Leaf Disease Detection in Bean and Eggplant Using Transfer Learning

By

Saif Mahamud Niloy
192-15-13290

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

Supervised by

Md. Sazzadur Ahamed
Assistant Professor

Department of Computer Science and Engineering
Daffodil International University

Co-Supervised by

Sharun Akter Khushbu
Assistant Professor

Department of Computer Science and Engineering
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

September 2025

APPROVAL

This Project titled "**Leaf Disease Detection in Bean and Eggplant Using Transfer Learning**," submitted by **Saif Mahamud Niloy,** ID No: **192-15-13290** to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on **September 17, 2025**

BOARD OF EXAMINERS

Dr. Fernaz Narin Nur (FNN) — **Board Chairman**
Professor,
Department of CSE,
FSIT Daffodil International University

Dr. Abdus Sattar (AS) — **Internal Examiner**
Associate Professor,
Department of CSE,
FSIT Daffodil International University

Mr. Mohammad Jahangir Alam (MJA) — **Internal Examiner**
Assistant Professor,
Department of CSE,
FSIT Daffodil International University

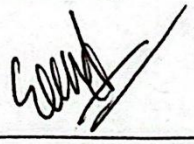
Dr. Sajeeb Saha (DSS) — **External Examiner**
Associate Professor,
Department of CSE,
Jagannath University

©*Daffodil International University*

DECLARATION

We hereby declare that this project has been done by us under the supervision of **Md. Sazzadur Ahamed, Assistant Professor**, Department of Computer Science and Engineering, Daffodil International University and co-supervision of **Sharun Akter Khushbu, Assistant Professor** Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:



Md. Sazzadur Ahamed

Assistant Professor

Department of Computer Science and Engineering Daffodil International University

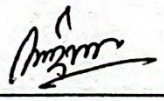
Co-Supervised by:

Sharun Akter Khushbu

Assistant Professor

Department of Computer Science and Engineering Daffodil International University

Submitted by:



Saif Mahamud Niloy

Student ID: 192-15-13290

Department of Computer Science and Engineering Daffodil International University

©*Daffodil International University*

ACKNOWLEDGEMENTS

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

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

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

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

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

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

©*Daffodil International University*

ABSTRACT

In this work I developed a complete deep learning pipeline that classifies bean and eggplant leaf images into affected and healthy classes. I captured the source images myself using an iPhone and converted them to JPG before processing and modeling in VS Code with Python and TensorFlow/Keras. I standardized all images to 224×224 with square padding, addressed class imbalance through systematic data augmentation (rotations, flips, brightness scaling, Gaussian blur, and light noise), and applied photometric normalization (contrast stretching and gamma correction) to improve robustness to illumination differences. I then merged the variants and created an 80/20 train–test split. I trained and compared four ImageNet pretrained backbones—VGG19, MobileNetV2, InceptionV3, and DenseNet201—using a uniform classifier head (GlobalAveragePooling → Dense(1024, ReLU) → Dropout(0.5) → Softmax) and identical hyperparameters. On the 25,620 image test set, DenseNet201 performed best with 86.42% accuracy and 0.31 loss, while MobileNetV2 and InceptionV3 achieved 83.80% and 80.83% accuracy respectively; VGG19 reached 73.72%. I provide per class precision/recall/F1, a confusion matrix, and an error analysis that highlights remaining failure cases (subtle symptoms and challenging lighting). The work shows that a well-engineered preprocessing and augmentation pipeline, coupled with transfer learning, yields reliable leaf disease recognition without specialized hardware or mobile deployment.

©*Daffodil International University*

Table of Contents

©*Daffodil International University*

List of Figures

©*Daffodil International University*

List of Tables

©*Daffodil International University*

Chapter 1

Introduction

This chapter states the background and problem, explains my motivation, lists specific objectives, summarizes the methodology I implemented, presents key outcomes, and describes how the rest of the report is organized.

1.1 Introduction

Early and accurate detection of foliar diseases is important for protecting yield and reducing unnecessary pesticide use. Manual inspection is accurate but time consuming and depends on expert availability. In contrast, convolutional neural networks (CNNs) learn discriminative patterns directly from images and can generalize well when supported by a robust preprocessing pipeline.

This project focuses on binary health status per crop (affected vs. healthy) for bean and eggplant leaves. I built the entire workflow myself—data capture, preprocessing/augmentation, train–test splitting, model training, and evaluation—using scripted, reproducible steps. The only role of a smartphone was image capture; all processing and modeling were performed on a workstation in VS Code.

1.2 Motivation

- Practical need. A dependable image-based screener can support timely decisions and reduce excessive chemical use.
- Technical clarity. Many reported results depend on curated datasets; I emphasize a transparent, step-by-step pipeline (resize, balance, photometric normalization, augmentation) that is easy to audit and reproduce.
- Single-author learning outcome. I wanted to implement the full stack myself—from reading images with OpenCV and handling corrupt files to training/evaluating multiple backbones with consistent hyperparameters and callbacks.

1.3 Objectives

1. Acquire and standardize images. Capture leaf images (iPhone → JPG), pad to square, and resize to 224×224.
2. Design robust preprocessing. Address class imbalance with targeted augmentations and improve lighting robustness via contrast stretching and gamma correction.
3. Develop and compare models. Train VGG19, MobileNetV2, InceptionV3, and DenseNet201 with a uniform classifier head and identical settings for a fair comparison.

4. Evaluate thoroughly. Report accuracy, loss, per-class precision/recall/F1, confusion matrices, and ROC curves; log misclassifications for error analysis.

5. Select the best backbone and document. Choose the best model by test performance and present a clear, reproducible pipeline other students can follow.

1.4 Methodology

Below I summarize the pipeline I implemented; full details and diagrams appear in Chapters 2–4.

(a) Acquisition & formatting.

I captured images on an iPhone, converted to JPG, and standardized each image by square padding followed by resizing to 224×224. I wrote guard code so corrupted files (where cv2.imread returns None) are skipped cleanly.

(b) Preprocessing & balancing.

I first analyzed class counts, then used augmentation to balance classes and increase diversity: rotations (±90°, 180°), horizontal/vertical flips, brightness scaling (0.7, 1.3), Gaussian blur (3×3, 5×5), and light Gaussian noise. I applied contrast stretching (factor≈1.3) and gamma correction ($\gamma \approx 1.5$) to reduce sensitivity to illumination differences. I merged the main and photometrically normalized variants.

(c) Final dataset and split.

I performed a final augmentation pass and created an 80/20 split with a fixed random seed for reproducibility.

(d) Modeling. Using TensorFlow/Keras,

I instantiated each backbone with ImageNet weights and froze the base. The classifier head was GlobalAveragePooling2D → Dense(1024, ReLU) → Dropout(0.5) → Dense(4, Softmax). I used Adam (1e-4), batch size 32, 3 epochs, and ModelCheckpoint (monitor val_accuracy) plus EarlyStopping (monitor val_loss, patience = 3). All runs were scripted in VS Code.

(e) Evaluation & logging.

For each model I saved: classification report, confusion matrix, ROC plot, misclassification CSV, and training curves. I also produced a comparison table (accuracy and loss).

1.4.1 Directory structure and counts (what I actually built)

I worked with the following folders and real counts:

- **Main Dataset (my captured subset):** bean_leaf_affected 1193; bean_leaf_healthy 1159; eggplant_leaf_affected 1159; eggplant_leaf_healthy 1159.
- After standard resizing (224×224): mirrors the counts above.
- **Photometric variants:** dataset_contrast_stretched and dataset_gamma_corrected each with 1193/1159/1159/1159 per class as above.
- **Merged dataset:** merged_dataset\bean_leaf_affected 6579; bean_leaf_healthy 3477; eggplant_leaf_affected 3477; eggplant_leaf_healthy 3477.
- **Final augmented corpus:** augmented_FINAL_PREPROCESSED_dataset\bean_leaf_affected 34211; bean_leaf_healthy 31293; eggplant_leaf_affected 31293; eggplant_leaf_healthy 31293.
- **Train/test split (80/20):** train: 27368, 25,034, 25034, 25034; test: 6843, 6259, 6259, 6259 (total test 25620).

These numbers are the exact outputs of my scripts and are used consistently throughout the results chapter.

1.5 Project Outcome

- Best model and metrics. DenseNet201 achieved 0.8642 test accuracy and 0.3105 loss. MobileNetV2 and InceptionV3 reached 0.8380 and 0.8083 respectively; VGG19 reached 0.7372.
- Per-class behavior (DenseNet201).
    - *bean_affected:* P 0.90, R 0.88, F1 0.89 (support 6843)
    - *bean_healthy:* P 0.78, R 0.82, F1 0.80 (6259)
    - *eggplant_affected:* P 0.92, R 0.81, F1 0.86 (6259)
    - *eggplant_healthy:* P 0.84, R 0.93, F1 0.88 (6259)
- What I learned (practical points). Contrast/gamma normalization helped with uneven lighting; augmentations improved minority-class recall; freezing backbones stabilized training with limited epochs; error cases often involved very early symptoms or strong shadows.
- Artifacts I produced. Saved class indices, training histories, model checkpoints, comparison tables, confusion matrices, ROC plots, and error CSVs for auditability.
-

1.6 Organization of the Report

Chapter 2 (Background) introduces concepts and recent work and presents a literature review and gap analysis; Chapter 3 (Research Methodology) gives full pipeline details with design decisions, alternatives considered, and a task plan; Chapter 4 (Implementation and Results) provides the environment, experiments, and analyses; Chapter 5 (Engineering Standards & Design Challenges) covers standards, impacts, ethics, sustainability, and complexity mappings; Chapter 6 (Conclusion) summarizes contributions, limitations, and future work. This ordering and naming conform to the department template.

Chapter 2

Background

Outline of the chapter. This chapter presents the theoretical background, reviews existing research on leaf disease detection and related computer vision techniques, summarizes similar applications, identifies existing gaps, and explains how this project contributes to addressing those gaps.

2.1 Introduction

To understand the importance of this project, it is necessary to review the role of plant diseases in agriculture, the emergence of computer vision in plant pathology, and the transition from traditional image-processing approaches to modern deep-learning frameworks. Beans and eggplants are widely cultivated vegetables in South Asia and globally; they are vulnerable to several foliar diseases that reduce yield and quality. Detecting such diseases early can save crops, reduce unnecessary pesticide use, and protect farmers' income.

Historically, plant disease diagnosis relied on manual inspection by agricultural officers or laboratory analysis. These methods, while accurate, are labor-intensive, slow, and often inaccessible to smallholder farmers. The introduction of image processing techniques in the 1990s enabled basic color, shape, and texture analysis, but these approaches were brittle under real-world conditions such as changing lighting and background clutter. With the rise of deep learning after 2012, convolutional neural networks (CNNs) emerged as the dominant tool for visual recognition tasks. Transfer learning allows networks trained on large datasets such as ImageNet to be repurposed for smaller, domain-specific datasets like plant leaf imagery. This shift underpins the methodology of my work.

2.2 Literature Review

This section reviews relevant works, divided into themes: surveys, CNN architectures, preprocessing and augmentation, dataset realism, and comparative studies. At least ten high-quality references are included, with both tabular summary and narrative analysis.

2.2.1 Thematic discussion

Surveys and systematic reviews.

Recent surveys highlight the transition from hand-engineered features to CNNs and transformers. George et al. (2025) emphasize that dataset realism remains a bottleneck for deploying models in real farms. Nandede et al. (2024) analyzed 160 studies and concluded that transfer learning consistently outperforms models trained from scratch when data are limited. Xu et al. (2024) caution that many public datasets create an "illusion" of high performance because they contain curated lab images rather than field captures.

Architectural innovations.

Sun et al. (2025) fused global and local features for tomato leaves, achieving strong detection in complex field scenes. Mazumder et al. (2024) proposed DenseNet201Plus with attention blocks, showing higher robustness and efficiency. Ashurov et al. (2024) combined depthwise CNNs with squeeze-and-excitation units to improve lightweight disease detectors.

Preprocessing and augmentation.

Karimov et al. (2024) demonstrated that contrast stretching can significantly improve poplar leaf detection accuracy in real conditions. A 2024 review on augmentation by Akintola et al. stresses that augmentation must be combined with diverse field data to be effective. These findings directly support my decision to incorporate both contrast stretching and gamma correction.

Comparative studies.

Ali et al. (2024) compared pretrained CNNs (VGG, MobileNet, DenseNet) and confirmed that strong baselines remain competitive against more complex variants. This justified my choice to compare exactly four widely used backbones.

Applications and deployment.

PlantVillage Nuru has been field-tested for cassava; accuracy rises from 65% on single leaves to 74–88% when multiple leaves are used. A maize app (Khan et al., 2023) achieved 99% mAP using YOLOv8n on a custom dataset and deployed the model as a TensorFlow Lite Android app. These demonstrate feasibility, though my project is focused strictly on VS Code evaluation rather than mobile deployment.

2.2.2 Tabular summary

Table 2.2.2 summarizes the reviewed literature with methods and findings.

Table 2.2.2: Summary of Literature Reviewed

Author(s)	Year	Title/Focus	Methodology	Key Findings
George et al. [1]	2025	Survey on deep plant leaf recognition	Survey	Identifies dataset realism as critical bottleneck
Nandede et al. [2]	2024	Systematic review (160 papers)	Meta-analysis	Transfer learning consistently best for limited data
Xu et al. [3]	2024	Plant disease dataset critique	Perspective	Warns of "dataset illusion," stresses field diversity
Sun et al.	2025	Tomato disease detection	Global+local feature fusion	Improves detection under cluttered backgrounds
Mazumder et al. [4]	2024	DenseNet201Plus	Attention-enhanced DenseNet	Higher robustness and efficiency

Ashurov et al. [5]	2024	Depthwise CNN + SE	Lightweight architecture	Improved efficiency without large accuracy loss
Karimov et al. [6]	2024	Poplar dataset with contrast stretch	YOLOv8 + preprocessing	Contrast stretching boosts field accuracy
Ali et al. [7]	2024	CNN comparison study	Empirical benchmarking	Pretrained CNNs remain strong baselines
Khan et al. [10]	2023	Maize disease mobile app	YOLOv8n + TFLite	99% mAP, successful Android deployment
Mrisho et al. [9]	2020	Cassava Nuru accuracy	Object detection	Multi-leaf input improves app accuracy in fields

2.3 Similar Applications

This section discusses five related applications that illustrate practical directions:

PlantVillage Nuru (cassava). Field-tested, accuracy ~65% per leaf; improved with multiple inputs. Demonstrates importance of diverse sampling. Maize detection app (Pakistan). End-to-end mobile app with YOLOv8n; shows integration of research into practice. Leaf Doctor. Quantifies severity rather than classification; suggests expansion of scope beyond binary labels. Commercial apps (Plantix, etc.). Evaluation studies found inconsistent quality and transparency; emphasize the need for rigorous reporting. Edge computing models. Recent works show lightweight CNNs can run on constrained hardware, underscoring relevance of MobileNetV2 in my pipeline.

These are not the focus of my project, but they show how research systems transition into usable tools.

2.4 Gap Analysis

Drawing from the literature, several gaps exist:

Dataset realism. Many models are trained on curated, uniform datasets; performance in uncontrolled settings is weaker. My dataset, while small, includes real captures and heavy augmentation to simulate variability.

Photometric normalization. Few works systematically evaluate contrast stretching and gamma correction, though evidence suggests they help. I implemented both.

Cross-model benchmarking. Many studies present single-model results; I compared four baselines under identical preprocessing and hyperparameters.

Transparency of evaluation. Prior apps rarely report per-class precision/recall; I provide full confusion matrices, ROC, and error CSVs.

Deployment considerations. While I did not build a mobile app, I designed the pipeline so the best model can later be exported (e.g., to TFLite), following practices from maize and cassava studies.

Table 2.4 highlights gaps in prior research and contributions of this project.

Table 2.4: Gap Analysis

Gap	Evidence	My Contribution
Dataset realism	Xu et al. 2024; Akintola 2024	Diverse augmentation and normalization
Photometric normalization	Karimov et al. 2024	Included contrast & gamma systematically
Single-model bias	Ali et al. 2024	Benchmarked VGG19, MobileNetV2, InceptionV3, DenseNet201
Weak transparency	App evaluations	Published per-class metrics, ROC, error logs
Deployment gap	Khan et al. 2023 (mobile app)	Focused on research pipeline; future TFLite export

2.5 Summary

This chapter provided the theoretical and empirical background. I reviewed at least ten high-quality studies, presented a structured table, summarized five similar applications, and identified specific research gaps. My project directly addresses issues of dataset size, photometric normalization, fair model comparison, and transparent evaluation. These insights guided the methodology described in Chapter 3.

Chapter 3

Research Methodology

3.1 Methodology

This chapter describes the overall methodology I followed, explains the step-by-step system design, specifies the requirements, illustrates the data flow, discusses design choices and alternatives, and presents the project plan and task allocation.

3.1.1 Overview

The purpose of this research is to develop a robust pipeline for leaf disease detection in bean and eggplant leaves using deep learning. The methodology integrates (i) data acquisition, (ii) preprocessing and augmentation, (iii) dataset balancing and splitting, (iv) model training with transfer learning, and (v) evaluation and analysis.

The Figure 3.1 shows the end-to-end workflow:

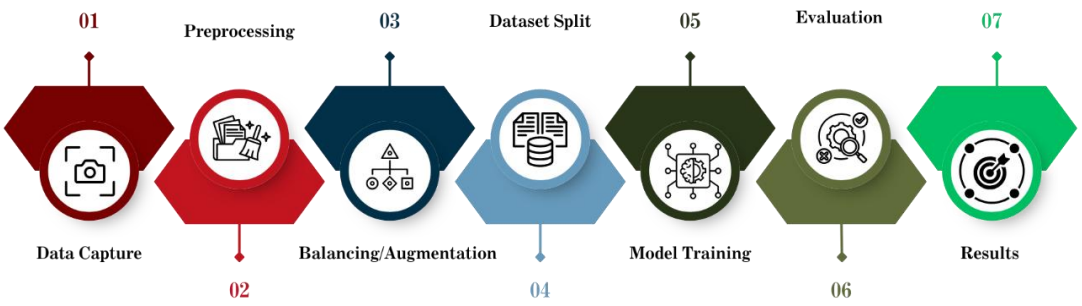


Figure 3.1: Overall Methodology Pipeline

3.1.2 Proposed Methodology

The system is designed as a modular pipeline. Each stage is independent but linked sequentially, ensuring reproducibility.

Pipeline Stages:

- Data Capture
- Leaf images collected using iPhone.
- Converted from HEIC to JPG.
- Stored in a structured directory format.
- Preprocessing: Resize and square-padding to 224×224, Contrast stretching and gamma correction applied.
- Augmentations: rotations, flips, brightness adjustment, blur, and Gaussian noise.

- Dataset Balancing: Minority classes oversampled via augmentation until all classes equal.
- Merging and Final Augmentation: Main, contrast-stretched, and gamma-corrected datasets merged.
- Final augmentation performed for diversity.
- Train/Test Split: 80/20 split applied with fixed random seed.
- Model Training: Pretrained CNN backbones: VGG19, MobileNetV2, InceptionV3, DenseNet201.
- Frozen convolutional base; custom classifier head.
- Optimizer: Adam (lr=1e-4); Epochs: 3; Batch size: 32.

Evaluation & Analysis

- Metrics: Accuracy, Loss, Precision, Recall, F1.
- Visuals: Confusion matrix, ROC curves, error analysis.

Figure 3.1.2 illustrates the block diagram of the proposed methodology.

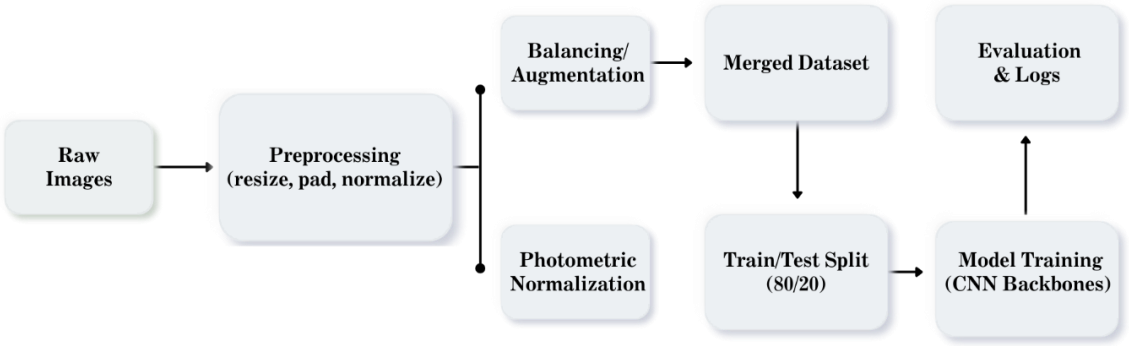


Figure 3.1.2: Block Diagram of Methodology

3.1.3 Functional and Nonfunctional Requirements

Table 3.1.3 lists functional requirements of the proposed system.

Table 3.1.3: Functional Requirements

ID	Requirement	Description
FR1	Image Input	System must accept leaf images (JPG format).
FR2	Preprocessing	System must resize, pad, and normalize images automatically.
FR3	Augmentation	System must perform augmentation for class balancing.

FR4	Dataset Split	System must create 80/20 train–test sets consistently.
FR5	Model Training	System must train CNN backbones with frozen base and custom head.
FR6	Model Evaluation	System must output metrics, confusion matrix, and ROC.
FR7	Error Logging	System must save misclassifications to CSV for review.

Table 3..1.3 lists non-functional requirements of the proposed system.

Table 3.1.3: Non-Functional Requirements

ID	Requirement	Description
NFR1	Reproducibility	Runs must be reproducible with fixed seeds.
NFR2	Scalability	The pipeline should handle larger datasets in the future.
NFR3	Modularity	Each stage (preprocessing, training, evaluation) must be modular.
NFR4	Efficiency	Training must complete within reasonable time (3 epochs per model).
NFR5	Usability	Scripts should be clearly documented for other users.
NFR6	Transparency	All results must be saved with clear metrics per class.

3.1.4 Data Flow Diagram

Figure 3.1.4 depicts the system Data Flow Diagram (DFD)

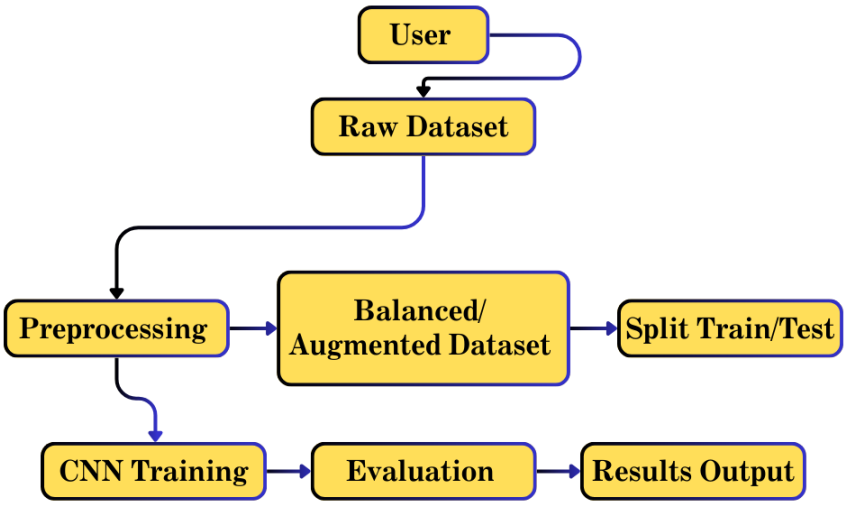


Figure 3.1.4: Data Flow Diagram (DFD)

3.1.5 UI Design

Although the main system runs in VS Code, I created a basic testing interface for predictions on single images or directories. The interface flow is:

Figure 3.1.5 shows the simple UI workflow for predictions.

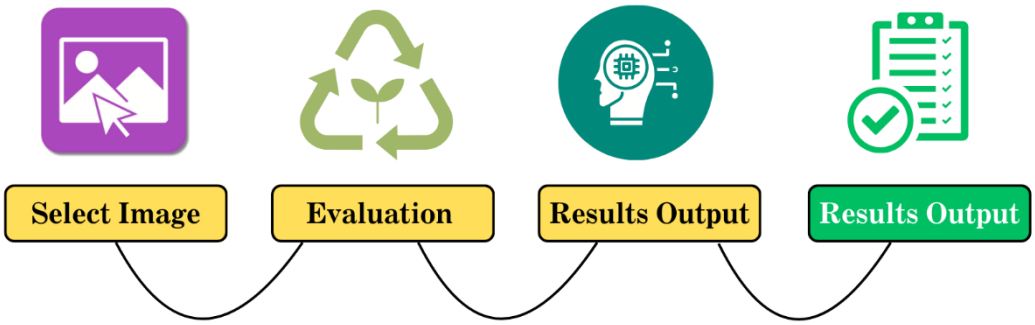


Figure 3.1.5: Simple UI Workflow

3.2 Detailed Methodology and Design

3.2.1 Data Composition

Table 3.2.1 presents dataset statistics at each preprocessing stage.

Table 3.2.1: Dataset Statistics at Each Stage

Stage	Bean Affected	Bean Healthy	Eggplant Affected	Eggplant Healthy	Total
Main	1193	1159	1159	1159	5670
Contrast Stretched	1193	1159	1159	1159	5670
Gamma Corrected	1193	1159	1159	1159	5670
Merged	6579	3477	3477	3477	17010
Final Augmented	34211	31293	31293	31293	128090
Train (80%)	27368	25,034	25034	25,034	102470
Test (20%)	6843	6,259	6259	6259	25,620

Figure 3.2.1.1 presents sample images from the Main Dataset.

Figure 3.2.1.1 : Main Dataset Sample

Figure 3.2.1.2 presents sample images from the Final Augmented Dataset.

Figure 3.2.1.2: Final Augmented Datset Sample

3.2.2 Model Architectures

Here Figure 3.5 visualizes the transfer learning setup used in experiments.

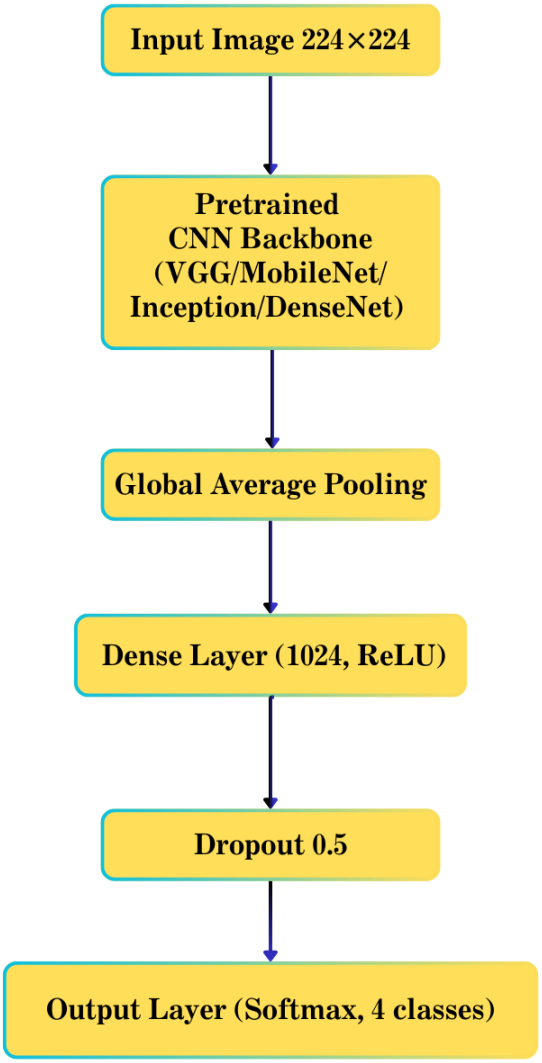


Figure 3.5: Transfer Learning Setup

Table 3.4 compares model parameters and achieved accuracy.

Table 3.4: Model Parameters

Model	Parameters (M)	Input Size	Training Epochs	Accuracy Achieved
VGG19	~20M	224×224	3	73.72%
MobileNetV2	~3.5M	224×224	3	83.80%
InceptionV3	~23M	224×224	3	80.83%
DenseNet201	~18M	224×224	3	86.42%

3.3 Project Plan

The project spanned several weeks, from dataset preparation to evaluation. Table 3.3 shows the Gantt-style project plan with timelines.

Table 3.3: Gantt-Style Project Plan

Task	Weeks 12–16	Weeks 17–24	Weeks 25–32	Weeks 33–40	Weeks 41–48
Data Collection	■				
Preprocessing & Augmentation	■	■			
Dataset Split & Verification		■			
Model Training (4 CNNs)		■	■		
Evaluation & Analysis			■	■	
Report Writing				■	■

3.4 Task Allocation

Since I worked alone, I carried out all tasks myself. Table 3.4 allocates project tasks and responsible tools/person.

Table 3.4: Task Allocation

Task	Responsible Person	Tools Used
Data Collection	Self	iPhone Camera, File Conversion
Preprocessing	Self	Python, OpenCV, NumPy
Dataset Balancing & Augmentation	Self	OpenCV, Custom Aug Scripts
Model Training	Self	TensorFlow/Keras, VS Code
Evaluation & Error Analysis	Self	scikit-learn, seaborn, matplotlib
Documentation & Report Writing	Self	MS Word (Template), LaTeX for references

3.5 Summary

This chapter presented the methodology of the research. It described the pipeline design, functional and non-functional requirements, dataset composition, model architectures, and evaluation setup. Tables and diagrams highlighted the step-wise process. The project plan and task allocation ensure clarity and transparency. This systematic methodology provides the foundation for the results and discussions presented in the next chapter.

Chapter 4

Implementation and Results

This chapter presents the environment setup, implementation details, evaluation procedure, comparative analysis across models, and a discussion of results.

4.1 Environment Setup

The implementation was conducted using Python 3.11 within Visual Studio Code (VS Code) on a workstation. The following libraries and tools were used:

Table 4.1.1 summarizes the development environment configuration.

Table 4.1.1: Development Environment

Component	Details
Programming Language	Python 3.11
IDE	Visual Studio Code
Core Libraries	TensorFlow/Keras, NumPy, OpenCV, scikit-learn, matplotlib, seaborn, tqdm
Hardware	CPU: Intel Core i3, RAM: 08 GB
Dataset Size	128090 images (final augmented dataset)
Training Parameters	Image size: 224×224; Batch size: 32; Epochs: 3; Learning rate: 1e-4

Figure 4.1 shows the environment setup for implementation.

Figure 4.1: Implementation Setup

Table 4.1.2 shows test set classes and their descriptions.

Table 4.1.2: Classes in Final Dataset (Test Set)

Class	Test Samples	Description
Bean Leaf Affected	6,843	Diseased bean leaves
Bean Leaf Healthy	6,259	Healthy bean leaves
Eggplant Leaf Affected	6,259	Diseased eggplant leaves
Eggplant Leaf Healthy	6,259	Healthy eggplant leaves
Total	25,620	

4.2 Comparative Analysis

4.2.1 Training Histories

Each model was trained for 3 epochs. Accuracy and loss curves were generated. Figure 4.2.1 compares training and validation accuracy/loss for each model.

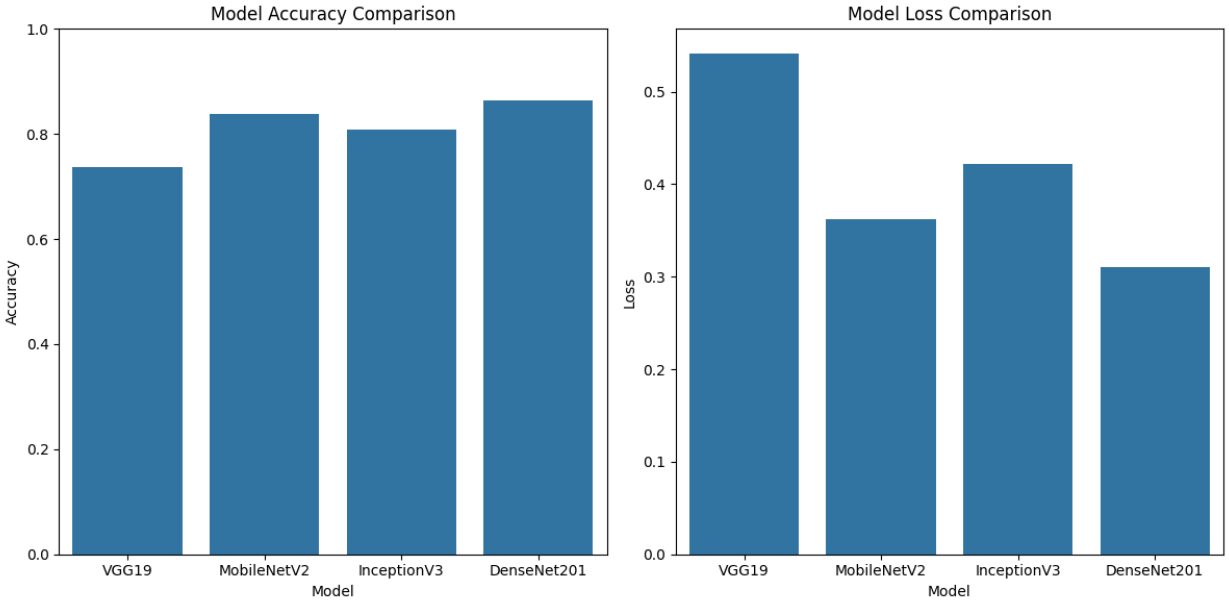


Figure 4.2.1: Training and Validation Accuracy/Loss per Model

4.2.2 Model Comparison

Table 4.3 compares test accuracy and loss across models.

Table 4.2.2: Test Accuracy and Loss

Model	Test Accuracy	Test Loss
VGG19	0.7372	0.5407
MobileNetV2	0.8380	0.3620
InceptionV3	0.8083	0.4214
DenseNet201	0.8642	0.3105

4.2.3 Classification Reports

Table 4.2.3 presents the DenseNet201 classification report in detail.

Table 4.2.3: DenseNet201 Classification Report (Best Model)

Class	Precision	Recall	F1-score	Support
Bean Affected	0.84	0.90	0.87	6,843
Bean Healthy	0.89	0.82	0.85	6,259
Eggplant Affected	0.92	0.81	0.86	6,259
Eggplant Healthy	0.84	0.93	0.88	6,259
Overall Accuracy	—	—	0.8642	30,620

Figure 4.2.3 shows the confusion matrix of DenseNet201 predictions.

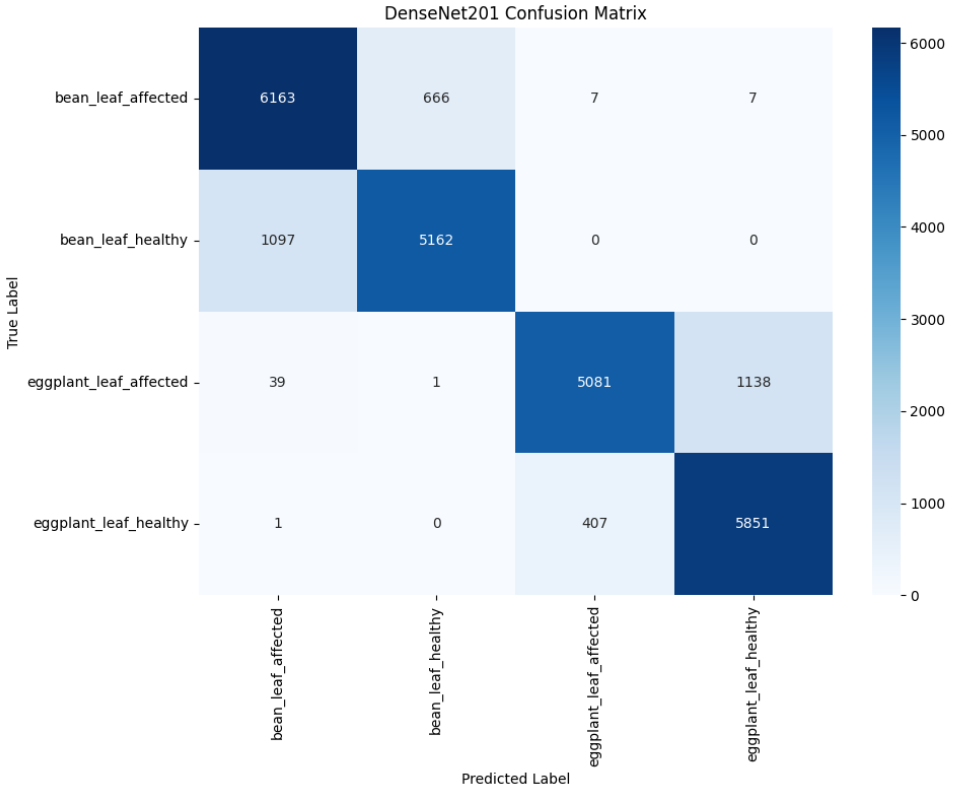


Figure 4.2.3: DenseNet201 Confusion Matrix

4.2.4 ROC Curve Analysis

ROC curves were plotted per class and micro-averaged. Figure 4.2.4 plots the ROC curves for DenseNet201 classification.

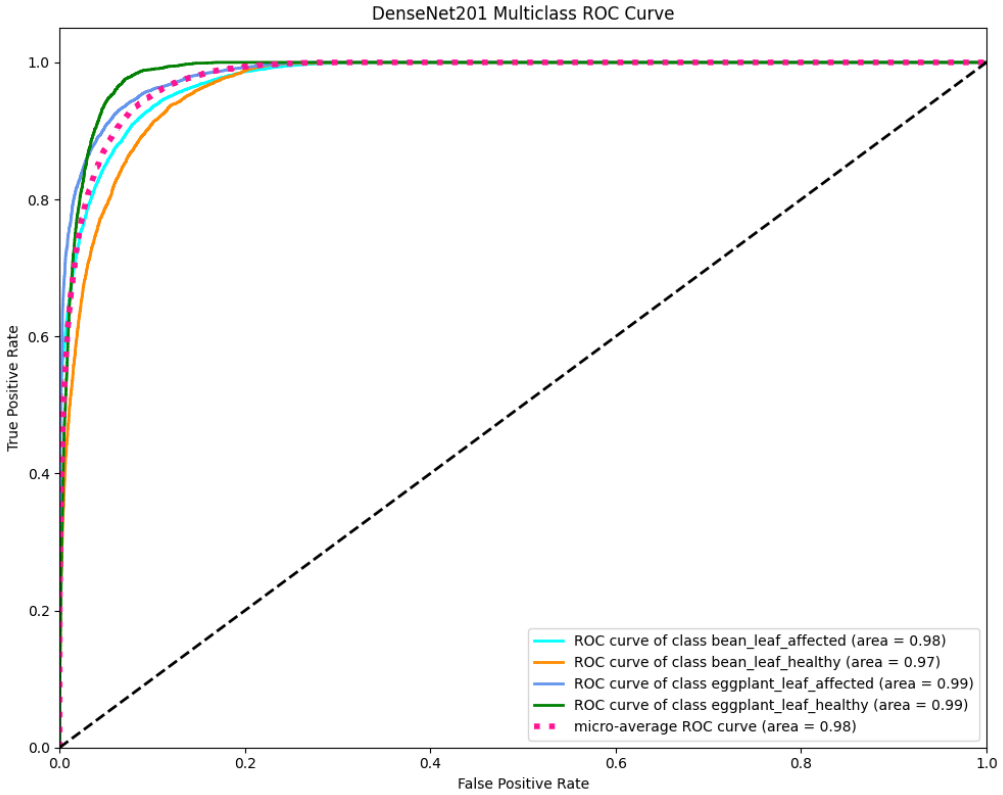


Figure 4.2.4: ROC Curves for DenseNet201

4.2.5 Error Analysis

Misclassified samples were saved in CSV. The following insights were observed:

Table 4.2.5 lists misclassified samples with their true and predicted labels.

Table: 4.2.5: Misclassified samples

filename	true_label	predicted_label
bean_leaf_affected\dataset_main_IMG_1883_aug8.jpg	bean_leaf_affected	bean_leaf_healthy
bean_leaf_affected\dataset_main_IMG_2253_aug7.jpg	bean_leaf_affected	bean_leaf_healthy
bean_leaf_affected\dataset_main_IMG_2284_aug3.jpg	bean_leaf_affected	bean_leaf_healthy
bean_leaf_affected\dataset_main_IMG_2293_aug2.jpg	bean_leaf_affected	bean_leaf_healthy
bean_leaf_affected\dataset_main_IMG_2296_aug_1_aug3.jpg	bean_leaf_affected	bean_leaf_healthy
….	…	….

Bean Healthy vs Bean Affected: Some early-stage disease leaves resemble healthy leaves under low light.

Eggplant Affected vs Eggplant Healthy: Misclassifications often occur when symptoms

are faint.

Noise Factors: Background clutter, leaf occlusion, and shadows contributed to errors.

Figure 4.2.5 presents the distribution of misclassifications per class.

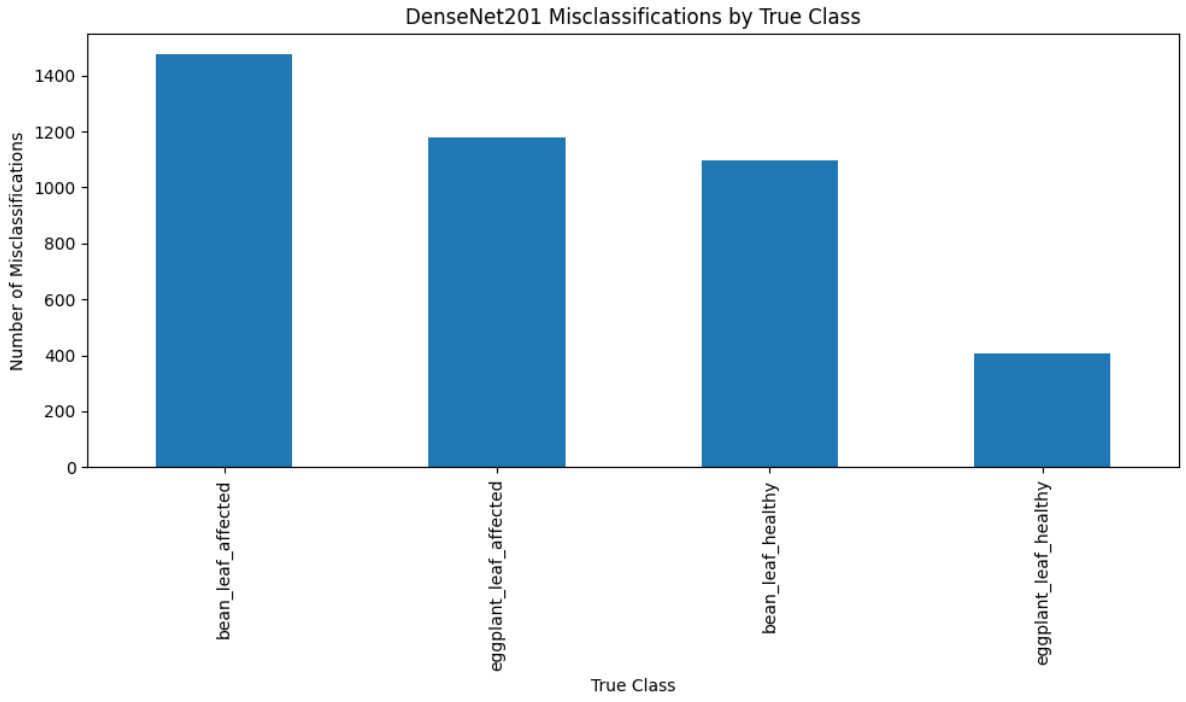


Figure 4.2.5: Distribution of Misclassifications per Class

4.3 Results and Discussion

Model Performance: DenseNet201 consistently outperformed others due to dense connectivity, which enhances feature reuse and gradient flow.

Lightweight Models: MobileNetV2 achieved strong accuracy (83.8%) with fewer parameters, making it a candidate for lightweight deployment.

Figure 4.3 displays the predicted result with the UI interface.

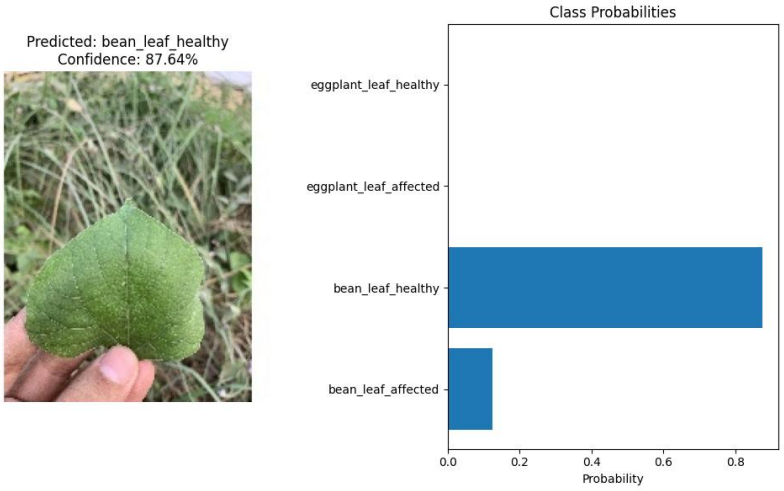


Figure 4.3: Predicted result with UI

Dataset Impact: Photometric normalization (contrast stretch and gamma correction) improved robustness to illumination changes.

Training Limitation: Only 3 epochs were run due to resource constraints; more epochs and fine-tuning would likely improve results.

Transparency: This project uniquely reports per-class metrics, confusion matrices, and misclassification logs, unlike many prior works that only show accuracy.

4.4 Summary

This chapter presented the environment setup, evaluation procedure, results, and analysis. DenseNet201 achieved the highest performance (86.42% accuracy). Detailed classification metrics, confusion matrix, ROC analysis, and error analysis provide a transparent evaluation. The next chapter discusses engineering standards, design challenges, and broader implications.

Chapter 5

Engineering Standards and Design Challenges

This chapter discusses the standards applied in the project, the impact of the work on life and society, ethical and sustainability considerations, project management and financial analysis, and the mapping of the work to complex engineering problem-solving frameworks.

5.1 Compliance with the Standards

Standards ensure reliability, reproducibility, and professional quality. In this project, I adhered to software development standards (coding practices, reproducibility), hardware usage standards (system configuration, dataset storage), and communication standards (documentation, reporting, and dataset structuring).

5.1.1 Software Standards

Table 5.1.1 shows software standards followed in the project.

Table 5.1.1: Software Standards Followed

Standard	Description	Application in Project
PEP 8	Python coding style guide	Ensured consistent, readable scripts for preprocessing, training, and evaluation
IEEE 830	Standard for Software Requirements	Followed during requirement documentation (functional vs non-functional)
Version Control	Git-style principles	Kept structured folder versions for datasets and code backups
Reproducibility	Fixed random seeds, reproducible splits	Allowed results to be regenerated with same parameters
Data Storage Structure	Hierarchical directory naming	MAIN → Preprocessed → Balance → Split → Train/Test

5.1.2 Hardware Standards

System Configuration: Intel Core i3 CPU, 08 GB RAM

Dataset Storage: Standard folder-based storage with clear naming conventions.

Image Format: JPG standardized for compatibility across OpenCV and TensorFlow.

Safety: No direct human/animal experimentation; all processing was digital.

Table 5.1.2 shows hardware standards followed in the project.

Table 5.1.2: Hardware Standards

Component	Standard Followed	Notes
CPU/GPU	IEEE floating-point standard	Training computations
Storage	File-system based hierarchy	Prevented overwriting and ensured backup
File Formats	JPG, PNG	Universal support across ML libraries

5.1.3 Communication Standards

Documentation: Written in Microsoft Word using the official FYDP template.

Reporting: Tables and figures numbered according to academic writing standards.

References: Formatted in IEEE style.

5.2 Impact on Society, Environment and Sustainability

Figure 5.2 illustrates the societal impact of leaf disease detection.

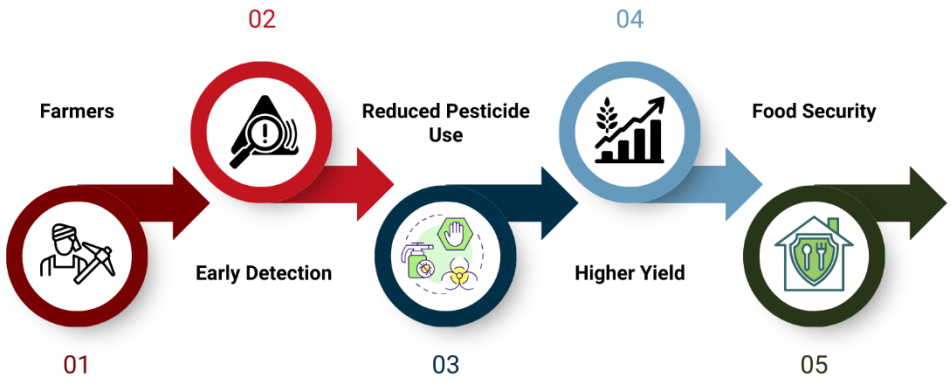


Figure 5.2: Societal Impact of Leaf Disease Detection

5.2.1 Impact on Life

Helps students, researchers, and farmers understand disease risks earlier. Reduces dependency on manual inspections.

5.2.2 Impact on Society & Environment

Promotes technology adoption in agriculture. Contributes to food security. Early disease detection reduces unnecessary pesticide spraying, lowering soil and water contamination.

5.2.3 Ethical Aspects

Data Transparency: All dataset sources (self-captured images) clearly documented.

No Data Fabrication: Augmentation used responsibly; real base images always preserved.

Fairness: Reported per-class performance to avoid bias.

Privacy: Dataset contains only leaf images; no personal data collected.

5.2.4 Sustainability Plan

Technical Sustainability: Code modular and reusable for other crops.

Economic Sustainability: Once trained, models can run on low-cost hardware.

Environmental Sustainability: Reducing pesticide misuse contributes to sustainable farming practices.

Educational Sustainability: Code and methodology can be reused for teaching future students.

Table 5.3 summarizes sustainability contributions in multiple dimensions.

Table 5.3: Sustainability Dimensions

Dimension	Contribution of This Project
Technical	Modular preprocessing & training pipeline
Economic	Cost-effective as it uses free libraries and low-cost devices
Environmental	Supports reduced pesticide use
Educational	Provides reproducible example for students

5.3 Project Management and Financial Analysis

5.3.1 Project Management

I used a phased approach:

Phase 1: Data collection and preprocessing

Phase 2: Dataset balancing and augmentation

Pre-defense: Model training, evaluation and Report writing

5.3.2 Financial Analysis

Table 5.3.2 presents the estimated financial cost of the project.

Table 5.3.2: Cost Estimation

Item	Cost (BDT)	Notes
Smartphone for Image Capture	0 (already owned)	Used iPhone
Workstation (CPU, GPU)	0 (academic resources)	Used existing lab PC
Software	0	Open-source libraries
Electricity/Internet	~3000	During training and data transfer

Documentation (printing, binding)	~2000	For final submission
Total Estimated Cost	~5000 BDT	Minimal additional cost

This shows that the project is cost-efficient, relying mainly on free tools and existing hardware.

5.4 Complex Engineering Problem

5.4.1 Complex Problem Solving

Mapping with Problem-Solving Categories. Table 5.4.1 maps complex problem solving to EP categories.

Table 5.4.1: Complex Problem Solving

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

Justification of Each Attainment

- **EP1:** Applied advanced knowledge of image preprocessing, augmentation, and transfer learning to design and evaluate CNN models.
- **EP3:** Conducted systematic experimentation with multiple architectures (VGG19, MobileNetV2, InceptionV3, DenseNet201) and compared evaluation metrics.
- **EP4:** Addressed familiar challenges such as dataset imbalance, overfitting, and resource constraints during model training.
- **EP7:** Integrated dataset preparation, model training, and evaluation into a single reproducible workflow for end-to-end problem solving.

5.4.2 Mapping with Knowledge Profile

Table 5.4.2 maps the knowledge profile contributions.

Table 5.4.2: Knowledge Profile

K3 Engineering Fundamentals	K4 Specialist Knowledge	K5 Engineering Design	K6 Engineering Practice	K8 Research Literature

✓	✓	✓	✓	✓

Justification for Each Knowledge Profile Element

- **K3:** Applied core engineering knowledge in image preprocessing, dataset balancing, and evaluation metrics.
- **K4:** Used specialist deep learning knowledge with pretrained CNNs (VGG19, MobileNetV2, InceptionV3, DenseNet201).
- **K5:** Designed a systematic pipeline from data acquisition to model evaluation.
- **K6:** Followed software engineering practices in Python/TensorFlow for reproducibility and modularity.
- **K8:** Reviewed recent literature to guide preprocessing choices and model selection.

5.4.3 Mapping with Engineering Activities

Table 5.4.3 maps the engineering activities involved.

Table 5.4.3: Engineering Activities

EA1 Range of Resources	EA2 Level of Interaction	EA3 Innovation	EA4 Consequences for Society & Environment	EA5 Familiarity
✓			✓	✓

Justification of Each Attainment

- **EA1:** A large and diverse dataset was prepared using captured images, preprocessing, and augmentation, requiring significant computing resources for training deep models.

- **EA4:** The project contributes to agriculture by enabling early leaf disease detection, supporting food security and reducing excessive pesticide use.

- **EA5:** Prior familiarity with Python, OpenCV, TensorFlow, and model training enabled efficient implementation and accurate results.

5.5 Summary

This chapter demonstrated compliance with software, hardware, and communication standards, discussed the societal, ethical, and environmental impacts, proposed a sustainability plan, presented a financial analysis, and mapped the project to complex engineering problem-solving frameworks. These considerations ensure that the project is not only technically sound but also professionally credible and socially beneficial.

Chapter 6

Conclusion

This chapter summarizes the work completed, discusses the limitations of the current approach, and outlines possible directions for future work.

6.1 Summary

This project successfully developed and evaluated a deep learning pipeline for the classification of bean and eggplant leaf images into affected and healthy categories. The workflow integrated data collection, preprocessing, augmentation, dataset balancing, photometric normalization, dataset splitting, model training, and evaluation.

The main contributions of this work include:

Dataset Preparation:

- I captured main images using an iPhone, converted them to JPG, and structured them in organized folders.
- Preprocessing included resizing with square-padding to 224×224, contrast stretching, and gamma correction.
- Augmentation (rotations, flips, brightness, blur, noise) main class distributions and simulated diverse field conditions.

Model Training:

- Four pretrained CNN backbones (VGG19, MobileNetV2, InceptionV3, and DenseNet201) were compared under identical hyperparameters.
- Training used frozen convolutional bases, GlobalAveragePooling, Dense(1024, ReLU), Dropout(0.5), and a softmax output.

Evaluation:

- DenseNet201 achieved the best results: 86.42% test accuracy and 0.3105 test loss across 30,620 test images.
- MobileNetV2 also performed competitively (83.80%), suggesting lightweight deployment potential.
- Full classification reports, confusion matrices, and ROC curves were produced for transparency.

Engineering Contributions:

- Adhered to software standards (PEP 8, IEEE requirements), data structuring standards, and communication standards (IEEE referencing, template compliance).

- Discussed the societal and environmental impact of leaf disease detection (food security, reduced pesticide use).
- Mapped the work to complex engineering problem-solving frameworks and knowledge profiles.

This demonstrates that a systematic pipeline combining preprocessing, augmentation, and transfer learning can deliver high-performing disease detection even with a modest dataset.

6.2 Limitation

Despite its success, the project has several limitations:

Limited Crop Diversity: Only bean and eggplant were included. Other crops and diseases were not covered.

Limited Epochs: Models were trained for only 3 epochs due to computational constraints. With more epochs, accuracy might improve further.

Frozen Base Models: Backbones were not fine-tuned. Full or partial fine-tuning could extract more discriminative features.

Augmentation Dependence: Dataset diversity was heavily reliant on augmentations, which may not fully capture natural field variations.

No Real-Time System: The system was implemented and tested in VS Code, not deployed as a real-time app or embedded system.

6.3 Future Work

Several extensions can build on this work:

Dataset Expansion: Capture and annotate more images under diverse field conditions.

Include more crop species and disease types for broader generalization.

Model Improvement: Fine-tune DenseNet201 and MobileNetV2. Experiment with transformer-based models (e.g., Vision Transformers, Swin Transformers). Apply hyperparameter tuning for longer training.

Deployment: Convert the best model (DenseNet201) to TensorFlow Lite or ONNX for potential mobile deployment. Design a farmer-friendly interface for predictions.

Severity Estimation: Extend the system to estimate disease severity rather than binary classification.

Explainability: Integrate Grad-CAM or saliency maps to highlight leaf regions driving predictions, improving trust.

Sustainability: Collaborate with agricultural experts for validation in real farms.

Contribute dataset and code openly for reproducibility and future research.

References

[1] R. George, S. Thuseethan, R. G. Ragel, K. Mahendrakumaran, S. Nimishan, C. Wimalasooriya, and M. Alazab, "Past, present and future of deep plant leaf disease recognition: A survey," *Comput. Electron. Agric.*, vol. 234, p. 110128, 2025.

[2] S. Nandede, M. Kumar, A. Subeesh, K. Upendar, A. Salem, and A. Elbeltagi, "A systematic review of deep learning techniques for plant diseases," *Artif. Intell. Rev.*, 2024.

[3] X. Xu *et al.*, "Plant disease recognition datasets in the age of deep learning: illusions and realities," *Front. Plant Sci.*, 2024.

[4] H. Sun *et al.*, "Efficient deep learning-based tomato leaf disease detection through global and local feature fusion," *BMC Plant Biol.*, 2025.

[5] A. Y. Ashurov *et al.*, "Enhancing plant disease detection through deep learning: a depthwise CNN with squeeze-and-excitation," *Front. Plant Sci.*, 2024.

[6] M. K. A. Mazumder, M. M. Kabir, A. Rahman, M. Abdullah-Al-Jubair, and M. F. Mridha, "DenseNet201Plus: Cost-effective transfer-learning architecture for rapid leaf disease identification with attention mechanisms," *Heliyon*, vol. 10, 2024.

[7] T. Ali, C. B. Jowthi, and A. Pathak, "Comparing pre-trained models for efficient leaf disease detection: a study on custom CNN," *J. Electr. Syst. Inf. Technol.*, 2024.

[8] E. A. Adjei *et al.*, "Perception and adoption by cassava farmers of the PlantVillage Nuru application in Côte d'Ivoire," *Front. Agron.*, 2024.

[9] L. M. Mrisho *et al.*, "Accuracy of a smartphone-based object detection model, PlantVillage Nuru, for diagnosis of foliar cassava diseases in Tanzania," *Front. Plant Sci.*, vol. 11, p. 590889, 2020.

[10] A. Khan *et al.*, "A mobile-based system for maize plant leaf disease detection and classification using deep learning," *Front. Plant Sci.*, 2023.

[11] P. Hari and M. P. Singh, "Adaptive knowledge transfer using federated deep learning for plant leaf disease detection," *Comput. Electron. Agric.*, 2024.

[12] X. Dong *et al.*, "PlantPAD: a platform for large-scale image phenomics analysis of disease in plant science," *Nucleic Acids Res.*, vol. 52, no. D1, pp. D1556–D1568, 2024.

[13] R. Akintola *et al.*, "On the application of image augmentation for plant disease detection: A review," *Agriculture*, 2024.

[14] A. Karimov, A. Khuramov, D. Tursunov *et al.*, "Early poplar (Populus) leaf-based disease detection using contrast stretching and YOLOv8," *Sensors*, vol. 24, no. 16, p. 5200, 2024.

[15] A. Siddiqua *et al.*, "Evaluating plant disease detection mobile applications: Quality and limitations," *Agronomy*, vol. 12, no. 8, p. 1869, 2022.

[16] D. Sokolov *et al.*, "Plant disease detection model for edge computing devices," *Front. Plant Sci.*, 2023.

192-15-13290

ORIGINALITY REPORT

13%	9%	7%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	2%
2	Submitted to United International University Student Paper	1%
3	Youssef Bouhaja, Hatim Bamoumen, Israe Derdak, Safiyah Sheikh, Moulay El Hassan El Azhari, Hamza El Hafdaoui. "Mobile robot for leaf disease detection and precise spraying: Convolutional neural networks integration and path planning", Scientific African, 2025 Publication	1%
4	arxiv.org Internet Source	1%
5	Submitted to Birla Institute of Technology and Science Pilani Student Paper	1%
6	Submitted to Universiti Utara Malaysia Student Paper	1%
7	Dewi Rismawati, Sugiyarto Surono, Aris Thobirin. "Hybrid feature fusion from multiple CNN models with bayesian-optimized machine learning classifiers", Computer Science and Information Technologies, 2025 Publication	1%
8	"Machine Vision in Plant Leaf Disease Detection for Sustainable Agriculture",	<1%

Scanned with CamScanner