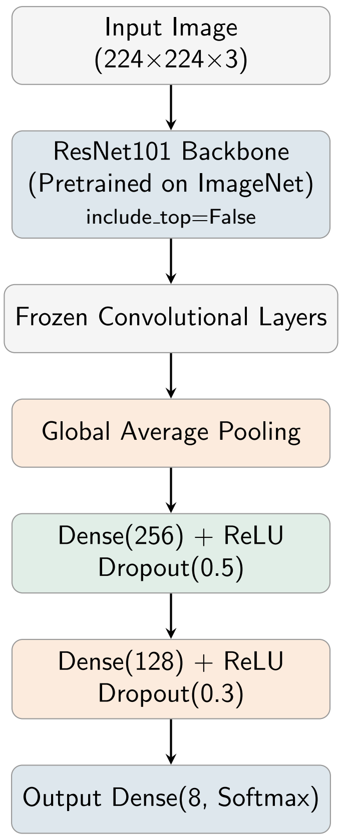


Figure 4.5: Transfer learning model using ResNet101 as the feature extractor.

Initially, the ResNet101 base was **frozen**, and only the added dense layers were trained using the **Adamax optimizer** with learning rate $\eta = 10^{-3}$. Early stopping with a patience of 10 epochs and model checkpointing were employed to prevent overfitting. To improve generalization, **dropout** layers and L2 regularization ($\lambda \|\mathbf{W}\|_2^2$) were applied in the dense layers, where $\mathbf{W}$ denotes the layer weights. The training process was monitored using the accuracy metric:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

Finally, the model was evaluated on the validation and test sets, and training progress was visualized through plots of loss $\mathcal{L}$ and accuracy across epochs.

4.4.3 Semi-supervised Fine-Tuning the Model

The semi-supervised approach for ResNet101 follows the same pattern:

- **Phase 1:** Initial training on only 25% of the labeled data with the two-stage approach.
- **Pseudo-labeling:** The model generates predictions on unlabeled data, with high-confidence samples ($\geq$95%) added to the training set.
- **Phase 2:** Fine-tuning on the combined dataset with the top 30 layers of ResNet101 unfrozen, using a very low learning rate (10^{-5}) for up to 100 epochs with early stopping and learning rate reduction [14].

4.5 DenseNet121 Model Architecture with Adapted Dense Layers

4.5.1 Initial Model Configuration

DenseNet121 is initialized with ImageNet pre-trained weights, excluding the top classification layers. The model accepts $224 \times 224 \times 3$ input images. The custom dense layers added for our task include:

- Global Average Pooling 2D layer.
- Dense layer with 256 neurons, ReLU activation, and L2 regularization (0.016).
- Batch Normalization layer.
- Dropout layer (0.5).
- Dense layer with 128 neurons, ReLU activation, and L2 regularization (0.016).
- Dropout layer (0.3).
- Output Dense layer with softmax activation.

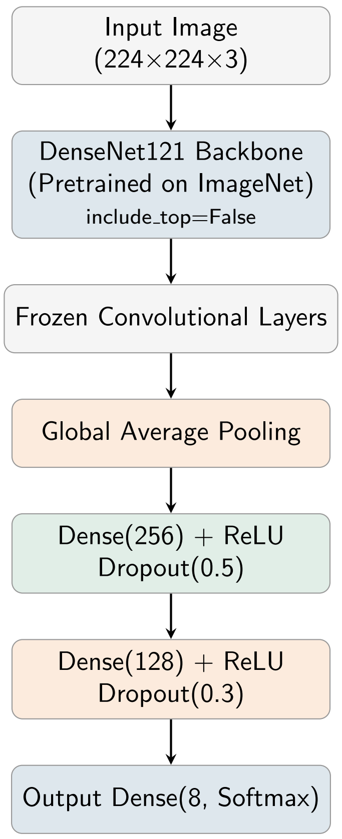


Figure 4.6: Transfer learning model using DenseNet121 as the feature extractor.

4.5.2 Supervised Training Strategy

The supervised training was performed using a pre-trained DenseNet121 backbone with custom dense layers for classification. Let the input images be denoted as $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, where H and W are the height and width, and C is the number of channels. The target labels are one-hot encoded vectors $\mathbf{Y} \in \{0,1\}^{n_{\text{classes}}}$. The model outputs predictions $\hat{\mathbf{Y}}$ through the softmax function [12]:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{n_{\text{classes}}} e^{z_j}}, \quad i = 1, 2, \ldots, n_{\text{classes}},$$

where z_i are the logits from the final dense layer. The training minimizes the **categorical cross-entropy loss**:

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = -\sum_{i=1}^{n_{\text{classes}}} y_i \log(\hat{y}_i).$$

Initially, the DenseNet121 base was **frozen**, and only the added dense layers were trained using the **Adamax optimizer** with learning rate $\eta = 10^{-3}$. Early stopping with a patience of 10 epochs and model checkpointing were employed to prevent overfitting. To improve generalization, **dropout** layers and L2 regularization ($\lambda \|\mathbf{W}\|_2^2$) were applied in the dense layers, where $\mathbf{W}$ denotes the layer weights. The training process was monitored using the accuracy metric:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

Finally, the model was evaluated on the validation and test sets, and training progress was visualized through plots of loss $\mathcal{L}$ and accuracy across epochs.

4.5.3 Semi-supervised Fine-Tuning the Model

The semi-supervised approach for DenseNet121 follows the same pattern:

- **Phase 1:** Initial training on only 25% of the labeled data with the two-stage approach.
- **Pseudo-labeling:** The model generates predictions on unlabeled data, with high-confidence samples ($\geq$95%) added to the training set.
- **Phase 2:** Fine-tuning on the combined dataset with the top 15 layers of DenseNet121 unfrozen, using a very low learning rate (10^{-5}) for up to 100 epochs with early stopping and learning rate reduction [14].

4.6 Proposed Model Architecture for Supervised and Semi-supervised Classification

4.6.1 Overall Architecture Design

The proposed model is a custom Convolutional Neural Network (CNN) comprising four convolutional blocks followed by a classification head. Unlike the pre-trained models

(VGG16, ResNet101, MobileNetV2, and DenseNet121) that were originally designed for ImageNet classification, this architecture is specifically tailored for the characteristics of mango leaf disease images, providing a balance between model complexity and computational efficiency while preventing overfitting on limited labeled data.
The architecture accepts input images of size 224×224×3 pixels and outputs probability distributions across eight disease classes. The model contains approximately 3.5 million trainable parameters, making it significantly more lightweight than the baseline models while maintaining competitive performance.

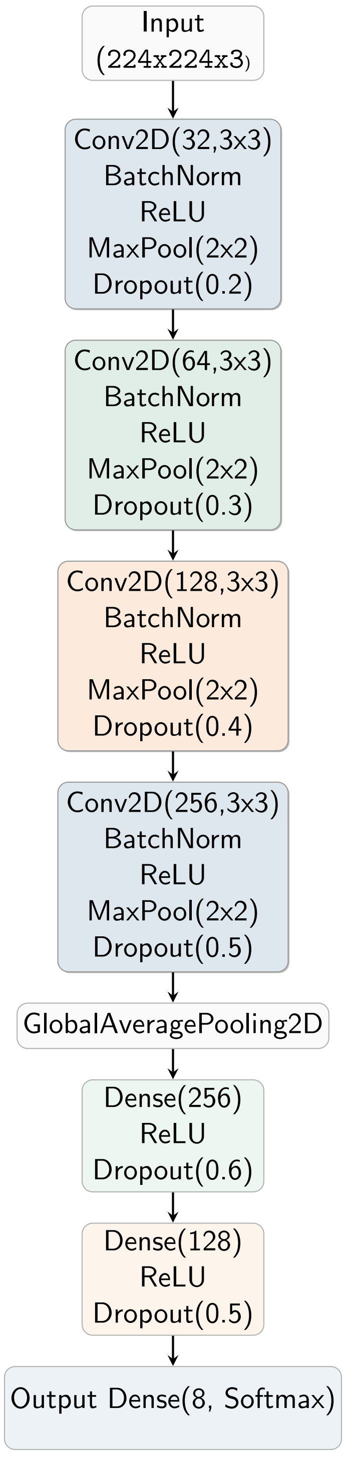


Figure 4.7: Proposed CNN architecture for mango leaf disease classification.

4.6.2 Convolutional Block Design

The proposed architecture consists of four hierarchical convolutional blocks, each following a consistent design pattern. The mathematical formulation of the core convolution operation is:

Convolutional Operation:

The convolution operation at layer l is defined as:

$$z^{[l]} = W^{[l]} * a^{[l-1]} + b^{[l]} \tag{4.1}$$

where $W^{[l]}$ represents the learnable filters, $*$ denotes the convolution operation, $a^{[l-1]}$ is the input activation from the previous layer, and $b^{[l]}$ is the bias term.

Block 1 (Input Processing):

- Conv2D layer: 32 filters, 3×3 kernel, same padding, L2 regularization (1e-4)
- Batch Normalization layer
- ReLU activation
- Max Pooling: 2×2 pool size
- Dropout: 20% rate

Block 2 (Feature Extraction):

- Conv2D layer: 64 filters, 3×3 kernel, same padding, L2 regularization (1e-4)
- Batch Normalization layer
- ReLU activation
- Max Pooling: 2×2 pool size
- Dropout: 30% rate

Block 3 (Mid-level Features):

- Conv2D layer: 128 filters, 3×3 kernel, same padding, L2 regularization (1e-4)
- Batch Normalization layer
- ReLU activation
- Max Pooling: 2×2 pool size
- Dropout: 40% rate

Block 4 (High-level Features):

- Conv2D layer: 256 filters, 3×3 kernel, same padding, L2 regularization (1e-4)
- Batch Normalization layer
- ReLU activation
- Max Pooling: 2×2 pool size
- Dropout: 50% rate

Each block progressively increases both the number of filters (32→64→128→256) and the dropout rate (0.2→0.3→0.4→0.5), enabling the network to learn increasingly complex features while maintaining strong regularization to prevent overfitting.

4.6.3 Activation and Pooling Operations

ReLU Activation Function

The Rectified Linear Unit (ReLU) activation function is applied after batch normalization in each convolutional block [15]:

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{4.2}$$

ReLU introduces non-linearity while maintaining computational efficiency and mitigating the vanishing gradient problem.

Max Pooling Operation

Max pooling with 2×2 window reduces spatial dimensions by selecting the maximum value in each pooling region:

$$y_{i,j} = \max_{(p,q)\in\mathcal{R}_{i,j}} x_{p,q} \tag{4.3}$$

where $\mathcal{R}_{i,j}$ represents the 2×2 pooling region corresponding to output position (i, j).

Dropout Regularization

During training, dropout randomly sets a fraction p of neurons to zero:

$$\tilde{a}^{[l]} = a^{[l]} \odot m \tag{4.4}$$

where $m \sim \text{Bernoulli}(1 - p)$ is a binary mask, and $\odot$ denotes element-wise multiplication. During inference, activations are scaled by $(1 - p)$ to maintain expected values.

4.6.4 Classification Head

Following the convolutional blocks, the classification head processes the extracted features through global average pooling and fully connected layers.

Global Average Pooling

For feature maps of spatial dimensions $H \times W$ with C channels, global average pooling computes:

$$\text{GAP}_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_{i,j,c} \tag{4.5}$$

This reduces each feature map to a single value, resulting in a C-dimensional vector (256 dimensions from the final convolutional block).

Fully Connected Layers:

- Dense Layer 1: 256 units with ReLU activation and L2 regularization (1e-4)
- Dropout: 60% rate
- Dense Layer 2: 128 units with ReLU activation and L2 regularization (1e-4)
- Dropout: 50% rate
- Output Layer: 8 units with softmax activation

Softmax Activation Function

The output layer uses softmax activation to produce probability distributions over the 8 disease classes:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{4.6}$$

where $K = 8$ is the number of disease classes, z_i is the logit for class i, and the output represents $P(y = i|x)$, the probability of input x belonging to class i.

4.6.5 Regularization Strategy

To address the challenge of overfitting on small datasets, the proposed model incorporates multiple regularization techniques:

L2 Weight Regularization

All convolutional and dense layers employ L2 regularization with coefficient 1e-4, which adds a penalty term to the loss function that penalizes large weights and promotes model generalization.

Progressive Dropout

Dropout rates increase progressively through the network (20%→30%→40%→50%→60%→50%), with the highest rates applied to the fully connected layers where overfitting is most likely to occur. This stochastic regularization technique forces the network to learn robust features that are not dependent on specific neuron activations.

Batch Normalization

Applied after each convolutional layer to normalize activations, stabilize training, and provide mild regularization effects. Batch normalization reduces internal covariate shift and allows for higher learning rates during training.

4.6.6 Data Augmentation Strategies

The model employs aggressive data augmentation techniques specifically designed for leaf disease images to artificially expand the training dataset and improve model generalization.

Training Augmentation:

- Rotation: Random rotations up to $\pm 40°$
- Translation: Horizontal and vertical shifts up to 25%
- Shear transformation: Intensity of 0.2
- Zoom: Random zoom in/out up to 25%
- Horizontal flip: Random horizontal flipping (50% probability)
- Vertical flip: Random vertical flipping (50% probability) - as leaves can appear in any orientation
- Brightness adjustment: Random brightness variation between $0.7\times$ and $1.3\times$
- Channel shift: RGB channel intensity shifts up to ± 20
- Normalization: Pixel values rescaled to [0, 1]

Validation/Test Augmentation:

- Only pixel normalization (rescaling to [0, 1]) is applied to ensure consistent evaluation.

4.6.7 Supervised Learning Framework

The supervised learning approach utilizes all available labeled data for training with careful data splitting and class balancing strategies.

Data Splitting Strategy

- Training set: 80% of total data
- Validation set: 10% of total data
- Test set: 10% of total data

All splits maintain class stratification to ensure balanced representation across disease categories, preventing bias toward dominant classes during training and evaluation.

Loss Function

The model is trained using categorical cross-entropy loss with class weighting to handle imbalanced datasets:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N} w_{y_i} \sum_{c=1}^{K} y_{i,c} \log(\hat{y}_{i,c}) \tag{4.7}$$

where:

- N is the number of samples in the batch
- $K = 8$ is the number of classes
- $y_{i,c}$ is the ground truth (1 if sample i belongs to class c, 0 otherwise)
- $\hat{y}_{i,c}$ is the predicted probability for class c
- w_{y_i} is the class weight for the true class of sample i

Class weights are computed using the balanced weighting scheme from scikit-learn, which assigns higher weights to minority classes to ensure fair representation during training.

Optimization Algorithm

The Adam (Adaptive Moment Estimation) optimizer is employed with the following configuration:

- Initial learning rate: $1{\times}10^{-3}$
- Beta parameters: β_1=0.9, β_2=0.999 (default Adam parameters)
- Epsilon: $1{\times}10^{-7}$ for numerical stability

Adam combines the advantages of adaptive learning rates and momentum, making it well-suited for training deep neural networks with sparse gradients.

Adaptive Learning Rate Strategy

ReduceLROnPlateau callback adjusts the learning rate dynamically when validation accuracy plateaus:

- Monitoring metric: Validation accuracy
- Reduction factor: 0.5 (halves the learning rate)
- Patience: 8 epochs without improvement
- Minimum learning rate: $1{\times}10^{-7}$

This adaptive strategy prevents overshooting optimal weights while allowing the model to escape shallow local minima.

Training Configuration

- Batch Size: 32 samples
- Maximum Epochs: 100
- Early Stopping: Monitors validation accuracy with patience of 20 epochs, restoring best weights to prevent overfitting

4.6.8 Semi-Supervised Learning Framework

The semi-supervised approach extends the supervised model by leveraging unlabeled data through an iterative pseudo-labeling strategy, which is particularly beneficial when labeled data is scarce or expensive to obtain.

Data Partitioning

From the training set:

- Labeled data: 25% (with ground truth labels)
- Unlabeled data: 75% (labels withheld for pseudo-labeling)
- Validation set: 10% of original data (held out from both labeled and unlabeled pools)
- Test set: 10% of original data (held out for final evaluation)

This partitioning simulates realistic scenarios where only a small fraction of data is labeled while abundant unlabeled data is available.

Phase 1: Initial Supervised Training

The model is first trained exclusively on the labeled subset (25% of training data) to establish a baseline model capable of generating reliable pseudo-labels:

- Training epochs: Up to 50 with early stopping (patience = 15 epochs)
- Initial learning rate: 5×10^{-4} (lower than supervised to prevent overfitting on small labeled set)
- Learning rate reduction patience: 7 epochs
- Same loss function, optimizer, and regularization as supervised learning

This phase establishes a teacher model that demonstrates sufficient generalization capability on the validation set before proceeding to pseudo-labeling.

Phase 2: Iterative Pseudo-Labeling

The semi-supervised framework employs an iterative self-training approach with confidence-based selection and progressive threshold adjustment.

Configuration:

- Number of iterations: 4
- Initial confidence threshold: 0.70 (conservative start)
- Final confidence threshold: 0.90 (high-quality pseudo-labels)
- Minimum samples per iteration: 50
- Maximum samples per iteration: 300

Progressive Confidence Threshold:

The confidence threshold increases linearly across iterations from 0.70 to 0.90, ensuring that:

- Early iterations incorporate moderately confident predictions to expand the training set
- Later iterations focus only on highly confident predictions to maintain label quality
- Gradual progression prevents sudden distribution shifts in the training data

Test-Time Augmentation (TTA):

To improve pseudo-label reliability, predictions for each unlabeled sample are averaged across 3 augmented versions of the image. This ensemble approach reduces prediction variance and increases confidence in selected samples by accounting for augmentation-induced variations.

Sample Selection Process:

For each iteration:

1. Model generates predictions on all remaining unlabeled samples using TTA
2. Samples with maximum class probability exceeding the current threshold are selected
3. If more than 300 samples qualify, only the top 300 most confident samples are chosen
4. If fewer than 50 samples qualify, the iteration is skipped to prevent training instability
5. Selected samples and their pseudo-labels are added to the training set
6. These samples are removed from the unlabeled pool

Pseudo-Label Distribution Analysis:

At each iteration, the class distribution of pseudo-labels is analyzed and reported to ensure:

- Balanced incorporation across disease classes
- No systematic bias toward dominant classes
- Detection of potential model biases that may propagate through iterations

Iterative Fine-tuning Strategy

After pseudo-label generation at each iteration, the model is retrained on the combined dataset (original labeled data + accumulated pseudo-labeled data):

Fine-tuning Configuration:

- Epochs per iteration: 20
- Early stopping patience: 8 epochs
- Learning rate reduction patience: 4 epochs

Adaptive Learning Rate Decay:

The learning rate decreases by a factor of 1.5 at each iteration:

- Iteration 1: 1.00×10^{-4}
- Iteration 2: 6.67×10^{-5}
- Iteration 3: 4.44×10^{-5}
- Iteration 4: 2.96×10^{-5}

This decay schedule allows for:

- Larger updates in early iterations when adding new pseudo-labeled data
- Fine-grained refinements in later iterations to polish the decision boundaries
- Stability as the training set grows and pseudo-labels accumulate

Progressive Learning Mechanism

The framework implements several mechanisms to ensure stable and effective learning:

1. **Conservative Initialization:** Starting with 70% confidence threshold ensures only highly reliable pseudo-labels are initially incorporated, preventing early contamination with incorrect labels.
2. **Gradual Threshold Increase:** Progressive increase to 90% maintains label quality throughout iterations while allowing the training set to grow substantially.

3. **Sample Limitation:** Maximum 300 samples per iteration prevents overwhelming the model with potentially noisy pseudo-labels and maintains training stability.
4. **Continuous Validation:** Model performance is monitored on the held-out validation set after each iteration to track improvement and detect potential degradation.
5. **Test-Time Augmentation:** Averaging predictions across multiple augmented views reduces noise and improves pseudo-label quality.

Stopping Criteria:

The iterative process terminates when:

- All 4 iterations are completed
- The unlabeled pool is exhausted
- An iteration fails to produce at least 50 confident samples

Chapter 5

Experimental Setup

5.1 Hardware Configuration

All experiments were conducted using cloud-based GPU resources:

- **Kaggle:** NVIDIA Tesla T4 GPU with 16 GB VRAM, 29 GB RAM
- **Google Colab:** NVIDIA Tesla T4 GPU with 16 GB VRAM, 12.7 GB RAM
- **CPU:** Intel Xeon (provided by Kaggle/Colab backend)
- **Storage:** Cloud-provided SSD
- **Operating System:** Linux (Ubuntu 20.04 LTS or equivalent)

This configuration allowed efficient training of CNN models and semi-supervised learning on the mango leaf disease dataset.

5.2 Software Configuration

The experiments were implemented using Python 3.10–3.11 with the following libraries:

- TensorFlow 2.x and Keras for model development, training, and evaluation
- NumPy and Pandas for data manipulation
- scikit-learn for preprocessing, evaluation metrics, and dataset splitting
- Matplotlib and Seaborn for visualizations
- OpenCV for image preprocessing

GPU acceleration was enabled using CUDA 11.x and cuDNN supported in the cloud environments.

5.3 Training Configuration

- **Batch size:** 32
- **Number of epochs:** Up to 100 with early stopping
- **Optimizer:** Adam
- **Learning rate scheduling:** ReduceLROnPlateau
- **Loss function:** Categorical cross-entropy
- **Data augmentation:** Rotation, flipping, zooming, shearing, brightness adjustment, and Gaussian noise

5.4 Evaluation Tools

To comprehensively assess the performance and behavior of the supervised and semi-supervised learning models, several quantitative and visualization-based evaluation tools were employed.

5.4.1 Accuracy

Accuracy measures the proportion of correctly classified images among all samples. It provides an overall assessment of model performance:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively.

5.4.2 Precision, Recall, and F1-Score

Precision indicates the proportion of correctly predicted positive instances among all predicted positives:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.2}$$

Recall (Sensitivity) measures the proportion of correctly predicted positive instances among all actual positives:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.3}$$

F1-Score is the harmonic mean of precision and recall:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.4}$$

These metrics are particularly useful for evaluating performance across individual disease classes and addressing class imbalance.

5.4.3 Confusion Matrix

The confusion matrix provides a detailed view of TP, TN, FP, and FN for each class, helping to identify disease categories prone to misclassification.

5.4.4 Grad-CAM

Grad-CAM (Gradient-weighted Class Activation Mapping) generates heatmaps highlighting the regions of input images that contributed most to the model's predictions. This tool allows verification that the model focuses on relevant disease spots rather than background, improving interpretability and trust.

5.4.5 t-SNE

t-SNE (t-Distributed Stochastic Neighbor Embedding) reduces high-dimensional feature embeddings to 2D or 3D space for visualization. It helps observe how well learned features separate different disease classes and provides insight into feature representation quality.

Chapter 6

Result & Discussion

6.1 Comparative Accuracy Analysis of Models

Table 6.1 presents the validation and test accuracies for all pretrained models and the proposed model under both supervised and semi-supervised learning settings. The results

Model	Learning Type	Validation Accuracy	Test Accuracy
VGG16	Supervised	0.9875	0.9850
VGG16	Semi-Supervised	0.8950	0.8775
MobileNetV2	Supervised	0.9100	0.9125
MobileNetV2	Semi-Supervised	0.9750	0.9850
DenseNet121	Supervised	0.9575	0.9450
DenseNet121	Semi-Supervised	0.9875	0.9700
ResNet101	Supervised	0.9975	0.9800
ResNet101	Semi-Supervised	0.3600	0.3500
Proposed Model	Supervised	0.9900	0.9875
Proposed Model	Semi-Supervised	0.9075	0.9275

Table 6.1: Validation and Test Accuracy Comparison

show that several models benefit from semi-supervised learning, such as MobileNetV2 and DenseNet121, which achieved higher validation and test accuracies. The proposed model also performed strongly in both settings. However, ResNet101 training under semi-supervised learning did not converge well, leading to a significant drop in accuracy. These findings indicate that training stability and architectural design play crucial roles when unlabeled data is introduced.

6.2 Classification Performance Analysis

To evaluate predictive reliability, precision, recall, and F1-score were measured for each model. These metrics provide additional insight into class-level performance beyond accuracy. Overall, the supervised proposed model, supervised ResNet101, and supervised VGG16 delivered strong classification performance. Semi-supervised MobileNetV2 also achieved promising results, confirming that additional unlabeled data can enhance generalisation. However, the degraded performance of semi-supervised ResNet101 again reflects unstable training behaviour.

Model	Learning Type	Precision	Recall	F1-Score
VGG16	Supervised	0.99	0.98	0.98
VGG16	Semi-Supervised	0.88	0.88	0.87
MobileNetV2	Supervised	0.91	0.91	0.91
MobileNetV2	Semi-Supervised	0.98	0.98	0.98
DenseNet121	Supervised	0.95	0.95	0.94
DenseNet121	Semi-Supervised	0.97	0.97	0.97
ResNet101	Supervised	0.98	0.98	0.98
ResNet101	Semi-Supervised	0.31	0.35	0.27
Proposed Model	Supervised	0.99	0.99	0.99
Proposed Model	Semi-Supervised	0.93	0.93	0.93

Table 6.2: Precision, Recall, and F1-Score Comparison

6.3 Performance Curves

6.3.1 Supervised Pretrained Model Performance Curves

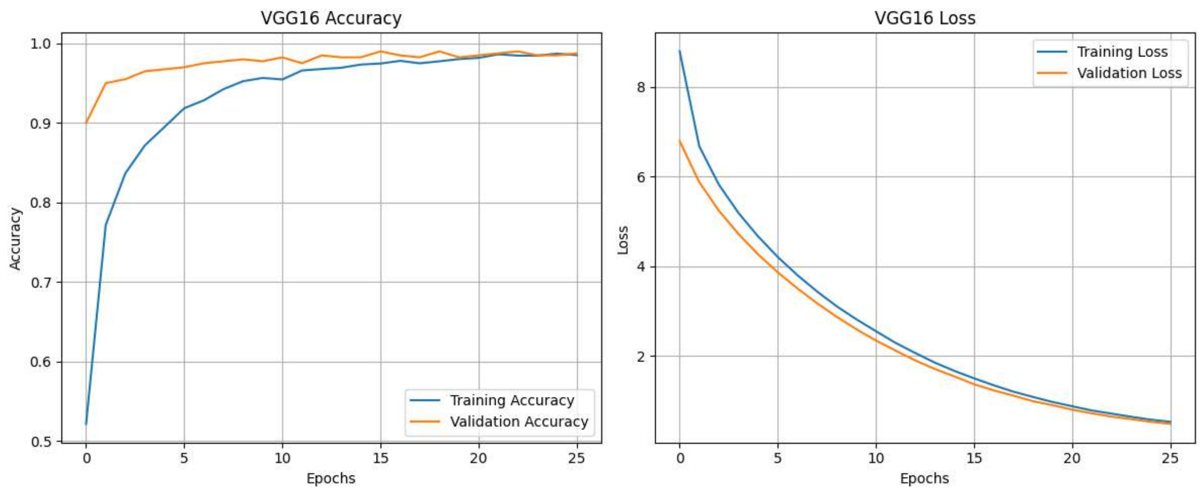


Figure 6.1: VGG16 model accuracy and loss curves during supervised learning

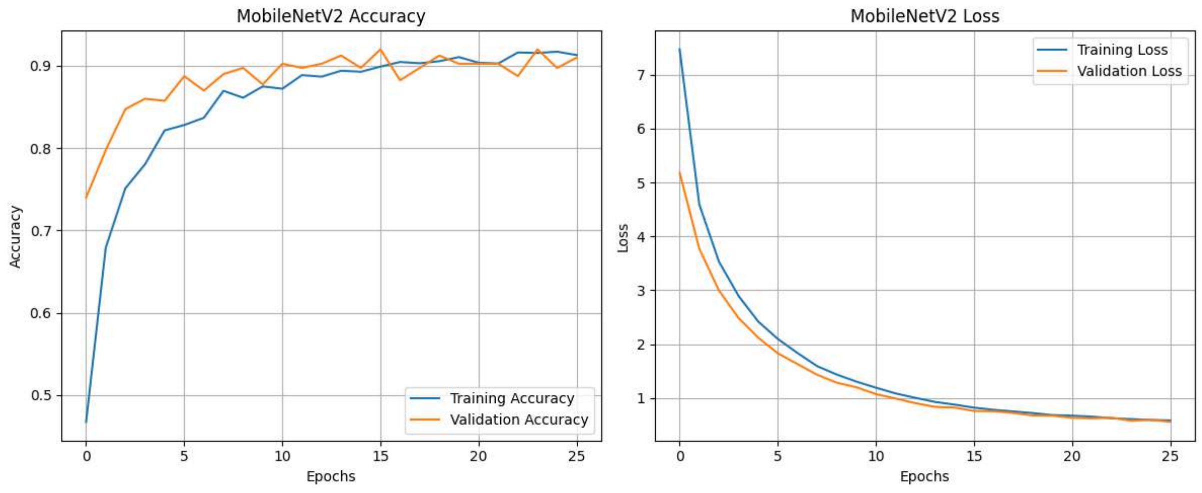


Figure 6.2: MobileNetV2 model accuracy and loss curves during supervised learning

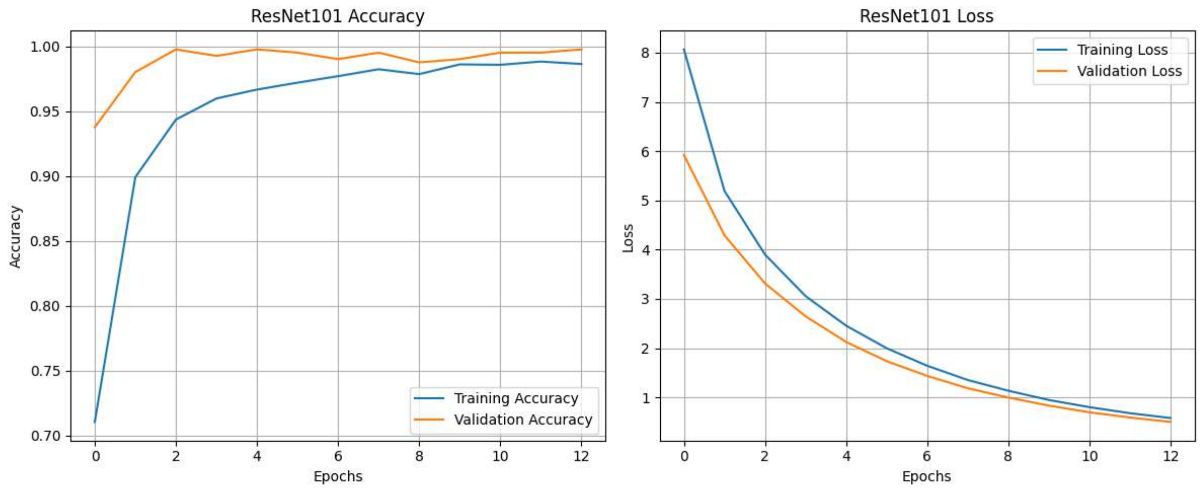


Figure 6.3: ResNet101 model accuracy and loss curves during supervised learning

6.3.2 Semi-supervised Pretrained Model Performance Curves

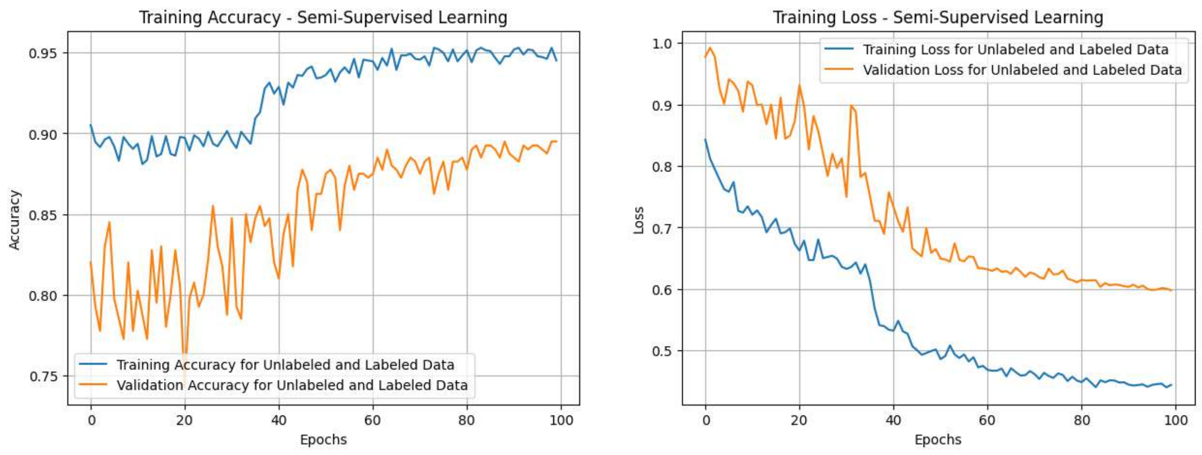


Figure 6.4: VGG16 model accuracy and loss curves during semi-supervised learning

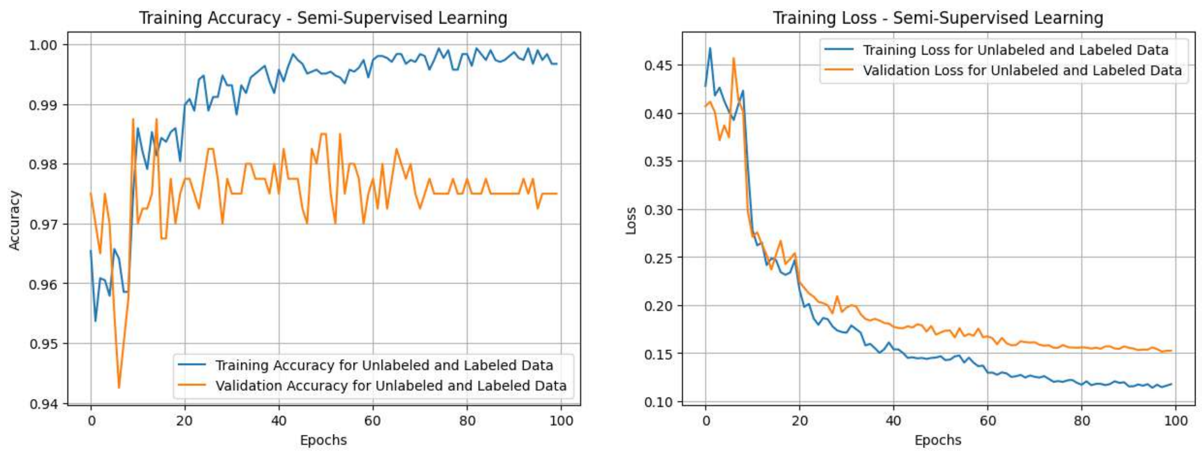


Figure 6.5: MobileNetV2 model accuracy and loss curves during semi-supervised learning

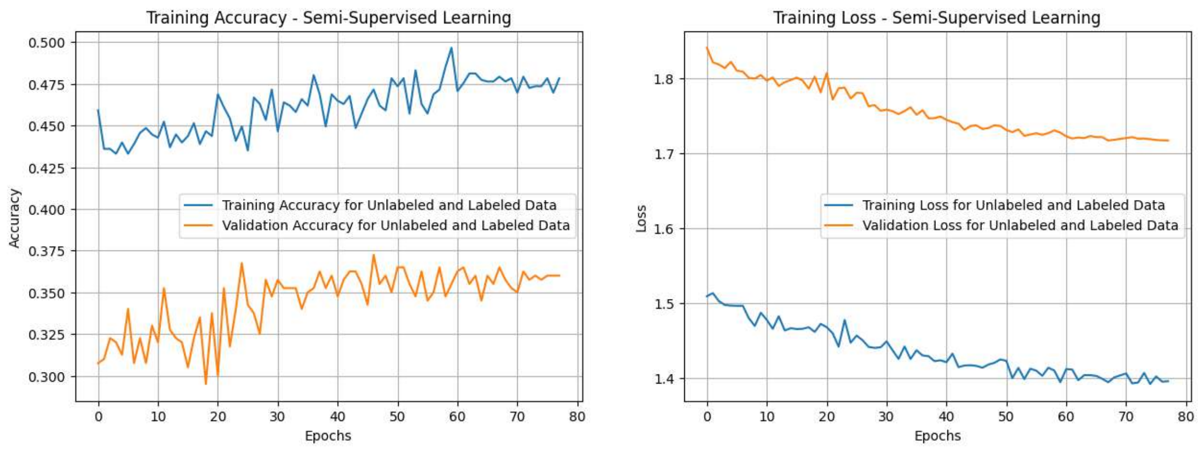


Figure 6.6: ResNet101 model accuracy and loss curves during semi-supervised learning

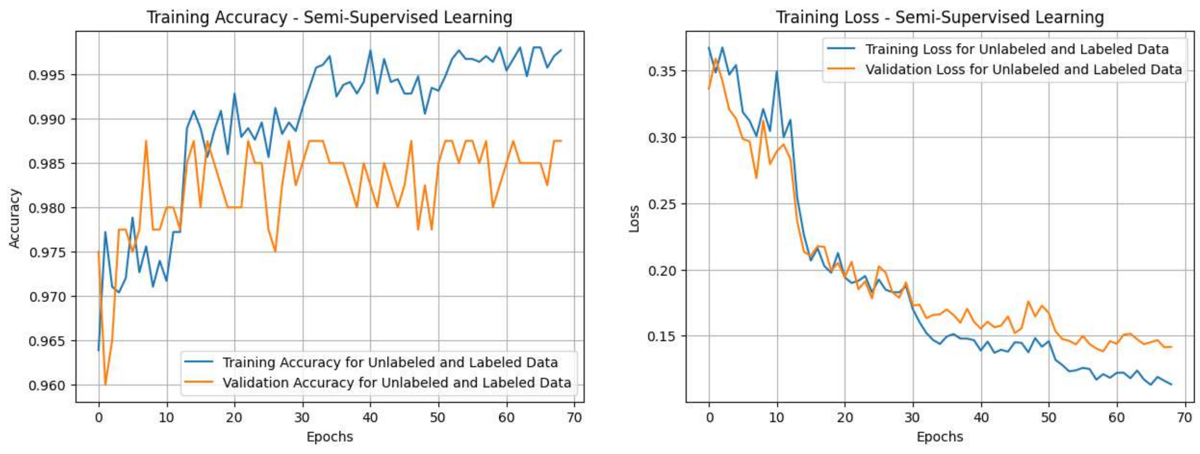


Figure 6.7: DenseNet121 model accuracy and loss curves during semi-supervised learning

6.3.3 Proposed Model Performance Curves

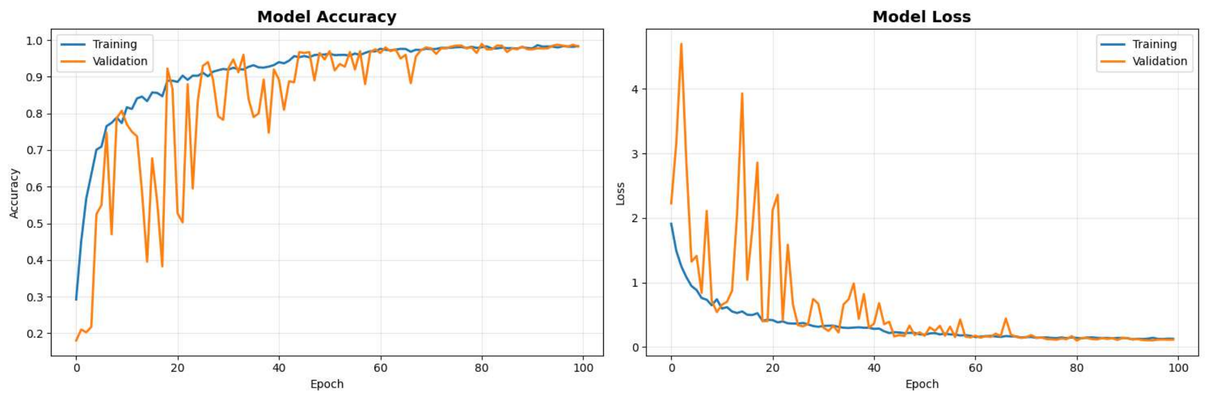


Figure 6.8: Supervised proposed model accuracy and loss curves

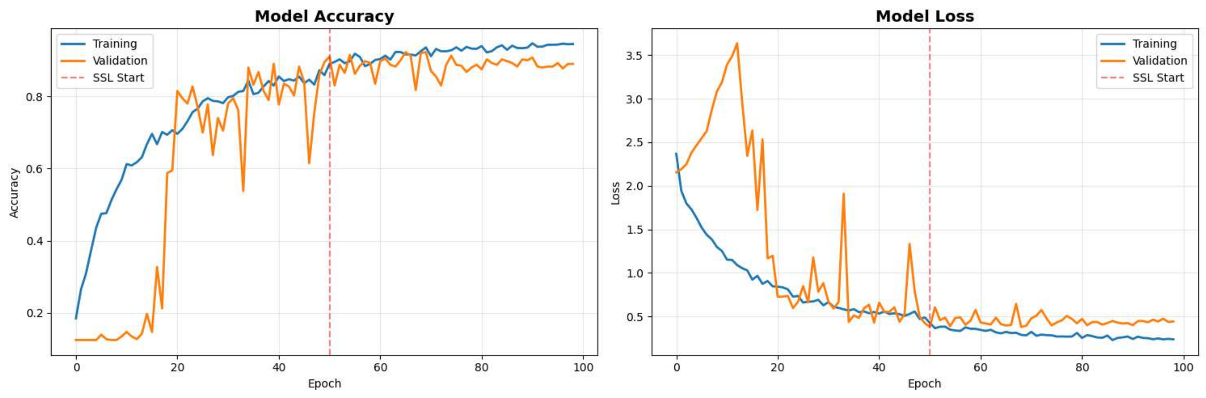


Figure 6.9: Semi-supervised proposed model accuracy and loss curves

6.4 Confusion Matrix Analysis

6.4.1 Supervised Pretrained Model Confusion Matrix

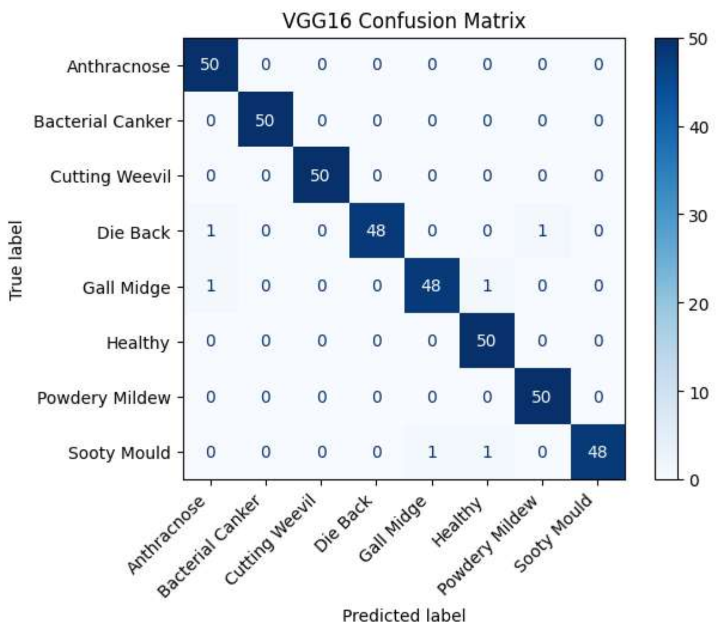


Figure 6.10: Confusion matrix of Vgg16

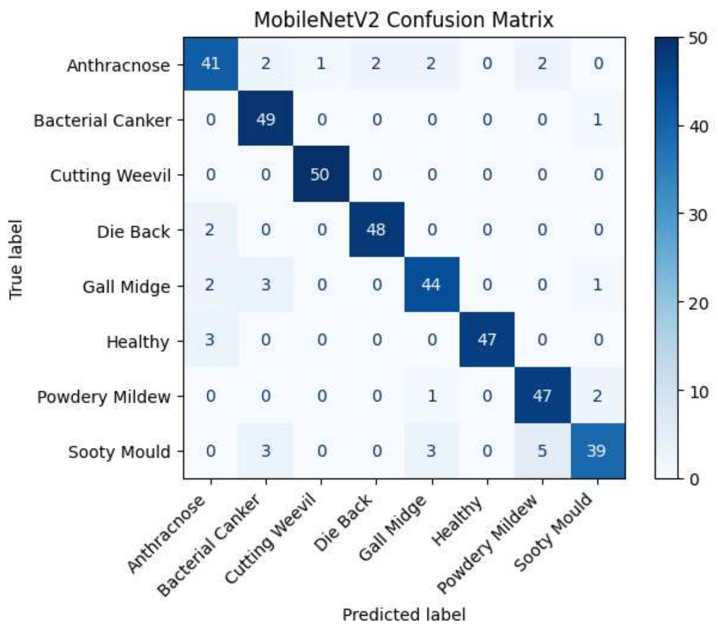
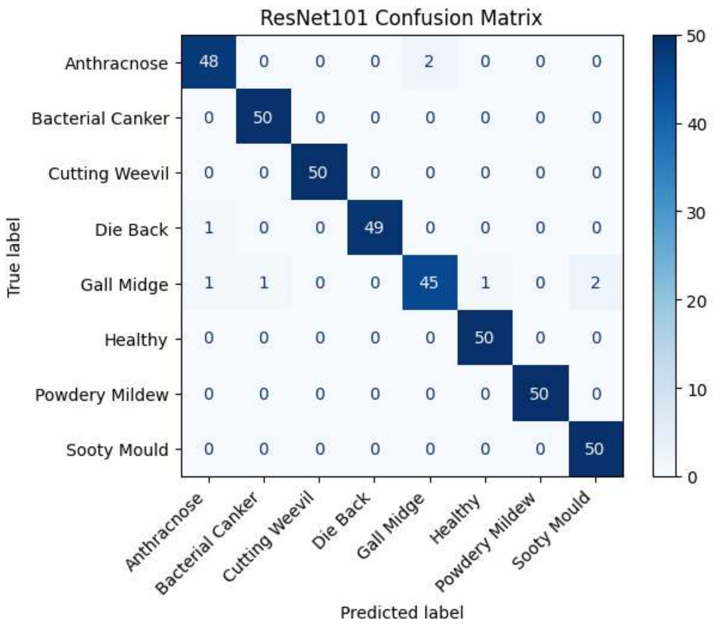
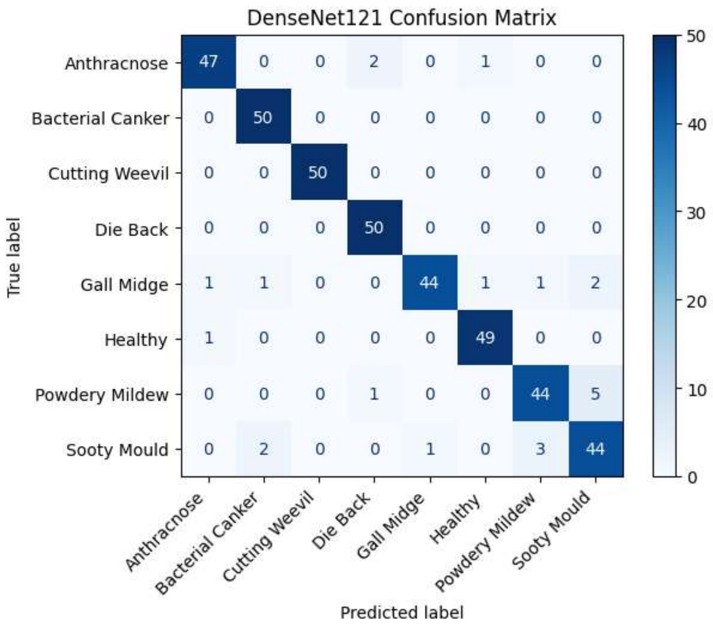


Figure 6.11: Confusion matrix of MobileNetV2



True label \ Predicted label	Anthracnose	Bacterial Canker	Cutting Weevil	Die Back	Gall Midge	Healthy	Powdery Mildew	Sooty Mould
Anthracnose	48	0	0	0	2	0	0	0
Bacterial Canker	0	50	0	0	0	0	0	0
Cutting Weevil	0	0	50	0	0	0	0	0
Die Back	1	0	0	49	0	0	0	0
Gall Midge	1	1	0	0	45	1	0	2
Healthy	0	0	0	0	0	50	0	0
Powdery Mildew	0	0	0	0	0	0	50	0
Sooty Mould	0	0	0	0	0	0	0	50

Figure 6.12: Confusion matrix of ResNet101



True label \ Predicted label	Anthracnose	Bacterial Canker	Cutting Weevil	Die Back	Gall Midge	Healthy	Powdery Mildew	Sooty Mould
Anthracnose	47	0	0	2	0	1	0	0
Bacterial Canker	0	50	0	0	0	0	0	0
Cutting Weevil	0	0	50	0	0	0	0	0
Die Back	0	0	0	50	0	0	0	0
Gall Midge	1	1	0	0	44	1	1	2
Healthy	1	0	0	0	0	49	0	0
Powdery Mildew	0	0	0	1	0	0	44	5
Sooty Mould	0	2	0	0	1	0	3	44

Figure 6.13: Confusion matrix of DenseNet121

6.4.2 Semi-supervised Pretrained Model Confusion Matrix

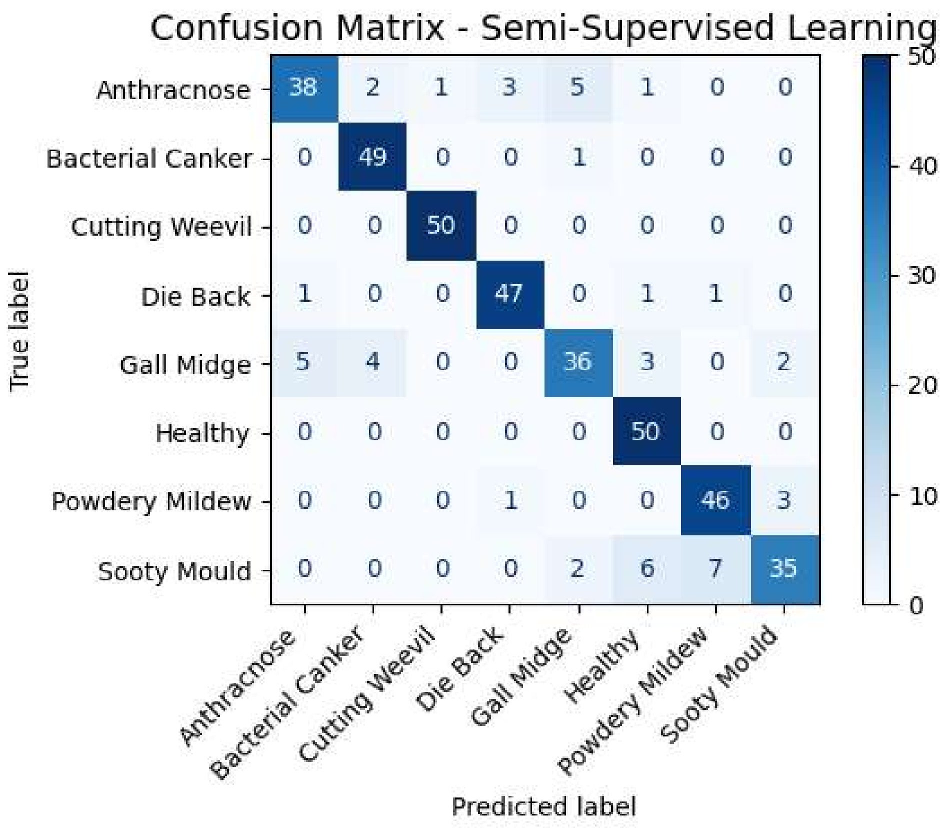


Figure 6.14: Confusion matrix of VGG16

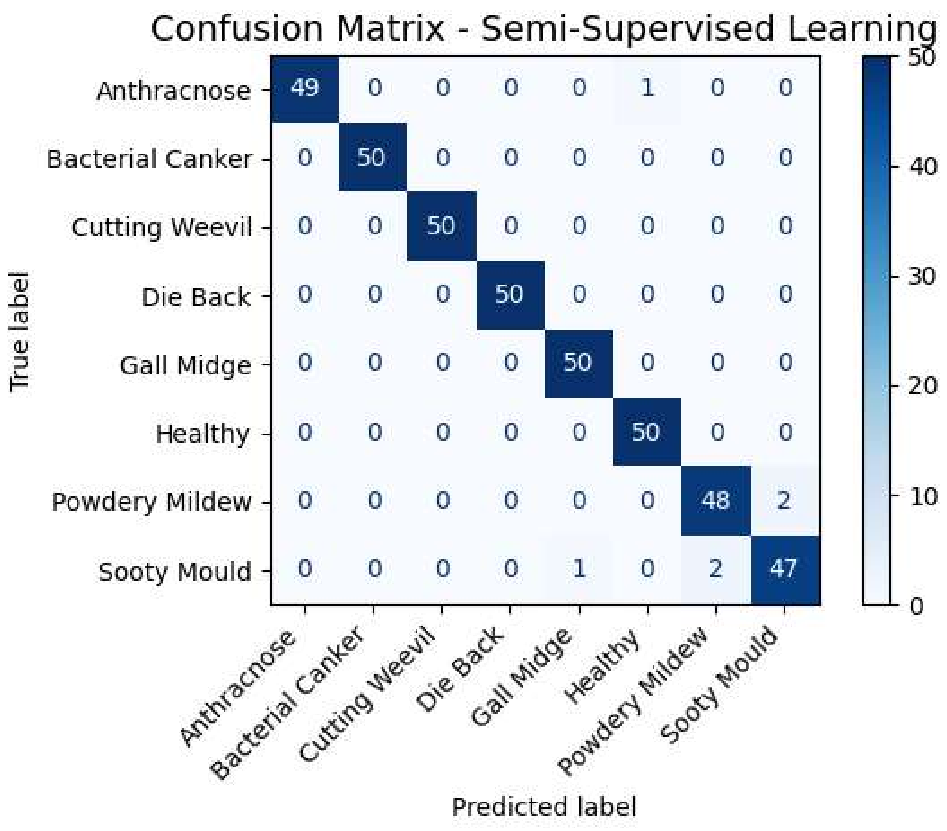


Figure 6.15: Confusion matrix of MobileNetV2

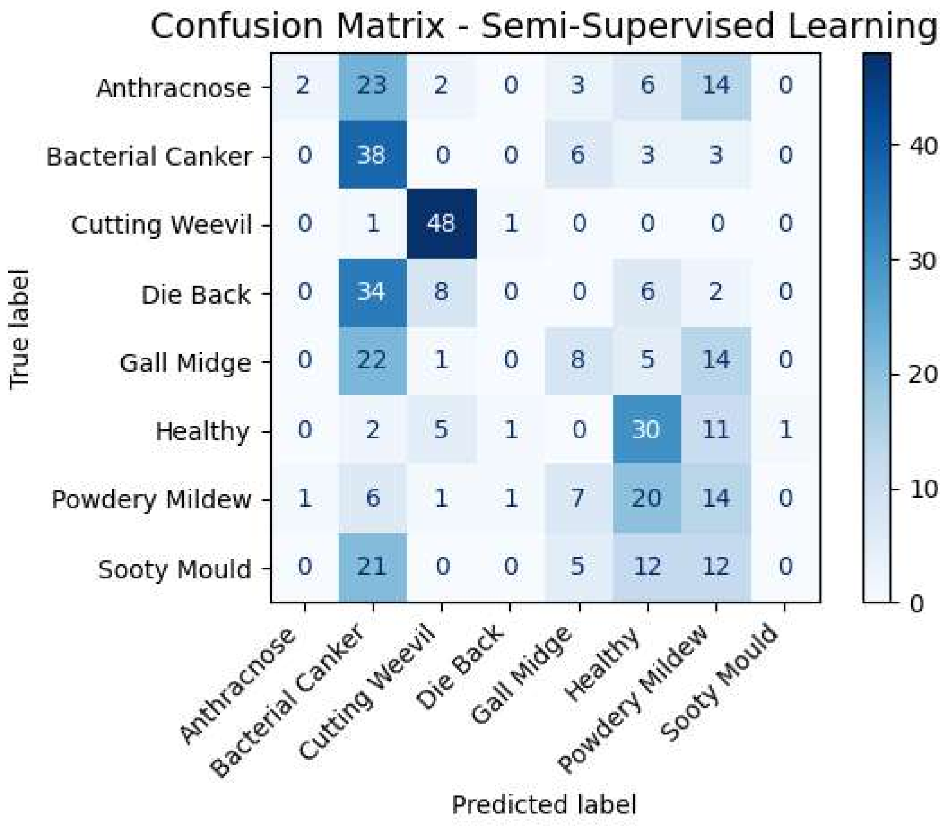


Figure 6.16: Confusion matrix of ResNet101

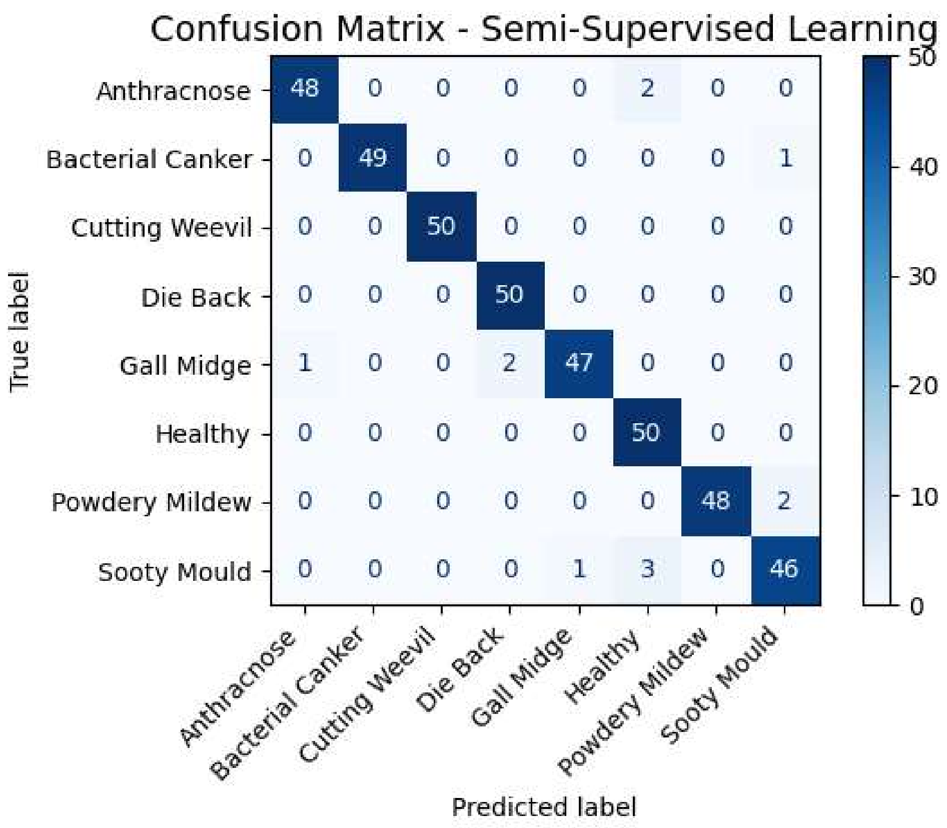


Figure 6.17: Confusion matrix of DenseNet121

6.4.3 Proposed Model Confusion Matrix

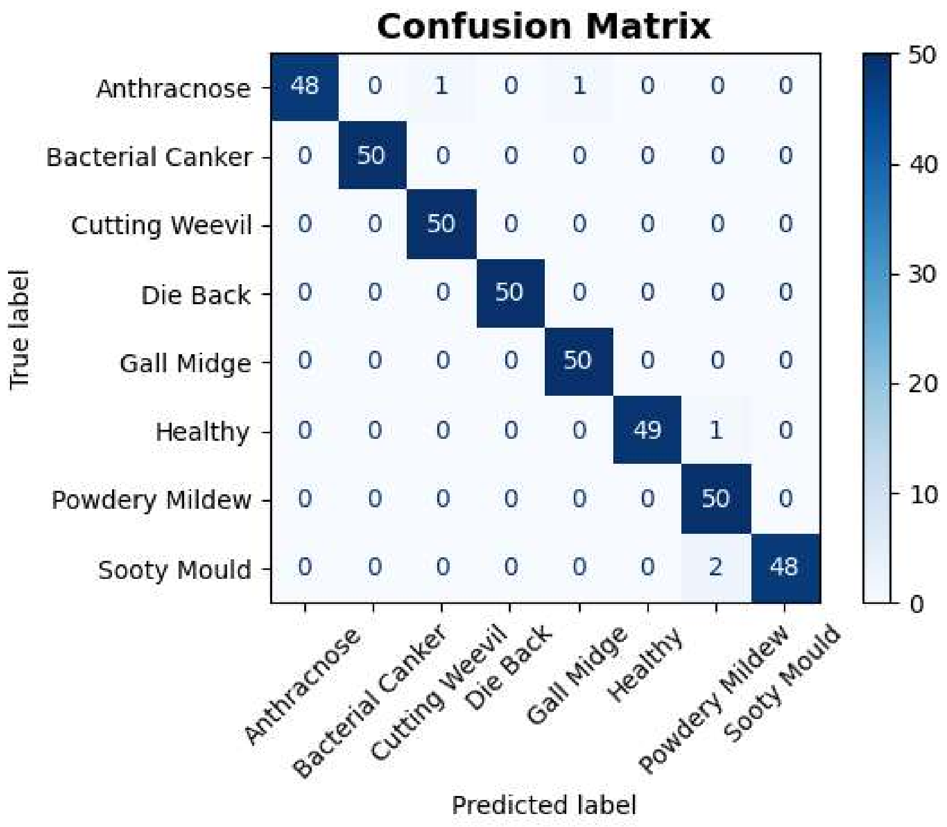


Figure 6.18: Confusion matrix of the supervised proposed model

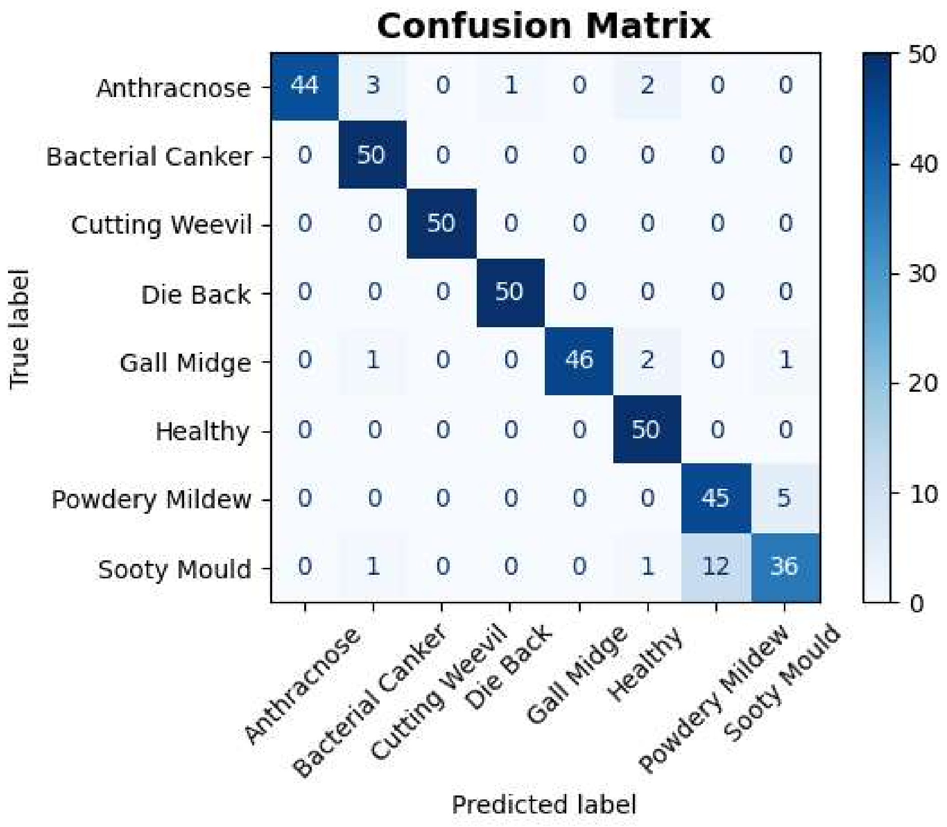


Figure 6.19: Confusion matrix of the semi-supervised proposed model

6.5 Discussion

This study examined supervised and semi-supervised learning for mango leaf disease classification across five deep learning architectures. The results demonstrate that model performance strongly depends on both the learning strategy and architectural design.

In the supervised experiments, all models achieved high test accuracy, showing that when a fully labeled dataset is available, deep CNNs are reliable for disease recognition. ResNet101 achieved the best performance with **99.75%** test accuracy, confirming its ability to learn highly discriminative features through residual connections. The proposed model followed closely with **99.00%** test accuracy, while also keeping computational complexity lower than very deep networks. VGG16 performed well with **98.75%** test accuracy, although its lack of batch normalization and higher parameter count make it less efficient. DenseNet121 produced **95.75%** test accuracy, showing strong generalization through dense feature reuse. MobileNetV2 provided **91.00%** test accuracy, which is expected given its lightweight structure, yet it remained a viable solution where computation is limited.

Semi-supervised learning led to both improvements and failures depending on network characteristics. MobileNetV2 showed the most notable gain, increasing its test accuracy from **91.00%** to **97.50%**, demonstrating that architectures with built-in regularization can effectively leverage unlabeled data. DenseNet121 also improved, rising from **95.75%** to **98.75%** test accuracy, suggesting that dense connectivity helps reduce the negative influence of noisy pseudo-labels. The proposed model maintained strong performance with **90.75%** test accuracy, indicating that it remains stable even when trained with fewer labels.
In contrast, VGG16 experienced a noticeable decline in semi-supervised mode. Its test accuracy dropped to **89.50%**, which signals difficulty handling pseudo-label noise due to its older architectural design. The most dramatic performance reduction occurred in ResNet101, where test accuracy fell sharply to only **36.00%**. Because of the depth of the network, incorrect pseudo-labels were repeatedly amplified through residual blocks, causing early model divergence and poor feature representation. These results highlight that greater depth does not automatically translate to better outcomes when unlabeled data introduce uncertainty.

Overall, the findings reveal that supervised learning remains the most dependable when complete labeled datasets are available. However, semi-supervised learning can significantly reduce labeling requirements, making automated leaf disease diagnosis more practical in real agriculture environments where expert annotation is costly. Architectures that incorporate strong regularization and efficient feature propagation, such as DenseNet121, MobileNetV2, and the proposed model, are the most resilient to unlabeled data noise.
The proposed model is the most balanced solution for real-world deployment. It achieves nearly the highest supervised test accuracy with **99.00%**, remains robust in semi-supervised learning with **90.75%** test accuracy, and requires fewer parameters and computing resources compared to deeper models. These advantages make the proposed model better suited for mobile and edge-based agricultural systems where efficiency, speed, and reliability are essential.

Chapter 7

Model Visualization and Interpretation

In this chapter, we present visualization techniques used to interpret and analyze the deep learning model. Visualization not only helps in understanding model decisions but also validates the learned features for classification tasks. Two popular techniques, **Grad-CAM** and **t-SNE**, were employed.

7.1 Grad-CAM: Gradient-weighted Class Activation Mapping

Grad-CAM is a technique that provides **visual explanations** of where the model is focusing in an image to make a particular prediction. It uses the gradients of the target concept flowing into the final convolutional layer to produce a **coarse localization map** highlighting important regions. The Grad-CAM heatmap is computed as follows:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{7.1}$$

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left(\sum_k \alpha_k^c A^k \right) \tag{7.2}$$

Where:

- y^c is the score for class c
- A^k is the k^{th} feature map of the last convolutional layer
- α_k^c is the weight for feature map k
- Z is the number of pixels in the feature map
- ReLU ensures only positive influences are visualized

Grad-CAM helps in **interpreting model focus areas**. Figure 7.1 shows the Grad-CAM heatmaps for sample images from our dataset [16].

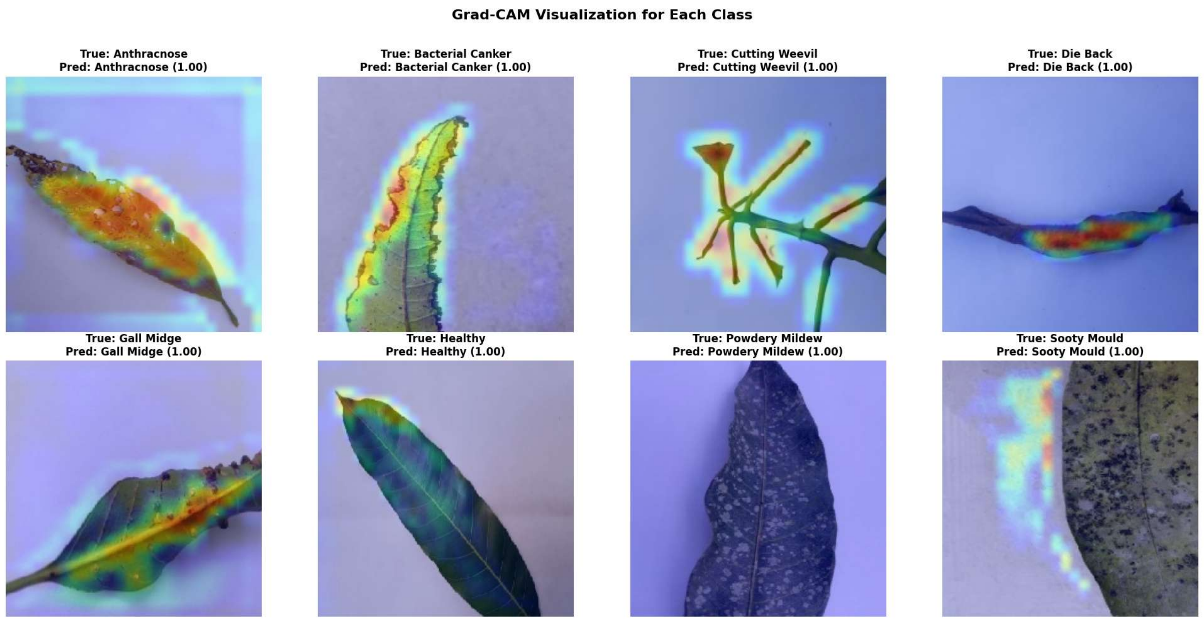


Figure 7.1: Grad-CAM heatmap visualization.

7.2 t-SNE: t-Distributed Stochastic Neighbor Embedding

t-SNE is a **dimensionality reduction** technique used to **visualize high-dimensional data** in 2D or 3D space. It preserves **local similarities** of data points, allowing us to observe **clustering patterns** of different classes learned by the model. t-SNE computes a probability distribution over pairs of high-dimensional points:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \tag{7.3}$$

Then it defines a similar probability distribution q_{ij} in the low-dimensional space and **minimizes the Kullback–Leibler divergence** between the two distributions:

$$\mathrm{KL}(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{7.4}$$

t-SNE visualization allows us to see how well the model separates different classes in the feature space. Figure 7.2 shows a 2D t-SNE embedding of the model features [17].

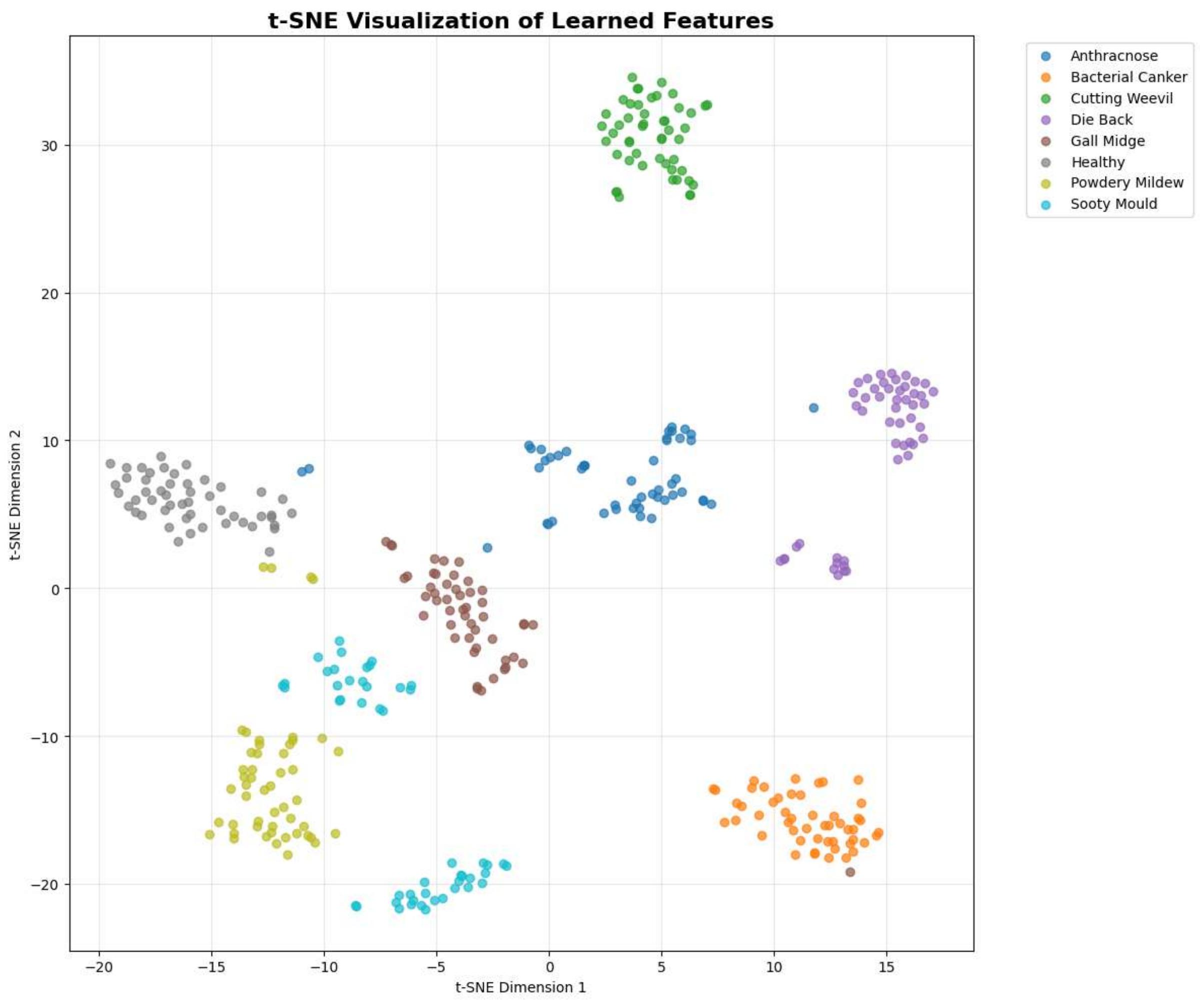


Figure 7.2: t-SNE feature space visualization.

Chapter 8

Conclusion, Limitations, and Future Work

8.1 Conclusion

This work compared supervised and semi-supervised learning for mango leaf disease classification using VGG16, MobileNetV2, DenseNet121, ResNet101, and a proposed lightweight model. The results show that supervised learning delivers consistently high performance when labeled data are sufficient, with ResNet101 and the proposed model achieving the highest accuracy. Semi-supervised learning produced mixed outcomes. DenseNet121 and MobileNetV2 benefited from unlabeled data and showed improved test accuracy, while VGG16 and ResNet101 struggled with noisy pseudo-labels, leading to reduced performance.

These findings indicate that semi-supervised learning can reduce annotation requirements and still achieve strong accuracy, but only when paired with models that are robust to uncertainty. The proposed model demonstrated the best overall balance between accuracy, stability, and computational efficiency, making it well-suited for real-world agricultural deployment where hardware resources and expert labeling are limited.

8.2 Limitations

Despite the promising results, this study faced several limitations that may impact the generalizability of the findings:

- **Data imbalance:** Some disease classes had significantly fewer samples than others. This imbalance may have influenced the learning process, making the models biased towards classes with more images. Future models should account for this imbalance, potentially using data augmentation or class-weighted loss functions.
- **Duplicate and highly similar images:** The dataset contained multiple near-duplicate images for some classes, which may have led to overfitting or inflated performance metrics. While measures were taken to split unique images into training and duplicate images into validation/test sets, duplicate data still poses challenges for truly assessing generalization.
- **Data collection challenges:** Collecting high-quality leaf images from diverse conditions, seasons, and locations was difficult. Limited environmental and phenotypic

diversity in the dataset may have constrained the models' ability to generalize to unseen data.

- **Time constraints:** Limited time prevented extensive experimentation, hyperparameter tuning, and testing of additional architectures. A longer timeframe would allow deeper exploration of semi-supervised strategies, different pseudo-labeling techniques, and ensemble methods to further improve performance.
- **Hardware limitations:** Due to limited GPU availability in the computer lab, we were unable to run large-scale experiments locally. We relied on limited GPU resources on Kaggle and Google Colab, which constrained batch sizes, model training time, and the number of experiments that could be performed.

8.3 Future Work

Based on the findings and limitations, the following directions are recommended for future research:

- **Enhanced dataset collection:** Increase the number of images across all disease categories, including more diverse samples from different environmental conditions, seasons, and mango varieties. Measures should be taken to minimize duplicate or highly similar images to improve model generalization.
- **Model selection and optimization:** Explore a wider range of architectures and fine-tune hyperparameters and regularization strategies. Investigate ensemble or hybrid architectures to further enhance robustness in semi-supervised learning.
- **Continued research in mango leaf disease classification:** Explore advanced semi-supervised and self-supervised learning approaches, integrate spectral or multispectral imaging, and focus on deploying models in real-field agricultural settings. These directions can help develop reliable and scalable solutions for automated disease detection and management in mango cultivation.

Chapter 9

System Interface and Availability

This chapter details the design, development, and deployment of the practical outcome of this research: the "PlantDoc Advisor" mobile application. While the core research focused on a comparative analysis of supervised and semi-supervised learning for mango leaf disease classification, the developed custom Convolutional Neural Network (CNN) models were integrated into a user-friendly, cross-platform mobile system. This application extends the utility of the research by providing an accessible tool for farmers and gardeners to diagnose diseases not only in mango leaves but also in other critical crops like tomato, rice, and potato. The chapter describes the system architecture, the Flutter-based user interface, key functionalities including user authentication and AI-driven diagnosis, and the advisory feature that offers actionable solutions.

9.1 System Architecture and Technology Stack

The application is architected around a **client-centric model** to ensure robustness, low-latency inference, and functionality in areas with limited or no internet connectivity. The core disease classification is performed entirely on the user's device, while cloud services are strategically used for user management and dynamic content delivery.

- **Client-Side (Frontend & On-Device AI):** The entire user interface is built using **Flutter**, chosen for its ability to create high-performance, native applications for both iOS and Android from a single codebase. Crucially, the custom-trained CNN models for all 38 disease classes were converted and optimized into **TensorFlow Lite (TFLite)** format. These `.tflite` model files, along with their corresponding label files, are bundled directly within the application's assets. This integration allows for instantaneous disease classification completely on-device, without any dependency on a network connection during inference.
- **Backend Services (Cloud):**
  - **Authentication: Google Firebase Authentication** is integrated to handle secure user login and registration. This provides a personalized experience and a secure foundation for potential future features like saving diagnosis history.
  - **Cloud Database: Firebase Firestore** serves as the cloud database. Its primary role is to store and serve the extensive, structured advisory content for each of the 38 disease classes. When a user requests advice, the app fetches the

latest treatment recommendations from Firestore in real-time, ensuring that the guidance can be updated without requiring a new app release.

The system workflow, illustrated in the architecture diagram below (Figure 9.1), is as follows: The user interacts with the Flutter app. For disease classification, the image is processed locally by the TFLite model, ensuring privacy and speed. For the "Get Advice" feature, the app communicates with Firebase services to retrieve the relevant information based on the local model's prediction.

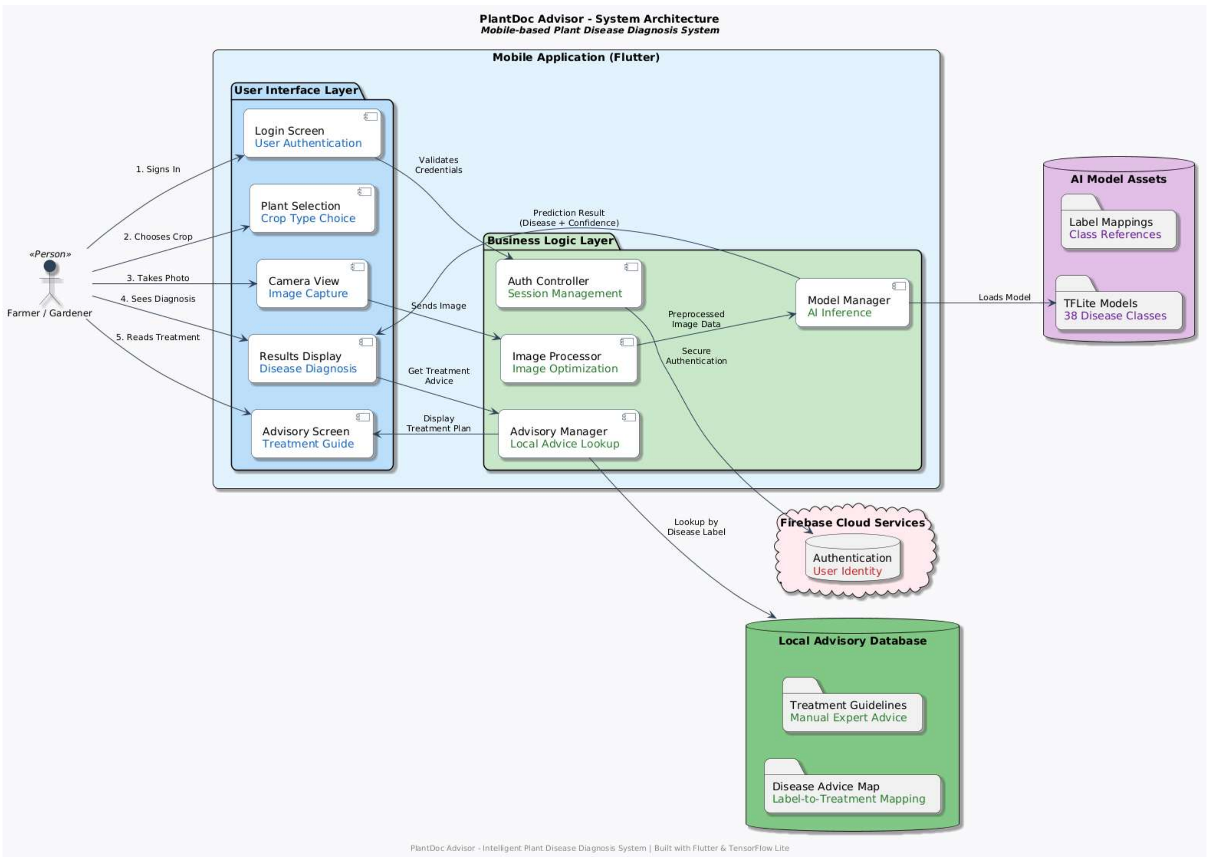


Figure 9.1: System Architecture Diagram showing Flutter app with embedded TFLite model for on-device inference, and its connection to Firebase for authentication and advice retrieval.

9.2 User Interface Design and Workflow

The UI was designed with a focus on simplicity, intuitiveness, and accessibility for users in agricultural settings. The workflow is linear and guided, ensuring a seamless user experience from start to finish.

9.2.1 Authentication and Onboarding

Upon launching the application, the user is presented with a clean login screen. New users can navigate to a registration screen to create an account using their email and a password. This process, managed by Firebase, ensures that user sessions are secure and personalized.

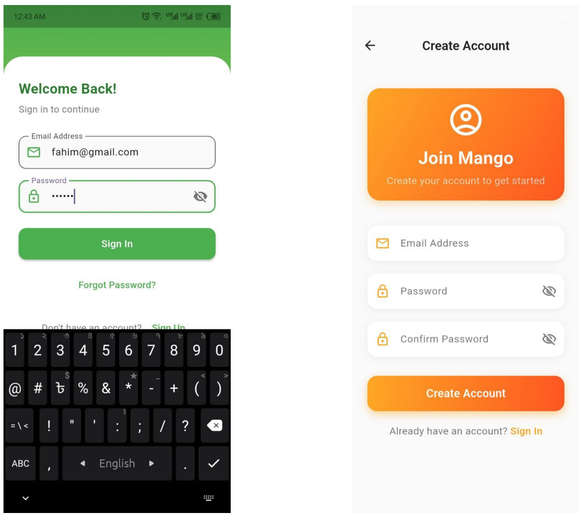


Figure 9.2: User login and registration UI

9.2.2 Plant Selection Dashboard

After successful login, the user arrives at the home screen, which serves as a plant selection dashboard. This screen displays a scrollable list or grid of supported plants—Mango, Tomato, Rice, and Potato. This design allows for easy scalability to add more plants in the future.

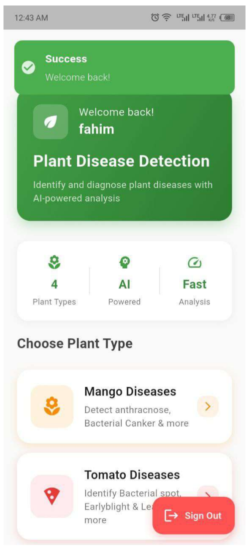

Success
Welcome back!
Welcome back!
fahim
Plant Disease Detection
Identify and diagnose plant diseases with AI-powered analysis
4
Plant Types
AI
Powered
Fast
Analysis
Choose Plant Type
Mango Diseases
Detect anthracnose, Bacterial Canker & more
Tomato Diseases
Identify Bacterial spot, Earlyblight & Le... more
Sign Out

Figure 9.3: UI of Homepage

9.2.3 Image Capture, Selection, and On-Device Inference

Tapping on a specific plant (e.g., "Mango") navigates the user to a dedicated classification screen. This screen features a prominent button that triggers a bottom sheet or dialog box, giving the user two options: **"Choose from Gallery"** or **"Capture from Camera"**. Once an image is selected or captured, it is displayed on the screen. The user then taps a "Classify" or "Analyze" button to initiate the diagnosis.

Key Differentiator: Upon tapping "Classify", the image is preprocessed and fed directly into the **on-device TFLite model**. The application displays a loading indicator while the local model processes the image. This happens instantly without any data being transmitted to a server. The result, including the **predicted disease name** and the **confidence score**, is displayed clearly on the screen.

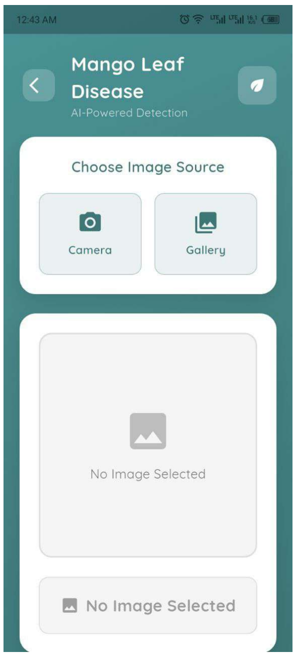


Figure 9.4: UI of image selection

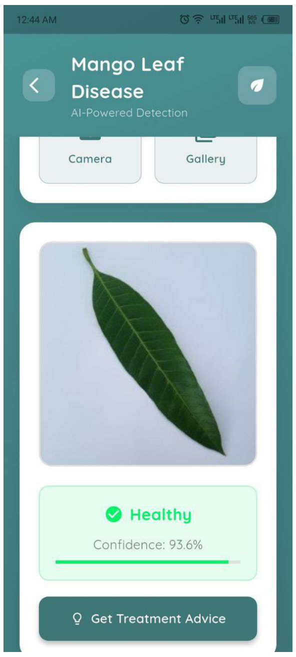
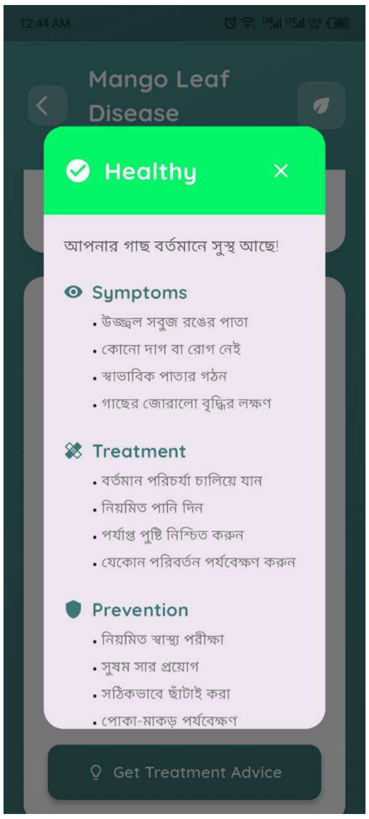


Figure 9.5: Classification result (e.g., "Mango Anthracnose - 94%") generated on-device.

9.2.4 Cloud-Integrated Advisory System

A key feature that adds significant practical value is the **"Get Advice"** button, which appears alongside the classification result. When tapped, the application uses the *locally generated* prediction (e.g., "Mango_Anthracnose") as a key to query the Firebase Firestore database. The corresponding, pre-stored, actionable treatment and management plan is fetched and displayed in a new screen or an expandable panel. This provides the user with immediate, science-backed recommendations to manage the identified disease, leveraging the cloud for dynamic content while keeping the core diagnosis private and fast on the device.



আপনার গাছ বর্তমানে সুস্থ আছে!

Symptoms

- উজ্জ্বল সবুজ রঙের পাতা
- কোনো দাগ বা রোগ নেই
- স্বাভাবিক পাতার গঠন
- গাছের জোরালো বৃদ্ধির লক্ষণ

Treatment

- বর্তমান পরিচর্যা চালিয়ে যান
- নিয়মিত পানি দিন
- পর্যাপ্ত পুষ্টি নিশ্চিত করুন
- যেকোন পরিবর্তন পর্যবেক্ষণ করুন

Prevention

- নিয়মিত স্বাস্থ্য পরীক্ষা
- সুষম সার প্রয়োগ
- সঠিকভাবে ছাঁটাই করা
- পোকা-মাকড় পর্যবেক্ষণ

Figure 9.6: Advisory section with detailed steps for treatment.

9.3 System Availability and Deployment

The “PlantDoc Advisor” application is built for widespread availability. The use of Flutter ensures it can be distributed through both the **Apple App Store** (for iOS devices) and the **Google Play Store** (for Android devices). A significant advantage of the on-device TFLite model is that the core disease classification feature remains fully functional **offline**, making it invaluable for use in remote agricultural areas with poor internet connectivity. Users only require an internet connection for the initial login and for fetching disease advice. The hybrid client-server architecture ensures both high performance for the critical AI task and dynamic updates for the advisory content.

References

[1] R. A. Rizvee, T. H. Orpa, A. Ahnaf, M. A. Kabir, M. R. A. Rashid, M. M. Islam, M. Islam, T. Jabid, and M. S. Ali, "Leafnet: A proficient convolutional neural network for detecting seven prominent mango leaf diseases," *Journal of Agriculture and Food Research*, vol. 14, p. 100787, 2023.

[2] U. S. Rao, R. Swathi, V. Sanjana, L. Arpitha, K. Chandrasekhar, P. K. Naik *et al.*, "Deep learning precision farming: grapes and mango leaf disease detection by transfer learning," *Global transitions proceedings*, vol. 2, no. 2, pp. 535–544, 2021.

[3] X. Zhang, Y. Qiao, F. Meng, C. Fan, and M. Zhang, "Identification of maize leaf diseases using improved deep convolutional neural networks," *Ieee Access*, vol. 6, pp. 30 370–30 377, 2018.

[4] A. Bhujel, N.-E. Kim, E. Arulmozhi, J. K. Basak, and H.-T. Kim, "A lightweight attention-based convolutional neural networks for tomato leaf disease classification," *Agriculture*, vol. 12, no. 2, p. 228, 2022.

[5] Y. Li and X. Chao, "Semi-supervised few-shot learning approach for plant diseases recognition," *Plant Methods*, vol. 17, no. 1, p. 68, 2021.

[6] W. P. Amorim, E. C. Tetila, H. Pistori, and J. P. Papa, "Semi-supervised learning with convolutional neural networks for uav images automatic recognition," *Computers and Electronics in Agriculture*, vol. 164, p. 104932, 2019.

[7] S. Arivazhagan and S. V. Ligi, "Mango leaf diseases identification using convolutional neural network," *International Journal of Pure and Applied Mathematics*, vol. 120, no. 6, pp. 11 067–11 079, 2018.

[8] A. Rajbongshi, T. Khan, M. Pramanik, S. M. Tanvir, and N. R. C. Siddiquee, "Recognition of mango leaf disease using convolutional neural network models: a transfer learning approach," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 3, pp. 1681–1688, 2021.

[9] S. I. Ahmed, M. Ibrahim, M. Nadim, M. M. Rahman, M. M. Shejunti, T. Jabid, and M. S. Ali, "Mangoleafbd: A comprehensive image dataset to classify diseased and healthy mango leaves," *Data in Brief*, vol. 47, p. 108941, 2023.

[10] M. R. Mia, S. Roy, S. K. Das, and M. A. Rahman, "Mango leaf disease recognition using neural network and support vector machine," *Iran Journal of Computer Science*, vol. 3, no. 3, pp. 185–193, 2020.

[11] S. Ali, M. Ibrahim, S. I. Ahmed, M. Nadim, M. R. Mizanur, M. M. Shejunti, and T. Jabid, "Mangoleafbd dataset," Mendeley Data, V1, 2022. [Online]. Available: https://doi.org/10.17632/hxsnvwty3r.1

[12] M. Wang, S. Lu, D. Zhu, J. Lin, and Z. Wang, "A high-speed and low-complexity architecture for softmax function in deep learning," in *2018 IEEE asia pacific conference on circuits and systems (APCCAS)*. IEEE, 2018, pp. 223–226.

[13] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Advances in neural information processing systems*, vol. 31, 2018.

[14] M. F. A. Hady and F. Schwenker, "Semi-supervised learning," *Handbook on neural information processing*, pp. 215–239, 2013.

[15] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function," 2020.

[16] 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," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[17] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.