A Report on

Plant Leaf Disease Detection System

Submitted for partial fulfilment of award of degree

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering (Data Science)

By

SHIKHAR SHARMA – 2100301540071

PIYUSH PANDEY – 2100301540052

UJJWAL SINGH – 2100301540079

YASH SRIVASTAVA – 2100301540087

Name of the guide

MS. RATI GOYAL

INDERPRASTHA ENGINEERING COLLEGE, GHAZIABAD,

Dr. A P J ABDUL KALAM TECHNICAL UNIVERSITY

LUCKNOW

May 2025

CERTIFICATE

This is to certify that the Project Report entitled, "**Plant Leaf Disease Detection System**" and submitted by **Shikhar, Piyush, Ujjwal,** and **Yash** in partial fulfilment of the requirement for the award of Bachelor of Technology degree in **Computer Science and Engineering (Data Science)** at **Inderprastha Engineering College, Ghaziabad** is an authentic work carried out by him under my supervision and guidance. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Ms. Rati Goyal	Dr. Sudhir Dawra
Assistance Professor	HOD CSE (Data Science)
Inderprastha Engineering College	Inderprastha Engineering College
Ghaziabad	Ghaziabad

Acknowledgement

We take this opportunity to thank our teachers and friends who helped us throughout the project. First and foremost, I would like to thank my guide for the project (Name of the Guide, designation, Department) for her/his valuable advice and time during development of project. We would also like to thank Dr. Sudhir Dawra (HOD, Computer Science and Engineering (Data Science)) for his constant support during the development of the project.

Shikhar Sharma	Piyush Pandey
2100301540071	2100301540052
Signature:	Signature:
Ujjwal Singh	Yash Srivastava
2100301540079	2100305140087
Signature:	Signature:

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:	Signature:
Shikhar Sharma	Piyush Pandey
2100301540071	2100301540052
Signature:	Signature:
Ujjwal Singh	Yash Srivastava
2100301540079	2100305140087

Abstract

Agriculture is essential for many nations' incomes. Diseases in plants caused by pathogens such as viruses, fungi, and bacteria cause global financial losses in agriculture. Effective disease management ensures crop quality and yield. Disease symptoms are often visible on various plant parts, with leaves being the most affected. Researchers have utilized computer vision, deep learning, few-shot learning, and soft computing techniques to automatically identify plant diseases using leaf images. These technologies help farmers act promptly to protect crops. By automating disease detection, these methods resolve limitations of traditional methods, enhancing both research speed and technology effectiveness. Additionally, molecular techniques have been developed to mitigate pathogenic threats. This review examines the use of machine learning, deep learning, and few-shot learning for automated plant disease detection, reviews diagnostic techniques and future advancements. The integration of these advanced techniques into agricultural practices not only aids in timely disease detection but also sup ports sustainable farming by reducing reliance on chemical treatments. By lever aging machine learning and molecular diagnostics, farmers can implement targeted interventions, minimizing environmental impact and improving resource efficiency. These innovations are vital in addressing the growing challenges of food security and climate change, ensuring the resilience of agricultural systems worldwide. The CNN model that we built achieves an accuracy of 81.83 %.

Keywords: *Deep learning, diagnosis, image processing, machine learning, and plant disease.*

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Abbreviation	Full form
KNN	K-Nearest Neighbour
SVM	Support Vector Machine
CNN	Convolutional Neural Network
GPDCNN	Global pooling dilated CNN
MCNN	Multilayer convolutional neural network
PSO	Particle Swarm Optimization
DCNN	Deep Convolution Neural Network
CCM	Color Co-occurrence Matrix
GLCM	Gray Level Co-occurrence Matrix
MER	Minimum Enclosing Rectangle
DWT	Discrete Wavelet Transform
SIFT	Scale Invariant Feature Transform
HOG	Histogram of Oriented Gradients
STL-10	Self-Taught Learning 10
ResNet	Residual Network
HSV	Hue Saturation Value
RGB	Red, Green, Blue
HDF	Hierarchical Data Format
AI	Artificial intelligence
MDP	Markov decision process
ML	Machine Learning
HTTP	HyperText Transfer Protocol
WSGI	Web Server Gateway Interface
PEP	Python Enhancement Proposal

BSD	Berkeley Source Distribution
AQR	Applied Quantitative Research
DBSCAN	Density-based spatial clustering of applications with noise
API	Application programming interface
RF	Random Forest
LR	Logistic Regression
VGG	Visual Geometry Group
JPEG	Joint Photographic Experts Group
BMP	Bitmap

CHAPTER 1
INTRODUCTION

The United Nations' Food and Agriculture Organization has reported a consistent increase in global hunger since 2015. Current estimates suggest that approximately 680 million people are undernourished, accounting for less than 9% of the global population. This represents an annual increase of 10 million people and a rise of around 120 million over the last decade. Moreover, over 85% of the global population depends on agriculture for sustenance, underscoring the critical need for efficient farming mechanisms. Plants also play a vital role in maintaining environmental balance by producing oxygen through photosynthesis. However, plant diseases, particularly those affecting leaves, can severely impact plant health and disrupt food production. A historical ex ample is the 1845 Irish Potato Famine, which caused 1.2 million deaths due to crop failure [1]. Laboratory techniques such as immunosorbent enzyme assays, isothermal amplification, and polymerase chain reaction (PCR) are commonly employed to detect plant diseases. Early detection, effective management, and prevention of plant diseases are essential. However, diagnosing diseases in large agricultural fields is challenging, requiring skilled personnel and visual inspection of plant leaves [2]. Farmers typically rely on their experience to identify symptoms, a process that is time-intensive, laborious, and demands specialized skills. Automated disease detection systems aim to assist non-experts, including non-pathologists and non-botanists. This review explores auto mated techniques utilizing image processing, machine learning, deep learning, and few shot learning for plant disease detection. Traditional machine learning approaches often lack robustness and are confined to controlled laboratory settings [3]. In contrast, deep learning has recently demonstrated remarkable success in classifying plant disease images. However, deep learning methods require extensive datasets, with images meticulously annotated by pathologists and botanists. These processes are resource-intensive and costly.

Few-shot learning (FSL) offers an alternative by enabling models to learn from limited labeled datasets [4], where the number of samples depends on the experiment's objectives and complexity.

Various pathogens contribute to plant diseases and can be identified using molecular techniques such as DNA analysis, PCR, MPG (Modified Panchayat Mixture), ELISA (Enzyme-Linked

Immunosorbent Assay), FISH (Fluorescence in Situ Hybridization), and IF (Immunofluorescence) methods. This review paper provides a comparative analysis of machine learning, deep learning, and few-shot learning in plant disease detection. It also examines segmentation, feature extraction, and classification techniques alongside molecular diagnostic tools [5]. A typical symptom can be seen in (Fig. 1), which depicts an infected leaf.

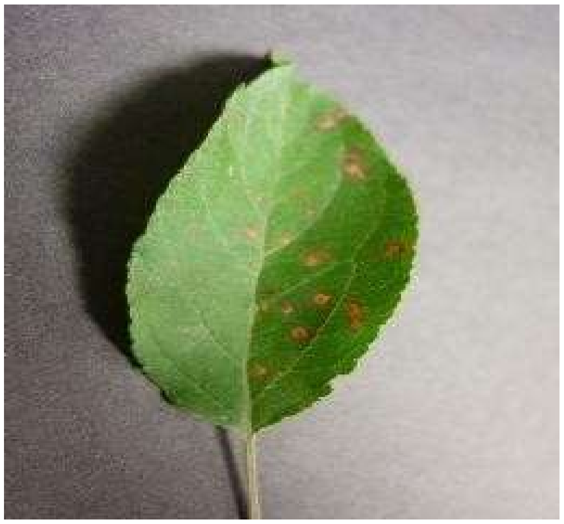


Fig. 1.1 Apple leaf infected with rust

1.1 Project Definition

The Plant Leaf Disease Detection project aims to develop an intelligent system that automatically identifies and classifies diseases present on plant leaves using artificial intelligence (AI) and computer vision techniques. The system will process images of plant leaves to detect visual symptoms of various diseases and accurately classify them into predefined categories. This project leverages deep learning models, particularly convolutional neural networks (CNNs), and may include alternative machine learning methods like SVM or XGBoost for comparative evaluation. The goal is to assist farmers and agricultural professionals in early disease detection, reduce crop loss, and support timely and effective disease management.

1.2 Background about the project idea

Agriculture plays a vital role in the global economy, particularly in countries where a significant portion of the population relies on farming for their livelihood. However, one of the major challenges faced by farmers is the early detection and effective management of plant diseases, which can severely impact crop yield and quality. Traditionally, identifying plant diseases requires expert knowledge, regular monitoring, and manual inspection, which may not be practical or accessible for all farmers, especially in remote or rural areas. With recent advancements in artificial intelligence (AI) and computer vision, it is now possible to automate the process of plant disease detection using image-based analysis. By training machine learning models—especially convolutional neural networks (CNNs)—on large datasets of plant leaf images, systems can learn to recognize disease patterns and classify them with high accuracy. This project aims to develop a robust and intelligent solution for detecting and classifying plant leaf diseases. The system uses images of affected leaves as input and predicts the type of disease present, if any. Such a tool can empower farmers with quick and accurate information, enabling timely intervention and minimizing crop damage. The implementation of this technology has the potential to improve agricultural productivity and contribute to sustainable farming practices.

1.3 Project Objective

The objective of the project "Plant Leaf Disease Detection and Classification Using AI and Computer Vision Techniques" is to develop an intelligent system capable of identifying and classifying plant leaf diseases accurately and efficiently. By leveraging advanced AI algorithms and computer vision, the system aims to assist farmers and agricultural professionals in detecting diseases at an early stage, reducing crop losses, and improving productivity.

Fig 1.2 Farmer’s Usage

1.4 Feasibility

1.4.1 Technical Feasibility

Availability of Technology: The project leverages readily available technologies, such as deep learning frameworks (e.g., TensorFlow, PyTorch), computer vision libraries (e.g., OpenCV), and cloud computing platforms for scalable deployment.

Data Requirements: Datasets of diseased and healthy plant leaves are widely available in agricultural research repositories. Additionally, smartphone cameras and digital imaging devices make it easy to collect further data.

Conclusion: Technically feasible with current tools, resources, and expertise.

1.4.2 Economic Feasibility

Development Costs: Initial costs include data acquisition, model training, and system development. These can be minimized using open-source tools and publicly available datasets.

Operational Costs: Cloud deployment and model updates will incur recurring expenses, but these are scalable based on user demand.

Conclusion: Economically feasible with potential for substantial ROI (Return of Investment).

1.4.3 Operational Feasibility

User Accessibility: The system can be deployed as a mobile application or web-based platform, making it easily accessible to farmers, agronomists, and agricultural stakeholders.

Ease of Use: Intuitive interfaces with image upload options and simple outputs (e.g. disease name, severity, and recommendations) will ensure widespread adoption.

Conclusion: Operationally feasible with user-friendly design and scalable deployment.

1.4.4 Legal and Ethical Feasibility

Data Privacy: Ensuring the anonymity and security of images uploaded by users will be crucial. Standard data protection measures will address privacy concerns.

Regulatory Compliance: The system must comply with regional agricultural policies and technology regulations, especially if pesticides or treatments are recommended.

Bias and Fairness: Ensuring the AI model is trained on diverse datasets to avoid bias is essential to ensure fair and accurate predictions.

Conclusion: Feasible with adherence to ethical AI practices and data protection laws.

Overall Feasibility: The project is technically, economically, operationally, and legally feasible. Its implementation can have a transformative impact on agriculture, offering a cost-effective, scalable, and accessible solution for plant disease management.

1.5 Phytopathology

Phytopathology refers to the study of plant pathogens, the diseases they cause, their mechanisms, and methods to control and mitigate their impact on plants. It serves as a comprehensive framework for understanding and managing a plant's life cycle. Derived from Greek, "Phytopathology" combines "Phyto" (plant), "Patho" (disease), and "Logo" (knowledge) [6]. Its core objectives include investigating the origins and causes of plant diseases, whether biotic or abiotic (etiology), understanding the mechanisms behind disease development (pathogenesis), examining interactions between plant pathogens and diseases (epidemiology), and developing strategies to reduce damage and manage losses, as depicted in Fig. 1.3.

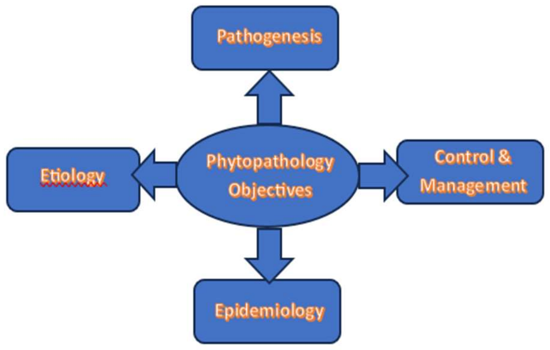


Fig. 1.3 Phytopathology Objectives

Phytopathology is a specialized subfield within agricultural science that integrates foundational knowledge from diverse disciplines such as microbiology, physiology, nematology, virology, anatomy, bacteriology, mycology, genetic engineering, botany, meteorology, climatology, and molecular biology, as illustrated in Fig. 1.4.

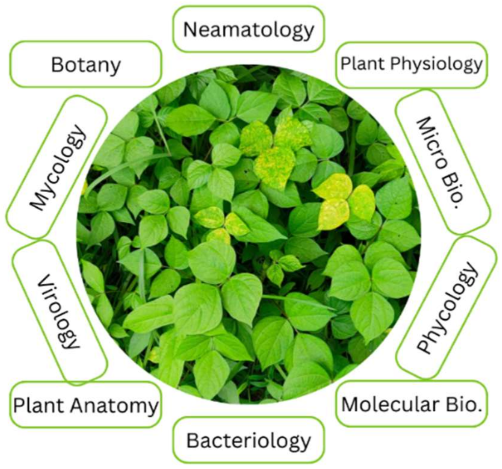


Fig. 1.4 Subdomains of phytopathology

1.5.1 Plant disease types & symptoms.

Bacterial Diseases. Bacterial infections in plants typically begin as water-soaked lesions that develop into small green blemishes. Over time, these lesions expand and dry into dead spots, as illustrated in Fig. 6. For instance, foliage may display water-soaked black blemishes, brown leaf spots, or yellow halos of uniform size. Under dry conditions, the blemishes often appear dappled. Bacterial wilt, a common issue in brinjal crops, causes the entire plant to collapse [7, 21].

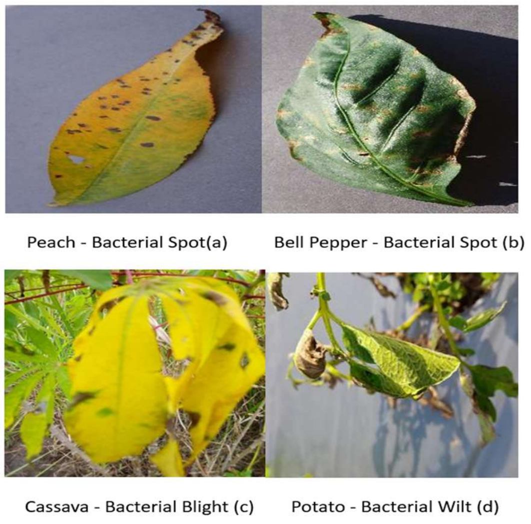

Peach - Bacterial Spot(a)
Bell Pepper - Bacterial Spot (b)
Cassava - Bacterial Blight (c)
Potato - Bacterial Wilt (d)

Fig. 1.5 Bacterial Diseases

Viral Diseases. Viral infections in plants are among the most challenging to diagnose, as they may exhibit no visible symptoms or mimic signs of herbicide damage or nutrient deficiencies [7].

Commonly observed viral disease include those transmitted by beetles, leafhoppers, aphids, and whiteflies, such as mosaic viruses, which manifest as green or yellow streaks on foliage, as shown in Fig. 1.6.

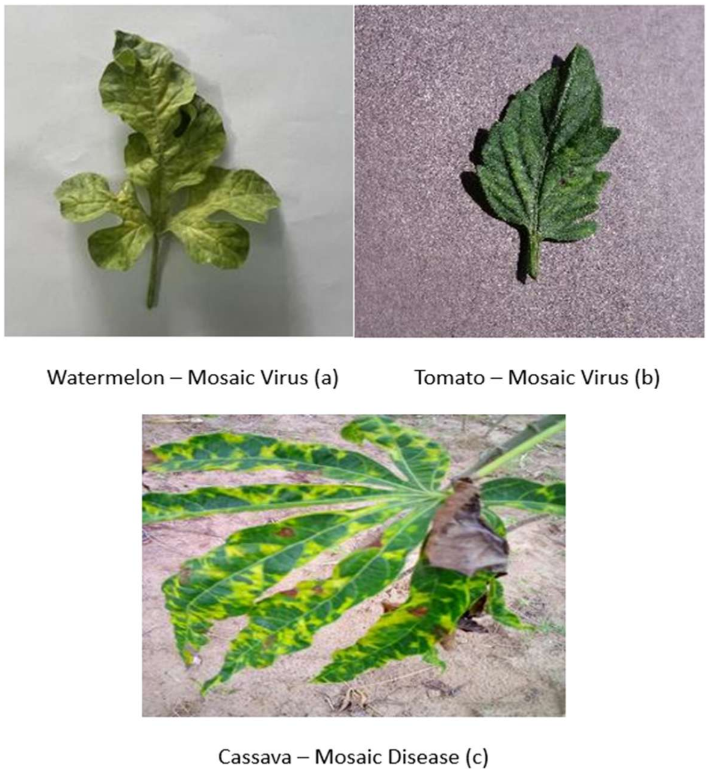

Watermelon – Mosaic Virus (a)
Tomato – Mosaic Virus (b)
Cassava – Mosaic Disease (c)

Fig. 1.6 Viral Diseases

Fungal Diseases. Fungal infections affect various parts of plants, including stems, leaves, seeds, and roots. Examples include sclerotium wilt, stem rust, blight, ergot, and carnal bunt. Late blight caused by Phytophthora fungus initially appears as gray green waterlogged blemishes on older leaves, as shown in Fig. 8 (a). Over time, these lesions darken, and white fungal growth may appear due to fluctuating wet and dry conditions [8]. Early blight caused by Alternaria fungus produces small brown blemishes with a characteristic concentric ring pattern, as shown in Fig. 8 (b). Rust fungi form spots on mature leaves that turn black over time, as illustrated in Fig. 1.7.

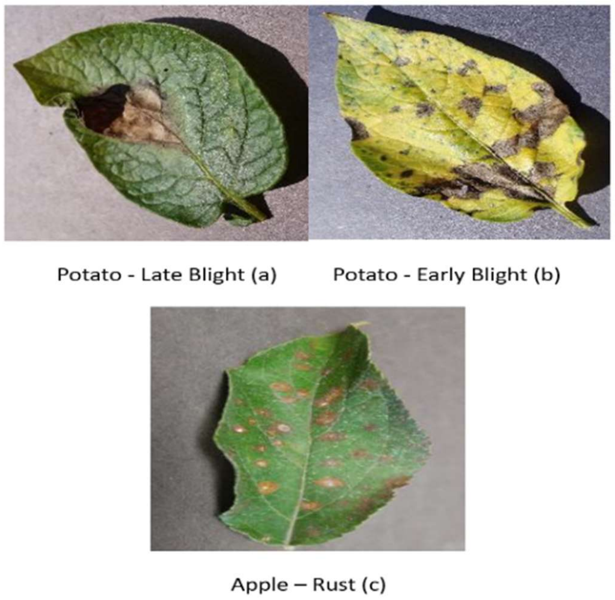


Fig. 1.7 Fungal Diseases

1.6 Importance of Plant Disease Detection

Accurate identification of plant diseases is essential for effective disease control. Without it, control measures may be misdirected, leading to wasted time, effort, and resources. In some cases, applying incorrect treatments can worsen the problem and cause further crop loss. Plant diseases are primarily caused by infectious agents such as fungi, bacteria, viruses, nematodes, and oomycetes. Since different pathogens can produce similar symptoms, precise diagnosis is critical for developing an effective management strategy. Additionally, it is important to distinguish between plant injuries and diseases—while injuries are typically caused by sudden external forces over a short period, diseases are the result of ongoing infections or infestations.

1.7 Techniques for disease detection

1.7.1 Machine Learning Methods

1.7.1.1 KNN

K-Nearest Neighbors (K-NN) is one of the simplest and most intuitive supervised machine learning algorithms. It assumes that similar data points exist close to each other in feature space. When presented with a new data point, the K-NN algorithm compares it to existing instances and assigns it the most common class among its 'K' nearest neighbors.

K-NN is a **lazy learner** algorithm, meaning it does not learn from the training data immediately. Instead, it stores the data and only performs computation when it is asked to classify a new instance. This makes K-NN a **non-parametric** method, as it does not make assumptions about the underlying data distribution.

Though K-NN can be used for both classification and regression problems, it is more commonly applied to classification tasks.

Example: During training, K-NN simply stores the dataset. When a new input is provided, the algorithm finds the 'K' closest points in the dataset and assigns the class that is most common among them.

1.7.1.2 Support Vector Machine

Support Vector Machine (SVM) is a powerful supervised learning algorithm primarily used for classification tasks, although it can also be used for regression. SVM's goal is to find the optimal hyperplane that best separates data points of different classes in an n-dimensional space.

The algorithm identifies the support vectors, which are the data points closest to the hyperplane and most influential in defining its position. These support vectors play a crucial role in the creation of the decision boundary.

Types of SVM:

- Linear SVM: Used when the data is linearly separable and can be divided using a single straight line (or hyperplane).
- Non-linear SVM: Used for datasets that are not linearly separable. In such cases, SVM applies kernel functions to transform the data into a higher dimension where it becomes linearly separable.

Key Concepts:

- Hyperplane: A decision boundary that separates different classes. The dimensionality of the hyperplane depends on the number of features. For example, with two features, the hyperplane is a straight line; with three, it becomes a plane.
- Support Vectors: The critical data points that are closest to the hyperplane and influence its orientation. They are the "support" of the SVM model.

1.7.1.3 Random Forest

Random Forest is a versatile and widely used supervised machine learning algorithm that can handle both classification and regression tasks. It is based on the concept of ensemble learning, which combines the predictions of multiple models to improve performance.

The Random Forest algorithm constructs multiple decision trees using different subsets of the training data and features.

The final prediction is made based on the majority voting (for classification) or averaging (for regression) of individual trees' outputs. This reduces the risk of overfitting compared to a single decision tree and improves accuracy and robustness.

Advantages:

- Handles large datasets with higher dimensionality
- Reduces overfitting
- Improves prediction accuracy

1.7.1.4 Logistic Regression

Logistic Regression is a popular supervised learning algorithm used primarily for binary and multiclass classification problems. It predicts the probability of a categorical dependent variable based on one or more independent variables.

Unlike linear regression, which provides continuous output, logistic regression uses the logistic (sigmoid) function to map predictions to probabilities between 0 and 1.

Key Characteristics:

- Predicts categorical outcomes such as Yes/No, True/False, or 0/1.
- Outputs probabilities rather than exact values.
- Often used in medical, financial, and social science applications.

Sigmoid Function (Logistic Function):

- Converts real-valued input into a probability score between 0 and 1.
- Produces an "S"-shaped curve.

- Ensures that predictions stay within logical bounds.
- A threshold (e.g., 0.5) is used to classify the output into one of two categories.

Assumptions in Logistic Regression:

- The dependent variable must be categorical.
- Independent variables should not exhibit multicollinearity.

Types of Logistic Regression:

1. Binomial Logistic Regression: The dependent variable has two categories (e.g., Pass/Fail, Yes/No).
2. Multinomial Logistic Regression: The dependent variable has three or more unordered categories (e.g., Cat, Dog, Sheep).
3. Ordinal Logistic Regression: The dependent variable has three or more ordered categories (e.g., Low, Medium, High).

1.7.2 Deep Learning Techniques

1.7.2.1 The CNN (Convolutional Neural Network) Technique:

CNNs are deep feed-forward neural networks designed to analyze multidimensional data. They learn feature representations by activating specific channels when detecting key spatial patterns in the input data [15, 17, 18]. The accuracy of CNN models depends on factors such as the number of training epochs and the implementation of convolution filters, typically with dimensions of 2 × 2 or 3 × 3. Several pre-trained architectures enhance CNN performance, including VGG16, VGG19, ResNet50, ResNet152, InceptionV3, InceptionNet, and DenseNet121.

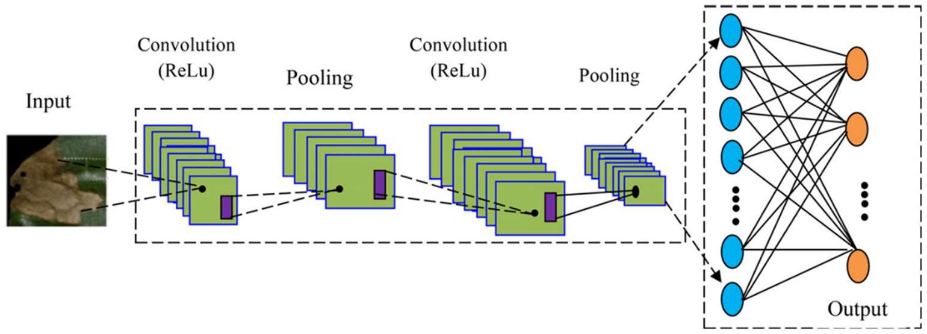


Fig. 1.8 CNN Image Processing Architecture

CHAPTER 2

LITERATURE REVIEW

This section provides a critical review of the existing literature on plant disease identification and classification. Research in this area can be broadly categorized into two approaches: machine learning (ML) and deep learning (DL) [6]. Early detection of crop health issues and diseases is crucial for effective management, including the use of fungicides, disease-specific treatments, and pesticide-based vector control. Beyond disease prevention, these measures can enhance agricultural productivity. Accurately distinguishing between healthy and diseased leaves is essential for reducing crop losses and increasing yield [7].

Machine learning techniques are increasingly being used for plant disease identification, as discussed in this section (Table 1). A comparison is presented between two widely adopted approaches—deep learning (DL) and machine learning (ML)—for plant disease detection using leaf image data [8]. Previously, many image-processing techniques relied on basic machine learning architectures [9]. However, deep learning networks are progressively becoming the industry standard for pattern recognition and image analysis. This comparison evaluates both approaches under standard conditions, considering three key factors: model design, processing capacity, and training data volume.

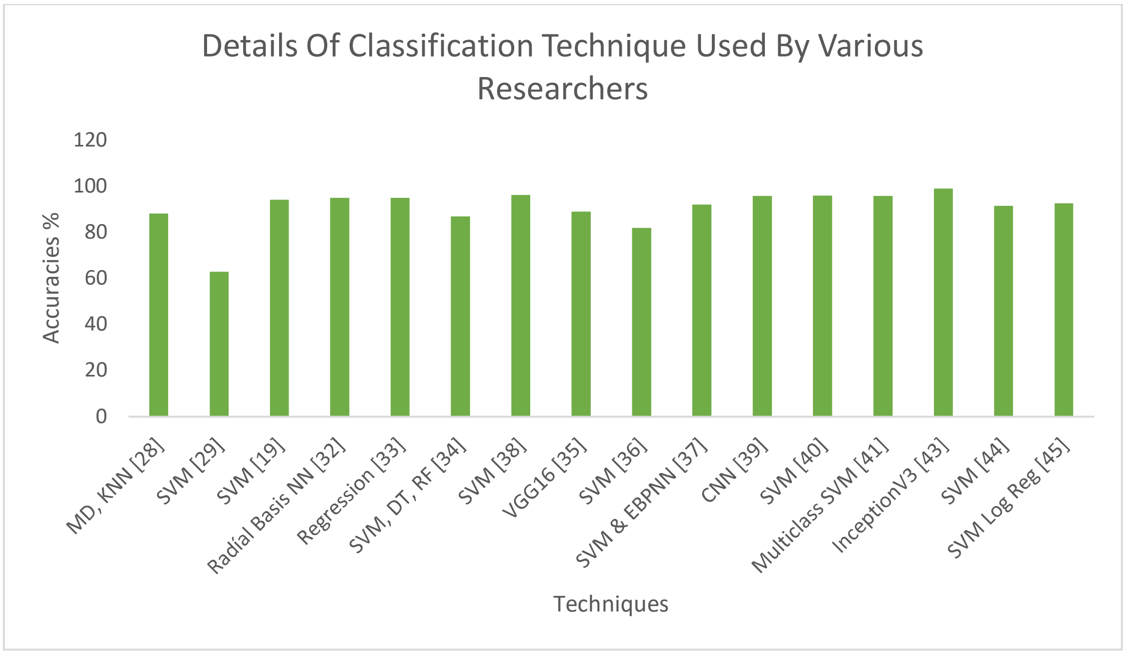


Fig 2.1 Classification Techniques by Various Researchers

Deep learning techniques are proving highly effective in agricultural image analysis. Study [10] demonstrated real-time insect detection and identification in soybean crops, achieving 98.75% accuracy and a processing speed of 53 frames per second using the YoloV5 model. This was accomplished with a custom dataset of crop insects, simplifying pest management for producers. Similarly, study [11] utilized a convolutional neural network (CNN) to classify plant leaf diseases from the PlantVillage dataset, achieving approximately 98% accuracy in both training and testing. This classification covered 15 categories, including healthy leaves and various diseases caused by bacteria and fungi. Shrivastava et al. [12] utilized transfer learning with deep convolutional neural networks (CNNs) to classify rice plant diseases. Their system identified four categories: rice blast, bacterial leaf blight, sheath blight, and healthy leaves. The classification process involved CNNs for feature extraction, followed by support vector machines (SVMs) for final classification. Aderghal et al. [13] employed cross-modal transfer learning with deep CNNs to classify Alzheimer's disease using imaging modalities. Their approach aimed to enhance accuracy and was compared favourably to existing methods. The classification relied on CNNs.

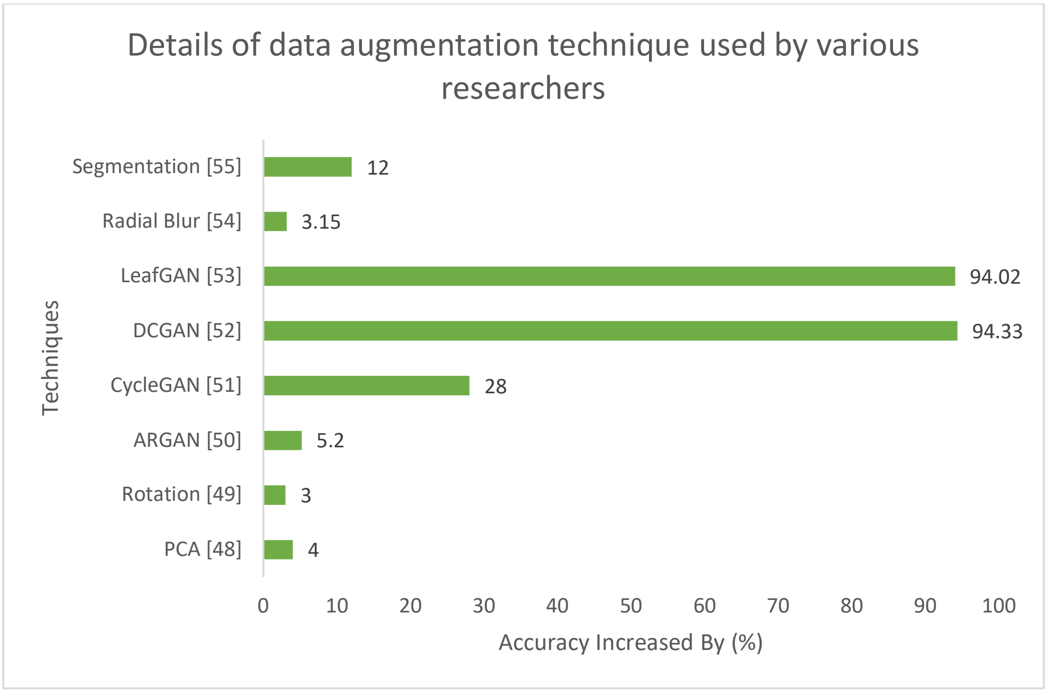


Fig 2.2 Data Augmentation Technique by Various Researchers

Years	**Literature Sources**	**Methodology for selecting features**	**Classification Approach**	**Datasets**	**Accuracy (%)**
2022	Aliyu M. Abdu et al.	SVM, DL	Supervised learning	Plantvillage dataset	95.30%

2022	Savvas Dimitriadis	Naïve Bayes	Supervised learning	Kaggle dataset	89.20%
2022	Asma Akhtar et al.	K-Mean, KNN, RNN	Unsupervised learning	Plant leaf dataset	87.42%
2021	Kiran R. Gavhale et al	ANN, CNN	Supervised learning	Leaf images	91.51%
2021	Jagadeh D. Pujari et al.	PNN, SVM, GLCM	Supervised learning/unsupervised	Image Data Set	96.51%
2021	Sharada P.Mohanty	Neural Network	Supervised learning	Image Net	83.13%
2020	Srdjan Sladojevic	SVM, Neural Network, Decision Tree	Supervised learning	Pictures of Plant Pictures Potato Village Tomato Pictures Apple Pictures Rice	94.31%
2020	P.Ferntions	Neural Network	Supervised learning	Alex Net, Google Net, Over-Feat, VGG	97%
2020	S. Sannakkiet et al.	SVM	Supervised learning	Image Dataset	85%
2019	TejalChandi waeetet al.	ANN SVM	Supervised learning	Manual Dataset	82.34%
2019	Yan Guo et al.	ANN SVM	Supervised learning	Plant Village	96.45%
2019	Yan Guo et al.	CNN	Supervised learning	Apple dataset	93.81%

Table 2.1 Literature Review

Author	Plant Name	Bacterial Disease	Viral Disease	Fungal Disease
Kianat et al. 2021 [9] Agarwal et al. 2021 [10]	Cucumber	Brown Blemnish, Angular Blemnish, Target Blemnish	Mosaic, Yellow Blemnish	Black Blemish, Gray Mold
Shrivastava et al. 2019 [11]	Rice	Streak, Blight	Black Dwarfed Streaked	Smut False

Chen et al. 2021 [12]				
Sun et al. 2021 [13]	Maize	Streak, Stalk	Crimson, Dwarf	Rust
Abbas et al. 2021 [14]	Tomato	Canker	Curl leaf yellow	Late/Early Blight

Table 2.2 Distinct disease in different plants

Plants	Diseases	Pathogens	Symptoms
Apple	Scab Rot Rust	PomiSpilocal MalorumSphaeropsis Sporangium	Brown-Gray on leaf Dark Brown on leaf Yellow pale on leaf
Cherry	Mildew	Clandestina	Gray powder on leaf
Corn	Gray Spot Rust Light blight	Cercospora Sorghipuccinia Tutcicasetosphaeria	Rectangle lesions Red pustules on leaf Elliptical lesions
Grape	Rot Measles Isariopsis blight	Bidwelliiguignardia Aleophilum Angulata brachypus	Red borders on leaf Necrotic stripping Coalesce lesions
Peach	Spot	Arboricola Xanthomonas	Clustered lesions
Potato	Early blight Late blight	Solani Alternaria Infestans phytophthora	Brown lesion Dark greeb spot
Tomato	Septoria spot Mosaic	Lycopersici Mosaic virus	Foliage Mottle green leaf
Orange	Green Citrus	Bacteria Motile	Precipitate Demolition

Strawberry	Scorch Fungus	Diplocarpon	Brown edges
Squash	Mildew	Xanthiipodosphaers	White powder

Table 2.3 Distinct plants, their disease and responsible pathogen

	Technique Used	Authors
Color Space Conversion	Enhancement Filtering, Background reduction RGB, HSV, HSI, YIQ, L*a*b, grayscale	Kaur et al. 2018 [22],
Image Enhancement Technique	Denoising Using mean and median filtering Illumination variation using histogram equalization	Goncharov et. al. 2019 [23]
Thresholding techniques	Thresholding, Entropy, classification of diseases and pests	M. Francisco et al. 2023 [24]
Clustering	k-means	Kaur et al. 2018b [22], Bashir et al. 2019 [19]
Feature Descriptor	GLCM, Wavelet Transform, Haralick feature, Gabor Transform, Local Binary Patterns, SURF	Bhagat & Kumar 2023 [25]
Texture Feature	GLCM Features	Deshapande et al. 2019 [26],
Color feature	Color co-occurrence matrix	Chouhan et al. 2019 [27]
Feature Descriptor	discrete wavelet transforms and SVM	S. M. Kiran et al. 2021 [28]

Table 2.4 Details of techniques used by various researchers

Author	Datasets	Techniques	Accuracy increased (%)
Bin et al. 2017 [48]	1053 – 13689	PCA	4.00
Srdjan et al. 2016 [49]	4483 – 33469	Rotation	3.00
Nazki et al. 2020 [50]	2789	ARGAN	5.20
Tain et al. 2019 [51]	TeaImage	CycleGAN	28.00
Wu et. al. 2020 [52]	GoogleNet	DCGAN	94.33
Liu et. al. 2020 [53]	Grape	LeafGAN	94.02
Lin et al. 2019 [54]	10820 – 32460	Radial Blur	3.15
Arnal Barbedo 2019 [55]	1567 – 46409	Segmentation	12.00

Table 2.5 Details of data augmentation technique is used by various researchers

CHAPTER 3
SOFTWARE REQUIREMENT ANALYSIS

Software Requirement Analysis is the process of understanding and documenting the needs and expectations of stakeholders for a software system. It involves identifying, analyzing, and documenting the functional and non-functional requirements that the software must fulfil to meet the needs of users and other stakeholders.

3.1 Functional Requirements

3.1.1 Image Upload and Input

- Users must be able to upload images of plant leaves via a web or mobile application.
- Support for common image formats (e.g., JPEG, PNG).
- Option to capture images directly using a device's camera.

3.1.2 Image Processing and Disease Detection

- Preprocess the uploaded image (e.g., resizing, noise reduction, normalization).
- Analyze the image using trained AI models to detect and classify diseases.
- Return disease classification results with confidence scores.

3.2 System Requirements

3.2.1 AI Model and Libraries

- Model: Pretrained Convolutional Neural Networks (CNNs) such as ResNet, MobileNet, or custom-trained models.
- Libraries: TensorFlow, PyTorch, or Keras for model development.

3.2.1.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of deep learning algorithm primarily used for image recognition and classification tasks. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input images through layers of convolutions, pooling, and fully connected layers.

The key components of a CNN include:

- Convolutional Layers: Extract feature maps by applying filters (kernels) to the input image.
- Pooling Layers: Reduce the dimensionality of feature maps, helping to retain essential information and improve computational efficiency.
- Fully Connected Layers: Act as classifiers that process the extracted features to produce the final output (e.g., predicted class).

CNNs are widely used in applications like facial recognition, medical image analysis, object detection, and plant disease detection due to their high accuracy and ability to handle large-scale image data.

3.2.1.2 ResNet

ResNet is a deep convolutional neural network architecture introduced by Microsoft in 2015. It addresses the problem of vanishing gradients in very deep networks by using residual blocks—shortcut connections that skip one or more layers. These connections allow the network to learn identity functions, making it easier to train very deep models. ResNet has been widely used in image classification tasks and has won the ImageNet competition. Common variants include ResNet-18, ResNet-50, and ResNet-101.

3.2.1.3 MobileNet

MobileNet is a lightweight deep learning model designed for mobile and embedded devices. It uses depthwise separable convolutions to reduce the number of parameters and computational cost, making it efficient for real-time applications on devices with limited processing power. MobileNet is commonly used in mobile vision applications like face detection, object recognition, and image classification.

3.2.1.4 Tensorflow

TensorFlow is an open-source machine learning and deep learning framework developed by Google Brain.

It is widely used in industry and academia for building, training, and deploying machine learning models at scale. TensorFlow supports both low-level operations (like building custom models layer by layer) and high-level APIs (like Keras) for ease of development.

TensorFlow allows users to build computational graphs where operations are represented as nodes. It supports both static and dynamic graphs, though the default is static. TensorFlow excels in production deployment, offering tools like TensorFlow Serving, TensorFlow Lite (for mobile), and TensorFlow.js (for the web).

Key Features:

- Scalable for both CPUs and GPUs
- Strong support for mobile and embedded devices
- TensorBoard for visualization and model debugging
- Large community and extensive documentation

3.2.1.5 PyTorch

PyTorch is an open-source deep learning framework developed by Facebook's AI Research (FAIR) lab. It is known for its dynamic computation graph, meaning the graph is built on-the-fly as operations are executed—making it more intuitive and easier to debug than static-graph frameworks.

PyTorch has become extremely popular among researchers and developers due to its Pythonic syntax, flexibility, and integration with other libraries. It also supports GPU acceleration for high-performance computing and has tools like TorchVision for computer vision tasks and TorchText for NLP.

Key Features:

- Dynamic computation graph (eager execution)
- Intuitive and flexible API
- Strong support for research and experimentation
- Seamless integration with NumPy and Python ecosystem
- Widely adopted in academia and increasingly in industry

3.2.1.6 Keras

Keras is a high-level neural networks API written in Python and capable of running on top of TensorFlow, Theano, or CNTK.

It is designed to enable fast experimentation with deep learning models. Keras simplifies the process of building and training neural networks through an easy-to-use and modular interface, making it ideal for beginners and rapid prototyping.

3.3 Cloud Services

Deployment on AWS, Azure, or Google Cloud for hosting and scalability.

3.3.1 Amazon Web Services

Amazon Web Services (AWS) is the world's leading cloud computing platform, launched by Amazon in 2006. It offers over 200 fully featured services across computing, storage, databases, networking, machine learning, artificial intelligence, Internet of Things (IoT), and more.

Key Services:

- EC2 (Elastic Compute Cloud) – for scalable virtual servers.
- S3 (Simple Storage Service) – for object storage.
- RDS (Relational Database Service) – for managed relational databases.
- Lambda – for serverless computing.
- SageMaker – for building, training, and deploying ML models.

Advantages:

- Extensive global infrastructure with data centers across multiple regions.
- Broadest and most mature set of cloud services.
- Deep support for enterprise-grade security, compliance, and identity management.
- Strong ecosystem and marketplace.

Use Cases:

- Web and mobile app hosting
- Big data processing
- Machine learning and AI workloads
- Cloud-native application development

3.3.2 Microsoft Azure

Microsoft Azure, launched in 2010, is a robust and enterprise-friendly cloud platform by Microsoft. It provides a comprehensive suite of cloud services for building, testing, deploying, and managing applications through Microsoft-managed data centres.

Key Services:

- Azure Virtual Machines – for scalable compute capacity.
- Azure Blob Storage – for unstructured data storage.
- Azure SQL Database – for fully managed relational databases.
- Azure Functions – for event-driven serverless computing.
- Azure Machine Learning – for building and deploying AI models.

Advantages:

- Seamless integration with Microsoft products like Windows Server, SQL Server, Active Directory, and Office 365.
- Strong support for hybrid cloud and on-premises environments via Azure Arc.
- Excellent developer tools and SDKs through Visual Studio and .NET.
- Strong enterprise and government adoption.

Use Cases:

- Enterprise cloud migration
- Hybrid cloud deployment
- DevOps and CI/CD automation
- Artificial intelligence and data analytics

3.3.3 Google Cloud Platform

Google Cloud Platform (GCP) is Google's suite of cloud computing services launched in 2011. It is known for its innovation in data analytics, artificial intelligence, and open-source technologies. GCP leverages Google's core infrastructure to provide highly scalable and high-performance services.

Key Services:

- Compute Engine – for customizable virtual machines.
- Cloud Storage – for scalable object storage.
- BigQuery – for fast, SQL-based analytics on big data.
- Cloud Functions – for lightweight, serverless computing.
- Vertex AI – for machine learning model development and deployment.

Advantages:

- Best-in-class data analytics and ML tools.
- Strong support for Kubernetes (Google originally developed Kubernetes).
- Cost-effective and developer-friendly tools.
- Integration with open-source and multi-cloud platforms like Anthos.

Use Cases:

- Real-time data processing and analytics
- Machine learning and AI model deployment
- Containerized application development with Kubernetes
- Web and mobile app hosting

3.4 Testing Frameworks

AI Model testing: Confusion matrix, accuracy metrics

3.4.1 Confusion matrix

A Confusion Matrix is a tabular summary of classification results. It compares the predicted labels with the actual labels and helps in understanding the types of errors made by the classifier. It is especially useful in binary and multi-class classification tasks.

Structure of a Confusion Matrix (Binary Classification):

	Predicted: Positive	Predicted: Negative
Actual: Positive	True Positive (TP)	False Negative (FN)
Actual: Negative	False Positive (FP)	True Negative (TN)

Terminologies:

- True Positive (TP): Model correctly predicts the positive class.
- True Negative (TN): Model correctly predicts the negative class.
- False Positive (FP): Model incorrectly predicts positive when it's actually negative (Type I Error).
- False Negative (FN): Model incorrectly predicts negative when it's actually positive (Type II Error).

Usefulness:

- Allows deeper analysis beyond overall accuracy.
- Helps identify imbalances and misclassification trends.
- Supports calculation of several performance metrics.

3.4.2 Accuracy metrics

Using the values from the confusion matrix, several performance metrics can be calculated to measure the model's effectiveness.

Accuracy

- Indicates the overall correctness of the model.
- Can be misleading in imbalanced datasets.

Precision

- Indicates how many predicted positives were actually positive.
- Important in cases where false positives are costly (e.g., spam detection).

Recall (Sensitivity or True Positive Rate)

- Indicates how many actual positives were correctly predicted.
- Important in cases where false negatives are costly (e.g., disease detection).

F1 Score

- Harmonic mean of precision and recall.
- Balances both false positives and false negatives.

Specificity (True Negative Rate)

- Measures the proportion of actual negatives correctly identified.

ROC Curve and AUC

- ROC Curve (Receiver Operating Characteristic): Plots TPR vs FPR at different threshold levels.
- AUC (Area Under Curve): Measures the area under the ROC curve; higher AUC indicates better model performance.

3.4.3 Robustness tests

Robustness testing evaluates how well a model maintains its performance under varying conditions or in the presence of noise and uncertainty. It ensures that the model is not only accurate on test data but also reliable in real-world scenarios.

Types of Robustness Tests:

Noise Tolerance Test

- Test model's performance after injecting noise into the input data.
- Useful in image, audio, or sensor data applications.

Adversarial Attacks

- Apply small, intentional perturbations to data to see if the model misclassifies.
- Ensures security and trustworthiness of models in critical applications.

Cross-Domain or Cross-Dataset Testing

- Evaluate the model on a slightly different dataset (different distribution, same task).
- Assesses model generalization.

Data Augmentation Tests

- Use augmented or synthetic data (rotation, flipping, scaling in images) to test how consistent the model's predictions are.

Model Drift Testing

- Over time, the data distribution may change (concept drift). Testing for robustness ensures the model remains reliable over time.

3.5 Other requirements

3.5.1 Integration Requirements

External APIs: Agricultural databases for updated disease management practices.

Data Sources: Integration with research datasets for model training and validation.

3.5.2 Hardware Requirements

Local Machine Requirements: A machine with at least 8GB RAM and a processor (Intel i3/i5/i7 or AMD Ryzen 5/7).

CHAPTER 4

SYSTEM ANALYSIS & DESIGN

4.1 Class Designs – Description

Class design refers to the blueprint or template for creating objects in an object-oriented program. It defines the attributes (data) and methods (functions) of each class that will be part of the system. In this project, class designs ensure a modular, reusable, and maintainable architecture.

- Purpose: To model real-world entities like users, images, classifiers, and diseases.
- Example: A DiseaseClassifier class might contain a method to load the trained model and another method to make predictions based on an input image.

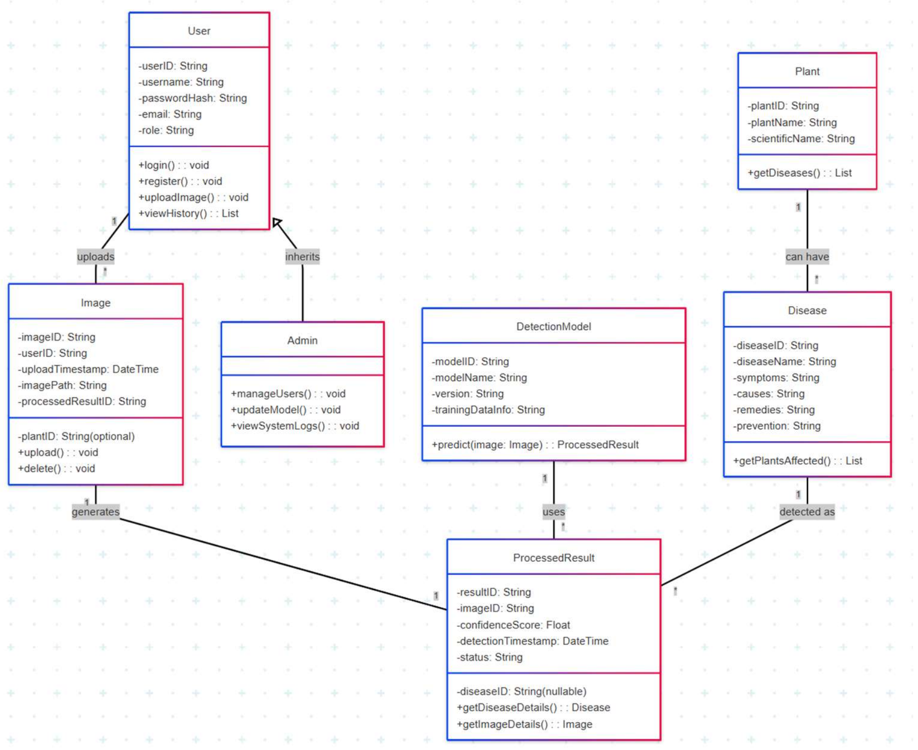


Fig 4.1 Class Diagram

4.2 Sequence Diagrams – Description

A sequence diagram is a UML behavioral diagram that shows how objects interact in a time sequence. It emphasizes the order in which messages are sent between system components.

- Purpose: To visualize the interaction between the user, system components (e.g., image processor, classifier), and the database over time.
- Example: The flow from image upload → preprocessing → prediction → fetching disease info → displaying results is captured step-by-step.

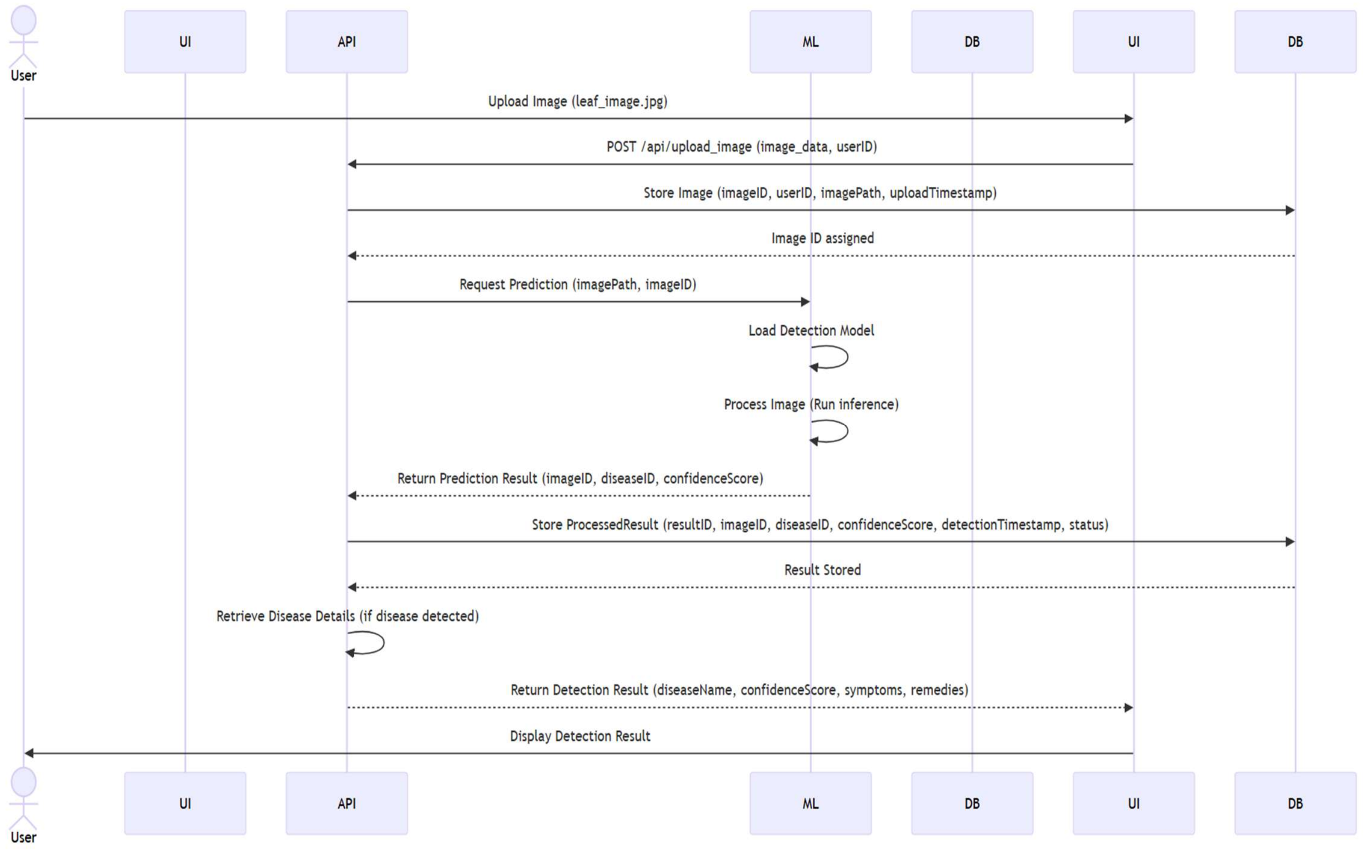

User
UI
API
ML
DB
UI
DB
Upload Image (leaf_image.jpg)
POST /api/upload_image (image_data, userID)
Store Image (imageID, userID, imagePath, uploadTimestamp)
Image ID assigned
Request Prediction (imagePath, imageID)
Load Detection Model
Process Image (Run inference)
Return Prediction Result (imageID, diseaseID, confidenceScore)
Store ProcessedResult (resultID, imageID, diseaseID, confidenceScore, detectionTimestamp, status)
Result Stored
Retrieve Disease Details (if disease detected)
Return Detection Result (diseaseName, confidenceScore, symptoms, remedies)
Display Detection Result
User
UI
API
ML
DB
UI
DB

Fig 4.2 Sequence Diagram

4.3 Activity Diagrams – Description

Activity diagrams represent workflow or business processes. They show the flow of control from one activity to another and are useful for modelling the logic of complex operations.

- Purpose: To describe the dynamic aspects of the system, especially processes like disease detection and result generation.
- Example: An activity diagram might illustrate how the image goes through preprocessing, model prediction, disease info retrieval, and result display.

4.4 Data Flow Diagrams (DFDs) – Description

A DFD illustrates how data flows through a system. It shows data sources, processes, storage, and destinations, helping to understand the system's input, processing, and output.

- Purpose: To represent the flow of data from user inputs (e.g., image upload) to the final output (e.g., disease result).
- Example: A level-1 DFD might show processes like "Upload Image", "Run Classifier", and "Fetch Disease Info", with data stores for user history and disease metadata.

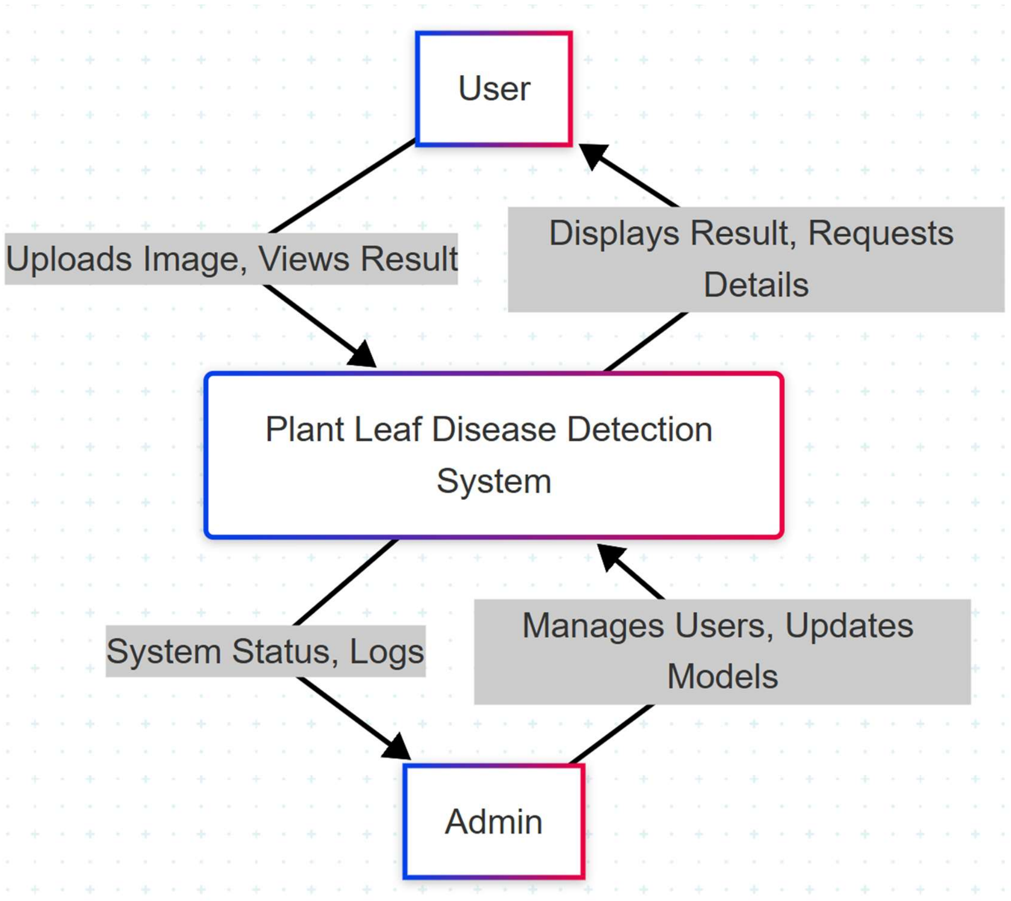

User
Uploads Image, Views Result
Displays Result, Requests Details
Plant Leaf Disease Detection System
System Status, Logs
Manages Users, Updates Models
Admin

Fig 4.3 DFD Level – 0

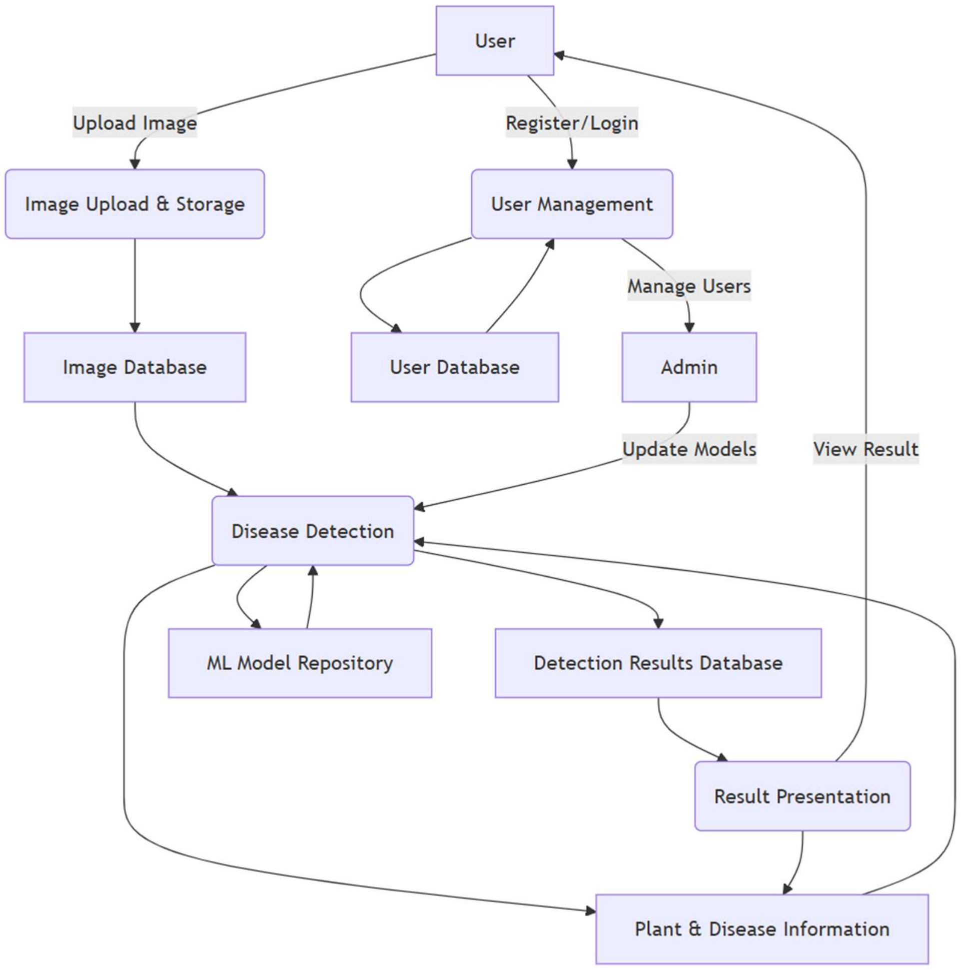


Fig 4.4 DFD Level – 1

4.5 Database Design

4.5.1 Entity-Relationship (E-R) Diagrams – Description

An ER diagram is a visual representation of the database schema. It shows entities (tables), their attributes (columns), and relationships among them.

- Purpose: To structure the project's database design by identifying how data is stored and connected.
- Example: Entities like Users, Predictions, and Diseases are connected, showing that each user can make multiple predictions and each prediction relates to one disease.

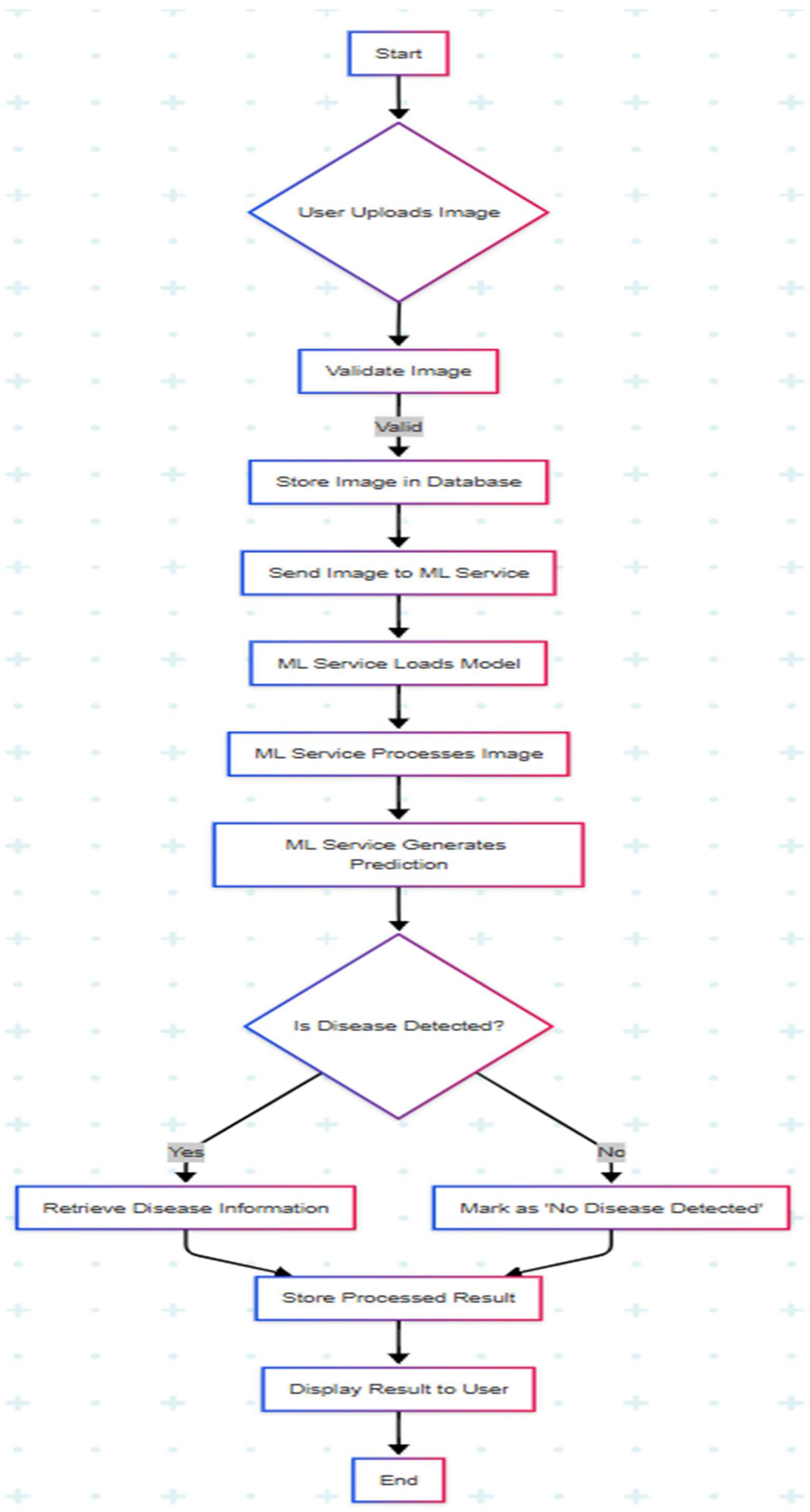


Fig 4.5 Activity Diagram

CHAPTER 5
IMAGE PROCESSING

Image processing involves transforming a physical image into a digital form, enabling various operations such as enhancement, analysis, or feature extraction. In image science, it refers to any technique where the input is an image—such as a photograph or a video frame—and the output may be a transformed image or a set of extracted features.

Although analog and optical techniques exist, digital image processing is the most widely used method. The process typically begins with capturing an image through devices like digital cameras or optical scanners. This digital image is then subjected to processing to enhance its quality or extract significant information.

This stage plays a critical role in many deep learning and computer vision systems, where proper preprocessing can greatly improve model effectiveness. In the entertainment sector, image processing is frequently used for editing tasks, such as object removal or insertion in images.

Images are usually treated as two-dimensional signals and are analysed using standard signal processing methods. Specific areas within an image, called Regions of Interest (ROIs), are often studied separately as they tend to contain key visual elements.

One practical example is defect detection, where image processing techniques help in identifying surface imperfections by focusing on the affected areas. Consequently, image processing has seen rapid growth across various industries, particularly in manufacturing.

The image processing workflow generally includes three primary steps:

1. Image acquisition using digital devices.
2. Image analysis and manipulation, which includes tasks like data compression, image enhancement, and detection of patterns that may not be visible to the human eye.
3. Output generation, where either a processed image or an analytical report is produced.

5.1 Introduction to Image Processing

5.1.1 Objective of Image Processing

Image processing serves several key functions, which can be categorized into five main areas:

1. Visualization – Making objects visible that are not easily seen.
2. Image enhancement and restoration – Improving image quality.
3. Pattern measurement – Determining the dimensions of specific elements in an image.
4. Image recognition – Identifying and classifying objects within an image.

5.1.2 Image Types

An image can be defined as a two-dimensional function where the spatial coordinates (x, y) indicate position, and the intensity value at each point denotes brightness or darkness at that location.

When both the spatial coordinates and intensity values are discretized, the result is known as a digital image. Digital image processing refers to the use of a computer to manipulate or analyze these digital images.

In a digital image, each pixel represents a specific location on the 2D grid and contains one or more numeric values, often called samples, which reflect its intensity or color.

In programming environments such as Python, image processing generally works with two main image types:

- Binary images – Each pixel is either black or white.
- Grayscale images – Pixels have varying shades of grey to represent intensity levels.

5.1.2.1 Binary Image

Binary images are stored as logical arrays and are also referred to as bi-level or two-level images—commonly known as black and white (B&W) images. Some input/output devices, such as laser printers and certain computer monitors, are designed specifically to handle only bi-level images. As illustrated in Figure 4.1, a binary image is composed of pixels that can take on only one of two distinct values: 0 or 1. These values represent black and white, respectively.

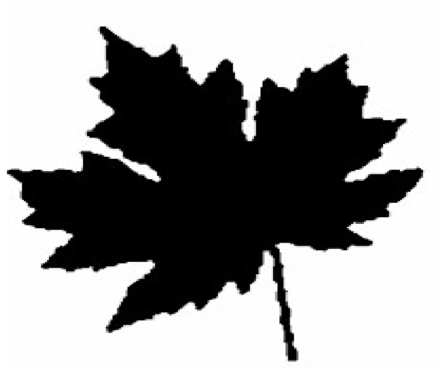


Fig 5.1 Binary Image

5.1.2.2 Grayscale Images

Grayscale images consist of varying shades from black (at the lowest intensity) to white (at the highest). Each pixel in a grayscale image holds a single intensity value, reflecting the light level at that point.

Such images are typically created by measuring the intensity of light across a single band of the electromagnetic spectrum, such as infrared, visible light, or ultraviolet. The pixel values represent these measurements.

In most digital systems, grayscale images are stored with 8 bits per pixel, allowing for 256 different shades of gray. This makes them ideal for visual display and detailed analysis.

A sample grayscale image is presented in Figure 5.2.

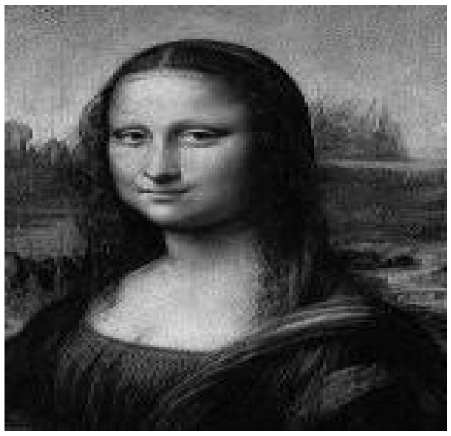


Fig 5.2 Gray Scale Image

5.1.3 Types of Image Processing

Image processing can be broadly categorized into two types:

- Analog Image Processing
- Digital Image Processing

5.1.3.1 Analog Image Processing

Analog image processing involves the manipulation of two-dimensional analog signals, often in physical formats such as printed images or photographs. It is mainly used in the fields of image science and computer science.

Unlike digital processing, analog techniques rely heavily on visual interpretation by experts. The process depends on the observer's knowledge, experience, and the specific characteristics of the image being analyzed.

Here, human judgment plays a critical role in evaluating visual data, often with minimal technological assistance.

5.1.3.2 Digital Image Processing

Digital image processing enables the manipulation of images using computer-based techniques. It is especially important in fields like satellite imaging, where raw data may contain various imperfections that must be corrected.

The digital process generally includes three main stages:

1. Pre-processing – Removing noise and correcting distortions.
2. Enhancement and display – Improving visual quality for better interpretation.
3. Information extraction – Identifying and isolating useful features from the image.

This method starts by converting a physical image into a digital form using devices like scanner-digitizers. Once digitized, the image can undergo numerous computational operations to produce an enhanced or modified version.

In essence, digital image processing refers to the application of algorithmic operations to a numerical image representation to generate the desired outcome.

5.2 Methodology for image processing

5.2.1 Data Preprocessing

The first step in the pipeline is data preprocessing. During this phase, 800 leaf images categorized as *Diseased* and *Healthy* are loaded into the system for training. The OpenCV library in Python is used to convert images from RGB to BGR format, as OpenCV processes images in the BGR color space.

5.2.2 Image Segmentation

Image segmentation involves partitioning a digital image into multiple segments or regions (sets of pixels). Each pixel is assigned a label such that pixels with the same label share certain characteristics.

5.2.3 Feature Extraction (Feature Mining)

Feature mining converts raw image data into mathematical features that preserve essential information. This approach significantly improves performance compared to applying machine learning directly to raw data.

Three types of descriptors are used for feature extraction:

- Color Features: Color Histogram
- Shape Features: Hu Moments
- Texture Features: Haralick Texture

5.2.4 Model Training

To prepare the data for model training:

- Image labels are encoded numerically based on their folder classification.
- The dataset is split into 80% for training and 20% for testing.
- Feature scaling is applied to standardize values with different magnitudes.
- Extracted features are saved in an HDF5 file for efficient access.

Five machine learning models are trained using the processed data:

- a) Random Forest
- b) Logistic Regression
- c) K-Nearest Neighbors (KNN)
- d) Naive Bayes
- e) Support Vector Machine (SVM)

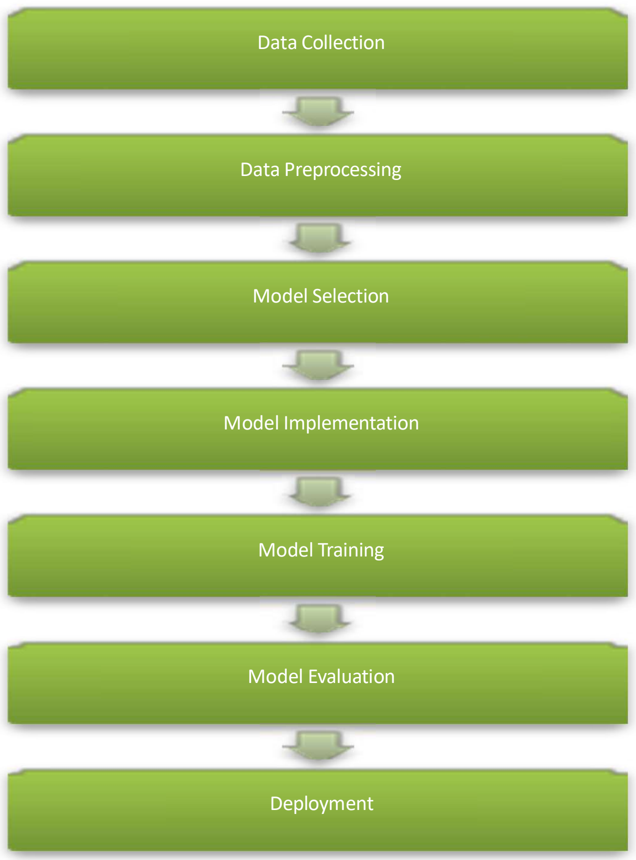


Fig 5.3 Flow Chart

5.2.5 Model Training and Evaluation

The dataset was split into 80% training, 10% validation, and 10% testing. All models were trained using TensorFlow and PyTorch frameworks. Models were assessed using the following performance metrics:

- Accuracy: The accuracy score of a model, often known as accuracy, is a classification statistic in DL and ML techniques that represents the proportion of correct predictions made by the model. [19]

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

- Precision: Precision is defined as the ability to identify only relevant objects. It is defined as the ratio of correctly classified positive outputs to total positive outputs.

$$Precision = \frac{TP}{TP + FP}$$

- Recall: A model's recall is defined as the model's ability to correctly identify True Positives. It is defined as the ratio of correctly classified positive outputs to correctly classified outputs.

$$Recall = \frac{TP}{TP + FN}$$

- F1 – Score: The f1 score is also introduced to assess the model's accuracy. The f1-score considers both the model's precision and recall [20]

$$F1 - Score = \frac{2(Precision * Recall)}{Precision + Recall}$$

5.2.6 Deployment on Local Web Application

A Flask-based web application was developed to enable real-time plant disease detection using the trained models.
- Frontend: HTML, CSS, JavaScript.

- Backend: Python (Flask) for model inference.

- User Input: Users can upload leaf images for instant classification.

5.3 Introduction to Machine Learning

Machine learning is a subset of Artificial Intelligence (AI) that focuses on recognizing patterns in data and developing models that are both intelligent and practical. In conventional computing, outputs are generated based on explicitly defined instructions. Machine learning, however, uses statistical techniques to allow systems to learn from input data and generate outputs within a specific range.

Machine learning tasks are commonly categorized into three major types, based on how the model learns and the kind of feedback it receives:

- a) Supervised Learning
- b) Unsupervised Learning
- c) Reinforcement Learning

a) Supervised Learning

In supervised learning, a model is built from a dataset that includes both input features and corresponding output labels.

Each data instance is typically represented as a feature vector, and the entire training dataset is structured as a matrix. The model learns to map inputs to outputs by optimizing an objective function, allowing it to generalize to new, unseen data.

b) Unsupervised Learning

Unsupervised learning deals with data that does not have labeled outputs. These algorithms identify underlying structures, patterns, or clusters within the dataset. Without supervision, the model detects inherent relationships in the data. A common application is clustering, where data points are grouped based on similarity.

c) Reinforcement Learning

Reinforcement learning involves training an agent to make decisions by interacting with its environment to maximize cumulative rewards. The environment is often modelled as a Markov Decision Process (MDP).

5.3.1 Algorithms Used

5.3.1.1 Random Forest Classifier

Random Forest (RF) is a widely used supervised learning algorithm suitable for both classification and regression tasks. Based on ensemble learning, it constructs multiple decision trees and aggregates their predictions to improve accuracy and reduce overfitting.

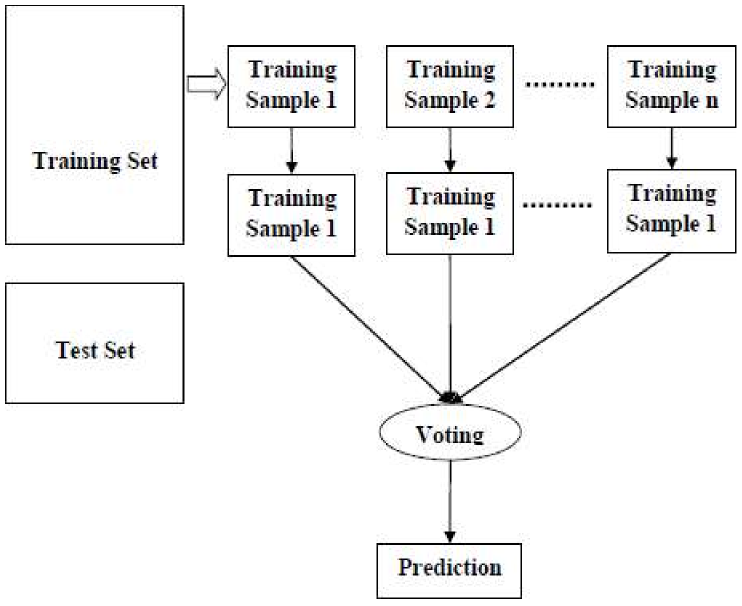


Fig 5.4 Working of Random Forest Algorithm

Working Steps:

1. Randomly select *K* data points from the training set.
2. Build a decision tree based on these selected points.
3. Define the number *N* of trees to be constructed.
4. Repeat steps 1–2 *N* times.
5. For a new data point, aggregate predictions from all trees and choose the class with the majority vote.

5.3.1.2 Logistic Regression

Logistic Regression is a fundamental supervised learning algorithm used for binary classification. It predicts the probability of an instance belonging to a particular class (e.g., 0 or 1). Rather than outputting discrete values, it provides probabilistic estimates between 0 and 1 using a logistic (sigmoid) function.

5.3.1.3 Support Vector Machine (SVM)

Support Vector Machine is a powerful supervised learning algorithm primarily used for classification, though it can also be applied to regression.

SVM aims to find the optimal hyperplane that best separates the data points of different classes in high-dimensional space.

5.3.1.4 K-Nearest Neighbors (KNN)

K-Nearest Neighbors is a simple yet effective supervised learning algorithm used for both classification and regression, though it is more commonly applied to classification problems.

KNN works by:

- Comparing a new data point with existing labeled data.
- Finding the K most similar instances (neighbors).
- Assigning the new point to the most frequent class among those neighbors.

5.3.1.5 Naive Bayes

Naive Bayes is a supervised learning algorithm based on Bayes' Theorem, often used for text classification and spam detection.

Despite its simplicity, Naive Bayes is highly effective. It assumes independence among features, which allows it to make fast and reliable predictions even on large datasets.

5.4 Technologies Used

5.4.1 Python

Python is a high-level, general-purpose programming language known for its simplicity and readability. It is an interpreted language that uses indentation to define code structure, enhancing clarity and maintainability. Originally developed by Guido van Rossum as a successor to the ABC language, Python was first released in 1991. Python also supports advanced concepts such as metaprogramming and aspect-oriented programming. Through extensions, it even allows for logic programming and design-by-contract paradigms.

5.4.2 Python Libraries

One of Python's greatest strengths is its comprehensive standard library, which includes tools and modules for a wide range of programming tasks. These include:

- Internet protocols (e.g., HTTP)
- GUI development
- Database access
- Regular expressions
- Numerical computations
- Unit testing

Modules like the Web Server Gateway Interface (WSGI) adhere to PEP specifications, while most others are defined by their source code, documentation, and test suites.

5.4.3 Pandas

Pandas is a powerful Python library designed for data manipulation and analysis. It provides flexible data structures such as DataFrames for handling structured data and time series.

Key Features:

- DataFrame object with integrated indexing for structured data
- Support for reading/writing data from various file formats
- Efficient handling of missing data

Developed by Wes McKinney in 2008 during his time at AQR Capital, the name "pandas" is derived from "panel data" and is also a play on "Python data analysis."

It is open-source software released under the BSD license.

5.4.4 Scikit-learn

Scikit-learn is an open-source machine learning library for Python that supports a wide array of algorithms for classification, regression, and clustering. These include:

- Support Vector Machines (SVM)
- Random Forests
- K-Means Clustering
- DBSCAN

It is designed to work seamlessly with other Python libraries like NumPy and SciPy, utilizing their capabilities for high-performance numerical computations.

5.4.5 NumPy

NumPy (Numerical Python) is a foundational library for numerical computing in Python. It offers support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on them efficiently.

Key Features:

- Optimized operations on arrays and matrices
- Fast computation using vectorization
- Interoperability with C and Fortran code

Developed by Jim Hugunin and other contributors, NumPy was designed to overcome performance issues of interpreted Python by providing highly efficient array operations.

CHAPTER 6
METHODOLOGY

6.1 Introduction

Globally, there are approximately 570 million farms, and over 90% of the world's population is connected to agriculture in some way. Farmers are responsible for producing nearly 80% of the food consumed worldwide. However, crop diseases continue to pose a significant threat to yield and quality in large-scale agriculture. These diseases can be caused by various pathogens such as bacteria, viruses, nematodes, and fungi, affecting both the visible and hidden parts of plants.

Additional factors, including climate change and the decline of pollinators, further impact food production, threatening farmers' incomes and global food security. To manage plant diseases, farmers often invest considerable time, money, and resources. Given the increasing environmental changes, early identification and diagnosis of plant diseases are more important than ever.

Accurately identifying disease types and understanding how to manage them remain persistent challenges in agriculture. Some plant diseases cannot be diagnosed solely through visible symptoms, making precise detection essential—not just for disease control but also for ensuring the sustainability of agriculture.

Early intervention using fungicides, pesticides, or disease-specific chemicals is more effective when an early warning system is in place. Recently, smartphone-based disease detection has become increasingly viable, thanks to the widespread availability of smartphones and advancements in machine learning and deep learning. Many studies have explored these methods, showing that traditional and modern machine learning techniques can significantly improve plant disease identification and classification.

6.1.1 Convolutional Neural Networks (CNN)

In image classification, images are interpreted as raw pixel data and categorized as objects. CNNs (Convolutional Neural Networks) help extract meaningful features from these raw images for accurate classification. Unlike traditional image recognition methods—where features must be defined manually—CNNs automatically learn and extract relevant features from raw data through training.

6.1.2 Transfer Learning

Transfer learning leverages a pre-trained model from one task as a starting point for a new but related task. This technique is particularly useful when the new task lacks sufficient data or when training a model from scratch is impractical due to high computational costs and time. By using the features learned from the initial task, transfer learning enables quicker convergence, better generalization, and improved performance in the new task. Pre-trained models such as AlexNet, InceptionNet, and VGG16 are often fine-tuned by modifying their final classification layers for new use cases.

6.1.2.1 Importance of Transfer Learning

Transfer learning is essential in scenarios where the target problem is complex and requires significant training resources. It allows the reuse of knowledge gained from large datasets, reducing both the amount of required data and the time needed for training a model on a new but related problem.

6.1.2.2 Benefits of Transfer Learning

- Faster Learning: The model already recognizes general patterns, leading to quicker and more efficient learning.
- Improved Accuracy: Leveraging prior knowledge often results in better performance on the target task.
- Handling Small Datasets: Transfer learning helps mitigate overfitting by using already-learned features from large-scale datasets.

6.1.3 ResNet50

ResNet50, short for "Residual Network with 50 layers," is a powerful deep learning model that addresses the limitations of traditional CNNs, particularly the vanishing gradient problem. Introduced by Microsoft in 2015, ResNet50 won the ImageNet competition that year and is widely regarded as one of the most effective pre-trained models for image classification—including plant disease detection.

ResNet allows for the training of very deep neural networks through the use of skip connections, which bypass one or more layers and feed the output of an earlier layer directly into a later one. This enables the model to retain essential features and avoid gradient degradation during training.

6.1.3.1 Residual Block

The fundamental building block of ResNet is the residual block. It works by summing the input x and the output of a layer f(x):

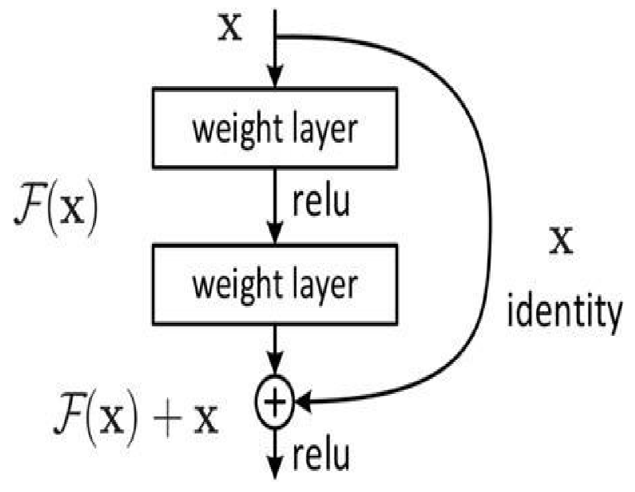


Fig 6.1 Residual Block

6.1.3.2 ResNet50 Architecture Overview

ResNet50 comprises the following:

- A 7×7 convolution layer with 64 filters, stride 2
- A max-pooling layer with stride 2
- Convolutional and identity blocks with:
    - 1×1, 3×3, and 1×1 convolution layers of varying filter sizes
    - Repeated layer structures:
        - 3 blocks of [64, 64, 256]
        - 4 blocks of [128, 128, 512]
        - 6 blocks of [256, 256, 1024]
        - 3 blocks of [512, 512, 2048]

Types of Blocks:

- Identity Block: Used when input and output dimensions are the same.
- Convolutional Block: Used when dimensions differ, including a 2D convolution to align them.

Each convolutional and identity block contains three convolutional layers, contributing to the model’s total of 25 million trainable parameters.

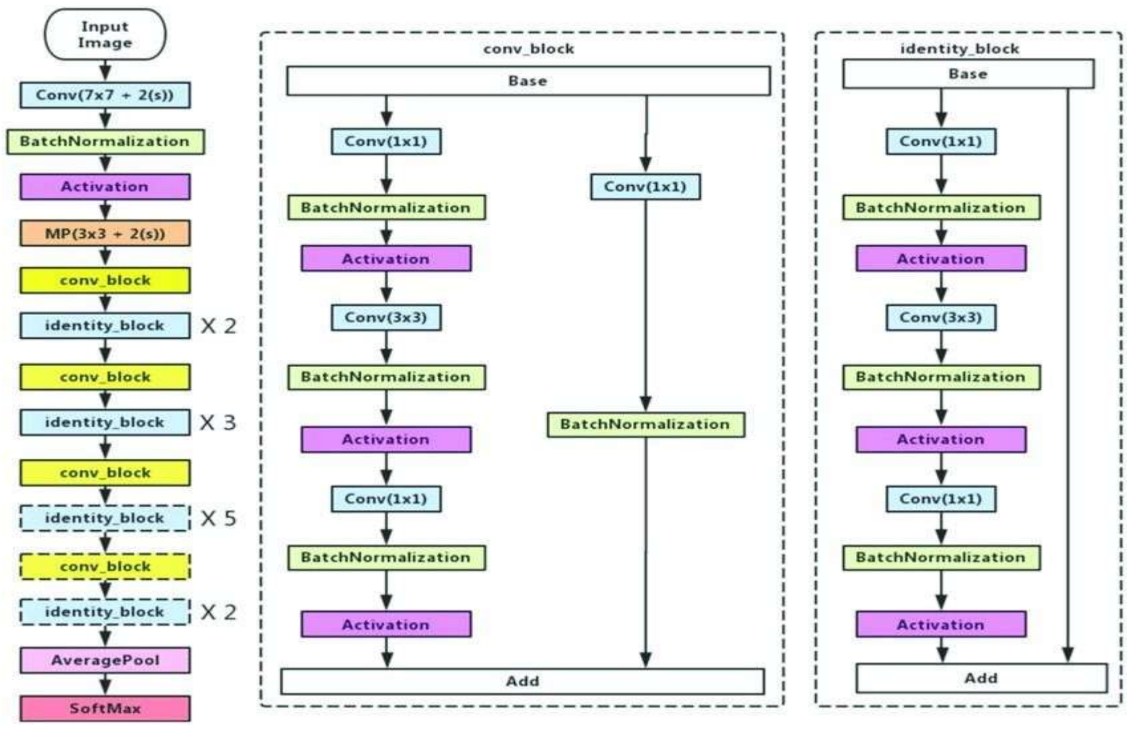


Fig 6.2 Block Diagram of ResNet50

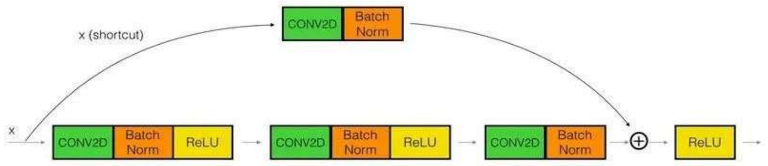


Fig 6.3 Convolutional Block

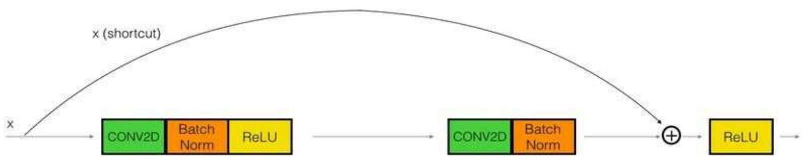


Fig 6.4 Identity Block

6.1.3.3 Key Features of ResNet50

- Efficient Deep Learning: Capable of training extremely deep networks (even over 100 layers) without increasing training error.
- Skip Connections: Help preserve learned features and avoid vanishing gradients.
- Fast Training: Due to reduced complexity in each block and optimized structure.

6.1.3.4 Advantages of ResNet

- Easily trains very deep networks without increasing error rates.
- Solves the vanishing gradient problem through identity mappings.
- Efficient in feature reuse, leading to better performance in tasks like plant disease detection.

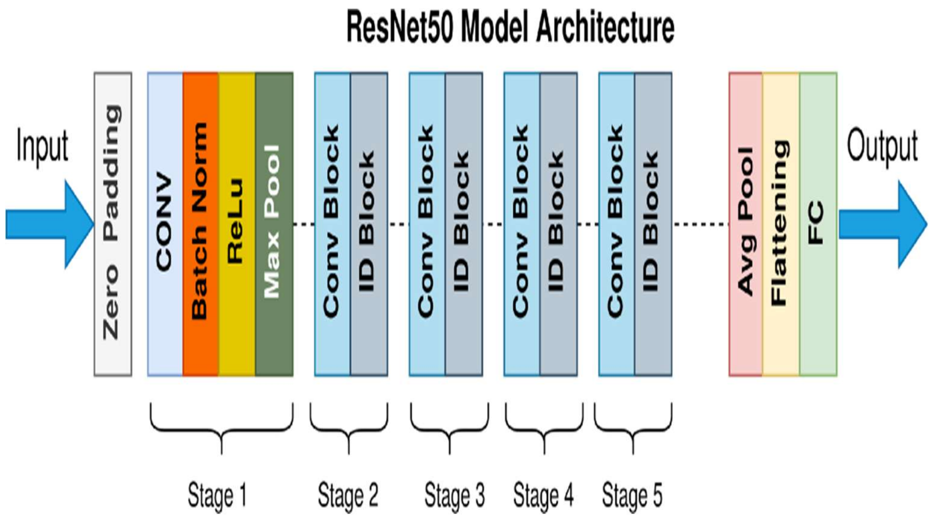


Fig 6.5 ResNet50 Architecture

6.2 The Dataset

6.2.1 Training Dataset:

The training dataset is the primary subset of the original data used to train and fit the machine learning (ML) model. During this phase, the model learns how to make predictions for a given task by processing the input data. In unsupervised learning, the dataset consists of unlabelled data points, meaning the model must identify patterns without any predefined outputs.

On the other hand, supervised learning involves labeled data, where each input is associated with a corresponding output. This helps the model learn to map inputs to predictions effectively. The quality and relevance of the training dataset significantly influence the model's accuracy and prediction capability.

6.2.2 Testing Dataset:

Once the model has been trained on the training dataset, it must be evaluated using a testing dataset. This dataset helps assess the model's performance and ensures it can generalize effectively to unseen or new data.

The testing dataset is a distinct subset of the original data, separate from the training set. Although it shares similar features and class distributions, it has not been used during the training phase. It acts as a benchmark to test how well the model performs in real-world scenarios.

6.2.3 Importance of Splitting the Dataset:

Splitting the dataset into training and testing sets is a crucial step in data preprocessing.

This division ensures that the model's performance can be evaluated reliably and that the results are not biased by the training data.

The idea is to train the model on one portion of the data and validate its performance on a separate portion. If both sets are not distinct, the model may memorize patterns from the training data, failing to generalize to new data—resulting in poor real-world performance.

6.2.4 Differences Between Training Data and Testing Data:

- Purpose: Training data is used to build and train the model, while testing data is used to evaluate the model's performance and generalization ability.
- Size: The training set is typically larger than the testing set. Common train-test splits include 80:20, 70:30, or 90:10 ratios.

6.3 Image Pre-Processing

Before images can be used in machine learning models, they must undergo pre-processing. This involves adjusting characteristics such as color, size, and orientation to improve image quality and enhance their suitability for analysis.

6.3.1 Importance of Image Pre-Processing

Pre-processing is a critical step for preparing image data to be used as input in machine learning models. For example, convolutional neural networks (CNNs) require all input images to be of the same size.

If the input images are too large, resizing them can significantly reduce training time without compromising performance.

6.3.2 Criteria for Image Pre-Processing

- For optimal results in machine learning projects, data must be in an appropriate format. Some algorithms, such as Random Forest, do not support missing values, so they must be handled during pre-processing.
- A dataset may be structured to support multiple machine learning or deep learning algorithms, allowing for a comparison of outputs and selection of the best-performing model.

6.3.3 Common Image Pre-Processing Techniques

1. Grayscale Conversion

   This process converts colored images into black-and-white (grayscale) format. It reduces computational complexity since most image recognition tasks do not require color information. Converting to grayscale decreases the number of pixels and simplifies calculations.

2. Standardization (Normalization)

   Also known as rescaling, this process adjusts pixel values to a predefined range—typically between 0 and 1. This ensures uniformity across all images, helping to maintain consistent performance and learning rates during training. It also prevents any one image from disproportionately influencing the model due to having a larger pixel value range.

3. Data Augmentation

   Data augmentation increases the variety and size of the dataset by applying small modifications to existing images, such as flipping, rotating, cropping, or shearing. This helps prevent the model from overfitting by enabling it to learn more robust features.

4. Image Rescaling/Standardizing

   Resizing images ensures they are consistent in dimensions. Additionally, converting pixel values to a mean of zero improves data consistency and quality, enhancing the model's learning process.

6.3.4 Steps for Loading a Custom Dataset for Deep Learning Models

1. Read the image file: Acceptable formats include JPEG, BMP, etc.
2. Resize the image: Match the dimensions required by the model input layer.
3. Convert pixel data: Cast pixel values to float data type.
4. Normalize pixel values: Scale pixel intensities from the 0–255 range down to 0–1.
5. Convert the data format: Ensure the image is represented as either a NumPy array or a tensor.

6.4 Data Augmentation

Data augmentation is a technique used in machine learning and deep learning to increase the volume and diversity of training data. It involves generating new training samples by applying various transformations to existing data—such as flipping, rotating, scaling, and cropping. The primary aim of this technique is to improve model performance and generalization by exposing the model to a broader range of scenarios.

By simulating different variations, data augmentation helps the model recognize patterns from various perspectives, lighting conditions, and orientations. It also helps reduce overfitting by supplying more diverse training examples.

6.4.1 Steps in Image Data Augmentation

There is no universal set of augmentation steps that guarantees improved model performance. In some cases, certain augmentations may even degrade performance, depending on the nature of the task.

1. Grayscale Conversion

   Converting coloured images to grayscale simplifies image data and reduces the input dimensions from three channels to one. This not only reduces computational complexity but also forces the model to focus on structural and shape-based features rather than color.

2. Random Flips

   Flipping images horizontally or vertically helps the model learn that objects can appear in different orientations. However, in tasks where orientation is important—such as reading text—flipping may not be appropriate.

3. Random Rotations

   This is crucial for applications where the orientation of the object can vary, such as mobile applications. However, rotation may result in loss of image data around the edges and requires careful handling to preserve important features.

4. Random Brightness Adjustments

   Adjusting image brightness randomly helps the model adapt to variations in lighting conditions. It simulates how the same object might appear under different illumination levels.

6.4.2 Image Augmentation Using Keras

The ImageDataGenerator class in Keras provides a convenient and efficient way to perform image augmentation in real-time during model training. It supports a variety of transformations such as normalization, rotation, brightness adjustments, and more.

6.4.2.1 Common Techniques in ImageDataGenerator

1. Random Rotation

   Images can be randomly rotated within a specified range (0° to 360°). Any empty pixels created as a result of rotation are filled based on the fill_mode parameter, which defaults to 'nearest'.

2. Random Shifting

   This involves translating the image either horizontally or vertically. The parameters width_shift_range and height_shift_range control the extent of the shift, which can be a float (fraction of the image size) or an integer (number of pixels).

3. Random Flipping

   Images can be flipped horizontally (horizontal_flip=True) or vertically (vertical_flip=True). This is effective for many object recognition tasks, though care should be taken with objects that have orientation-specific meanings.

4. Random Zooming

   Zooming in or out is controlled by the zoom_range parameter. If a float is provided, the zoom occurs within [1 - zoom_range, 1 + zoom_range]. This helps the model generalize to different scales of the object.

6.4.2.2 Methods in ImageDataGenerator

1. flow_from_directory()
   This method loads images directly from a directory and applies augmentations on-the-fly. It expects images to be organized into subfolders by class. Key parameters include:
   - directory: Path to the main folder containing class subfolders.
   - target_size: Desired image size.
   - color_mode: 'rgb' for color images or 'grayscale' for black-and-white.
   - batch_size: Number of images per batch.
2. flow_from_dataframe()
   This method loads images based on file paths and labels specified in a Pandas DataFrame. It's especially useful when images are stored in a single folder. Key parameters include:
   - dataframe: Contains file names and labels.
   - directory: Path to the folder containing images.
   - x_col: Column name with image file names.
   - y_col: Column name with labels.
   - class_mode: 'binary' or 'categorical' depending on the task.

6.5 Feature Extraction Using ResNet50

Feature extraction is a critical step in the dimensionality reduction process, where raw data is transformed into a more compact and manageable format. This is especially useful when working with high-dimensional datasets that contain a vast number of variables, which can demand extensive computational resources.

Feature extraction helps by isolating the most significant features and transforming the data into a lower-dimensional space, while still preserving the most relevant information.

6.5.1 Importance of Feature Extraction

When dealing with large datasets, it becomes essential to minimize computational load without compromising on the quality of information. Feature extraction plays a key role in achieving this by reducing data redundancy and highlighting the most important patterns.

6.5.2 Constructing the ResNet50 Model

ResNet50, a powerful convolutional neural network architecture pre-trained on ImageNet, is commonly used for feature extraction in deep learning workflows. It can be initialized using the following code:

```
tf.keras.applications.resnet50.ResNet50(
    include_top=False,
    weights='imagenet',
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=14,
    **kwargs
)
```

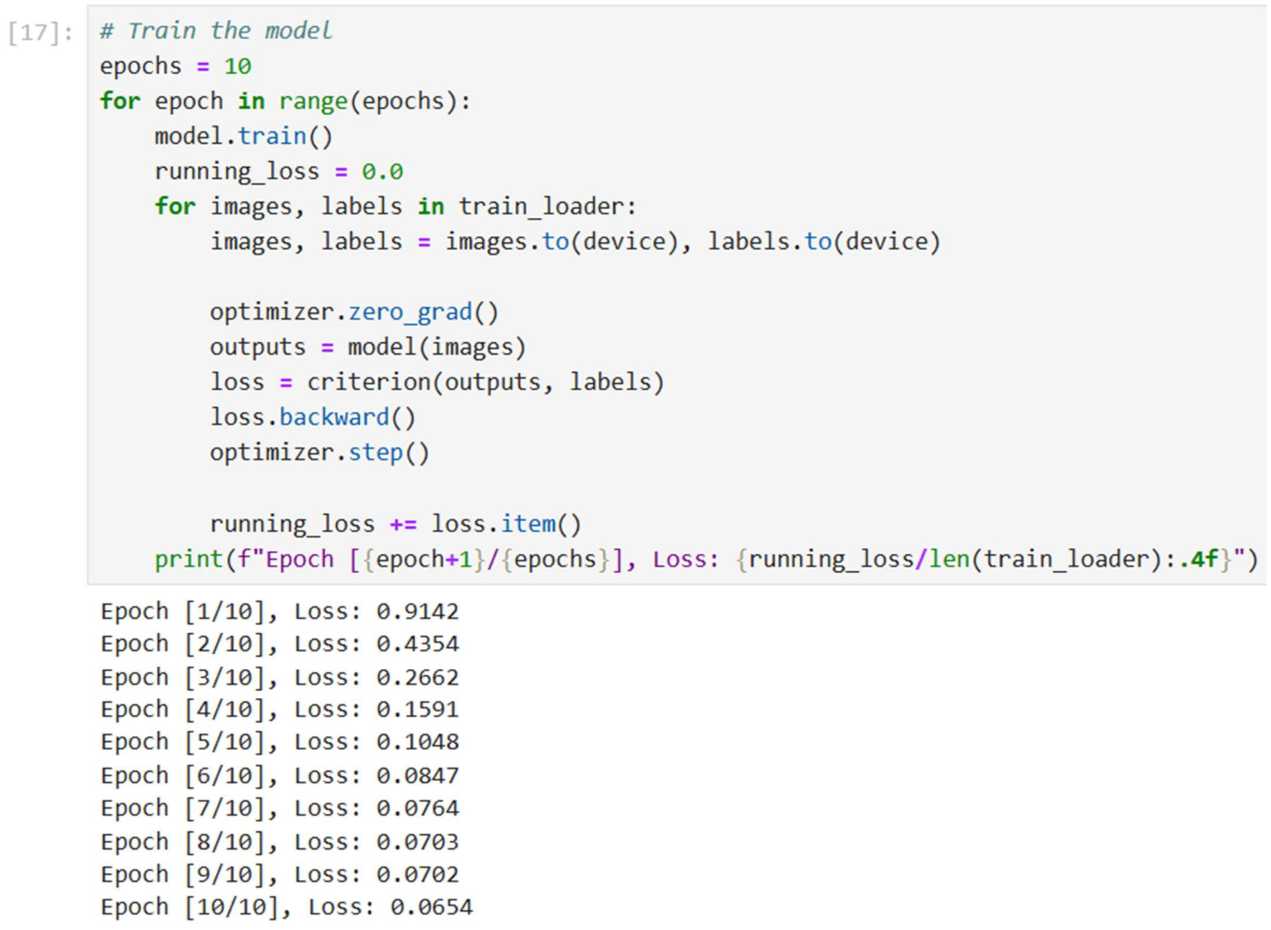
6.6 Training the Model

Training of deep learning models in Keras can be performed using either model.fit() or model.fit_generator() (note: in modern Keras, fit() now also supports generators). Both serve the purpose of fitting the model to training data, but their usage depends on the dataset size and augmentation needs:

- model.fit() is used when the dataset fits entirely into memory and no augmentation is required.
- model.fit_generator() is suited for large datasets and when using real-time data augmentation via generators.

Training Using model.fit()

After building the model, training is initiated by passing an image data iterator to the fit() function. The iterator may include augmented images generated in real-time. The model learns from data provided in batches and updates weights accordingly.



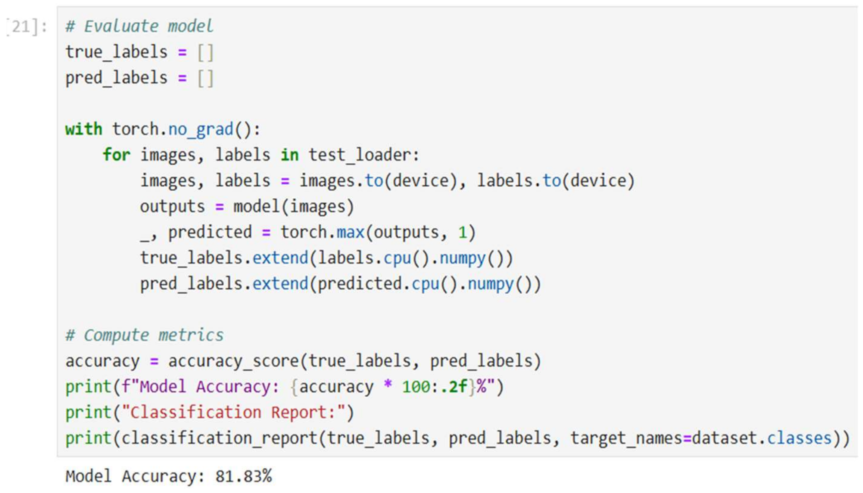
```
[17]: # Train the model
      epochs = 10
      for epoch in range(epochs):
          model.train()
          running_loss = 0.0
          for images, labels in train_loader:
              images, labels = images.to(device), labels.to(device)

              optimizer.zero_grad()
              outputs = model(images)
              loss = criterion(outputs, labels)
              loss.backward()
              optimizer.step()

              running_loss += loss.item()
          print(f"Epoch [{epoch+1}/{epochs}], Loss: {running_loss/len(train_loader):.4f}")
```

```
Epoch [1/10], Loss: 0.9142
Epoch [2/10], Loss: 0.4354
Epoch [3/10], Loss: 0.2662
Epoch [4/10], Loss: 0.1591
Epoch [5/10], Loss: 0.1048
Epoch [6/10], Loss: 0.0847
Epoch [7/10], Loss: 0.0764
Epoch [8/10], Loss: 0.0703
Epoch [9/10], Loss: 0.0702
Epoch [10/10], Loss: 0.0654
```

Fig 6.6 Training Model



```
[21]: # Evaluate model
      true_labels = []
      pred_labels = []

      with torch.no_grad():
          for images, labels in test_loader:
              images, labels = images.to(device), labels.to(device)
              outputs = model(images)
              _, predicted = torch.max(outputs, 1)
              true_labels.extend(labels.cpu().numpy())
              pred_labels.extend(predicted.cpu().numpy())

      # Compute metrics
      accuracy = accuracy_score(true_labels, pred_labels)
      print(f"Model Accuracy: {accuracy * 100:.2f}%")
      print("Classification Report:")
      print(classification_report(true_labels, pred_labels, target_names=dataset.classes))
```

```
Model Accuracy: 81.83%
```

Fig 6.7 Evaluate Model

CHAPTER 7
RESULTS

The final System is going to predict which disease does a leaf possess with an accuracy of 81.83%.

7.1 Result of Leaf Images

Loading Original Images – For training purposes, a total of 1,600 leaf images are used. These images are categorized into two classes: *Healthy* and *Diseased*. Before processing, each image is converted from RGB (Red-Green-Blue) format to BGR (Blue-Green-Red) format.

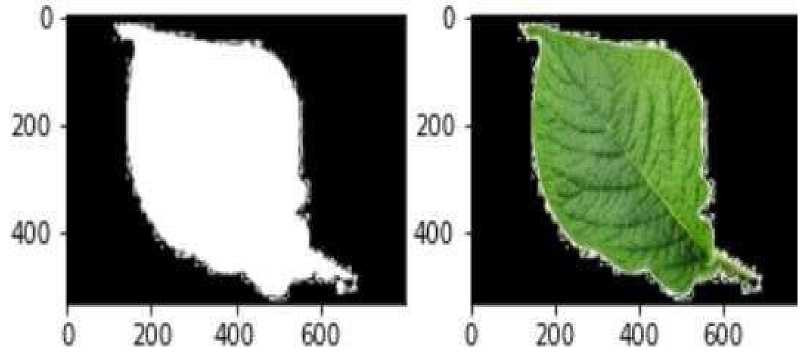


Fig 7.1 Healthy Leaf

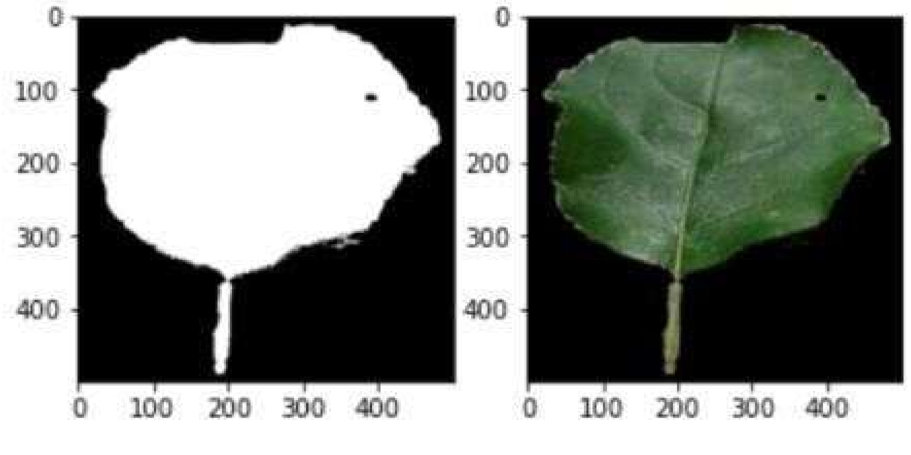


Fig 7.2 Diseased Leaf – Low

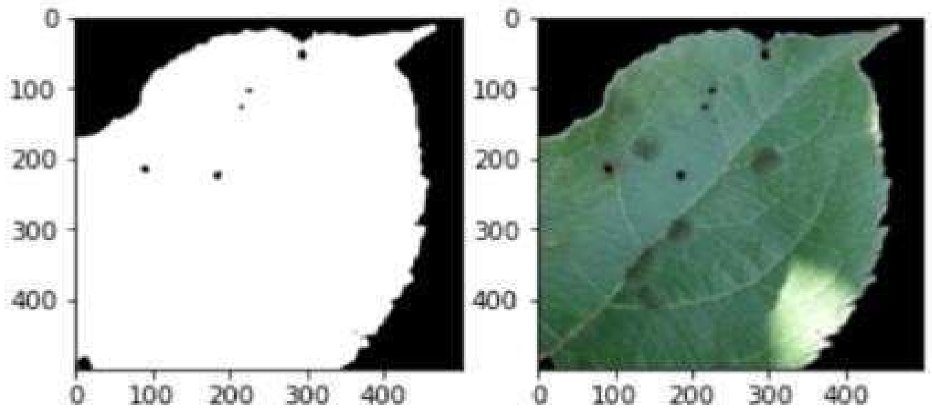


Fig 7.3 Diseased Leaf – Medium

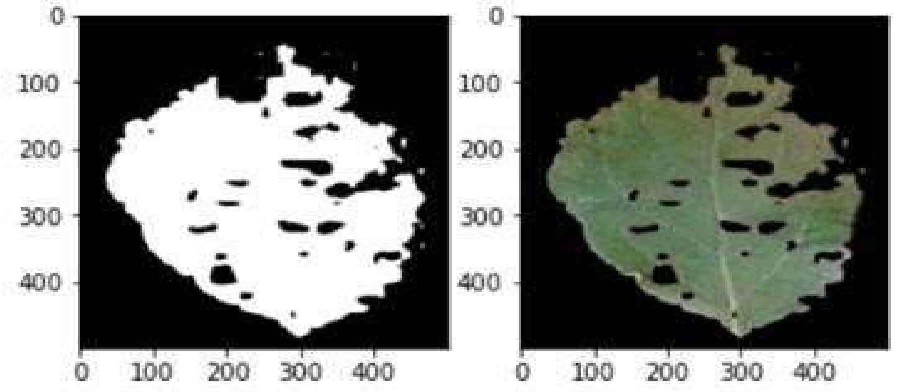


Fig 7.4 Diseased Leaf – High

7.2 Experiment

We experimented with several machine learning algorithm, but confined to build the CNN model mainly due to computational limitations. Table 6.1 shows the classification report of the CNN model for each disease class.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Apple___alternaria_leaf_spot	0.96	0.8	0.87	56
Apple___black_rot	0.77	0.91	0.83	124
Apple___brown_spot	0.78	0.84	0.81	43
Apple___gray_spot	0.8	0.52	0.63	79
Apple___healthy	0.81	0.65	0.72	514

Apple___rust	0.8	0.62	0.7	248
Apple___scab	0.56	0.45	0.5	244
Bell_pepper_bacterial_spot	0.85	0.76	0.81	199
Bell_pepper___healthy	0.84	0.88	0.86	296
Cassava_bacterial_blight	0.25	0.17	0.2	217
Cassava_brown_streak_disease	0.32	0.28	0.3	438
Cassava__green_mottle	0.22	0.13	0.16	477
Cassava___healthy	0.27	0.26	0.27	516
Cassava___mosaic_disease	0.73	0.84	0.78	2632
Cherry___healthy	0.89	0.97	0.93	171
Cherry___powdery_mildew	0.98	0.9	0.94	210
Coffee___healthy	0.67	0.63	0.65	158
Coffee___red_spider_mite	0.0	0.0	0.0	33
Coffee___rust	0.44	0.53	0.48	120
Corn___common_rust	0.98	0.89	0.93	238
Corn___gray_leaf_spot	0.78	0.64	0.7	103
Corn___healthy	0.87	0.97	0.92	232
Corn_northern_leaf_blight	0.86	0.9	0.88	197
Grape___Leaf_blight	0.98	0.98	0.98	295
Grape___black_measles	0.93	0.82	0.87	534
Grape___black_rot	0.97	0.93	0.95	316
Grape___healthy	0.93	0.92	0.93	341
Peach___bacterial_spot	0.85	0.92	0.88	460
Peach___healthy	0.92	0.67	0.77	72
Potato___bacterial_wilt	0.63	0.59	0.61	114
Potato___early_blight	0.88	0.88	0.88	526
Potato___healthy	0.75	0.69	0.72	455
Potato___late_blight	0.76	0.8	0.78	417
Potato___leafroll_virus	0.98	0.76	0.86	105
Potato___mosaic_virus	0.75	0.84	0.79	133
Potato___nematode	0.27	0.21	0.24	14
Potato___pests	0.36	0.35	0.36	122
Potato___phytophthora	0.51	0.61	0.56	69
Potato_spindle_tuber_viroid	0.81	1.0	0.89	17

Table 7.1 Classification Report

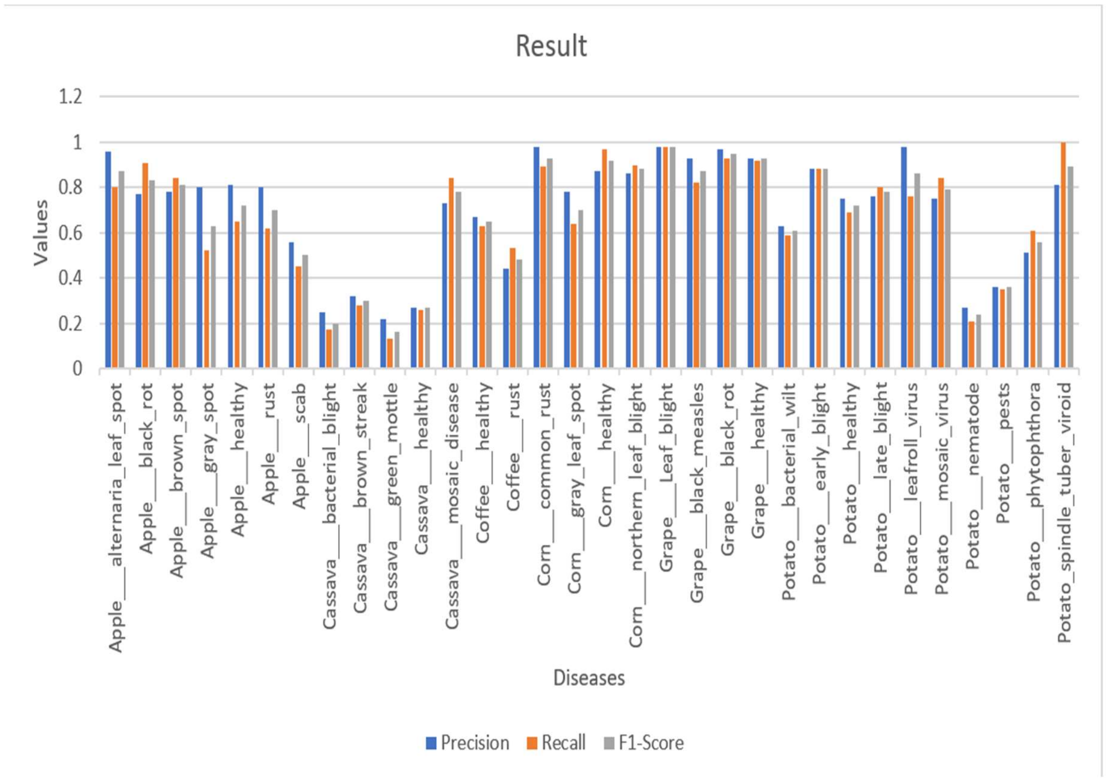


Fig 7.5 Visualization of classification

7.3 The Webpage

This how a webpage is going to look after deploying the model.

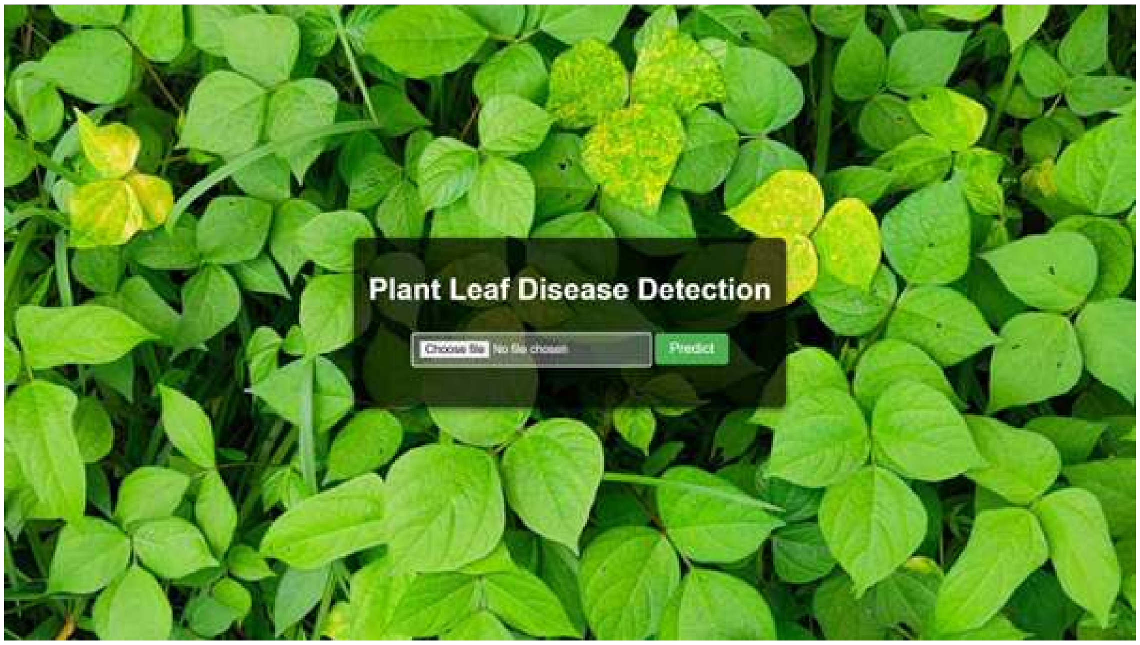


Fig 7.6 Webpage

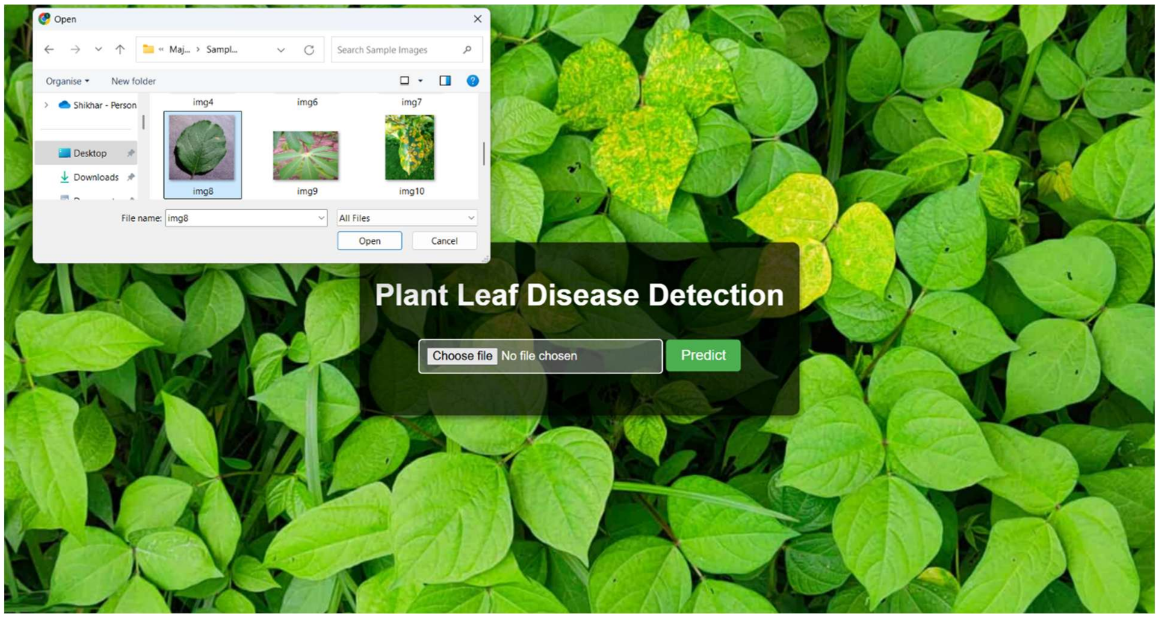


Fig 7.7 Choosing the image files

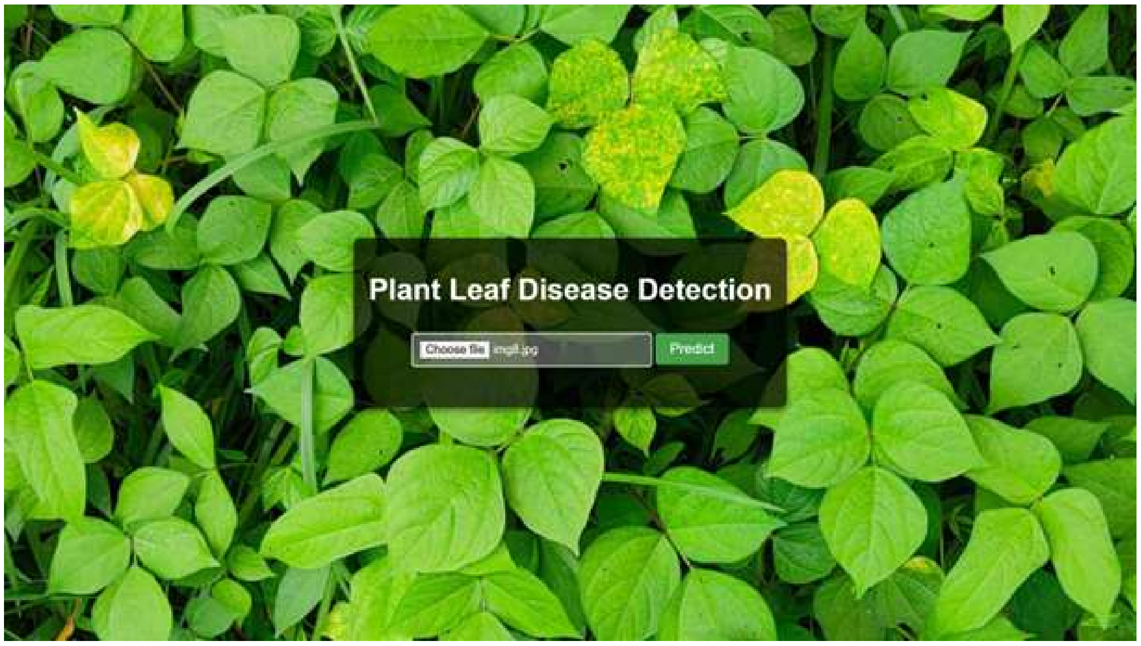
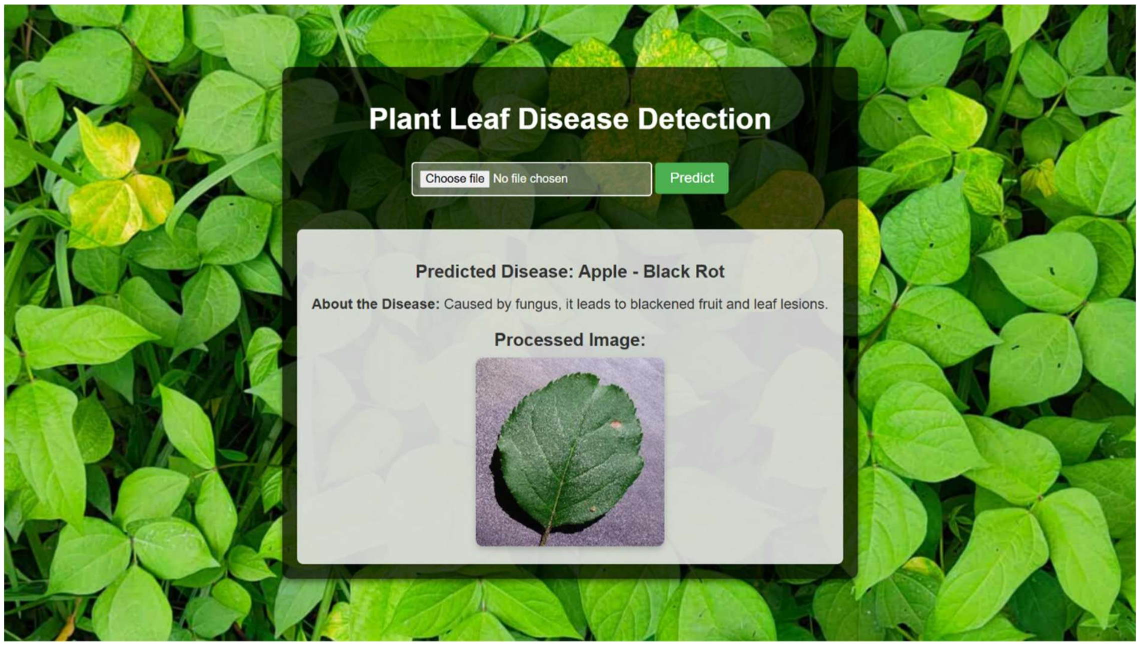


Fig 7.8 Click Predict



Predicted Disease: Apple - Black Rot

About the Disease: Caused by fungus, it leads to blackened fruit and leaf lesions.

Processed Image:

Fig 7.9 Output with leaf and its disease

CONCLUSION

Plant leaf disease detection using machine learning algorithms has shown significant potential in recent years. These algorithms are capable of accurately classifying plant leaves as either healthy or diseased, enabling early diagnosis and treatment. This early intervention helps prevent crop damage and minimizes yield loss.

In this project, we explored several machine learning algorithms including Logistic Regression, K-Nearest Neighbors (KNN), Naive Bayes, Support Vector Machines (SVMs), and Random Forests for the purpose of plant disease detection. We assessed their performance on different datasets by evaluating key metrics such as accuracy, precision, recall, and F1-score.

Additionally, we investigated the application of ResNet-50, a deep convolutional neural network, for plant disease detection using video datasets. ResNet-50 excels in extracting high-level features from images, making it particularly effective for this task. Our experiments demonstrated that ResNet-50 achieved high accuracy in identifying plant diseases from video frames.

Overall, our findings indicate that the Random Forest classifier performed the best among traditional machine learning models in terms of accuracy and generalization on image datasets, while ResNet-50 outperformed other models on video datasets. These results suggest that both algorithms can serve as reliable and efficient tools for automated plant disease detection.

With the advancement of technology, the demand for automated plant monitoring and management systems is growing. In agriculture, yield losses are often caused by diseases that are detected only in advanced stages. This late detection leads to significant losses in terms of productivity, time, and cost. The proposed system addresses this challenge by detecting diseases at an early stage—as soon as they appear on the leaf. This early detection helps reduce crop loss and lessens reliance on expert diagnosis. Furthermore, it can assist individuals with limited knowledge of plant diseases by providing clear disease identification. Achieving these objectives involves extracting relevant features associated with different plant diseases.

The emergence of plant pathogens poses a significant threat to global food security, ecosystems, and economies. Factors such as globalization, increased mobility, vectors, climate change, and pathogen evolution have accelerated the spread of invasive plant pathogens. To address agricultural losses, the development of automated approaches for plant disease detection and classification is urgently needed.

This review explores the use of machine learning (ML), deep learning (DL), and few-shot learning (FSL) for automated plant disease recognition. It highlights key methodologies, including acqui sition, preprocessing, segmentation, feature extraction, and classification. While many studies rely on RGB images, some have adopted hyperspectral imaging for plant leaves, which offers the advantage of detecting microscopic symptoms without requiring la beled datasets. These automated techniques have facilitated timely advancements in research. Additionally, the review examines molecular diagnostic tools and state-of the-art techniques for plant disease detection. The methods discussed are highly sensitive, specific, and capable of rapid detection.

REFFERENCES

1. FAO, IFAD, UNICEF, WFP, WHO, The State of Food Security and Nutrition in the World 2023. Geneva, Switzerland, Jul. 2023.

2. "State of Agriculture in India," accessed Aug. https://prsindia.org/files/policy/policy_analytical_reports 9, 2023.

3. C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," Electron. Markets, vol. 31, no. 3, pp. 685–695, Apr. 2021, doi: 10.1007/s12525-021-00475-2.

4. K. Kc, Z. Yin, D. Li, and Z. Wu, "Impacts of background removal on convolutional neural networks for plant disease classification in-situ," Agriculture, vol. 11, no. 9, p. 827, Aug. 2021, doi: 10.3390/agriculture11090827.

5. J. Liu and X. Wang, "Plant diseases and pests detection based on deep learning: A review," Plant Methods, vol. 17, no. 1, pp. 1–18, Dec. 2021.

6. Y. Zhang, C. Song, D. Zhang, Deep learning-based object detection improvement for tomato disease, IEEE Access 8 (2020) 56607–56614, doi: 10.1109/ACCESS.2020.2982456.

7. G. Sambasivam, G.D. Opiyo, A predictive machine learning application in agriculture: Cassava disease detection and classification with imbalanced dataset using convolutional neural networks, Egypt. Inform. J. 22 (1) (2021) 27–34, doi: 10.1016/j.eij.2020.02.007.

8. I. Ahmed, P.K. Yadav, An automated system for early identification of diseases in plant through machine learning, in: Soft Computing: Theories and Applications: Proceedings of SoCTA 2021, Springer Nature Singapore, 2022, pp. 803–814.

9. M.H. Saleem, J. Potgieter, K.M. Arif, Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers, Plants 9 (10) (2020) 1–17, doi: 10.3390/plants9101319.

10. Tirkey D, Singh KK, Tripathi S. Performance analysis of AI-based solutions for crop disease identification detection, and classification. Smart Agric Technol. 2023.

11. Jasim MA, Al-Tuwaijari JM. Plant leaf diseases detection and classification using image processing and deep learning techniques. Int Comput Sci Soft Eng Conf. 2020.

12. V. K. Shrivastava, M. K. Pradhan, S. Minz, and M. P. Thakur, "Rice plant disease classification using transfer learning of deep convolution neural network," The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XLII-3/W6, pp. 631–635, 2019.

13. K. Aderghal, A. Khvostikov, A. Krylov, J. Benois-Pineau, K. Afdel, and G. Catheline, "Classification of Alzheimer disease on imaging modalities with deep CNNs using cross modal transfer learning," in 2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS), pp. 345–350, IEEE, Karlstad, Sweden, 2018.

14. A. Abdalla, H. Cen, L. Wan et al., "Fine-tuning convolutional neural network with transfer learning for semantic segmentation of ground-level oilseed rape images in a field with high weed pressure," Computers and Electronics in Agriculture, vol. 167, Article ID 105091, 2019.

15. Kumar R, Chug A, Singh AP, Singh D. A systematic analysis of machine learning and deep learning based approaches for plant leaf disease classification: a Review. J Sensors. 2022

16. Tiwari V, Joshi RC, Dutta MK. Dense convolutional neural networks based multiclass plant disease detection and classification using leaf images. Ecol Inform. 2021.

17. Shoaib M, et al. An advanced deep learning models-based plant disease detection: a review of recent research. Front Plant Sci. 2023.

18. Ahmed I, Yadav PK. A systematic analysis of machine learning and deep learning based approaches for identifying and diagnosing plant diseases. Sustain Oper Comput. 2023;4:96–104.

19. Plant disease detection and classification techniques: a comparative study of the performances Wubetu Barud Demilie

20. Liu J, Wang X. Plant diseases and pests detection based on deep learning: a review. Plant Methods. 2021;17(1):1–18

21. Plant Leaf Disease Detection, Classification, and Diagnosis Using Computer Vision and Artificial Intelligence: A Review ANUJA BHARGAVA, AASHEESH SHUKLA, OM PRAKASH GOSWAMI, MOHAMMED H.ALSHARIF, PEERAPONG UTHANSAKUL, AND MONTHIPPA UTHANSAKUL

22. V. K. Vishnoi, K. Kumar, and B. Kumar, ''Plant disease detection using computational intelligence and image processing,'' J. Plant Diseases Protection, vol. 128, no. 1, pp. 19–53, Aug. 2020,

23. B. Liu, Y. Zhang, D. He, and Y. Li, ''Identification of apple leaf diseases based on deep convolutional neural networks,'' Symmetry, vol. 10, no. 1, p. 11, Dec. 2017.

24. R. Chen, H. Qi, Y. Liang, and M. Yang, ''Identification of plant leaf diseases by deep learning based on channel attention and channel pruning,'' Frontiers Plant Sci., vol. 13, Nov. 2022.

25. H. Nazki, S. Yoon, A. Fuentes, and D. S. Park, ''Unsupervised image translation using adversarial networks for improved plant disease recognition,'' Comput. Electron. Agricult., vol. 168, Jan. 2020, Art. no. 105117.

26. Y. Tian, G. Yang, Z. Wang, E. Li, and Z. Liang, ''Detection of apple lesions in orchards based on deep learning methods of CycleGAN and YOLOV3 dense,'' J. Sensors, vol. 2019, pp. 1–13, Apr. 2019.

27. Q. Wu, Y. Chen, and J. Meng, ''DCGAN-based data augmentation for tomato leaf disease identification,'' IEEE Access, vol. 8, pp. 98716–98728, 2020.

28. B. Liu, C. Tan, S. Li, J. He, and H. Wang, ''A data augmentation method based on generative adversarial networks for grape leaf disease identification,'' IEEE Access, vol. 8, pp. 102188–102198, 2020.

29. Z. Lin, S. Mu, A. Shi, C. Pang, and X. Sun, ''A novel method of maize leaf disease image identification based on a multichannel convolutional neural network,'' Trans. ASABE, vol. 61, no. 5, pp. 1461–1474, 2018.

30. J. G. Arnal Barbedo, "Plant disease identification from individual lesions and spots using deep learning," Biosyst. Eng., vol. 180, pp. 96–107, Apr. 2019.

31. S. M. Kiran and D. N. Chandrappa, ‘‘Plant disease identification using discrete wavelet transforms and SVM,’’ J. Univ. Shanghai Sci. Technol., vol. 23, no. 6, pp. 108–114, 2021. Available: https://jusst.org/wp-content/uploads/2021/06/Plant-Disease-Identification-Using-Discrete-Wavelet-Transforms-and-SVM-1.pdf

32. R. R. Patil and S. Kumar, ‘‘Rice-fusion: A multimodality data fusion framework for rice disease diagnosis,’’ IEEE Access, vol. 10, pp. 5207–5222, 2022.

33. D. Zhang, X. Zhou, J. Zhang, Y. Lan, C. Xu, and D. Liang, ‘‘Detection of rice sheath blight using an unmanned aerial system with high-resolution color and multispectral imaging,’’ PLoS ONE, vol. 13, no. 5, May 2018, Art. no. e0187470.

34. V. K. Shrivastava and M. K. Pradhan, ‘‘Rice plant disease classification using color features: A machine learning paradigm,’’ J. Plant Pathol., vol. 103, no. 1, pp. 17–26, Oct. 2020, doi: 10.1007/s42161-020-00683-3.

35. A. K. Rath and J. K. Meher, ‘‘Disease detection in infected plant leaf by computational method,’’ Arch. Phytopathol. Plant Protection, vol. 52, nos. 19–20, pp. 1348–1358, Dec. 2019.

36. L. Hallau, M. Neumann, B. Klatt, B. Kleinhenz, T. Klein, C. Kuhn,M. Röhrig, C. Bauckhage, K. Kersting, A. Mahlein, U. Steiner, andE. Oerke, ‘‘Automated identification of sugar beet diseases using smart- phones,’’ Plant Pathol., vol. 67, no. 2, pp. 399–410, Feb. 2018.

37. J. D. Pujari, R. Yakkundimath, and A. S. Byadgi, "SVM and ANN based classification of plant diseases using feature reduction technique," Int. J. Interact. Multimedia Artif. Intell., vol. 3, no. 7, p. 6, 2016.

38. R. D. L. Pires, D. N. Gonçalves, J. P. M. Oruě, W. E. S. Kanashiro,J. F. Rodrigues, B. B. Machado, and W. N. Gonçalves, ‘‘Local descriptors for soybean disease recognition,’’ Comput. Electron. Agricult., vol. 125, pp. 48–55, Jul. 2016.

39. S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, ‘‘Deep neural networks based recognition of plant diseases by leaf image classification,’’ Comput. Intell. Neurosci., vol. 2016, pp. 1–11, May 2016.

40. P. Jolly and S. Raman, ‘‘Analyzing surface defects in apples using Gabor features,’’ in Proc. 12th Int. Conf. Signal-Image Technol. Internet-Based Syst. (SITIS), Nov. 2016, pp. 178–185.

41. M. Sharif et al., "Detection and classification of citrus diseases in agriculture based on optimized weighted segmentation and feature selection," Computers and Electronics in Agriculture, vol. 150, pp. 220–234, Jul. 2018.

42. N. Sengar, M. K. Dutta, and C. M. Travieso, ‘‘Computer vision based technique for identification and quantification of powdery mildew dis- ease in cherry leaves,’’ Computing, vol. 100, no. 11, pp. 1189–1201, Nov. 2018.

43. S. M. Javidan, A. Banakar, K. A. Vakilian, and Y. Ampatzidis, ‘‘Diagnosis of grape leaf diseases using automatic K-means clustering and machine learning,’’ Smart Agricult. Technol., vol. 3, Feb. 2023, Art. no. 100081, doi: 10.1016/j.atech.2022.100081.

44. S. Zhang, Y. Zhu, Z. You, and X. Wu, "Fusion of superpixel, expectation maximization, and PHOG for recognizing cucumber diseases," Computers and Electronics in Agriculture, vol. 140, pp. 338–347, Aug. 2017.

45. G. Geetharamani and A. Pandian, ‘‘Identification of plant leaf diseases using a nine-layer deep convolutional neural network,’’ Comput. Electr. Eng., vol. 76, pp. 323–338, Jun. 2019

46. M. Mehdipour Ghazi, B. Yanikoglu, and E. Aptoula, ‘‘Plant identification using deep neural networks via optimization of transfer learning parameters,’’ Neurocomputing, vol. 235, pp. 228–235, Apr. 2017.

47. B. Liu, Y. Zhang, D. He, and Y. Li, ‘‘Identification of apple leaf diseases based on deep convolutional neural networks,’’ Symmetry, vol. 10, no. 1, p. 11, Dec. 2017.

48. R. Chen, H. Qi, Y. Liang, and M. Yang, ‘‘Identification of plant leaf dis- eases by deep learning based on channel attention and channel pruning,’’ Frontiers Plant Sci., vol. 13, Nov. 2022.

49. H. Nazki, S. Yoon, A. Fuentes, and D. S. Park, ‘‘Unsupervised image translation using adversarial networks for improved plant disease recog- nition,’’ Comput. Electron. Agricult., vol. 168, Jan. 2020, Art. no. 105117.

50. Y. Tian, G. Yang, Z. Wang, E. Li, and Z. Liang, ‘‘Detection of apple lesions in orchards based on deep learning methods of CycleGAN and YOLOV3-dense,’’ J. Sensors, vol. 2019, pp. 1–13, Apr. 2019.

51. Q. Wu, Y. Chen, and J. Meng, ‘‘DCGAN-based data augmenta- tion for tomato leaf disease identification,’’ IEEE Access, vol. 8, pp. 98716–98728, 2020.

52. B. Liu, C. Tan, S. Li, J. He, and H. Wang, ‘‘A data augmentation method based on generative adversarial networks for grape leaf disease identifi- cation,’’ IEEE Access, vol. 8, pp. 102188–102198, 2020.

53. Z. Lin, S. Mu, A. Shi, C. Pang, and X. Sun, ‘‘A novel method of maize leaf disease image identification based on a multichannel convolutional neural network,’’ Trans. ASABE, vol. 61, no. 5, pp. 1461–1474, 2018.

54. J. G. Arnal Barbedo, "Plant disease identification from individual lesions and spots using deep learning," Biosyst. Eng., vol. 180, pp. 96–107, Apr. 2019/

55. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand,M. Andreetto, and H. Adam, ‘‘MobileNets: Efficient convolutional neural networks for mobile vision applications,’’ 2017, arXiv:1704.04861.

56. Lv, W., & Wang, X. (2020). Overview of hyperspectral image classification. Journal of Sensors, 2020, 1–13. https://doi.org/10.1155/2020/4817234