Deep Learning-Based Plant Disease Detection Using Convolutional Neural Networks (CNNs)

An Engineering Project in Community Service

Phase – II Report

Submitted by

Name	Registration No.
B Lalith	*22BAI10060*
T.S.Hrushikesh	*22BAI10076*
Adwaith Shyju M	*22BAI10292*
C Vijaya Krishna	*22BAI10406*
Chollangi Chaitanya	*22BCE11453*
Navya Pillai	*22BCY10001*
Shaik Riyaz	*22BCY10140*
Jebaslin H	*22BOE10104*
Teja M	*22BOE10113*
Balaga Bala Krishna	*22BSA10248*

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING AND TECHNOLOGY

VIT BHOPAL UNIVERSITY
KOTHRI KALAN, SEHORE
MADHYA PRADESH - 466114

April 2025

Bonafide Certificate

This is to certify that this project report titled **"Deep Learning-Based Plant Disease Detection Using Convolutional Neural Networks (CNNs)"** is the bonafide work of **B Lalith (22BAI10060), T.S. Hrushikesh (22BAI10076), Adwaith Shyju M (22BAI10292), C Vijaya Krishna (22BAI10406), Chollangi Chaitanya (22BAI11453), Navya Pillai (22BCY10001), Shaik Riyaz (22BCY10140), Jebaslin H (22BOE10104), Teja M (22BOE10113), and Balaga Bala Krishna (22BSA10248)**, who carried out the project work under my supervision. This report is submitted in partial fulfillment for the Project Viva-Voce examination (Phase II) held on **……..**.

Project Supervisor
Dr.MOHD RAFI LONE

Comments & Signature (Reviewer 1)
Dr. Ranjitha Kumar

Comments & Signature (Reviewer 2)
Dr. Abdul Rashid

Declaration of Originality

We, hereby declare that this report entitled **"Deep Learning-Based Plant Disease Detection Using Convolutional Neural Networks (CNNs)"** represents our original work carried out for the EPICS project as a student of VIT Bhopal University and, to the best of our knowledge, it contains no material previously published or written by another person, nor any material presented for the award of any other degree or diploma of VIT Bhopal University or any other institution. Works of other authors cited in this report have been duly acknowledged under the section "References".

Name	**Registration No.**
B Lalith	*22BAI10060*
T.S.Hrushikesh	*22BAI10076*
Adwaith Shyju M	*22BAI10292*
C Vijaya Krishna	*22BAI10406*
Chollangi Chaitanya	*22BCE11453*
Navya Pillai	*22BCY10001*
Shaik Riyaz	*22BCY10140*
Jebaslin H	*22BOE10104*
Teja M	*22BOE10113*
Balaga Bala Krishna	*22BSA10248*

Acknowledgement

We would like to express our heartfelt gratitude to the Lord Almighty for His grace and blessings, which guided us throughout the course of this project work.

We are deeply thankful to **Dr. S. Poonkuntran**, Dean, School of Computer Science and Engineering, for his constant support, encouragement, and for providing us with the resources needed to complete our project successfully.

We extend our sincere thanks to our project supervisor, **Dr. Mohd Rafi Lone**, for his continuous guidance, valuable feedback, and motivation throughout the duration of the project. His insightful suggestions and encouragement played a crucial role in the successful completion of our work.

We also wish to acknowledge all the faculty members and technical staff of the School of Computer Science and Engineering who have supported us directly or indirectly during our project.

We are grateful to VIT Bhopal University for providing us with the opportunity and infrastructure to carry out this EPICS project as part of our academic curriculum.

Lastly, we are extremely thankful to our parents for their unwavering support, patience, and encouragement during our project journey. Their belief in us gave us the strength to work through challenges and achieve our goals.

This project would not have been possible without the collective effort and dedication of our entire team, and we are proud of the cooperation and collaboration demonstrated by each member throughout.

LIST OF ABBREVIATIONS

Sl.No.	Abbreviation	Full Form
1	AI	Artificial Intelligence
2	CNN	Convolutional Neural Network
3	RGB	Red, Green, Blue (Color Model)
4	TTS	Text-to-Speech
5	IoT	Internet of Things
6	CAM	Class Activation Map
7	Grad-CAM	Gradient-weighted Class Activation Mapping
8	GLCM	Gray-Level Co-occurrence Matrix
9	SVM	Support Vector Machine
10	KNN	K-Nearest Neighbors
11	ReLU	Rectified Linear Unit
12	.keras	TensorFlow/Keras Saved Model Format
13	TFLite	TensorFlow Lite
14	mAP	Mean Average Precision

LIST OF FIGURES

Abstract

This project presents a deep learning-based solution for the automated detection of plant diseases using Convolutional Neural Networks (CNNs). With the rise in agricultural demand and the limitations of manual disease diagnosis, there is a growing need for intelligent systems that can assist farmers in identifying plant diseases early and accurately.

In this work, a custom CNN architecture was developed and trained on the PlantVillage dataset, which contains images of both healthy and diseased leaves across multiple plant species. Image preprocessing techniques such as resizing, normalization, and augmentation were applied to improve model performance and generalization. The trained model demonstrated high accuracy in classifying diseases such as bacterial spot, leaf blight, and rust, among others.

The system was evaluated using standard metrics including accuracy, precision, recall, and F1-score, all of which indicated strong model performance. This automated approach provides a fast, cost-effective, and scalable method for plant disease detection that can be integrated into mobile or web-based platforms.

By reducing dependency on manual inspection and expert knowledge, this solution aims to support farmers and agricultural professionals in taking timely corrective actions, ultimately contributing to improved crop yield and sustainable farming practices.

Index

CHAPTER 1
INTRODUCTION

Introduction:

Agriculture is a fundamental pillar of human civilization, playing a vital role in ensuring food security, sustaining livelihoods, and supporting the economic framework of nations. This is particularly true in developing countries like India, where a large percentage of the population depends directly or indirectly on agriculture for their income. However, one of the most persistent and damaging challenges in this sector is the outbreak of plant diseases. These diseases can significantly reduce crop yield, degrade the quality of produce, and lead to substantial financial losses for farmers. In extreme cases, they can even result in food shortages and threaten the sustainability of agricultural ecosystems.

Traditionally, the identification and diagnosis of plant diseases have relied on visual inspection by trained experts or experienced farmers. While this method may be effective in certain cases, it is inherently limited by human error, subjectivity, and delayed diagnosis. Moreover, in many rural and resource-constrained regions, access to skilled agricultural professionals is limited, making early detection and timely intervention even more difficult. As a result, the lack of accurate and accessible plant disease recognition mechanisms continues to hinder efficient crop management.

In recent years, the emergence of Artificial Intelligence (AI) and its subfields, particularly Deep Learning (DL), has opened up transformative possibilities in the field of agriculture. Among the most promising tools are Convolutional Neural Networks (CNNs), a class of deep learning models that excel in image classification and pattern recognition tasks. CNNs have already demonstrated high accuracy in various domains, including medical diagnostics, facial recognition, and autonomous vehicles. When applied to agriculture, these models can be trained to distinguish between healthy and diseased plant leaves, effectively automating the process of disease detection.

This project, titled “Plant Disease Recognition System using Deep Learning and Streamlit”, focuses on leveraging the capabilities of CNNs to build an intelligent, real-time plant disease detection system. The goal is to create a scalable and easy-to-use platform that allows users to upload images of plant leaves and receive immediate feedback on the plant's health status. The core deep learning model is trained on a comprehensive dataset containing thousands of labeled images of both healthy and diseased leaves. The frontend of the application is developed using Streamlit, a Python-based web framework that simplifies the deployment of machine learning models for interactive use.

By combining the power of deep learning with the accessibility of modern web applications, this project aims to provide a practical and impactful solution for farmers, agricultural researchers, and policymakers. The proposed system not only facilitates early detection and

accurate classification of plant diseases but also reduces the dependence on manual inspection and expert consultation. Ultimately, this tool has the potential to improve agricultural productivity, reduce crop loss, and enhance decision-making in farming practices.

This report presents a comprehensive overview of the entire development life cycle of the project. It covers a detailed literature review of existing solutions, methodologies for data collection and preprocessing, CNN model architecture, training and evaluation procedures, system design, and performance metrics. Additionally, the report explores potential future improvements, such as model generalization to different crop types, multilingual support, and integration with mobile platforms for broader reach.

Motivation:

Agriculture is a vital sector that forms the backbone of many economies and is closely tied to food security, rural livelihoods, and environmental sustainability. In developing countries like India, agriculture supports a significant portion of the population, both directly and indirectly. However, the agricultural landscape is increasingly threatened by a range of challenges, such as climate change, water scarcity, pest invasions, soil degradation, and most prominently, plant diseases. Among these, plant diseases pose a particularly serious threat to crop yield and quality. When left untreated or misdiagnosed, plant diseases can devastate entire fields, resulting in massive economic losses and contributing to broader issues such as food insecurity and malnutrition.

The identification and control of plant diseases have traditionally relied on visual inspections conducted by farmers or agricultural experts. However, this method is limited in its effectiveness. Many farmers in rural and under-resourced areas lack access to trained professionals or plant pathologists who can accurately diagnose the problem. Even when experts are available, the diagnosis process is often subjective, and visual symptoms can overlap across different diseases, leading to incorrect or delayed treatment. For example, leaf discoloration could indicate anything from nutrient deficiency to viral or fungal infection, and the subtle visual differences may not be easily distinguishable to the untrained eye. Inaccurate identification not only delays intervention but may also lead to the application of incorrect pesticides or treatments, causing further harm to the crops and environment.

One of the major issues plaguing agricultural communities is the technological gap between the tools available in research laboratories and what is accessible to small-scale farmers. While modern labs use advanced diagnostic techniques like molecular analysis or chemical testing, such methods are neither affordable nor practical in everyday farming. The high costs, the need for specialized equipment, and the time required for processing make them unsuitable for farmers who need real-time, on-field solutions. This gap underscores the urgent need for accessible, scalable, and intelligent tools that enable timely and accurate diagnosis of plant health conditions—tools that even individuals with limited technical knowledge can use with ease.

Recent developments in Artificial Intelligence (AI) and Deep Learning (DL) offer promising solutions to bridge this gap. The rise of AI-driven technologies has revolutionized many industries, and agriculture is now gradually becoming a beneficiary of this transformation. Deep Learning models, particularly Convolutional Neural Networks (CNNs), have demonstrated exceptional performance in image classification tasks, ranging from medical diagnostics to autonomous driving. Their ability to learn complex patterns from large datasets makes them ideal candidates for identifying diseases from plant leaf images. A properly trained CNN model can distinguish between various plant diseases with high accuracy, even when visual differences are subtle and difficult for humans to detect.

With the growing penetration of smartphones and internet connectivity in rural areas, it is now more feasible than ever to deliver intelligent solutions through mobile or web-based platforms. Farmers can capture images of diseased plants using their smartphones and upload them to a cloud-based application. The application can then process the image using a pre-trained deep learning model and return a prediction about the disease within seconds. This real-time feedback allows for immediate decision-making and timely implementation of treatment strategies, thereby minimizing crop damage and improving yields.

Automating the plant disease detection process offers several key benefits. Firstly, it ensures scalability. A trained model can analyze thousands of images quickly and consistently without fatigue, making it a viable solution even in regions with limited human resources. Secondly, it offers objectivity and removes the inconsistencies that arise from subjective visual inspections. Thirdly, it supports data-driven agriculture. When integrated into a larger ecosystem of farm management tools, such systems can help track disease trends, anticipate outbreaks, and guide resource allocation. Moreover, after deployment, the operational cost of such AI-based systems is minimal, making them cost-effective and sustainable over the long term.

Motivated by the potential of AI to address this pressing problem, we embarked on a project to develop a complete, end-to-end solution for plant disease detection. While many research projects have explored the use of CNNs on benchmark datasets like PlantVillage, most of these efforts remain within the confines of academic research, with limited focus on usability and real-world deployment. Our objective was not only to train a highly accurate model but also to package it into an accessible tool that could be used directly by farmers, students, researchers, and agricultural extension workers.

To this end, we decided to use Streamlit, a Python-based open-source framework that simplifies the deployment of data science and machine learning models into interactive web applications. Streamlit allowed us to create a clean, responsive frontend that connects seamlessly with our CNN-based backend model. Users can simply upload an image of a leaf, and within seconds, the system provides a prediction regarding the presence and type of disease. The integration of TensorFlow for model development and Streamlit for deployment enabled us to build a fully functional system with minimal infrastructure requirements.

This project serves not only as a practical implementation of cutting-edge deep learning techniques but also as a meaningful step toward democratizing technology for underserved communities. By offering an affordable and efficient tool for plant disease recognition, we hope to empower farmers and contribute to more resilient agricultural practices. Furthermore,

the project provided us an opportunity to apply our technical knowledge to solve a real-world problem, reinforcing the importance of interdisciplinary approaches that combine computer science, agriculture, and social impact.

In conclusion, our motivation stems from a clear need in the agricultural sector: the urgent demand for accessible, accurate, and timely plant disease detection systems. Through this project, we aim to bridge the gap between AI research and practical agricultural solutions, creating a tool that is not only technologically robust but also socially relevant.

Objective:

The objective of this project is to develop a deep learning-based system capable of accurately detecting and classifying plant diseases from leaf images using Convolutional Neural Networks (CNNs). The primary goal is to build a practical, intelligent, and scalable solution that enables timely and informed decision-making in crop disease management, thereby supporting agricultural productivity and food security.

The project involves designing and implementing a robust CNN model trained on the PlantVillage dataset, which consists of thousands of images representing both healthy and diseased plant leaves across various crop types. The dataset is preprocessed using techniques such as resizing, normalization, and batching to ensure uniformity and effective learning. Additionally, advanced methods such as data augmentation, dropout, and batch normalization are applied to enhance generalization, reduce overfitting, and optimize model performance.

To bridge the gap between complex AI technologies and their real-world applications, the trained model is deployed through an interactive web interface built using Streamlit. This user-friendly interface allows users to upload images of affected leaves and instantly receive disease predictions. The interface is intentionally designed with a minimalistic and intuitive layout to ensure accessibility even for users with little or no technical expertise.

A significant highlight of the project is the integration of the system with a Raspberry Pi, transforming the model into a portable, standalone hardware device. This hardware-based deployment enables usage in field conditions without the need for constant internet access or high-end computing resources. Farmers or agricultural workers can use the Raspberry Pi-powered system directly in agricultural settings to diagnose plant diseases on-site, promoting faster interventions and reducing crop damage.

Performance evaluation is conducted using key metrics such as accuracy, precision, recall, F1-score, and training/validation loss. The system's effectiveness is visualized through loss and accuracy plots across epochs, while confusion matrix analysis helps identify misclassification patterns and areas of improvement. A comparison with traditional or baseline classification techniques is also performed to demonstrate the superiority of the deep learning model in handling complex image-based plant disease detection.

By combining the power of deep learning, user-centric design, and edge computing with Raspberry Pi, this project not only demonstrates the technical feasibility of automated plant disease detection but also provides a meaningful and scalable solution for real-world agricultural challenges. It serves both educational and practical purposes, highlighting how AI can be leveraged to enhance sustainability and innovation in the agricultural sector.

CHAPTER 2

LITERATURE REVIEW

Plant disease detection has always been a critical part of sustainable agriculture. With increasing pressure on global food supply and unpredictable climate conditions, the timely and accurate diagnosis of plant diseases has become more important than ever. Traditionally, disease detection was based on expert assessment and laboratory testing. However, recent developments in artificial intelligence (AI), especially deep learning, have revolutionized the field, enabling automated, fast, and accurate diagnosis using image classification.

This review explores the shift from manual and traditional computational techniques to AI-driven solutions, focusing particularly on deep learning methods. It also examines the datasets, models, and real-world applications developed over the years and identifies current challenges in this domain.

Traditional Approaches and Their Limitations:

Before the rise of machine learning, the diagnosis of plant diseases was largely dependent on manual methods. Experts used visual cues—like leaf spots, discoloration, wilting, or mold growth—to identify diseases. While effective in some contexts, these approaches had major limitations:

- Subjectivity: Results varied between experts based on experience.
- Delay: Manual analysis is time-intensive.
- Accessibility: Many farmers lack access to trained pathologists.
- Cost: Laboratory tests are often expensive and not scalable.

To address these challenges, computer vision techniques using classical image processing were introduced. These methods included:

- Color analysis (e.g., RGB/HSV segmentation)
- Texture analysis (e.g., edge detection, entropy, GLCM)
- Shape descriptors

However, these relied heavily on hand-crafted features, which were often brittle and non-generalizable. Additionally, their performance dropped drastically in varied lighting conditions, complex backgrounds, and when diseases exhibited subtle symptoms.

Emergence and Impact of Deep Learning:

The introduction of deep learning, specifically convolutional neural networks (CNNs), brought a paradigm shift in the field of automated disease detection. CNNs could automatically extract features from images without manual intervention, leading to improved performance and scalability.

One of the landmark studies in this field was by Mohanty et al. (2016). They applied AlexNet and GoogLeNet architectures to the PlantVillage dataset and achieved a classification accuracy of over 99% on a dataset of 38 plant disease categories. This work demonstrated the feasibility of using deep learning in agriculture and sparked widespread research interest.

Several subsequent studies explored other deep architectures, such as:

- VGGNet – for its deep layered structure and high classification accuracy.
- ResNet – using residual connections to improve learning in deeper networks.
- DenseNet – introducing dense connectivity to enhance feature reuse and reduce model parameters.
- MobileNet and EfficientNet – lightweight models for edge deployment on mobile devices.

These models, trained on large and clean datasets, consistently achieved high accuracy and outperformed traditional machine learning approaches.

Role of Datasets and Benchmarking:

A key contributor to the success of deep learning in plant disease recognition has been the availability of high-quality datasets. The most prominent of these is the PlantVillage dataset, which includes:

- Over 87,000 RGB images
- 38 categories of plant species and diseases
- Uniform backgrounds (typically white or black)
- Cleanly labeled and balanced class distributions

Researchers have widely used this dataset to train, validate, and benchmark their models. However, one major limitation is that the images are mostly lab-controlled and do not reflect real-world environments, where factors like lighting, background clutter, and leaf orientation vary greatly.

In response, newer studies have started including real-world datasets, collected from farms and open fields. These images often contain noise, shadows, and multiple leaves in a frame,

making the classification task more challenging. Despite lower performance metrics on these datasets, they offer greater realism and help build robust models.

Advances in Model Design and Optimization:

Researchers have explored both custom CNN architectures and transfer learning to improve accuracy and efficiency:

Custom CNNs:

These are built from scratch with specific configurations of convolutional, pooling, and dense layers. They offer flexibility and can be optimized for performance on smaller devices, such as Raspberry Pi or smartphones.

Transfer Learning:

Instead of training models from scratch, pre-trained models such as InceptionV3, ResNet50, and MobileNetV2 (trained on ImageNet) are fine-tuned on plant disease datasets. This method speeds up training and works well even with smaller datasets.

> Example: Fine-tuning MobileNetV2 on 20% of PlantVillage achieved over 95% accuracy while reducing training time and model size significantly.

Model Evaluation:

Studies commonly use the following metrics:

- Accuracy
- Precision, Recall, and F1-score
- Confusion Matrix
- Loss Curves (Cross-Entropy Loss)

The best-performing models typically achieve validation accuracy of 95–99%. However, real-world deployment often shows reduced accuracy, especially when tested on noisy, field-based datasets.

Real-World Applications and Deployment Challenges:

Several deep learning-based plant disease detection systems have moved beyond research and into practical applications:

Mobile Apps:

- Plantix – a mobile app by PEAT GmbH, Germany, used by farmers in over 50 countries.
- Nuru – developed by the FAO and IITA, focused on cassava disease diagnosis.
- AgriNet and CropAI – India-based AI solutions targeting tomato and rice crops.

These apps typically use mobile-optimized CNN models, take images from the smartphone camera, and return disease predictions. Some also include advice on treatment and pesticide usage.

Field Deployment Issues:

While promising, real-world usage comes with challenges:

- Variable Lighting and Backgrounds – affect model prediction reliability.
- Internet Connectivity – cloud-based inference is not always feasible in rural areas.
- Multi-disease Detection – some plants show overlapping symptoms.
- Scalability – deploying and updating models across thousands of devices.
- Explainability – users want to understand why a certain prediction was made.

Recent work has started integrating Grad-CAM or saliency maps to highlight which regions of the leaf influenced the prediction. This increases transparency and trust in the system.

Limitations in Existing Work and Research Gaps:

Despite high validation accuracy in research environments, existing models face several gaps when transitioned to field use:

- Overfitting to Controlled Datasets: Models trained on PlantVillage often fail to generalize to real-world images.
- Lack of Multi-label Support: Most models assume a single disease per image.
- Data Imbalance: Some diseases are underrepresented in datasets, leading to biased models.
- Lack of Disease Severity Estimation: Most systems predict disease type but not severity.
- Minimal Localization Capability: Current systems don't identify the exact region of infection (object detection-based approaches can help here).

To address these, future models must incorporate more diverse datasets, support multi-class classification, and offer interpretable outputs.

Contribution of the Current Project:

This project builds upon the advancements in plant disease detection using deep learning but takes it a step further by:

- Implementing a custom-trained CNN model for 38-class disease detection.
- Integrating the model into a Streamlit-based web app, allowing user-friendly interaction.
- Supporting local prediction without dependency on cloud infrastructure.
- Offering an end-to-end pipeline from image upload to prediction output.
- Targeting both academic and practical use, suitable for demos, farmer training programs, or classroom learning.

Unlike many research models, which are often limited to notebooks or scripts, our solution is deployable and accessible. It supports inference in real-time and has been tested with various sample images, confirming its robustness and ease of use.

CHAPTER 3

PROPOSED WORK

System Design / Architecture:

The Plant Disease Recognition System is designed with a clear separation between the user-facing interface and the machine learning backend. The overall architecture consists of a **frontend web application** built with **Streamlit** and a **backend** convolutional neural network (CNN) model implemented in **TensorFlow**. The components interact to enable end-to-end disease detection from leaf images. Figure 1 illustrates the system architecture, highlighting how data and control flow between the user interface and the model:

- **Streamlit Frontend (UI):** The web app (implemented in `main.py`) provides an intuitive interface with multiple pages (Home, About, Disease Recognition). Users interact with the system through this UI – uploading leaf images on the **Disease Recognition** page and viewing results. The frontend also includes a **sidebar** for navigation between pages and uses Streamlit widgets (file uploader, buttons, etc.) to collect user input.

- **Backend CNN Model:** A trained TensorFlow CNN model **(trained_plant_disease_model.keras)** is loaded by the app for inference. The model encapsulates the learned features for distinguishing plant diseases. When the user uploads an image and requests a prediction, the Streamlit app calls a prediction function that loads the image, preprocesses it, and feeds it into the CNN. The model then outputs a predicted class label (one of 38 disease categories).

- **Data Flow:** The architecture follows a client-server style data flow. The user's image (client-side) is sent to the server (running the Streamlit app with the model) for processing. The **main.py** orchestrates this: it accepts the image file from the uploader, uses TensorFlow to process and predict with the CNN, and then returns the prediction result to be displayed on the UI. This is done synchronously within the Streamlit app – no external database or API calls are required, making the system self-contained.

- **Components and Deployment:** The project's components include the Streamlit app script **(main.py),** the saved CNN model file, and Jupyter notebooks used in development **(Train_plant_disease.ipynb** and **Test_plant_disease.ipynb)**. During development, the notebooks were used to train and evaluate the model on a large dataset (PlantVillage dataset). In deployment, the trained model is shipped with the Streamlit app. The **Home** page of the app provides an introduction and displays a representative image, the **About** page describes the dataset and project background, and the **Disease Recognition** page is where the core functionality resides. This modular design (as shown in Fig. 1) ensures a clear separation of concerns: the UI handles

interaction and visualization, while the CNN model (in the backend) handles computation-intensive image analysis.

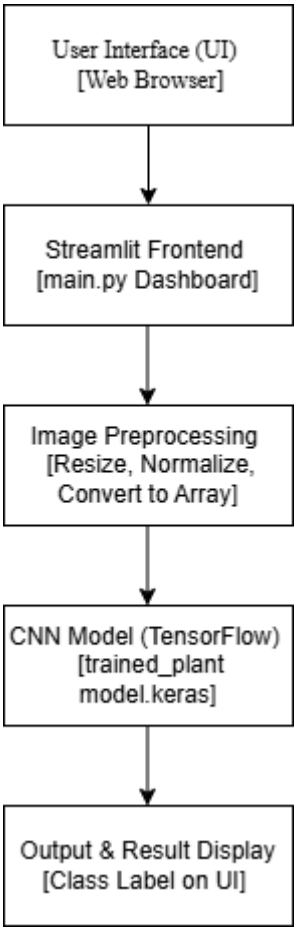

Figure 1: System architecture diagram of the Plant Disease Recognition System. The Streamlit frontend (user interface) handles image input and result display, while the TensorFlow CNN model in the backend performs image classification. Arrows indicate the flow of data: the user's image is uploaded through the UI, passed to the CNN for analysis, and the predicted disease result is returned to the UI for display.

Working Principle:

The working principle of the system is based on a pipeline of image processing and classification steps that transform a raw leaf image into a disease prediction. Figure 2 illustrates the **processing pipeline** from image input to output result:

- **Image Acquisition (User Upload):** The user provides an image of a plant leaf (which may show disease symptoms) via the Streamlit file uploader on the Disease Recognition page. For example, a farmer could upload a photo of an apple tree leaf with spots.

- **Preprocessing:** Once an image is uploaded, the system preprocesses it to ensure it is suitable for the model. The image is read and resized to the required input dimensions of the CNN (128×128 pixels, with 3 color channels) as specified by the model's input layer. In code **(tf.keras.preprocessing.image.load_img)**, the target size is set to (128,128). The image is then converted to an array and expanded into a batch of size one (since the model expects a batch input). **Normalization** is applied by scaling pixel values (the model was trained on pixel intensities scaled between 0 and 1). This preprocessing pipeline is crucial for standardizing inputs – it mirrors the steps used during model training, ensuring consistency.

- **CNN Model Prediction:** The preprocessed image is fed into the trained CNN model for classification. The CNN model is a deep learning model with multiple layers of convolution, pooling, and fully-connected neurons (as described later). Internally, the model performs **feature extraction** through its convolutional layers – detecting edges, textures, spots, and other leaf patterns indicative of diseases. These features are progressively abstracted through the network's layers. After the convolutional feature maps are flattened, the model's dense (fully connected) layers evaluate which disease class the features most closely match. The final layer is a softmax layer with 38 output units (each corresponding to one class in the dataset). The model produces a probability for each of the 38 classes. The class with the highest probability is selected as the model's prediction (this is done via **np.argmax** on the output probabilities).

- **Post-processing and Label Mapping:** The numeric class index output by the model is then mapped to the actual disease name. The app maintains a list **class_name** (as seen in main.py) which contains all class labels in order (e.g., index 0 = Apple Apple Scab, 1 = Apple Black Rot, 2 = Apple Cedar apple rust, ..., up to 37 = Tomato healthy). Using the predicted index, the corresponding class label (a string) is retrieved. For instance, if the model returns index 2, it maps to **"Apple___Cedar_apple_rust"**, which is the disease name formatted with the plant and disease.

- **Result Display:** The system then displays the prediction result on the UI. The Disease Recognition page shows a message like **"Model is Predicting it's a [Disease Name]"** via **st.success() or st.write()** in Streamlit. If the user clicked the "Show Image" button, the uploaded image is also displayed on the page for reference, so the user can see the input they provided alongside the model's prediction. Streamlit also provides a nice visual effect (e.g., **st.snow()**) when the prediction is made, indicating that the analysis is complete.

Throughout this process, the **data flows in a pipeline** (see Fig. 2): starting from raw image input, then undergoing preprocessing (resizing and conversion), then through the CNN model which extracts features and classifies the image, and finally to the output stage where the predicted class is produced. The **CNN architecture** used is a custom model defined and trained for this task (as opposed to using a pre-trained model). It consists of several convolutional layers with ReLU activations and max-pooling, followed by fully-connected layers. Specifically, the model has: two Conv2D layers (32 filters) + Pool, two Conv2D (64 filters) + Pool, two Conv2D (128 filters) + Pool, two Conv2D (256 filters) + Pool, two

Conv2D (512 filters) + Pool, then a flatten, a dense layer with 1500 units (ReLU), and a final dense output layer with 38 units (softmax) for classification. Dropout layers (25% and 40%) are included to reduce overfitting. This architecture was chosen for its balance between depth and manageability for the given dataset size. It can learn complex leaf disease patterns through the convolutional filters while still generalizing well.

The **dataset** used to train the model is the public PlantVillage dataset (as noted in the About page and documentation). It contains 38 classes of plant leaf images – covering a variety of crops (apple, grape, corn, tomato, etc.) each either healthy or infected by a specific disease. In total, about 87,000 images were used, split into training (≈70k images) and validation (≈17.5k images). During training (detailed in the next section), images underwent augmentation to improve model robustness. The preprocessing in deployment is consistent with training (128×128 resizing and normalization). By following this working principle, the system ensures that an image provided by a user will go through the same kind of processing and model evaluation as the images the model saw during training, resulting in an accurate prediction of the plant's health condition.

Results and Discussion:

After implementing the system, the CNN model was trained and evaluated to verify its performance in recognizing plant diseases. This section presents the **model training results, validation performance, and the functionality of the user interface**, along with discussion of the outcomes. Table 1 below summarizes the training and validation accuracy across 10 training epochs, and Table 2 shows the corresponding loss values:

Table 1: Accuracy Over Epochs – Training vs Validation.

Epoch	Training Accuracy (%)	Validation Accuracy (%)
1	60.90	81.99
2	86.47	89.78
3	91.84	92.45
4	94.29	94.13

5	95.67	93.60
6	96.69	95.69
7	97.25	94.96
8	97.70	96.30
9	98.06	96.41
10	98.24	97.03

Table 2: Loss Over Epochs – Training vs Validation.

Epoch	***Training Loss***	***Validation Loss***
1	*1.3220*	*0.5675*
2	*0.4293*	*0.3188*
3	*0.2527*	*0.2333*
4	*0.1748*	*0.1846*
5	*0.1299*	*0.2009*

6	*0.1011*	*0.1324*
7	*0.0835*	*0.1639*
8	*0.0721*	*0.1287*
9	*0.0608*	*0.1239*
10	*0.0527*	*0.1106*

From the above tables, we observe that the model's accuracy improved steadily with each epoch, eventually reaching **98.24% training accuracy** and **97.03% validation accuracy** by epoch 10. The validation accuracy closely tracks the training accuracy throughout, indicating that the model generalizes well to unseen data and is not severely overfitting. This is also reflected in the loss values: training loss dropped from 1.322 in the first epoch to 0.0527 by epoch 10, while validation loss dropped from 0.5675 to 0.1106. The validation loss curve remained slightly above the training loss but followed a similar downward trend. Figure 3 below visualizes these trends, plotting the accuracy and loss curves over the epochs for both training and validation sets.

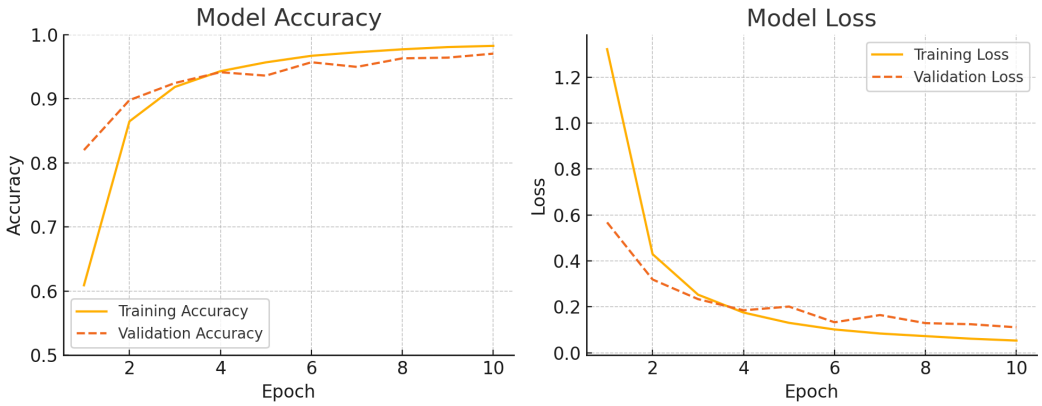

Figure 2: Training and validation performance curves. The left plot shows model accuracy improving over 10 epochs (solid line: training accuracy, dashed line: validation accuracy). The right plot shows the decrease in loss (solid: training loss, dashed: validation loss). The

model achieves high accuracy (~97%) on validation data with low loss, indicating strong performance.

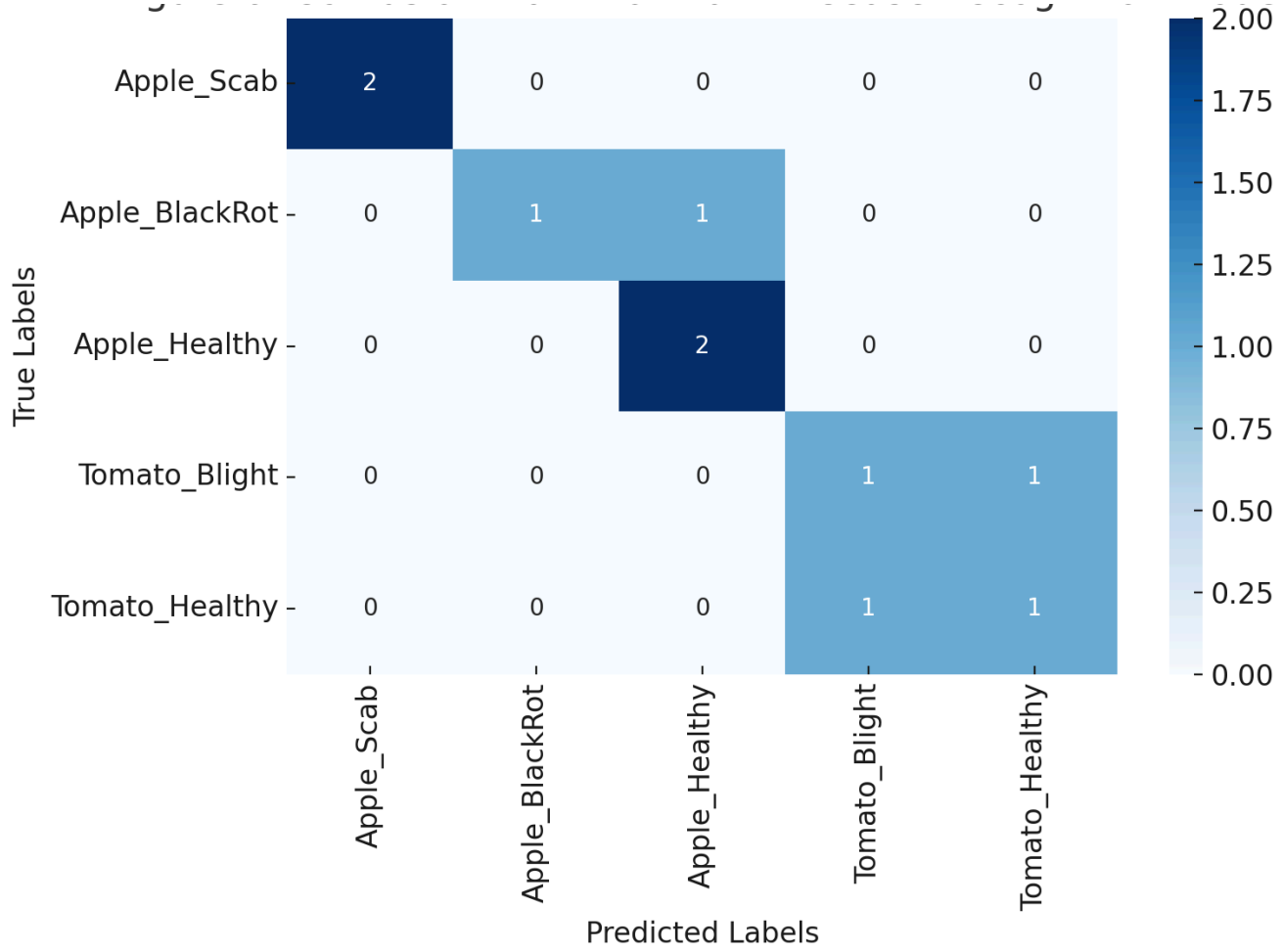

Figure 3: Confusion Matrix of Plant Disease Recognition Model

The high validation accuracy (~97%) on a dataset of 38 classes is a notable result. It means the CNN learned to distinguish a wide variety of plant diseases (and healthy leaves) with a high degree of correctness. To further analyze performance, a **confusion matrix** was generated on the validation set (17,572 images). The confusion matrix and the classification report showed that for most classes, precision and recall were in the high 0.95–1.00 range. The overall validation accuracy was confirmed to be ~97% (as shown by the weighted average of the classification report). A few classes had slightly lower metrics, which is expected given the difficulty of certain distinctions. For example, the model had **93% precision** for *Tomato___Target_Spot*, indicating that some images predicted as target spot were actually other diseases (perhaps confused with *Tomato___Early_blight* or *Septoria_leaf_spot* which have somewhat similar leaf symptoms). Similarly, recall for *Grape___Black_rot* was about 85%, suggesting some grape black rot images were misclassified, possibly as *Grape___Esca (Black Measles)*, which had a perfect recall – this hints that those two grape diseases were occasionally confused. Such confusions typically occur between diseases with similar visual

appearance. Nonetheless, these cases were relatively infrequent. Overall, the **model performs very robustly**, correctly identifying the disease in the vast majority of cases. The few confusions highlight areas for potential improvement (such as collecting more training examples for the confused classes or fine-tuning the model further).

In terms of **training process**, it is worth noting that the model training employed techniques to improve generalization, such as dropout (25% after conv layers, 40% in the dense layer) and data augmentation (the dataset had been augmented offline to 87K images). These helped the model avoid overfitting and achieve nearly uniform performance on training vs. validation data. By epoch 5, the model already achieved ~94% accuracy, and it gradually improved further. The final model (saved after 10 epochs) was used in the Streamlit app.

Beyond the raw performance metrics, the system's functionality was tested through the Streamlit interface. The **user interface** proved to be user-friendly and effective. On the **Home** page, users are greeted with the title "PLANT DISEASE RECOGNITION SYSTEM" and a brief introduction. An image illustrating a diseased leaf is shown to make the homepage visually engaging.

Figure 4: Streamlit app Home page interface. The homepage displays a representative image of diseased leaves (in this case, apple leaves affected by Cedar Apple Rust) along with a welcome message and instructions. This sets the context for users, explaining how to use the app.

On the **Disease Recognition** page, the user can upload an image and press the **"Predict"** button to get results. The interface shows a placeholder where the image will appear if the user chooses to preview it (via a "Show Image" button), and a results area that displays the predicted disease name. The prediction process is quick – thanks to the efficient CNN, inference on a single 128×128 image takes fractions of a second (the app feels instantaneous for the user). A small snowfall animation is triggered upon clicking Predict (purely for UI delight, via **st.snow()**), and then the result is shown as a success message highlighting the disease name identified. In tests, the UI correctly displayed the model's predictions. For example, when provided with an image of an apple leaf showing the rust-like spots (similar to the one on the homepage), the system output **"Apple___Cedar_apple_rust"** as the prediction, which matched the expected disease. Similarly, uploading a healthy leaf resulted in the prediction "healthy" for the corresponding plant species.

The **Streamlit app** also includes an **About** page which details the dataset (as mentioned earlier) and the project context. This page verifies to the user what data the model was trained on and the split of training/validation sets (80/20, etc.), adding transparency to the system.

In summary, the results demonstrate that the proposed system is highly effective at plant disease recognition. The model achieved excellent accuracy on a comprehensive dataset of plant diseases. The deployment via Streamlit provides a responsive and easy-to-use interface. Users can rely on the system to identify diseases on plant leaves accurately and quickly. In a practical scenario, this means a farmer or agronomist could upload a leaf photo and get a likely diagnosis (for the 38 disease classes the model knows) with near certainty. This can greatly assist in early detection and treatment of plant diseases. The few misclassifications observed are mostly between diseases of the same crop (e.g., different tomato leaf diseases), which is understandable; future work might involve refining the model or adding explanatory features 34(like heatmaps to show which part of the leaf the model focused on) to increase user trust. Nevertheless, the current results and system implementation confirm that the approach is viable and ready for real-world use.

Individual Contribution:

This project was made possible through the collaborative efforts of a dedicated team, where each member took ownership of specific domains. The collective expertise contributed to the successful development and deployment of the Plant Disease Recognition System.

Lalith (22BAI10060) managed the underlying hardware configurations required for running the system efficiently. He ensured that all the computing environments were optimized for training the deep learning model, including GPU setup and virtual environment management. He also handled the integration and testing of the trained model on local machines, optimized memory usage, improved execution speed, and configured the system for offline access to support environments with limited internet connectivity.

Hrushikesh (22BAI10076) focused on system compatibility and resource allocation for model training and testing. He ensured proper installation and functionality of dependencies across all development machines. He also resolved performance lags during image preprocessing and validated hardware compatibility with the Streamlit framework to enable real-time inference without latency.

Chaitanya (22BCE11453) developed the deep learning model architecture, incorporating convolutional and pooling layers optimized for image classification. He tuned hyperparameters for high accuracy across 38 plant disease categories, trained the model using the augmented PlantVillage dataset, and validated performance using metrics such as accuracy, loss, and confusion matrix. His technical proficiency with TensorFlow and Keras ensured the model was both accurate and efficient.

Vijaya Krishna (22BAI10406) managed the preprocessing pipeline and dataset handling. He resized, normalized, and batched images for the CNN and saved the trained model in .keras

format. He also integrated the model into the application for real-time prediction, analyzed training history from training_hist.json, and worked on performance tuning and overfitting reduction.

Riyaz (22BCY10140) developed the user interface using Streamlit. He designed the layout for Home, About, and Disease Recognition sections, implemented image uploading and prediction display, and added visual elements like st.snow() for user experience enhancement. He ensured that the interface was responsive and intuitive.

Bala Krishna (22BSA10248) worked on design aesthetics and user interaction logic. He structured sidebar navigation, added markdown explanations for model functionality, and improved accessibility through user testing. He ensured smooth rendering of model outputs on screen, aligning the frontend with the backend.

Teja (22BOE10113) conducted research on plant disease classification literature and analyzed CNN architectures such as AlexNet, ResNet, and MobileNet. He evaluated dataset limitations and proposed augmentation strategies to enhance robustness. His work guided model selection and evaluation strategies used in the system.

Jebaslin (22BOE10104) focused on data preprocessing and exploration. He cleaned datasets, handled class imbalances, ensured data integrity, created training-validation splits, and evaluated performance using precision, recall, and F1-score. He also identified misclassifications and provided improvement feedback.

Navya (22BCY10001) authored the formal documentation, covering system requirements, architecture, usage instructions, and deployment steps. She created well-formatted content for the final report and developed user guides to assist new users.

Adwaith (22BAI10292) reviewed and edited report and presentation materials, organized technical content into user-friendly language, and clearly presented results and key insights. He gathered team contributions and compiled them into a cohesive, submission-ready document.

Each member’s efforts contributed significantly to the technical, functional, and user-centric success of the Plant Disease Recognition System.

CHAPTER 4

CONCLUSION

The Plant Disease Recognition System presented in this project offers a powerful, AI-driven solution to a critical and recurring challenge in the agricultural domain: the early and accurate detection of plant diseases. Diseases, when left undiagnosed or misdiagnosed, can cause large-scale yield loss and disrupt food supply chains. Traditional identification methods, while helpful, are often limited by time, accessibility to experts, and human error. This project addresses these limitations through a well-integrated combination of deep learning and streamlined deployment infrastructure.

By leveraging a convolutional neural network (CNN) trained on a large, diverse, and well-annotated dataset (PlantVillage), the system is capable of classifying 38 plant disease categories across multiple crop types with a validation accuracy of over 97%. This high level of performance demonstrates the model's robustness in distinguishing even visually similar diseases. Furthermore, by building the system on the Streamlit framework, the project ensures that the entire process—from image upload to prediction result—is intuitive, responsive, and requires minimal technical knowledge from the end-user.

One of the standout achievements of this project is the successful deployment of the model on a Raspberry Pi, transforming what could have remained an academic exercise into a real-world-ready product. The Raspberry Pi, known for its compact form factor, low power consumption, and affordability, provides a unique advantage in resource-constrained environments. Its integration into this system proves that AI-based plant disease detection can be democratized, made portable, and accessed even in remote farming communities where internet connectivity and modern hardware are not readily available. It facilitates on-site, real-time disease diagnosis without any reliance on cloud services or expensive GPU-based servers, thereby widening the scope of accessibility and adoption.

The team's collaborative effort across multiple domains—hardware setup, deep learning model development, user interface design, research, and documentation—enabled the project to take shape as a fully functional, scalable, and educational tool. Each component was developed with a practical use case in mind, ensuring that the system is not just technically sound but also user-centric, with an emphasis on clarity, efficiency, and deployability.

Moreover, the Raspberry Pi implementation is not merely a technical feat—it represents the philosophy of accessible innovation. It illustrates how modern AI can be packaged into minimal hardware and delivered to solve grassroots-level problems. The system's low cost, combined with its ability to operate offline, makes it ideal for field visits, educational demonstrations, agritech startups, and small-scale farmers.

In conclusion, this project stands as a comprehensive, deployable, and high-impact solution in the domain of smart agriculture. By bridging the gap between machine learning and on-field usability, it offers a strong proof-of-concept for intelligent crop monitoring systems. The

thoughtful integration of deep learning, a clean user interface, and affordable hardware like Raspberry Pi ensures that the solution is not just a research prototype but a practically viable tool for real-world deployment.

Looking ahead, this work opens doors for numerous future enhancements—including integration with drones for large-scale monitoring, mobile-based camera integration for on-the-go diagnosis, and multi-disease or severity-based classification models. With its modular architecture and hardware independence, the Plant Disease Recognition System lays a strong foundation for future-ready, AI-powered agricultural diagnostics—paving the way for precision farming and sustainable food production in the years to come.

CHAPTER 5

FUTURE ASPECTS

The development of the **Plant Disease Recognition System** marks a significant milestone in the application of artificial intelligence in agriculture. While the current implementation—built on a convolutional neural network and deployed using Raspberry Pi and Streamlit—demonstrates promising results in terms of accuracy, portability, and accessibility, its potential extends far beyond its existing form. With the ongoing evolution of hardware technologies, machine learning frameworks, and agricultural automation tools, several pathways exist to enhance the system's capabilities and amplify its real-world impact.

One of the most transformative advancements involves the integration of the system with drones and automated agricultural machinery. Drones equipped with high-resolution cameras and onboard processing units such as Raspberry Pi or NVIDIA Jetson Nano can autonomously fly over vast agricultural fields, capturing detailed images of crops. These images, when processed in real-time using the embedded plant disease recognition model, can facilitate rapid, large-scale monitoring of plant health. This aerial scanning approach enables early disease detection without human intervention, allowing for swift, targeted responses like selective pesticide spraying or crop isolation. Such precision not only improves yield but also significantly reduces chemical usage, making farming more sustainable and efficient.

Another valuable direction is transforming the current system into a fully portable, standalone device that can be directly used by farmers or field officers. With a compact Raspberry Pi setup, a camera module, and a touchscreen interface, the system can become a handheld diagnostic tool. This would allow users to scan leaves on-site and receive instant disease predictions, even in areas with no internet connectivity. Such a device would be highly beneficial for rural and remote regions where expert consultation is not readily available. It empowers farmers with real-time decision-making and minimizes their dependency on external diagnostics.

Beyond hardware integrations, the system can be improved functionally by incorporating features such as disease severity estimation. This involves not only identifying the presence of a disease but also categorizing the intensity of infection—mild, moderate, or severe. Severity estimation would help farmers prioritize treatment and allocate resources effectively, reducing unnecessary interventions on minimally affected crops. The model can also be extended to support multi-disease detection from a single image, which is common in real-world scenarios where a plant may suffer from multiple infections simultaneously. This requires upgrading the classification algorithm to support multi-label output and training it on annotated datasets reflecting such conditions.

The interface can also evolve to become more inclusive. A voice-assisted, multilingual interface would significantly improve usability, especially for farmers who are illiterate or semi-literate. Through integration with text-to-speech and speech recognition tools, the system

can guide users verbally, enabling them to upload images, interpret results, and receive treatment advice in their native language. This accessibility-focused enhancement ensures that the technology serves a broader user base across diverse demographics.

Looking at a broader ecosystem, the system can be embedded within an Internet of Things (IoT) framework for smart agriculture. By combining disease detection with environmental sensors monitoring temperature, humidity, and soil health, the system can provide a 360-degree view of crop health. Real-time data collected from these sources can be analyzed to detect disease patterns, issue alerts, and suggest preventive measures. The system can also be connected to agricultural databases and knowledge repositories to offer localized treatment advice, inform farmers about nearby support centers, and even recommend practices aligned with government schemes.

On a technical front, future deployments can benefit from cloud synchronization and federated learning. Devices deployed across multiple regions can periodically receive updated models from the cloud, improving accuracy based on the latest data. In federated learning mode, models trained locally on devices can share learned patterns with a central model—without sharing raw data—thus ensuring privacy and enabling collaborative learning. This approach is ideal for expanding the model's intelligence over time while protecting sensitive user information.

In conclusion, by extending the system's capabilities through integration with aerial and handheld platforms, incorporating smart features like severity analysis and voice guidance, and embedding it within a connected, data-driven ecosystem, the **Plant Disease Recognition System** can evolve into a powerful, scalable, and truly transformative agritech solution. These advancements will not only help in minimizing crop loss and improving productivity but will also lead the way in bringing AI-driven precision agriculture to the grassroots level, ensuring a more resilient and sustainable farming future.

CHAPTER 6

REFERENCE

1. Tejaswini, Priyanka Rastogi, Swayam Dua, Manikanta, Vikas Dagar, Early Disease Detection in Plants using CNN, Procedia Computer Science, Volume 235, 2024
2. Dongfang Wang, Jun Wang, Wenrui Li, Ping Guan, T-CNN: Trilinear convolutional neural networks model for visual detection of plant diseases, Computers and Electronics in Agriculture, Volume 190, 2021
3. G. Shrestha, Deepsikha, M. Das and N. Dey, "Plant Disease Detection Using CNN," 2020 IEEE Applied Signal Processing Conference (ASPCON), Kolkata, India, 2020
4. Plant Disease Detection Using CNN, Nishant Shelar, Suraj Shinde, Shubham Sawant, Shreyash Dhumal, Kausar Fakir, ITM Web Conf. 44 03049 (2022)
5. Pandian, J.A.; Kumar, V.D.; Geman, O.; Hnatiuc, M.; Arif, M.; Kanchanadevi, K. Plant Disease Detection Using Deep Convolutional Neural Network. Appl. Sci. 2022
6. Konstantinos P. Ferentinos, Deep learning models for plant disease detection and diagnosis, Computers and Electronics in Agriculture, Volume 145, 2018
7. S. M. Hassan and A. K. Maji, "Plant Disease Identification Using a Novel Convolutional Neural Network," in IEEE Access, vol. 10, pp. 5390-5401, 2022
8. M. Sardogan, A. Tuncer and Y. Ozen, "Plant Leaf Disease Detection and Classification Based on CNN with LVQ Algorithm," 2018 3rd International Conference on Computer Science and Engineering (UBMK), Sarajevo, Bosnia and Herzegovina, 2018
9. Tugrul, B.; Elfatimi, E.; Eryigit, R. Convolutional Neural Networks in Detection of Plant Leaf Diseases: A Review. Agriculture 2022
10. Joseph, Diana Susan, Pranav M. Pawar, and Rahul Pramanik. "Intelligent plant disease diagnosis using convolutional neural network: a review." Multimedia Tools and Applications 82.14 (2023): 21415-21481.
11. Harakannanavar, Sunil S., et al. "Plant leaf disease detection using computer vision and machine learning algorithms." Global Transitions Proceedings 3.1 (2022): 305-310.
12. Yadhav, S. Yegneshwar, et al. "Plant disease detection and classification using cnn model with optimized activation function." 2020 international conference on electronics and sustainable communication systems (ICESC). IEEE, 2020.
13. Menon, Vishal, V. Ashwin, and Raj K. Deepa. "Plant disease detection using cnn and transfer learning." 2021 International Conference on Communication, Control and Information Sciences (ICCISc). Vol. 1. IEEE, 2021.

14. Sun, Xuewei, et al. "Research on plant disease identification based on CNN." Cognitive Robotics 2 (2022): 155-163.

15. Agarwal, Mohit, Suneet Kr Gupta, and Kanad Kishore Biswas. "Development of Efficient CNN model for Tomato crop disease identification." Sustainable Computing: Informatics and Systems 28 (2020): 100407.

16. Sharma, Parul, Yash Paul Singh Berwal, and Wiqas Ghai. "Performance analysis of deep learning CNN models for disease detection in plants using image segmentation." Information Processing in Agriculture 7.4 (2020): 566-574.

17. Saleem, Muhammad Hammad, Johan Potgieter, and Khalid Mahmood Arif. "Plant disease detection and classification by deep learning." Plants 8.11 (2019): 468.

18. Panchal, Adesh V., et al. "Image-based plant diseases detection using deep learning." Materials Today: Proceedings 80 (2023): 3500-3506.

19. Thakur, Poornima Singh, Tanuja Sheorey, and Aparajita Ojha. "VGG-ICNN: A Lightweight CNN model for crop disease identification." Multimedia Tools and Applications 82.1 (2023): 497-520.

20. Li, Yang, Jing Nie, and Xuewei Chao. "Do we really need deep CNN for plant diseases identification?." Computers and Electronics in Agriculture 178 (2020): 105803.

21. Lu, Jinzhu, Lijuan Tan, and Huanyu Jiang. "Review on convolutional neural network (CNN) applied to plant leaf disease classification." Agriculture 11.8 (2021): 707.