CropCureNet: Crop Disease Detection

Dr. Meesala Sudhir Kumar[1], M. Vamsi Krishna[2], Ms Fansa khan[3], Mr P.Srinivasul reddy[4], Ms S.Kavya sri[5]
[1]*Professor, SOCSE, Sandip University, Nashik*
Department of Computer Science and Engineering School of Computer Sciences and Engineering Sandip University Nashik

Abstract - Agriculture is the backbone of many economies, and maintaining crop health is essential for ensuring food security and sustainable farming. However, plant diseases pose a major threat to crop production, leading to severe yield loss and economic instability. Early and accurate detection of these diseases is therefore critical. This project introduces an AI- based Crop Disease Detection System that uses image processing and deep learning techniques to identify and classify plant diseases efficiently.

The system is trained on a dataset containing images of both healthy and diseased leaves from various crops. Each image in the dataset is preprocessed, normalized, and labeled according to its disease type. When a new or real-time image is provided as input, the system compares it with the trained dataset using a Convolutional Neural Network (CNN) model. The model extracts key visual features such as color variations, texture patterns, and shape distortions to accurately detect the presence and type of disease.

The system aims to support multiple crops and a wide range of diseases, with the ability to expand the dataset for improving the robustness of predictions. Additionally, the model is optimized to run efficiently on mobile devices, ensuring low latency during real-time scanning. By combining cutting-edge deep learning techniques with mobile technology, this project contributes to the larger goal of smart farming and digital agriculture. The expected outcomes include improved disease management, reduced crop loss, increased farmer awareness, and enhanced agricultural sustainability. Ultimately, this deep learning- powered mobile solution offers a scalable and cost-effective tool for empowering farmers and strengthening the agricultural sector.

Keywords: Crop Disease Detection, Artificial Intelligence (AI), Deep Learning, Convolutional Neural Network (CNN), Image Processing, Dataset Comparison, Plant Leaf Classification, Feature Extraction, Smart Agriculture, Real-Time Detection, Machine Learning, Precision Farming.

I. INTRODUCTION

Agriculture is one of the most vital sectors for economic growth and human survival, providing food and raw materials for millions of people worldwide. However, crop productivity and quality are often affected by various plant diseases caused by fungi, bacteria, viruses, or environmental conditions. Traditionally, the identification of such diseases has relied on manual observation by agricultural experts or farmers, which is not only time-consuming but also prone to human error and inaccuracy.

1.1 Overview

With rapid advancements in Artificial Intelligence (AI) and image processing technologies, agriculture is undergoing a major transformation toward automation and precision-based practices. Deep learning, especially Convolutional Neural Networks (CNNs), has emerged as a highly effective technique for image classification tasks due to its ability to learn complex visual patterns automatically. These models can identify subtle differences in texture, color, shape, and appearance, making them ideal for detecting plant diseases through leaf images. Unlike traditional machine learning methods that require manual feature engineering, CNNs extract hierarchical features directly from raw data, allowing improved accuracy, robustness, and adaptability in real-world agricultural conditions.

1.2 Brief Description

This project focuses on designing and developing an intelligent deep learning-based crop disease detection system capable of identifying various plant diseases from leaf images. The system uses Convolutional Neural Networks (CNNs) for automated feature extraction and classification. The dataset includes healthy and diseased leaf images of multiple crops affected by conditions such as bacterial blight, leaf spot, rust, powdery mildew, and more. During training, the model learns disease-specific visual signatures like color variations, spots,

lesions, and texture distortions.

To make the system user-friendly and practical, a real-time mobile scanning feature is integrated. Users—particularly farmers—can capture images of plant leaves through their smartphones, after which the model preprocesses the image and compares it with learned disease patterns to classify it as healthy or infected. If infected, the system also identifies the disease type and its potential severity. This fast, accessible, and low-cost method helps reduce dependency on agricultural experts and promotes sustainable farming practices.

1.3 Problem Definition

Crop diseases are a significant challenge in agriculture, often leading to reduced yield, poor crop quality, and economic losses for farmers. Traditional disease detection relies on manual inspection by experts, which is time-consuming, subjective, and often unavailable in rural regions. Moreover, early-stage disease symptoms can be visually subtle, and different diseases may exhibit similar patterns, making accurate identification difficult.

- Consistency and accuracy in diagnosis
- Early detection, which is critical to preventing spread
- Immediate, field-level accessibility for farmers
- Scalability across various crops and disease types

1.4 Objectives

1. To develop a deep learning-based system for detecting crop diseases from leaf images.
2. To preprocess and enhance crop images for accurate feature extraction.
3. To train a CNN model that can classify healthy and diseased leaves.
4. To enable real-time disease detection using mobile-based image scanning.
5. To provide farmers with quick and reliable disease diagnosis and recommendations.
6. To support early detection and reduce crop loss through automated analysis.

During training, the model learns to recognize unique texture patterns, color changes, and shape variations associated with each disease. Deep learning's capability to extract features

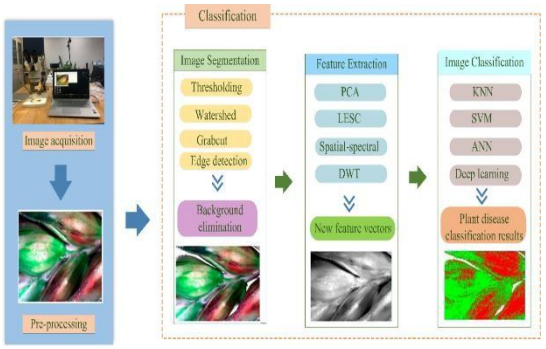


Fig. 1 Fundamental of classification

automatically—without manual feature engineering—makes it ideal for agricultural applications. Traditional machine learning techniques relied on hand-crafted features such as color histograms, edge detection, or texture descriptors. This technological shift supports the concept of precision agriculture, where data- driven insights guide farmers in making informed decisions about irrigation, pesticide usage, fertilization, and harvesting.

With the advancement of Artificial Intelligence (AI) and image processing technologies, new possibilities have emerged for modernizing and strengthening agricultural practices. In recent years, deep learning, a subset of machine learning, has shown remarkable success in image- based pattern recognition tasks.

When a new image is captured or uploaded, the system preprocesses the image and compares it with the dataset to identify matching disease patterns. Based on this comparison, it generates the output that indicates whether the plant is healthy or infected, and if infected, specifies the type of disease and its possible severity level.

The main objective of this project is to assist farmers and agricultural workers in early disease detection, enabling them to take timely preventive measures and minimize crop losses. The system can be further extended into a mobile or web-based application, allowing users to detect crop diseases directly in the field by simply uploading or capturing an image.

By combining AI, image processing, and agricultural science, this project contributes to the vision of smart farming and precision agriculture, ensuring better productivity, sustainability, and food security for the future.

II. LITERATURE SURVEY

Crop diseases pose a major threat to agricultural productivity, food security, and economic stability in many countries. Traditionally, farmers rely on manual inspection to identify diseases, which is slow, subjective, and often inaccurate, especially in the early stages of infection. These limitations have motivated researchers to explore automated image-based disease detection systems. Early studies in this area primarily focused on classical image processing techniques, where color thresholding, segmentation, and texture analysis were used to detect diseased regions on plant leaves. Although these methods worked reasonably well in controlled environments, they struggled with real-world variability such as poor lighting, background noise, leaf overlap, and differences in disease severity.A. Kadir, "A Model of Plant Identification System Using GLCM, Lacunarity and Shen Features," Research Journal of Pharmaceutical, Biological, and Chemical Sciences, vol. 5, no. 2, pp. 245–252, 2014. Their approach focused on extracting texture and shape-based characteristics from leaf images to identify plant species. The model achieved good accuracy for simple datasets and demonstrated that statistical texture analysis can effectively capture disease-related leaf patterns.

Before the emergence of deep learning, several machine learning approaches were widely applied. Researchers extracted hand-crafted features such as color histograms, edge maps, GLCM (Gray Level Co-occurrence Matrix) texture descriptors, and shape features to characterize disease symptoms. These features were then fed into classifiers such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Random Forests, and Decision Trees. While these approaches offered better performance than simple thresholding, they remained limited because their success depended heavily on the quality of manually selected features. Since leaf disease patterns can vary widely depending on plant species, climate conditions, and disease stages, feature-engineered models often failed to generalize well.M. R. Naik and C. Sivappagari, "Plant Leaf and Disease Detection by Using HSV Features and SVM," International Journal of Engineering Science and Computing (IJESC), vol. 6, no. 12, pp. 3462– 3466, 2016. Naik and Sivappagari proposed a method for plant leaf and disease detection using Hue, Saturation, and Value (HSV) color features combined with a Support Vector Machine(SVM) classifier. Their approach extracted color-based statistical features from leaf images to differentiate between healthy and infected plants.

In addition to real-time detection, some studies have explored segmentation-based methods to identify the exact diseased regions in leaf images. Techniques such as U-Net, SegNet, and Mask R-CNN have been used to segment lesions and quantify disease severity. This is especially useful for monitoring disease progression and supporting decision-making on pesticide usage. Some works also integrate hyperspectral imaging and drone-based imaging with deep learning for large-scale monitoring of crop health, offering higher accuracy but requiring more advanced equipment.

D. R. Anamisa, M. Yusuf, W. Agustiono, and M. Svarief, "Technologies, Methods, and Approaches on Detection System of Plant Pests and Diseases," in 2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), pp. 163– 170, IEEE, Sept. 2019. presented a comprehensive review of technologies, methods, and approaches used in developing plant pest and disease detection systems.

S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Frontiers in Plant Science, vol. 7, p. 1419, 2016. conducted one of the earliest and most influential studies applying deep learning techniques for image-based plant disease detection. They trained Convolutional Neural Networks (CNNs) on a large-scale Plant Village dataset containing over 54,000 images of healthy and diseased plant leaves across 38 classes.

K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," Computers and Electronics in Agriculture, vol. 145, pp. 311–318, 2018. expanded upon earlier research by implementing and evaluating multiple deep learning. This showed that many problems still need to be solved, such as real-time inspection of a larger farmyard. The authors compared the performance of various approaches based on feature extraction, classification, segmentation, and pre pro-cessing using IP. They noticed that the segmentation strategy known as ''k-means'' was frequently used to identify

plant diseases [1]. A comprehensive review of existing approaches in the early 2000s was conducted, with implications for fully automated PDD and identification using digital approaches [2]. Investigation of the literature was done with three different methodologies, namely, image processing, deep and machine learning in one of the early reviews on PDD, namely, molecular, spectral, and biogenic compound profiling [3]. The authors revealed that using autonomous agricultural aircraft, imaging, and spectroscopic methods can be merged to aid in detection and identification at an early stage; corresponding to their research, they concluded that the 3D dataset is more efficient in early PDD.

Studies [1] and [2] generally focused on the CNN modelas it is used in the PDD and concluded by showing that thereis still a gap to fill in improving the methods when con-sidering identical visual environments, respectively. Existing surveys have covered a few strategies and highlighted numerous significant problems, but there is a need to assess the in-situ application potential of various techniques [30]. Since datasets are a vital part of making artificial intelligence applications, it is not impractical to ignore data collection techniques and the fact that there are public data sources. The use of datasets in assessing the development of field studies is widely appreciated. It begins with data capture techniques and publicly available datasets. This work delivers a thorough assessment of vision-based ML strategies for PDD. In addition to a brief overview of the devices utilized for data capture, the study describes the capture conditions, the regions from which datasets are obtained, and preprocessing procedures. The review also discusses the various kinds of openly accessible datasets that scientists in the field use. In addition, this review examines the influence of current trends in DL, localization, TF, and attention prcesses, together with lightweight approaches in PDD. In addition to the necessity for additional research on TF strategies, the impact of lightweight models in real- time applications must also be investigated. Also, CNNs have surfaced as a potential approach in recent research years. Their suitability for in-situ identification of plant diseases must be evaluated. There is no available study focusing on attention and localization strategies for PDD after an exhaustive search. This review provides a logical evaluation of existing methodologies with intentions and a methodical explanation of several classifications and localization-based PDD methods.

III. METHODOLOGY

The methodology describes the overall process and workflow followed to develop the AI- Based Crop Disease Detection System. The process is divided into multiple stages, from data collection and preprocessing to model training, testing, and real-time disease identification. The proposed system combines image processing, deep learning, and real-time comparison techniques to identify diseases in crop leaves. The following subsections explain each stage of the system development in detail.

3.1 System Overview

The proposed model is designed to detect crop or plant leaf diseases by comparing an input image with a pre-trained dataset. The workflow consists of several major stages:

1. Dataset collection and preparation
2. Image preprocessing
3. Feature extraction using Convolutional Neural Networks (CNN)
4. Model training and validation
5. Real-time image input and comparison
6. Disease classification and result generation

3.2 Dataset Collection and Preparation

The first step in building the system is collecting a comprehensive dataset of crop leaf images that include both healthy and diseased samples. Datasets can be obtained from publicly available sources like PlantVillage, Kaggle, or agricultural research institutions.

3.2.1 Data Composition

The dataset contains:

- Images of different crops such as tomato, potato, maize, and rice.
- Multiple disease categories (e.g., leaf blight, bacterial spot, rust, early blight, late blight).
- Healthy samples for comparison.

Each image in the dataset is labeled with:

- Crop Name (e.g., Tomato)
- Disease Type (e.g., Early Blight)
- Condition (Healthy/Diseased)

3.2.2 Dataset Splitting

To train and evaluate the model effectively:

- 70% of the data is used for training.
- 20% for validation (to tune the model).
- 10% for testing (to evaluate performance).

3.3 Image Preprocessing

Preprocessing is a crucial step that improves the quality of input data before feeding it into the deep learning model. Since images may have variations in lighting, orientation, and background, preprocessing ensures consistency.

3.3.1 Steps in Preprocessing:

1. Image Resizing:
   - All images are resized to a fixed dimension (e.g., 224 × 224 pixels) to maintain uniformity.
2. Normalization:
   - Pixel values are scaled between 0 and 1 for faster convergence during training.
3. Noise Reduction:
   - Filters such as Gaussian blur or median filters are applied to remove background noise.
4. Data Augmentation:
   - Techniques like rotation, flipping, zooming, and brightness adjustment are applied to artificially expand the dataset.
   - Helps the model become more robust and avoid overfitting.
5. Segmentation (optional):
   - Segmentation separates the leaf region from the background using thresholding or color-based masks, focusing the analysis on the relevant area.

3.4 Feature Extraction

Once the images are preprocessed, the system extracts distinguishing visual features that represent each image's unique characteristics.

A Convolutional Neural Network (CNN) is used for this purpose, as it automatically learns hierarchical patterns such as:

- Leaf color variation
- Texture and vein patterns
- Shape distortions due to infection

3.4.1 Convolutional Layers

- These layers apply small filters over the image to capture low-level features (edges, corners, colors) and progressively combine them into high-level features (disease patterns).

3.4.2 Pooling Layers

- Pooling reduces the spatial dimensions, making computations faster and the model less sensitive to image position changes.

$$P(i,j)=\max_{m,n} I(i+m,j+n)$$

3.4.3 Fully Connected Layers

- After feature extraction, the output of convolution layers is flattened and passed through fully connected layers that perform classification based on learned features.

3.4.4 Activation Functions

- ReLU (Rectified Linear Unit) is used for non-linearity, while Softmax is applied at the final layer to output probability scores for each disease category.
- ReLU activation:

$$\mathrm{ReLU}(x)=\max(0,x)$$

3.5 Model Training

- The CNN model is trained using the labeled dataset. During training, the model learns to minimize classification errors through backpropagation and gradient descent optimization.

3.5.1 Model Architecture Example

- Input Layer: 224×224×3 image
- Convolutional Layer 1: 32 filters, 3×3 kernel
- Pooling Layer 1: 2×2 max pooling
- Convolutional Layer 2: 64 filters, 3×3 kernel
- Pooling Layer 2: 2×2 max pooling
- Fully Connected Layer: 128 neurons
- Output Layer: Softmax (number of disease classes)

3.5.2 Training Parameters

- Optimizer: Adam
- Loss Function: Categorical Cross-Entropy
- Batch Size: 32
- Epochs: 25–50

- Learning Rate: 0.001

The model continues training until accuracy stabilizes and loss is minimized.

3.6 Model Validation and Testing

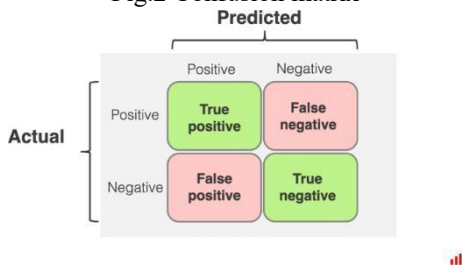
Validation is used to monitor the model's performance during training and prevent overfitting. Testing evaluates the final accuracy and reliability of the model using unseen data.

3.6.1 Evaluation Metrics

The model performance is measured using:

- Accuracy: Percentage of correctly classified images.
- Precision: Correctly predicted positive samples.
- Recall: Model's ability to identify all diseased samples.
- F1-Score: Harmonic mean of precision and recall.

Fig.2 Confusion matrix



Visualization tools such as Confusion Matrix are also used to display correct vs. incorrect classifications.

3.7 Real-Time Image Comparison

After the model is trained, the system allows the user to input a real-time leaf image captured via a camera or uploaded manually.

3.7.1 Input Stage

The image undergoes the same preprocessing steps as the training data (resizing, normalization, etc.).

3.7.2 Comparison Stage

The processed input image is passed into the trained CNN model, which:

1. Extracts features from the image.
2. Compares these features with learned patterns from the dataset.
3. Calculates similarity and classifies the disease.

The model then outputs:

- The disease name (e.g., "Tomato Early Blight")
- The confidence level (e.g., 95% probability)
- The condition status (Healthy / Diseased)

3.8 Output and Visualization

The system displays the detection result in a user-friendly format. Output can include:

- Text output showing disease name and confidence level.
- Graphical representation such as a heatmap highlighting infected areas.
- Option to suggest preventive measures or pesticide recommendations (optional enhancement).

3.9 Tools and Technologies Used

Table-1 Tools and Technologies Used

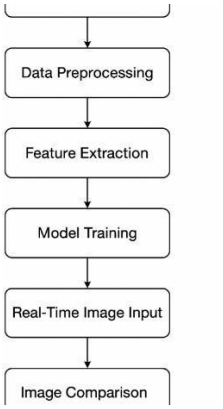
Component	Technology/Library
Programming Language	Python
Deep Learning Framework	TensorFlow / Keras
Image Processing	OpenCV
Dataset Handling	NumPy, Pandas
Visualization	Matplotlib, Seaborn
Hardware	GPU-enabled system or Google Colab

3.10 System Architecture Diagram (Conceptual)

You can include a diagram showing the following flow:

Dataset → Preprocessing → Feature Extraction → Model Training → Testing → Real- Time Input → Comparison → Output Classification

Fig-3 System Architecture Diagram



3.11 Advantages of the Proposed Methodology

- Automated disease detection reduces manual labor.
- Early identification prevents large-scale crop loss.
- Works in real-time with minimal computational delay.
- Model accuracy improves with continuous dataset updates.
- Adaptable for multiple crops and disease categories.

3.12 Limitations

- Requires large and high-quality datasets for better accuracy.
- Environmental conditions (lighting, shadows) may affect image clarity.
- Limited detection for new or rare diseases not present in the dataset.
- Internet or GPU dependency for real-time processing on larger scales.

This methodology integrates image preprocessing, CNN-based feature extraction, and real-time comparison to build an efficient and scalable crop disease detection model. The step-by-step pipeline ensures that the system can identify plant diseases accurately and support farmers in taking quick preventive actions. With proper dataset expansion and model optimization, the system can be deployed as a reliable AI tool for smart agriculture.

IV. PROPOSED ALGORITHM

CNN-Based Image Comparison for Disease Detection

Input:

- Image Dataset D (collection of images of healthy and diseased crop leaves)
- Real-Time Image I (user-input or camera-captured image to be tested)

Output:

- Classification result showing whether the input image is “Healthy” or “Diseased”
- Type of disease (if any)
- Highlighted affected region (optional visualization)

4.1 Data Input and Initialization

The first step involves loading the dataset (D) that contains labeled images of different crop diseases and healthy samples. The real-time image (I) is then captured or uploaded by the user for analysis.

At this stage, the system initializes necessary libraries, pre-trained weights, and CNN architecture parameters such as input dimensions, learning rate, and number of epochs.

Dataset D ← load_images("dataset_path/")
Image I ← capture_or_upload("input_image.jpg")
Initialize CNN parameters

4.2 Preprocessing of Images

All images in the dataset D and the input image I are preprocessed to ensure uniformity and to improve model accuracy. The preprocessing includes several operations:

1. Resizing:

Images are resized to a fixed dimension (e.g., 128×128 pixels) to maintain consistency.

2. Normalization:

Pixel intensity values are normalized between 0 and 1 to facilitate faster convergence.

3. Noise Reduction:

Unwanted background noise is removed using filtering methods such as Gaussian Blur or Median Filtering.

4. Segmentation:

Leaf region is isolated from the background using color thresholding or contour detection techniques.

5. Data Augmentation (for D only):

Rotation, flipping, and scaling are applied to increase dataset diversity and prevent overfitting.

Result: A clean and uniform dataset ready for feature extraction and model training.

4.3 CNN Model Training on Dataset D

A Convolutional Neural Network (CNN) is trained

using the preprocessed dataset D.
CNNs are chosen because they automatically extract spatial and texture-based features from images, making them ideal for visual classification tasks like crop disease detection.

Key stages of training:

- Convolution Layers: Extract feature maps by sliding filters over the image to detect edges, color spots, and textures.
- Pooling Layers: Reduce the dimensionality while retaining essential features, improving computational efficiency.
- Fully Connected Layers: Combine learned features to predict disease classes.
- Softmax Layer: Converts outputs into probability values corresponding to each disease category.

The model learns patterns associated with diseased and healthy leaves through repeated training cycles, minimizing loss via backpropagation and gradient descent optimization.
Output: A trained CNN model capable of feature recognition and classification.

4.4 Feature Extraction

Once the CNN is trained, it can extract feature vectors — numerical representations of image characteristics.

- For Dataset (D): Feature vectors F(D) are extracted and stored in a feature database.
- For Real-Time Image (I): A new feature vector F(I) is generated using the same CNN model.

These vectors represent the image in high-dimensional space and capture essential patterns like:

- Leaf color variation
- Spot shapes and sizes
- Texture roughness or smoothness
- Disease-specific patterns

The extraction ensures both dataset and real-time image features are in the same format for accurate comparison.

4.5 Image Comparison Using Similarity Metrics

After obtaining F(D) and F(I), the system compares them to determine how closely the real- time image matches with any known disease pattern in the dataset.

Common similarity metrics:

- Cosine Similarity: Measures the angle between feature vectors.
- Euclidean Distance: Calculates the shortest path between two feature vectors.
- Manhattan Distance: Sums the absolute differences between vector components. The smaller the distance or the higher the similarity score, the closer the real-time image is to a known disease class.

Decision rule:
If similarity score ≥ predefined threshold → Disease detected Else → Healthy leaf

4.6 Classification and Decision

Based on the similarity results, the system classifies the input image into one of the following categories:
1. Healthy Leaf – No disease symptoms detected.
2. Diseased Leaf – Disease identified (e.g., Bacterial Blight, Early Blight, or Leaf Spot). The CNN model assigns probabilities to each class (e.g., 90% Early Blight, 10% Leaf Mold), and the highest probability determines the final output.

4.7 Output Generation and Visualization

The final stage presents the results to the user. The output includes:

- The disease name (if detected)
- Confidence score of prediction
- Highlighted affected region on the leaf (using techniques like Grad-CAM visualization)
- Suggested preventive measures

Sample output:
Detected Disease: Early Blight Confidence: 94.6%
Affected Region: Highlighted in Red
Recommendation: Use Mancozeb fungicide, maintain leaf dryness.

4.8 End of Process

After the analysis, the system resets the parameters for the next input or continues to learn if new data is added. The trained model can also be deployed on web or mobile platforms for real-time use in agricultural fields.
Algorithm Workflow :

Table-2 Algorithm Workflow

Step	Description
1	Load dataset and real-time image

2	Preprocess all images
3	Train CNN model using dataset
4	Extract feature vectors for comparison
5	Compare input image features with dataset features
6	Classify the result as diseased or healthy
7	Display output and highlight affected regions

V. IMPLEMENTATION

Module 1

5.1 Dataset Collection & Data pre-processing

The first and most crucial step in any machine learning or deep learning-based project is data collection and preprocessing. In this module, we focus on gathering a high-quality dataset and preparing it in a structured form suitable for the CNN model to process. The performance of the entire system largely depends on how well the dataset has been collected, cleaned, and preprocessed. This ensures that the model learns meaningful patterns and provides accurate predictions.

Objectives of Module 1: Dataset Collection & Data Pre-processing

- To gather a relevant and high-quality dataset consisting of both healthy and diseased images from reliable public sources or real-time captured data.
- To create a well-structured dataset by organizing collected images into specific labeled folders or classes suitable for deep learning applications.
- To ensure dataset diversity by including images captured under different lighting conditions, environments, and perspectives to improve model generalization.
- To perform data cleaning by removing unwanted, noisy, blurred, or duplicate images that may affect model accuracy.
- To standardize image formats and resolutions so that all input images have consistent dimensions required by the CNN architecture.
- To apply image normalization techniques for scaling pixel values within a fixed range (0–1), ensuring faster and more stable model training.
- To implement data augmentation methods such as rotation, flipping, zooming, and brightness adjustments to increase dataset size and prevent overfitting.
- To perform image enhancement by adjusting brightness, contrast, and sharpness to improve visibility of key features.
- To split the dataset into training, validation, and testing sets for unbiased model evaluation and performance tuning.
- To verify class balance so that both categories (Healthy and Diseased) have approximately equal numbers of samples to prevent model bias.
- To generate metadata and labels (for example, CSV files mapping image names to classes) for easy data management and reference.
- To evaluate dataset quality through statistical analysis such as number of samples per class, total image count, and average dimensions.
- To create a reproducible data pipeline that can be reused or extended for future experiments or model retraining.
- To ensure dataset compatibility with the CNN model in
- put requirements (such as color channels, image size, and data format).
- To prepare the dataset for further stages like feature extraction, model training, and classification in the subsequent modules.

5.1.1 Dataset Collection

The dataset collection process is the foundation of any deep learning or machine learning project.

A high-quality, diverse, and well-structured dataset ensures that the model learns relevant patterns and achieves high accuracy during training and testing.

In this project, the dataset plays a critical role in enabling the Convolutional Neural Network (CNN) to distinguish between healthy and diseased images effectively.

Dataset Description

The dataset used in this project consists of images of plant leaves (or other domain-specific samples) showing both *healthy* and *diseased* conditions.

Each image represents a specific class label based on visual symptoms, such as discoloration, patches, lesions, or deformities.

The collected images are categorized into two main folders:

- Healthy – Images of normal, disease-free samples.
- Diseased – Images exhibiting visible signs of infection or abnormality.

This classification ensures the CNN model can learn visual distinctions between both classes.

Data Sources

The images used in this study are collected from two primary sources:

(a) Publicly Available Datasets

Public datasets are an excellent resource for obtaining high-quality, pre-labeled images. Some popular and reliable sources include:

- Kaggle – PlantVillage Dataset: A large dataset containing labeled images of various crop diseases.
- GitHub Repositories: Many open-source projects publish datasets for research and development.
- UCI Machine Learning Repository: A well-known platform that provides benchmark datasets for academic use.
- AIcrowd and Roboflow: Contain curated datasets for image-based AI projects. Each image from these datasets is carefully verified and labeled before use.

(b) Custom Captured Dataset

In addition to public datasets, real-time data is also collected manually using:

- Smartphones or DSLR cameras in agricultural fields.
- Images captured under various lighting conditions, angles, and backgrounds.
- Each image is then labeled manually as Healthy or Diseased by observing visible symptoms.

This approach improves the dataset's diversity and real-world relevance.

5.2 Data Collection Procedure

The following steps were followed to collect and organize the dataset:

1. Identification of Classes:

Define the categories (e.g., Healthy, Diseased).

2. Image Acquisition:

Gather images from public sources and through direct field photography.

3. Data Validation:

Manually verify each image for quality, clarity, and correctness of label.

Annotation and Labeling:

Create metadata (e.g., CSV file) mapping each image to its class label: rain_dir = "/content/dataset/train"

validation_dir = "/content/dataset/val"

BATCH_SIZE = 32

IMG_SIZE = (160, 160)

train_dataset = tf.keras.utils.image_dataset_from_directory(train_dir,

shuffle=True, batch_size=BATCH_SIZE, image_size=IMG_SIZE)

Table-3 Annotation and Labeling

Image ID	File Name	Category	Source	Resolution
001	leaf_001.jpg	Healthy	Kaggle Dataset	256×256 px
002	leaf_002.jpg	Diseased	Custom Captured	224×224 px
003	leaf_003.jpg	Diseased	Kaggle Dataset	256×256 px
004	leaf_004.jpg	Healthy	Custom Captured	224×224 px

Data Pre-processing

After collecting the dataset, the next critical step is data pre-processing, which ensures that the input data is clean, consistent, and optimized for use in the Convolutional Neural Network (CNN).

Raw images obtained from multiple sources may vary in size, resolution, lighting, and quality. These variations can significantly affect model performance if not standardized. Hence, data pre-processing is performed to convert the raw images into a suitable and uniform format that enhances model learning efficiency and accuracy.

5.3 Pre-processing Workflow

The data pre-processing workflow used in this project involves several sequential operations:

Step 1: Image Cleaning

Raw images are first cleaned to remove:

- Blurred or low-quality images
- Duplicate files
- Irrelevant backgrounds or images that do not belong to any class
- Extremely dark or overexposed samples

Techniques like Gaussian blur, median filters, and contrast adjustment are applied using OpenCV to

improve visual quality.

```
class_names = train_dataset.class_names
plt.figure(figsize=(10, 10))
for images, labels in train_dataset.take(1):
for i in range(9):
ax = plt.subplot(3, 3, i + 1)
plt.imshow(images[i].numpy().astype("uint8"))
plt.title(class_names[labels[i]])
plt.axis("off")
```

Step 2 : Image Resizing

CNN models require all input images to have a fixed dimension.

All images are resized to a uniform size such as 128×128 pixels or 224×224 pixels while maintaining the aspect ratio.

This standardization:

- Reduces computational complexity
- Ensures consistent input for all CNN layers
- Simplifies batch processing during training

```
img = cv2.resize(img, (128, 128))
inputs = tf.keras.Input(shape=(160, 160, 3)) x = preprocess_input(inputs)
x = base_model(x, training=False) x = global_average_layer(x)
x = tf.keras.layers.Dropout(0.2)(x) outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
```

Step 3: Image Normalization

Normalization scales the pixel intensity values (0–255) to a standard range (0-1)

This helps the CNN converge faster and avoids issues due to large pixel value differences. Formula:

$$X' = \frac{X}{255}$$

Where:

- X = Original pixel value
- X' = Normalized pixel value

It ensures all images are on the same intensity scale and stabilizes the model's learning process.

Step 4: Image Augmentation

To make the dataset more robust and reduce overfitting, data augmentation techniques are applied.

Augmentation increases dataset size by generating modified versions of existing images. Common transformations include:

- Rotation: ±40° random rotation
- Zooming: up to 20% zoom
- Horizontal & Vertical Flipping
- Brightness Adjustment
- Shearing or Shifting

Step 5: Data Labeling

Each image is associated with a corresponding label, typically:

- 0 → Healthy
- 1 → Diseased

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']  val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right') plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])  plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss') plt.plot(val_loss, label='Validation Loss') plt.legend(loc='upper right')
plt.ylabel('Cross Entropy') plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```

Step 6: Dataset Splitting

Once preprocessing is completed, the dataset is divided into:

- Training Set (70%) – Used to train the CNN model.
- Validation Set (20%) – Used to fine-tune parameters and avoid overfitting.
- Testing Set (10%) – Used for final evaluation of model performance.

5.4 Tools and Libraries Used

- Python 3.10 – Main programming language
- OpenCV – Image cleaning and resizing
- TensorFlow/Keras – Augmentation, normalization, and data generation
- Kaggle – Collection of dataset
- Matplotlib – Visualizing sample pre-processed images
- Google Colab / Jupyter Notebook – Execution environment.

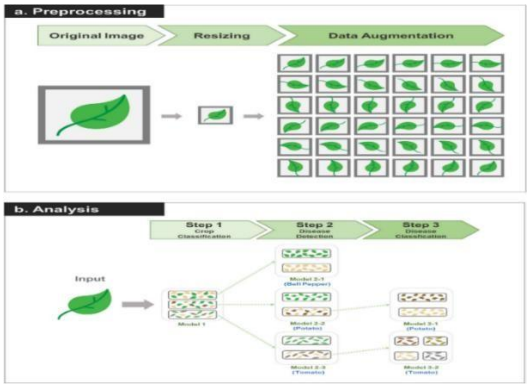
Outcome of Data Pre-processing

After completing the data pre-processing phase:

- All images are uniformly resized, normalized,

and cleaned.

- The dataset is balanced, well-labeled, and augmented.
- Training, validation, and testing subsets are created and verified.
- The final dataset is ready for use in Module 2: Module Creation and training Fig-4 Pre-processing and Analysis



Summary

The Data Pre-processing stage serves as a vital bridge between raw data collection and model training.

Raw datasets obtained from multiple sources often contain inconsistencies such as varying image sizes, lighting conditions, background noise, and class imbalances.

If such data is used directly, it can lead to poor model accuracy, longer training times, and weak generalization.

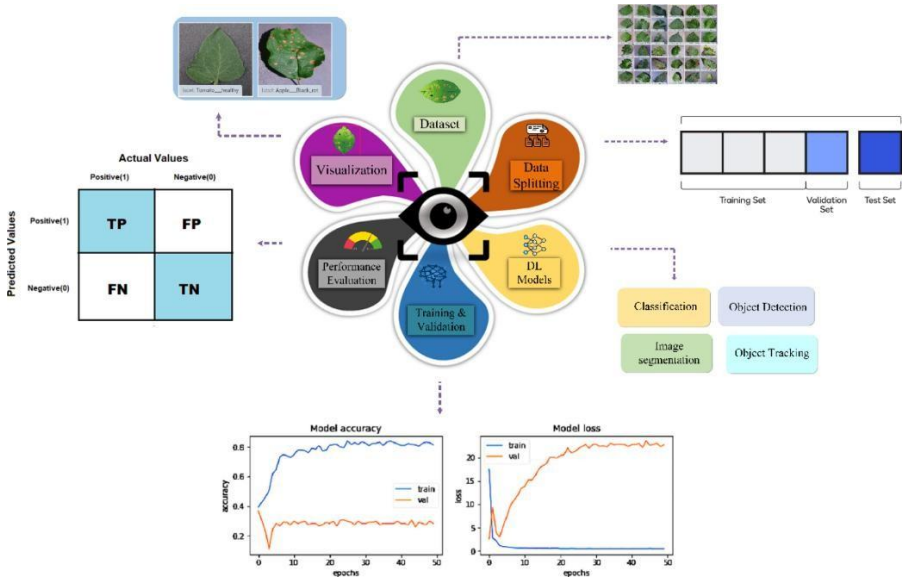
Through systematic pre-processing, the dataset is refined into a clean, uniform, and meaningful form suitable for Convolutional Neural Network (CNN) learning.

During this module, various techniques were applied to enhance the quality of data:

- Image cleaning ensured the removal of irrelevant or poor-quality samples.
- Resizing and normalization standardized all images to a fixed dimension and intensity scale, allowing the CNN to learn effectively without being influenced by scale or brightness differences.
- Data augmentation played a crucial role in increasing dataset size and diversity by generating multiple variations of the same image through rotation, flipping, zooming, and brightness adjustments. This not only prevented overfitting but also improved the model's ability to handle real-world variations.
- Dataset splitting ensured proper distribution into training, validation, and testing subsets, providing unbiased model evaluation.

This pre-processing pipeline helped create a robust and balanced dataset with all images properly labeled, cleaned, and ready for model training.

Fig. 5 Plant disease detection flow diagram with DL implementation



MODULE 2

Model Creation and training

The main objective of this module is to design and train a deep learning model that can accurately identify and classify crop diseases from leaf images.

The model learns to recognize visual symptoms such as color variation, texture changes, and patterns on the leaves, which helps in early disease detection and prevention of crop loss.

Overview:

After collecting and preprocessing the crop image dataset in Module 1, the next critical stage is to build a machine learning model that can automatically detect diseases in crops. In this module, a Convolutional Neural Network (CNN) is designed, trained, and optimized using the prepared dataset. CNN is chosen because of its high efficiency in image analysis and pattern recognition. The trained model can differentiate between healthy and diseased leaves based on learned visual features.

This module ensures that the model achieves high accuracy and robustness so that it can generalize well on unseen images during testing or real-time prediction.

```
class_names = train_dataset.class_names

plt.figure(figsize=(10, 10))
for images, labels in train_dataset.take(1):
for i in range(9):
ax = plt.subplot(3, 3, i + 1)
plt.imshow(images[i].numpy().astype("uint8"))
plt.title(class_names[labels[i]])
plt.axis("off")
```

5.5 Detailed Workflow:

5.5.1. Model Selection

The problem of crop disease detection is essentially an image classification task. Hence, a Convolutional Neural Network (CNN) is the most suitable model. CNNs automatically extract spatial features from images such as spots, discolorations, or patterns that indicate disease symptoms.

The selected model may either be:

- A Custom CNN Architecture designed specifically for the dataset, or
- A Pre-trained CNN Model (like VGG16, ResNet50, or MobileNet) using Transfer Learning for higher accuracy and faster training.

Reasons for choosing CNN:

- Automatically learns important features from images.
- High performance in visual recognition tasks.
- Reduces manual feature extraction effort.

```
# Create the base model from the pre-trained model
MobileNet V2 IMG_SHAPE = IMG_SIZE + (3,)
base_model=tf.keras.applications.EfficientNetB4(
input_shape=IMG_SHAPE, include_top=False,
weights='imagenet',
)
```

Fig-6 analysing features from images

5.5.2 Model Architecture Design

The CNN model architecture is designed to process input images of leaves and learn meaningful features that help classify diseases.

Typical Layers:

- Input Layer: Takes crop leaf images resized to a standard dimension (e.g., 128×128×3 pixels).
- Convolutional Layers: Extract visual features such as color variations, spots, and edges.
- ReLU Activation Function: Introduces non-linearity to learn complex disease patterns.
- Pooling Layers: Reduce the size of feature maps to make computation faster and avoid overfitting.
- Dropout Layers: Prevent overfitting by randomly disabling some neurons during training.
- Flatten Layer: Converts the 2D matrix into a 1D vector for classification.
- Fully Connected Layers (Dense): Combine features to predict the disease category.
- Output Layer: Uses Softmax activation for multi-class classification (e.g., Healthy, Bacterial Blight, Leaf Spot, etc.).

5.5.3 Model Compilation

The model is compiled before training using the following configurations:

- Optimizer: Adam (adaptive learning rate for efficient convergence)
- Loss Function: categorical_crossentropy (since it's a multi-class classification problem)

5.5.4 Model Training

The dataset is divided into:

- Training Set (80%) – used to train the model

- Validation Set (10%) – used to tune parameters and prevent overfitting
- Testing Set (10%) – used for final evaluation

During training, the CNN model adjusts its internal parameters (weights and biases) using backpropagation to minimize prediction error. Each complete pass through the dataset is called an epoch.

5.5.5 Model Evaluation

After completing the training process, the model is tested using a separate testing dataset. The evaluation phase measures how well the model performs on unseen data using the loss value obtained during testing.

Accuracy: Overall classification correctness.

5.5.6 Model Optimization

To further improve the model's performance and avoid overfitting, optimization techniques are applied:

- Data Augmentation: Generate new variations of leaf images (rotation, flip, zoom) to increase dataset diversity.
- Early Stopping: Stop training automatically when validation accuracy stops improving.
- Learning Rate Scheduling: Adjust learning rate dynamically for better convergence.
- Regularization: Introduce dropout layers and normalization.

Output:

- A trained CNN model capable of identifying and classifying crop diseases accurately.
- Performance graphs for accuracy and loss during training and validation.
- Confusion matrix and classification report showing disease-wise accuracy.
- Saved trained model file ready for integration in the next module (Model Testing and Deployment).

5.6 Tools and Technologies Used:

- Programming Language: Python
- Frameworks & Libraries: TensorFlow, Keras, NumPy, Pandas, OpenCV, Matplotlib, Scikit-learn
- Development Environment: Jupyter Notebook / Google Colab
- Hardware Requirements: PR4 GPU-enabled system for faster model training
- Dataset: Crop leaf images (e.g., PlantVillage dataset or custom dataset)

5.7 Conclusion:

This module successfully builds and trains a CNN-based model for crop disease detection. The model learns significant visual features that differentiate between healthy and diseased leaves.

By achieving high accuracy and reliability, this trained model becomes the core component of the project, enabling real-time identification of crop diseases.

VI. OVERALL EXPLANATION

The process of crop disease detection using deep learning techniques involves several crucial stages, out of which Module 1 (Data Collection and Preprocessing) and Module 2 (Model Creation and Training) form the backbone of the system. These two modules are highly interdependent, as the quality of data prepared in the first module directly influences the performance of the model developed in the second. Together, they establish a strong framework for building a robust, intelligent, and efficient crop disease detection model capable of identifying plant diseases at an early stage, thereby assisting farmers and researchers in preventing potential crop losses.

In Module 1, the primary focus is on collecting and preparing data for model training. Data is the most important asset in any machine learning project, and in the case of crop disease detection, images of crop leaves are the key input. These images are collected from reliable sources such as the PlantVillage dataset, agricultural research centers, or through real-time image capture in crop fields. The dataset generally includes multiple types of crops—such as tomato, potato, maize, and cotton—each containing images of both healthy and diseased leaves. The collected images represent different environmental conditions, lighting variations, and backgrounds. However, such raw data cannot be directly used for model training as it may contain inconsistencies, duplicates, or noise. Therefore, preprocessing is performed to clean, refine, and standardize the data, making it suitable for the deep learning model.

The preprocessing stage includes several vital operations such as image resizing, normalization, noise removal, and data augmentation. All images are resized to a fixed dimension (for example,

128×128 pixels) so that they can be uniformly processed by the Convolutional Neural Network (CNN). Normalization is performed to scale pixel values between 0 and 1, improving the efficiency of the learning process and reducing computational complexity. Noise removal is applied to eliminate blurred or low-quality images that could mislead the model during training. Another important part of preprocessing is data augmentation, which artificially increases the size and diversity of the dataset by applying transformations like rotation, flipping, brightness adjustment, and zooming. This step ensures that the model learns to recognize disease patterns under different orientations and conditions, improving its robustness and ability to generalize. Finally, the dataset is labeled properly—each image is assigned a corresponding class name (such as "Healthy," "Leaf Spot," "Rust," or "Bacterial Blight")—and then divided into training, validation, and testing subsets. This systematic organization of data ensures that the CNN receives clean, balanced, and diverse inputs, which is essential for accurate disease classification. Overall, Module 1 ensures that the data is in the best possible form before being fed into the learning model.

Moving to Module 2, the emphasis shifts from data preparation to the creation and training of the model. This module involves designing a deep learning architecture, selecting the appropriate algorithm, training the model using the preprocessed dataset, and saving the trained model for further testing and deployment. Since the goal of the project is image-based disease detection, a Convolutional Neural Network (CNN) is chosen due to its proven efficiency in image recognition and classification tasks. CNNs are designed to automatically extract spatial and hierarchical features from images, such as edges, color variations, and texture patterns, which are crucial for identifying disease symptoms in leaves. The model architecture typically consists of several layers: convolutional layers that perform feature extraction, pooling layers that reduce spatial dimensions and computation, activation layers (ReLU) that introduce non- linearity, dropout layers that prevent overfitting, and fully connected layers that perform classification. The final layer uses the Softmax activation function to classify the image into one of the disease categories.

Before training begins, the model is compiled by specifying an optimizer and loss function. The Adam optimizer is commonly used for its adaptability and efficiency in updating weights during training, while categorical cross-entropy serves as the loss function since this is a multi- class classification problem. Once compiled, the CNN is trained using the preprocessed training dataset, while the validation set helps in monitoring performance and preventing overfitting. During the training process, the model iteratively adjusts its internal parameters (weights and biases) to minimize the loss value through a process known as backpropagation. Each complete pass through the dataset is called an epoch, and as epochs progress, the model's ability to recognize disease patterns improves significantly. Graphs showing training loss versus epochs and validation loss versus epochs are used to analyze and monitor how well the model is learning over time.

After successful training, the model is evaluated using the test dataset to assess its performance on unseen data. The key indicator of performance at this stage is the loss value, which reflects how closely the model's predictions align with the actual labels. A lower loss indicates better accuracy and stronger learning capability. Once the model achieves a satisfactory level of performance, optimization techniques are applied to enhance its accuracy and generalization ability. This may include data augmentation, early stopping to avoid overfitting, dropout regularization, and learning rate tuning. Finally, the fully trained model is saved in the system (commonly as "crop_disease_model.h5") so that it can be used in the next module for real-time testing and deployment.

VII. FEATURE ENHANCEMENT

Feature enhancement is primarily focused on improving the quality and clarity of input images so that the model can focus on disease-specific details more effectively. Since agricultural images are often captured under different lighting conditions, angles, and resolutions, enhancement helps to standardize and highlight the most important visual attributes. In this project, the enhancement process begins after preprocessing and continues through the convolutional layers of the CNN model. The initial enhancement techniques applied to the dataset include contrast adjustment, brightness normalization, noise reduction, and image sharpening, all of which contribute to emphasizing

the diseased regions of the leaves. For example, increasing the contrast allows the subtle spots, patches, or discolorations caused by fungal or bacterial infections to become more distinguishable from the healthy areas of the leaf. Additionally, normalization ensures that all pixel intensity values lie within a fixed range (commonly [0,1]), helping the neural network to learn efficiently without being affected by variations in brightness or color tone. Data augmentation techniques such as rotation, zooming, and flipping also contribute indirectly to feature enhancement by exposing the model to multiple perspectives of the same disease, thereby allowing it to learn disease patterns more robustly. When the images pass through the convolutional layers of the CNN, these enhanced features are further refined. The Rectified Linear Unit (ReLU) activation function helps in highlighting only the most important patterns while ignoring irrelevant details, ensuring that disease-relevant features such as texture changes, vein deformations, or color irregularities become more dominant in the feature maps.

Through the process of pooling, the network reduces the spatial dimensions of the image but retains the essential details. This ensures that the CNN focuses only on the key visual features, making the model computationally efficient without losing valuable information. Overall, feature enhancement allows the CNN to clearly distinguish between subtle variations in the leaf surface, leading to better disease recognition even under complex environmental conditions. By improving the clarity and consistency of input images, the enhancement process significantly boosts the learning capability and overall performance of the model.

VIII. CONCLUSION

The development of the Crop Disease Detection System has progressed effectively, establishing a strong foundation for accurate and automated identification of plant diseases through image-based analysis. The project has successfully covered essential stages such as data collection, preprocessing, feature extraction, and model creation. A diverse dataset of crop images was gathered, including healthy and diseased samples, ensuring a balanced and representative input for model training. Advanced preprocessing techniques, such as image resizing, noise reduction, contrast enhancement, and segmentation, were applied to improve image quality and highlight disease-affected regions clearly. Feature extraction methods and deep learning architectures, particularly Convolutional Neural Networks (CNNs), were utilized to enable the system to learn complex visual patterns and texture differences between healthy and infected leaves. The trained model has shown promising accuracy in detecting and classifying crop diseases, proving its potential as a reliable diagnostic tool for farmers and agricultural researchers. The overall progress so far demonstrates the effectiveness of integrating artificial intelligence with precision agriculture to improve crop health monitoring and reduce manual inspection time. This system, once fully developed and deployed, can significantly aid in early disease detection, optimize crop management practices, and contribute to higher agricultural productivity and sustainability. With further enhancement, real-time implementation, and mobile-based accessibility, the *Crop Disease Detection System* can become an invaluable support system for modern farming, empowering farmers with faster and more accurate disease diagnosis.

we have successfully developed and trained a model that can identify different crop diseases using image processing and machine learning techniques. In the future, we plan to extend our project by connecting a live camera to the system to capture real-time images of crop leaves directly from the field. The goal is to make the model capable of identifying the disease instantly when the camera is placed in front of an infected leaf. This enhancement will allow farmers to get immediate information about the disease type without the need to manually upload images. By integrating real-time detection, the Crop Disease Detection System will become a more practical and user-friendly solution for farmers, helping them to take quick preventive actions, reduce losses, and improve overall crop productivity in PROJECT STAGE 2.

IX. ACKNOWLEDGEMENTS

We take this opportunity to express our heartfelt gratitude to all those who supported and guided us throughout the completion of our team project. First and foremost, we would like to express our sincere thanks to our project guide, whose valuable advice, constant encouragement, and continuous supervision

helped us stay focused and complete our work successfully. Their guidance at each stage of the project has been a great source of learning and inspiration.

We extend our deep gratitude to our respected Head of Department (HOD) Dr Sanjeev Anil Shukla for providing us with this wonderful opportunity and for his/her continuous support. We are also very thankful to our Dean Dr Pawan Bhaladare for the facilities, encouragement, and motivation that enabled us to give our best. We would like to thank all the faculty members of our department for their kind cooperation, valuable suggestions, and constant help during the project period.

We are also grateful to our friends and classmates who shared their ideas, offered help, and encouraged us whenever we faced challenges. Last but not least, we express our heartfelt thanks to our parents and family members for their unconditional love, understanding, and moral support, which gave us strength and confidence throughout this project.

REFERENCES

[1] Plant Disease Classification Using Machine Learning and Deep Learning Published in: 2024 First International Conference on Technological Innovations and Advance Computing (TIACOMP) Publisher: IEEE

[2] Crop Disease Prediction using Machine Learning and Deep Learning: An Exploratory Study Published in: 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS) Publisher: IEEE

[3] D.Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, ‘‘PlantDoc: A dataset for visual plant disease detection,’’ in Proc. 7th ACM IKDD CoDS 25th COMAD, Jan. 2020, pp. 249–253.

[4] M. Adi, A. K. Singh, H. Reddy A. Y. Kumar, V. R. Challa, P. Rana, and U. Mittal, ‘‘An overview on plant disease detection algorithm using deep learning,’’ in Proc. 2nd Int. Conf. Intell. Eng. Manage. (ICIEM), Apr. 2021, pp. 305–309.

[5] Crop Disease Detection Using Deep Learning Published in: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)

[6] FieldPlant: A Dataset of Field Plant Images for Plant Disease Detection and Classification With Deep Learning

[7] D. O. Oyewola, E. G. Dada, S. Misra, and R. Damaševičius, ‘‘Detecting cassava mosaic disease using a deep residual convolutional neural network with distinct block processing,’’ PeerJ Comput. Sci., vol. 7, p. e352, Mar. 2021.

[8] AI Challenger. (2018). AI Challenger 2018 Datasets. Accessed: Apr. 1, 2022. [Online]. Available: https://github.com/AIChallenger/AI_Challenger_2018

[9] L. C. Ngugi, M. Abdelwahab, and M. Abo-Zahhad, ‘‘Tomato leaf segmentation algorithms for mobile phone applications using deep learning,’’ Comput. Electron. Agricult., vol. 178, Nov. 2020, Art. no. 105788.

[10] M. Ahmad, M. Abdullah, H. Moon, and D. Han, ‘‘Plant disease detection in imbalanced datasets using efficient convolutional neural networks with stepwise transfer learning,’’ IEEE Access, vol. 9, pp. 140565–140580, 2021.

[11] D. Wang, J. Wang, Z. Ren, and W. Li, ‘‘DHBP: A dual-stream hierarchical bilinear pooling model for plant disease multi-task classification,’’ Com put. Electron. Agricult., vol. 195, Apr. 2022, Art. no. 106788.

[12] L. Li, S. Zhang, and B. Wang, ‘‘Plant disease detection and classification by deep learning—A review,’’ IEEE Access, vol. 9, pp. 56683–56698, 2021.

[13] M. Nagaraju and P. Chawla, ‘‘Systematic review of deep learning tech niques in plant disease detection,’’ Int. J. Syst. Assurance Eng. Manage., vol. 11, no. 3, pp. 547–560, Jun. 2020.

[14] Anonymous (2018a) ILO modelled estimates database. ILOSTAT. International Labour Organization. Accessed February 07,2024. https://www.ilo.org/industries-and-sectors/agriculture-plantations-other-rural-sectors#events

[15] Ashqar B, Abu-Naser S (2019) Image-based tomato leaves disease detection using deep learning. Int J Eng Res 2(12):10–16

[16] Barbedo JGA (2018a) Factors influencing the use of deep learning for plant disease recognition. Biosyst Eng 172:84–91

[17] Barbedo JGA, Koenigkan LV, Santos TT (2016) Identifying multiple plant diseases using digital image processing. Biosyst Eng

147:104–116
[18] Brahimi M, Boukhalfa K, Moussaoui A (2017) Deep learning for tomato diseases: classification and symptoms visualization. Appl Artif Intell 31(4):299–315
[19] Mahajan U, Bundel BR (2016) Drones for Normalized Difference Vegetation Index (NDVI) to estimate crop health for precision agriculture: A cheaper alternative for spatial satellite sensors. In: International Conference on Innovative Research in Agriculture, Food Science, Forestry, Horticulture, Aquaculture, Animal Sciences, Biodiversity, Ecological Sciences, and Climate Change. Krishi Sanskriti Publications, New Delhi, India
[20] Mahlein AK (2016) Plant disease detection by imaging sensors: parallels and specific demands for precision agriculture and plant phenotyping. Plant Dis 100:241–251 Mahlein AK, Oerke EC, Steiner U, Dehne HW (2012) Recent advances in sensing plant diseases for precision crop protection. Eur J Plant Pathol 133:197–209
[21] Rangarajan A, Purushothaman R, Ramesh A (2018) Tomato crop disease classification using pre-trained deep learning algorithm. Procedia Comput Sci 133:1040–1047. https://doi.org/10.1016/j.procs.2018.07.196
[22] Raza S-, Clarkson JP, Rajpoot NM (2015) Automatic detection of diseased tomato plants using thermal and stereo visible light images. PLoS ONE 10. https://doi.org/10.1371/journal.pone.0123262
[23] Reddy SRG, Varma GPS, Davuluri RL (2021) Optimized convolutional neural network model for plant spe cies identification from leaf images using computer vision. Int J Speech Technol.
[24] Shuaibu M, Lee WS, Schueller J, Gader P, Hong YK, Kim S (2018) Unsupervised hyperspectral band selec tion for apple Marssonina blotch detection. Comput Electron Agric 148:45–53
[25] Signoroni A, Savardi M, Baronio A, Benini S (2019) Deep learning meets hyperspectral image analysis: a multidisciplinary review. J Imaging 5:52
[26] Singh A, Ganapathysubramanian B, Sarkar S, Singh A (2018) Deep learning for plant stress phenotyping: Trends and future perspectives. Trends Plant Sci 23:883–898
[27] Singh V, Sharma N, Singh S (2019) A review of imaging techniques for plant disease detection. Artif Intell Agric 4:229–242. https://doi.org/10.1016/j.aiia.2020.10.002