A Project Report on

Plant Disease Detection Using CNN

Submitted in partial fulfillment for award of

Bachelor of Technology

Degree

in

Computer Science and Engineering

By

V. Rakesh (Y20ACS584) **P. Kethana Sai (Y20ACS531)**

S. Appala Swamy (Y20ACS555) **Y. Hussain Valli (Y20ACS588)**

Under the guidance of

Mr. Krishna Kishore Thota, M.Tech

Asst. Professor

Department of Computer Science and Engineering

Bapatla Engineering College

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

BAPATLA – 522 102, Andhra Pradesh, INDIA

2023-2024

Department of
Computer Science and Engineering

CERTIFICATE

This is to certify that the project report entitled **Plant Disease Detection Using CNN** that is being submitted by V. Rakesh (Y20ACS584), P. Kethana Sai (Y20ACS531), S. Appala Swamy (Y20ACS555), Y. Hussain Valli (Y20ACS588) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide
Mr. Krishna Kishore Thota
Asst. Prof.

Signature of the HOD
Dr. M. Rajesh Babu
Assoc. Prof. & Head

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

V. Rakesh (Y20ACS584)

P. Kethana Sai (Y20ACS531)

S. Appala Swamy (Y20ACS555)

Y. Hussain Valli (Y20ACS588)

Acknowledgement

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Mr. Krishna Kishore Thota**, Asst. Prof., Department of CSE, for his valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, Assoc. Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr. Nazeer Shaik** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

V. Rakesh (Y20ACS584)

P. Kethana Sai (Y20ACS531)

S. Appala Swamy (Y20ACS555)

Y. Hussain Valli (Y20ACS588)

Abstract

Modern agriculture faces numerous challenges, from combating diseases that threaten food security. These diseases instigated by a varied spectrum of pathogens encompassing fungi, bacteria, and viruses, have the potential to inflict significant reductions in crop yields, economic harm, and disruptions to the integrity of food supply chains. So, we here present a comprehensive overview of advancements in plant leaf disease detection through the application of deep learning methodologies.

Existing agricultural practices often rely on outdated methods for disease detection, yield prediction, and nutrient management. Farmers face challenges in identifying crop diseases accurately. Moreover, these methods may not detect diseases in their early stages, when intervention is most effective. To overcome these hurdles, we have explored the potential of deep learning, particularly Convolutional Neural Networks (CNNs), for automated and enhanced plant leaf disease detection.

Automatic detection and categorization of diverse agricultural diseases are required for reliable diagnosis under this regard. Deep learning's CNN network category is mostly utilized for picture categorization. Convolutional Neural Networks (CNNs), a category of artificial neural networks, demonstrate proficiency in discerning and acquiring intricate features from image data. The primary objective of the proposed study is to discover a way to address the problem of detecting 38 distinct kinds of plant illnesses by using easiest technique to obtain better results than the standard models.

Key Words: Agriculture, CNN, Deep Learning, Plant Disease Detection, Automation

Table of Contents

List of Tables

List of Figures

1 Introduction

A disease detection model for plants has the potential to greatly benefit farmers and improve crop productivity. In addition to classifying the plants, the model could also provide valuable information such as prevention measures and supplements.

1.1 Introduction

Diseases are very detrimental to the health of plants, and that in turn affects them development. India has made significant strides in pesticide, fungicide, and herbicide advancement and research. But each year, owing to unknown factors, plants fall to a variety of recognized illnesses, resulting in the loss of countless tons of yield. The assault of such different forms of crop diseases causes a significant reduction of crop yield both qualitatively and quantitatively.

Traditional methods for detecting diseases require manual inspection of plants by experts. This process needs to be continuous, and can be very expensive in large farms, or even completely unavailable to many small farm holders living in rural areas. This is why many attempts to automate disease detection have been made in the last few decades.

Current technological advancements have made the detection and diagnosis of plant diseases conceivable and achievable, hence paving the road for improved plant management in the event that a plant is infected. The suggested approach for the identification of plant leaf diseases concentrates on 14 plant species and 38 varieties or categories. Image Classification, Voice Recognition, and Processing of Natural Language has all shown exceptional performance over recent years due to Deep

Learning. Utilizing a CNN to address the issue of identifying plant diseases yields excellent results.

CNN is acknowledged as the most effective Object Recognition technique. It is utilized for the creation of a predictive model that is operated on the input picture and changes the input in order to identify the output labels. In addition to classifying the plants, the model could also provide valuable information such as prevention measures and supplements. This could include recommendations for specific pesticides or fungicides to apply, cultural practices to reduce the risk of disease, or nutritional supplements to improve plant health.

1.2 Problem Statement

The agricultural sector is vital for global food security, yet it faces challenges like plant diseases, leading to crop losses and economic strain. Prompt disease identification is crucial, but manual inspection is slow and error-prone. Modern technologies, especially CNNs, offer automated solutions for accurate disease detection, leveraging visual symptoms captured in images.

1) CNNs, known for their prowess in image classification, present a promising avenue for automating plant disease detection.
2) By swiftly identifying visual symptoms, these deep learning techniques can expedite diagnosis, aiding in effective disease management.
3) Automation through CNNs can alleviate the labor-intensive and error-prone nature of manual inspection, enhancing crop protection and agricultural productivity.

4) The suggested solution to crop disease diagnosis is substantially less costly and needs shorter effort for predicting than existing deep learning-based systems.

1.3 Project overview/Specifications

The project aims to develop a system for detecting diseases in plants using computer vision techniques. This system will analyze images of plant leaves to identify symptoms of diseases accurately and efficiently. The ultimate goal is to provide early detection of diseases, enabling timely intervention and minimizing crop losses.

1.3.1 Objective

The primary objective of our project is to assist farmers in identifying diseases affecting their plant leaves and to offer optimal solutions for disease management. By leveraging modern technologies like deep learning, we strive to enhance farming productivity and cultivate a renewed interest in agricultural practices. Through automated disease detection and tailored recommendations, we aim to streamline the farming process, reducing the time and financial burden on farmers. Ultimately, our endeavor seeks to empower farmers, mitigate crop losses, and foster sustainable agricultural practices for a thriving farming community.

1.3.2 Goal

To propose a model that accurately identify diseases present in plant leaves, providing the best possible solutions for effective management. By leveraging advanced algorithms, it aims to reduce the burden on farmers while boosting productivity. By automating disease diagnosis and offering tailored solutions, the software seeks to streamline farming operations and enhance agricultural outcomes. Ultimately, it aims to empower farmers with efficient tools for disease management, contributing to overall agricultural sustainability and success. In addition to classifying the plants, the proposed model could also provide valuable information such as prevention measures and supplements. This could include recommendations for specific pesticides or fungicides to apply, cultural practices to reduce the risk of disease, or nutritional supplements to improve plant health.

1.4 Deep Learning

Deep learning, a subset of machine learning, emulates the functioning of the human brain through artificial neural networks composed of interconnected nodes called neurons. These techniques specialize in constructing complex models with multiple hidden layers, enabling the extraction of intricate patterns and features from data. Among the myriad architectures, Convolutional Neural Networks (CNNs) excel in image recognition tasks by efficiently capturing spatial hierarchies, while Recurrent Neural Networks (RNNs) excel in sequential data analysis, making them suitable for tasks like speech recognition and language modeling.

1.4.1 Key Components and Hyperparameters

When constructing models using deep learning, several hyperparameters need to be carefully tuned to ensure optimal performance. Some of the key hyperparameters include:

Learning rate: This hyperparameter controls the step size taken by the optimizer during each iteration of training. Too small a learning rate can result in slow convergence, while too large a learning rate can lead to instability and divergence.

Batch Size: This hyperparameter defines the number of samples we use in one epoch to train a neural network. Choosing an appropriate batch size is crucial in training deep learning models. Larger batch sizes require more memory, both on the GPU/TPU and the CPU. If you have limited memory, you might need to decrease the batch size. Smaller batch sizes can sometimes lead to better generalization, meaning the model performs better on unseen data. This is because smaller batches introduce more randomness into the optimization process.

Epochs: This hyperparameter represents the number of times the entire training dataset is passed through the model during training. Increasing the number of epochs can improve the model's performance but may lead to overfitting if not done carefully.

Number of layers: This hyperparameter determines the depth of the model, which can have a significant impact on its complexity and learning ability.

Number of nodes per layer: This hyperparameter determines the width of the model, influencing its capacity to represent complex relationships in the data.

Architecture: This hyperparameter determines the overall structure of the neural network, including the number of layers, the number of neurons per layer, and the connections between layers. The optimal architecture depends on the complexity of the task and the size of the dataset.

Activation function: This hyperparameter introduces non-linearity into the model, allowing it to learn complex decision boundaries. Common activation functions include sigmoid, tanh, and Rectified Linear Unit (ReLU).

Dropout: Dropout is a regularization technique commonly used in neural networks to prevent overfitting and improve the performance of the model.

Loss Function: This hyperparameter compute error between actual and prediction values and measure models performance. Hyperparameters are fine tuned to minimise the loss function.

Optimizer: a model's optimizer is the algorithm that updates the weights of the model during training, using output from the loss function along with other model parameters.

1.4.2 Popular Activation Functions

Sigmoid Activation Function: The sigmoid activation function, also known as the logistic function, is a commonly used non-linear activation function in neural networks. It maps the input values to a range between 0 and 1, making it suitable for binary classification tasks where the output represents probabilities.

The Equation (1.1) represents the Sigmoid activation function formula.

$$f(x) = \frac{1}{1 + e^{-x}} \quad \textbf{1-1}$$

where:

x is the input to the neuron.

e is the base of the natural logarithm (Euler's number).

ReLU Activation Function: The rectified linear unit (ReLU) or rectifier activation function increases the complexity of the neural network by introducing non-linearity, which allows the network to learn more complex representations of the data. The ReLU function sets all negative values to zero.

The Equation (1.2) represents the ReLU activation function formula.

$$f(x) = max(0, x) \tag{1-2}$$

where:

x is the input to the neuron.

Softmax Activation Function: The softmax activation function is commonly used in the output layer of neural networks, particularly in multi-class classification tasks. It converts the raw output scores of a neural network into probabilities, ensuring that the sum of the probabilities across all classes adds up to 1.

The Equation (1.3) represents the Softmax activation function formula.

$$softmax(Z_i) = \frac{e^{Z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{1-3}$$

where:

Z_i is the raw output score (logit) for class i.

K is the total number of classes.

e is the base of the natural logarithm (Euler's number).

Tanh Activation Function: The hyperbolic tangent (tanh) activation function is a non-linear function commonly used in neural networks, particularly in the hidden layers. It shares similarities with the sigmoid activation function but produces output values in the range $[-1,1]$ $[-1,1]$, making it centered around zero.

The Equation (1.4) represents the tanh activation function formula.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1-4}$$

where:

x is the input to the neuron.

e is the base of the natural logarithm (Euler's number).

2 Literature Survey

In this section we provide relevant background on previous work on Plant Disease Detection. Recently, several methods have been proposed for predicting diseases. Many of these methods are based on Convolutional Neural Networks and its pre trained models.

In the paper [1] proposed by Robert G. de Luna, Elmer P. Dadios, Argel A. (2019) a neural network was developed for efficiently identifying diseases in tomato plants. This research created a new method for the effective identification of diseases in tomato plants. The study focused on a specific tomato variety called Diamante Max. The methodology aimed to diagnose Phroma Rot, Leaf Miner, and Target Spot diseases by collecting both damaged and healthy leaves for data collection. Convolutional Neural Networks (CNN) were employed to determine the prevalence of tomato illnesses on the observed plants. The F-RCNN-trained abnormality detection model achieved an 80% confidence score, while the Transfer Learning illness identification model demonstrated a precision of 95.75%.

In the paper [2] proposed by X.-P. Fan, J.-P. Zhou, and Y. Xu (2020) added a batch standardization layer to the convolutional layer of the Faster R-CNN model, introduced a central cost function to construct a mixed cost function, and used a stochastic gradient descent algorithm to optimize the training model. They used 9 kinds of corn leaf diseases with complex backgrounds in the field as the research object. Under the same experimental environment, the improved method had an average accuracy increase of 8.86%, and a single image detection time was reduced by 0.139s; compared with the SSD algorithm, the average accuracy was 4.25% higher, and a single image detection time was reduced by 0.018 s.

In the paper [3] proposed by Bin Liu, Peng Jiang, Yuehan Chen, Dongjian He, Chunquan Lian (2019) addresses the detection of five apple leaf diseases, namely brown spot, mosaic, aria leaf spot, rust, and grey spot. Deep learning techniques are employed, specifically enhanced Convolutional Neural Networks (CNNs), to improve the detection of these diseases. Image annotation and data augmentation techniques are applied to construct a comprehensive dataset. The model, termed INAR-SSD, is trained and tested on a dataset comprising 26,377 photos of apple leaf diseases. Experimentally, the INAR-SSD model achieves a detection accuracy of 78.80%.

In the paper [4] proposed by Rekha Chahar, Priyanka Soni (2016) mainly focuses on the detection of diseases in leaf images of various plants, vegetables, fruits, and flowers, crucial for agriculture. The primary objective is to ascertain the health status and identify infectious diseases based on area identification in leaf images. The research utilizes arbitrarily obtained leaf photos from the internet for various plants, emphasizing the importance of remote disease detection for agricultural management.

3 Proposed System

The Deep learning model was proposed to implement a plant disease detection system using Convolutional Neural Networks (CNNs). The model will analyze images of plant leaves to accurately identify diseases present in the plants. Once a disease is detected, the system will provide remedial measures such as prevention strategies and recommended supplements based on the detection results.

3.1 Dataset

A dataset [8] is a structured collection of data that is organized and stored for easy access, retrieval, and analysis. In the context of machine learning and data science, a dataset typically refers to a set of observations or examples used to train, validate, or test a model. It consists of multiple instances or samples, each containing one or more features or attributes.

In this project, images were taken from the new plant disease dataset from Kaggle. This dataset is recreated using offline augmentation from the original dataset which is plant village dataset. This dataset consists of about 87K RGB images of healthy and diseased crop leaves which is categorized into 38 different classes. A new directory containing 33 test images is created later for prediction purpose. The plants that are considered in this dataset are Orange, Grape, Raspberry, Tomato, Peach, Apple, Cherry, Soybean, Pepper bell, Corn, Potato, Blueberry, Squash and Strawberry.

This dataset consists of following directories:

1) Train (70295 images)

2) Test (33 images)

3) Valid (17572 images)

The following Figure 3.1 shows the images present in each class label of the dataset.

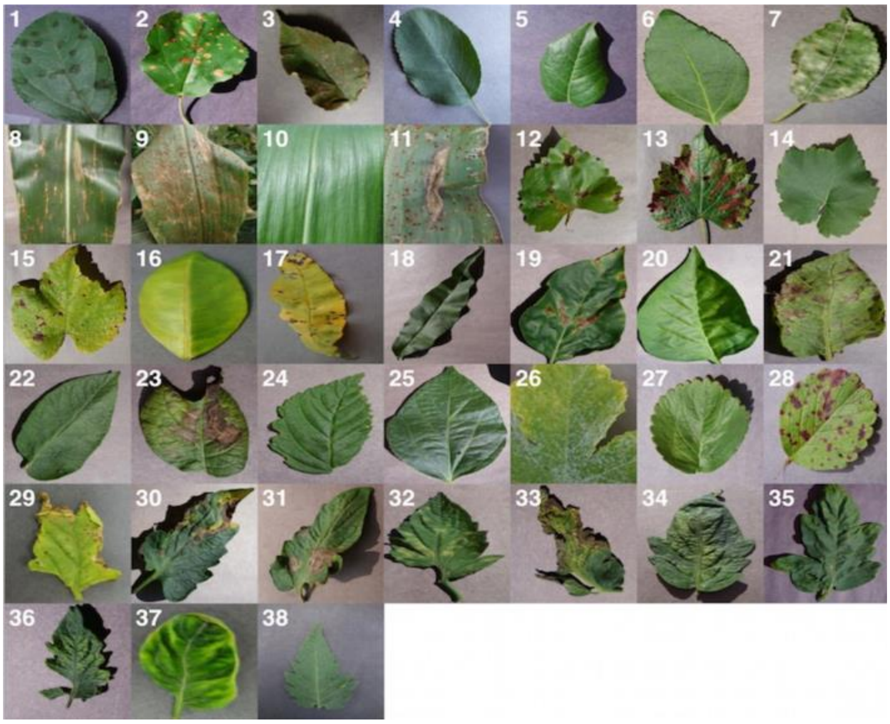

Figure 3.1 Samples from the dataset

The Table 3.1 describes the number of images present in train and validation directories for every class label present in the dataset. This table also defines total number of images used for training and validation purpose.

Table 3.1 Number of images in each class of the dataset

Class Name	Train	Validation
Apple___Apple_scab	2017	504
Apple___Black_rot	1987	497
Apple___Cedar_apple_rust	1760	440
Apple___healthy	2008	502
Blueberry___healthy	1816	454
Cherry_(including_sour)___healthy	1826	456
Cherry_(including_sour)___Powdery_mildew	1683	421

Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	1642	410
Corn_(maize)___Common_rust_	1907	477
Corn_(maize)___healthy	1859	465
Corn_(maize)___Northern_Leaf_Blight	1908	477
Grape___Black_rot	1888	472
Grape___Esca_(Black_Measles)	1920	480
Grape___healthy	1692	423
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1722	430
Orange___Haunglongbing_(Citrus_greening)	2010	503
Peach___Bacterial_spot	1838	459
Peach___healthy	1728	432
Pepper,_bell___Bacterial_spot	1913	478
Pepper,_bell___healthy	1988	497
Potato___Early_blight	1939	485
Potato___healthy	1824	456
Potato___Late_blight	1939	485
Raspberry___healthy	1781	445
Soybean___healthy	2022	505
Squash___Powdery_mildew	1736	434
Strawberry___healthy	1824	456
Strawberry___Leaf_scorch	1774	444
Tomato___Bacterial_spot	1702	425
Tomato___Early_blight	1920	480
Tomato___healthy	1926	481
Tomato___Late_blight	1851	463
Tomato___Leaf_Mold	1882	470
Tomato___Septoria_leaf_spot	1745	436
Tomato___Spider_mites Two-spotted_spider_mite	1741	435
Tomato___Target_Spot	1827	457
Tomato___Tomato_mosaic_virus	1790	448
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1961	490
Total	**70296**	**17572**

3.2 Preprocessing

To preprocess and create a dataset for training a model on image data. Firstly, the function resizes all images to a uniform size of 256x256 pixels for ensuring consistency across the dataset. Labels are inferred from the directory structure, where each subdirectory within the "train" and “valid” directories represents a distinct class, and images within those subdirectories are labeled accordingly. These labels are encoded in a categorical format, facilitating multi-class classification tasks. Additionally, the dataset is batched into groups of 32 images per batch, aiding in efficient model training. The data is shuffled randomly after each epoch (shuffle=True) to prevent the model from learning spurious patterns from the order of the data. Images are loaded in the RGB color space (color_mode="rgb"), and bilinear interpolation is employed for resizing (interpolation="bilinear") to ensure smooth image transformations. While basic preprocessing steps are applied, such as resizing, batching, label encoding, and shuffling, we do not explicitly include data augmentation techniques or validation data splitting because the dataset is already augmented.

3.3 Model

Deep learning carries out the machine learning process using an artificial neural network that is composed of several levels arranged in a hierarchy. The model is based on deep networks where the flow of information starts from the initial level, where the model learns something simple and then the output of which is passed to layer two of the network and input is combined into something that is a bit more complex and passes it on to the third level. This process continues as each level in the network produces something more complex from the input it received from the ascendant level.

3.3.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are a specialized type of Artificial Neural Network (ANN) designed primarily for processing matrix-like data. They are particularly effective in tasks related to computer vision, such as image recognition and classification. They are especially effective in tasks where the spatial relationships between data points matter, like recognizing patterns in images, text and time-series data as well. The complexity in terms of no. of parameters to be learned and the training time exponentially grows in case of ANNs.

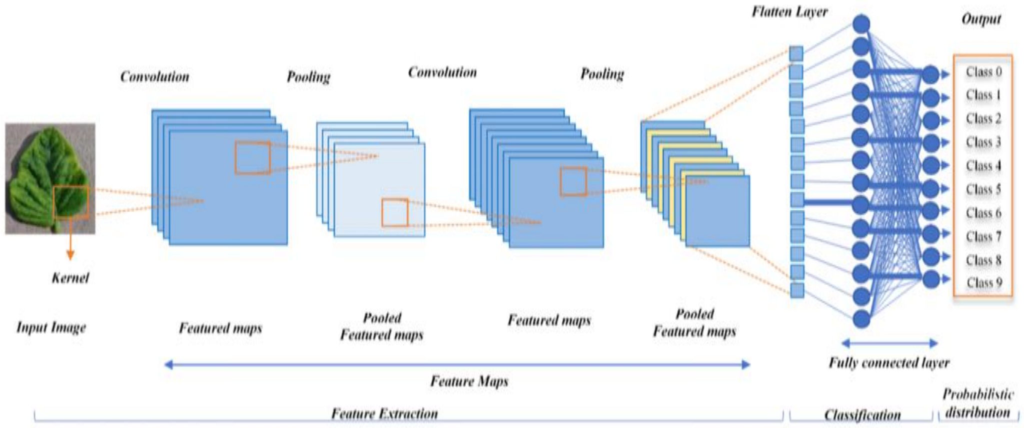

Figure 3.2 CNN Architecture

CNNs consist of multiple layers-Convolution layers, Pooling layers and Fully connected layers.

Convolutional Layers: These layers apply a set of filters (small kernels/matrices) to the input data. Each filter learns to detect specific features (like edges or textures) within the data.

The input consists of 3 channels/matrices of values corresponding to the RGB channels. The filter for convolution consists of 3 kernels, one each for each channel with all the kernels sharing the same bias. The convolution of each kernel with the corresponding channel is performed and the resulting three values along with bias are summed up as shown in the Figure.

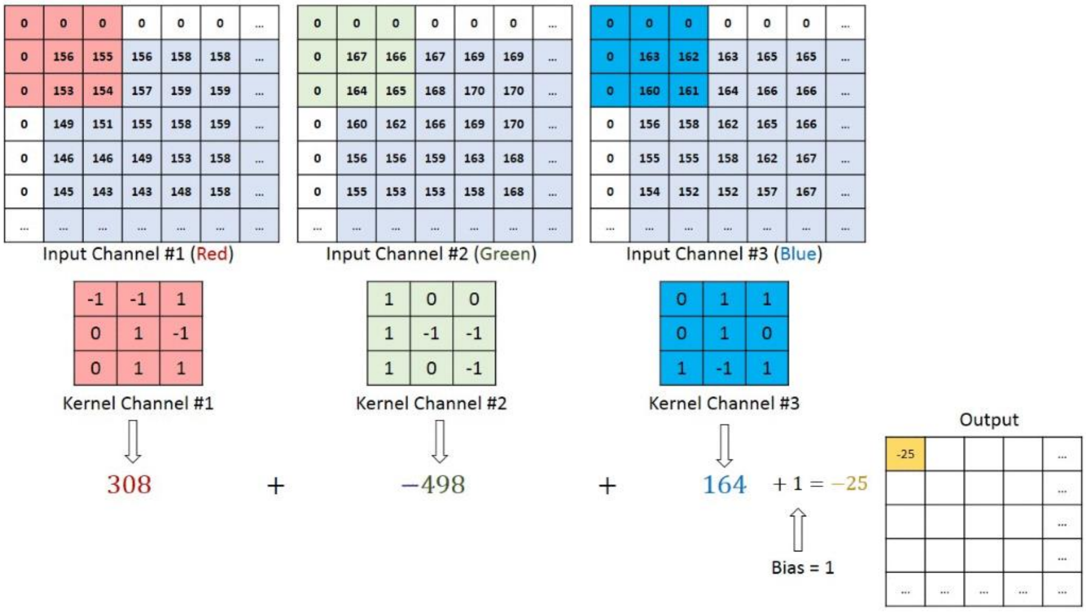

Figure 3.3 Convolution Operation

Pooling Layers: Pooling layers are often used in CNNs to reduce the spatial dimensions of the feature maps generated by the convolutional layers. Common pooling operations include max pooling and average pooling, which help in reducing computational complexity and controlling overfitting. The following figure shows the examples of Max. pooling and average pooling. Pooling requires the window size to be selected.

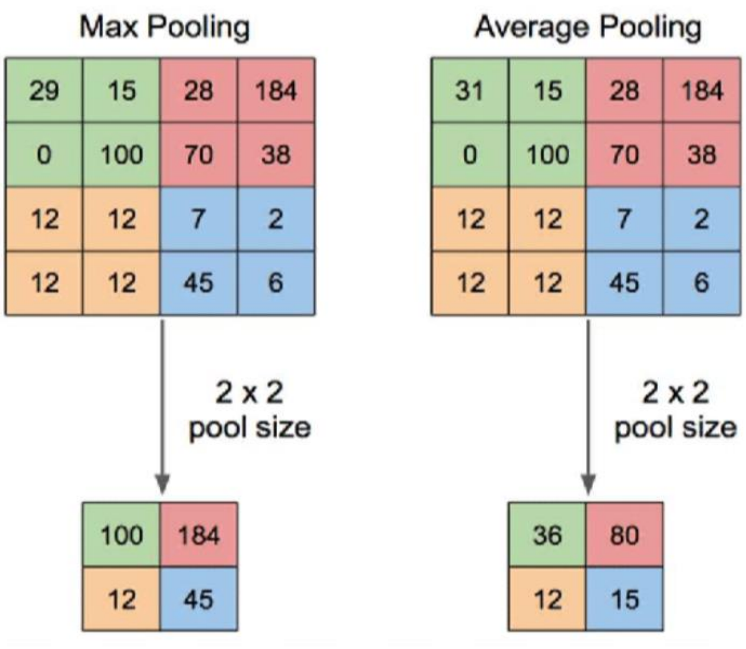

Figure 3.4 Pooling Operation

Fully Connected Layers: These are traditional densely connected neural network layers that process the features learned by previous layers for tasks like classification. The flattened one-dimensional output of the pooling layers is fed to the first fully connected layer. These layers are often followed by a softmax activation function for classification tasks.

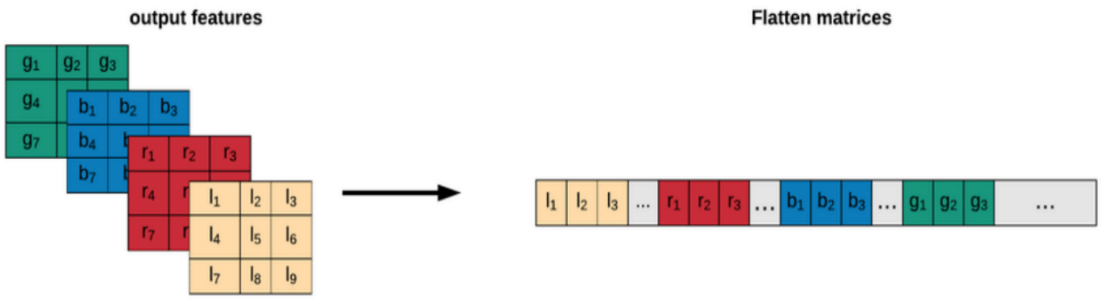

Figure 3.5 Flatten Layer

3.4 Proposed Architecture

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 32)      896

 conv2d_1 (Conv2D)           (None, 254, 254, 32)      9248

 max_pooling2d (MaxPooling2  (None, 127, 127, 32)      0
 D)

 conv2d_2 (Conv2D)           (None, 127, 127, 64)      18496

 conv2d_3 (Conv2D)           (None, 125, 125, 64)      36928

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 64)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 62, 62, 128)       73856

 conv2d_5 (Conv2D)           (None, 60, 60, 128)       147584

 max_pooling2d_2 (MaxPoolin  (None, 30, 30, 128)       0
 g2D)

 conv2d_6 (Conv2D)           (None, 30, 30, 256)       295168

 conv2d_7 (Conv2D)           (None, 28, 28, 256)       590080

 max_pooling2d_3 (MaxPoolin  (None, 14, 14, 256)       0
 g2D)

 conv2d_8 (Conv2D)           (None, 14, 14, 512)       1180160

 conv2d_9 (Conv2D)           (None, 12, 12, 512)       2359808

 max_pooling2d_4 (MaxPoolin  (None, 6, 6, 512)         0
 g2D)

 dropout (Dropout)           (None, 6, 6, 512)         0

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 1500)              27649500

 dropout_1 (Dropout)         (None, 1500)              0

 dense_1 (Dense)             (None, 38)                57038

=================================================================
Total params: 32418762 (123.67 MB)
Trainable params: 32418762 (123.67 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3.6 Model Summary

The model begins with a series of Conv2D layers, each followed by a Rectified Linear Unit (ReLU) activation function, serving to extract hierarchical features from the input data. Subsequently, MaxPooling2D layers reduce the spatial dimensions of the feature maps, aiding computational efficiency and mitigating overfitting. Dropout layers are strategically incorporated to randomly deactivate a portion of neurons during training, promoting generalization and preventing overreliance on specific features. Following these convolutional and pooling layers, a Flatten layer reshapes the output into a one-dimensional array, facilitating the transition to the fully connected layers. The Dense layers at the end of the network connect every neuron in one layer to every neuron in the subsequent layer, culminating in an output layer with 38 neurons, likely corresponding to the number of classes in the classification task. The summary also provides insights into the model's complexity, revealing a total of 32,418,762 parameters, all of which are trainable, indicative of a sophisticated architecture capable of capturing intricate patterns in the data.

3.5 Model Working

In the Working Model we are going to discuss the basic Design of the project and detailed information about each step in Design.

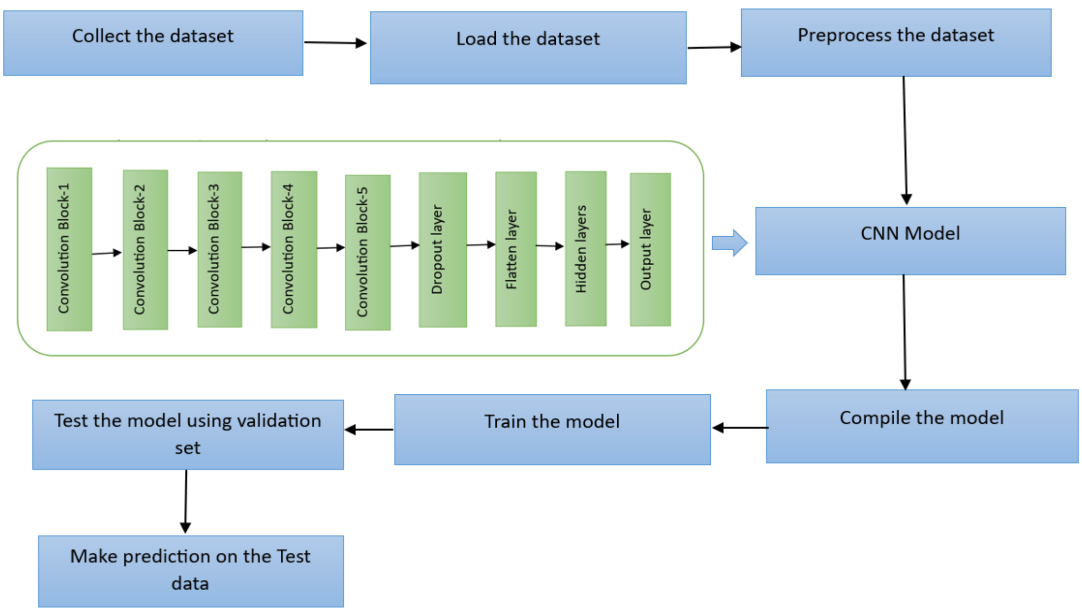

Figure 3.7 Working flow of the Model

1. **Collect the Dataset:** This step involves gathering image data that will be used to train the CNN model. The dataset should be relevant to the classification task you intend to perform. For instance, if you want to train a model to classify between apple and orange leafs, you'll need a collection of images containing apple and orange leafs.
2. **Load the Dataset:** The collected dataset is loaded into the system for further processing.
3. **Preprocess the Dataset:** Preprocessing is crucial to ensure the data is consistent and suitable for training the model. Common preprocessing techniques for image data include resizing, batching, shuffling etc.
4. **Define CNN Model Architecture:** Here, we designed the CNN architecture, specifying the layers and connections that will process the image data and extract features. The architecture determines the model's capacity to learn complex patterns from the images. Common CNN layers include convolutional layers, pooling layers, activation functions, and fully connected layers.
5. **Compile the Model:** This step involves configuring the training process by specifying the optimizer (algorithm used to adjust model weights) and the loss function (function used to measure the model's prediction error). Common choices include optimizers like Adam or SGD (Stochastic Gradient Descent) and loss functions like categorical cross-entropy for multi-class classification problems.
6. **Train the Model:** The preprocessed data and compiled model are used for training. Training involves iterating through the dataset multiple times (epochs). In each iteration (batch), the model predicts outputs for a subset of

images, calculates the loss based on the difference between predictions and true labels, and adjusts its internal weights using the optimizer to minimize the loss. This process helps the model learn to map the input images to their corresponding categories.

7. **Test the Model using Validation Set:** A validation set, separate from the training data, is used to monitor the model's performance during training. The model's accuracy (percentage of correct predictions) or other metrics like precision, recall, and F1-score are evaluated on the validation set.
8. **Evaluate Model (Test Set):** Once training is complete, the final model's performance is assessed using a separate test set. The test set is another unseen data partition used for a final evaluation of the model's generalizability on real-world data.

   If the model's performance on the test set is satisfactory, you can save the trained model for future use. This allows you to use the model for image classification tasks on new, unseen images without retraining the entire model.

9. **Deploy the Model (Optional):** This step involves deploying the trained model to a production environment where it can be used for real-world image classification tasks. Deployment can involve integrating the model into web applications, mobile apps, or standalone software systems.

3.6 Advantages of Proposed System

1. **Automation:** The proposed system automates the process of disease detection, reducing the need for manual labor and expertise.

2. **Accuracy:** CNNs have shown superior performance in image classification tasks, leading to more accurate disease diagnosis compared to traditional methods.
3. **Scalability:** The system can be easily scaled to handle large volumes of data and accommodate additional plant species and diseases.
4. **Real-time detection:** With the deployment of the system on mobile devices, farmers can receive instant feedback on the health status of their plants, enabling timely intervention.
5. **Adaptability:** CNN models can adapt to different environmental conditions and variations in plant appearance, making them suitable for diverse agricultural settings.

4 System Design

System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.

4.1 Use Case Diagram

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. Actors are the external entities that interact with the system. The use cases are represented by either circles or ellipses. The Figure 4.1 shows the use case representation of the system.

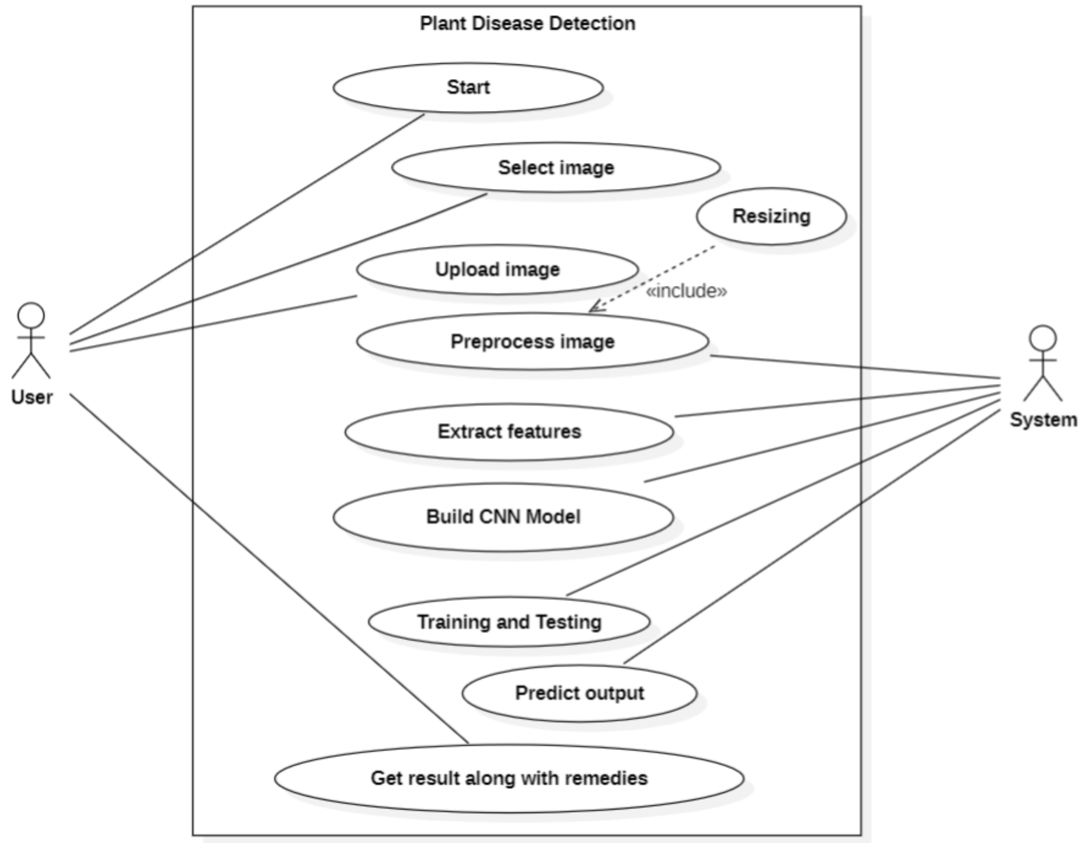

Figure 4.1 Use Case Diagram

4.2 Class Diagram

Class diagrams give an overview of a system by showing its classes and the relationships among them. Class diagrams are static – they display what interacts but not what happens when they do interact. In general a class diagram consists of some set of attributes and operations. Operations will be performed on the data values of attributes. The Figure 4.2 shows the class diagram representation of the system.

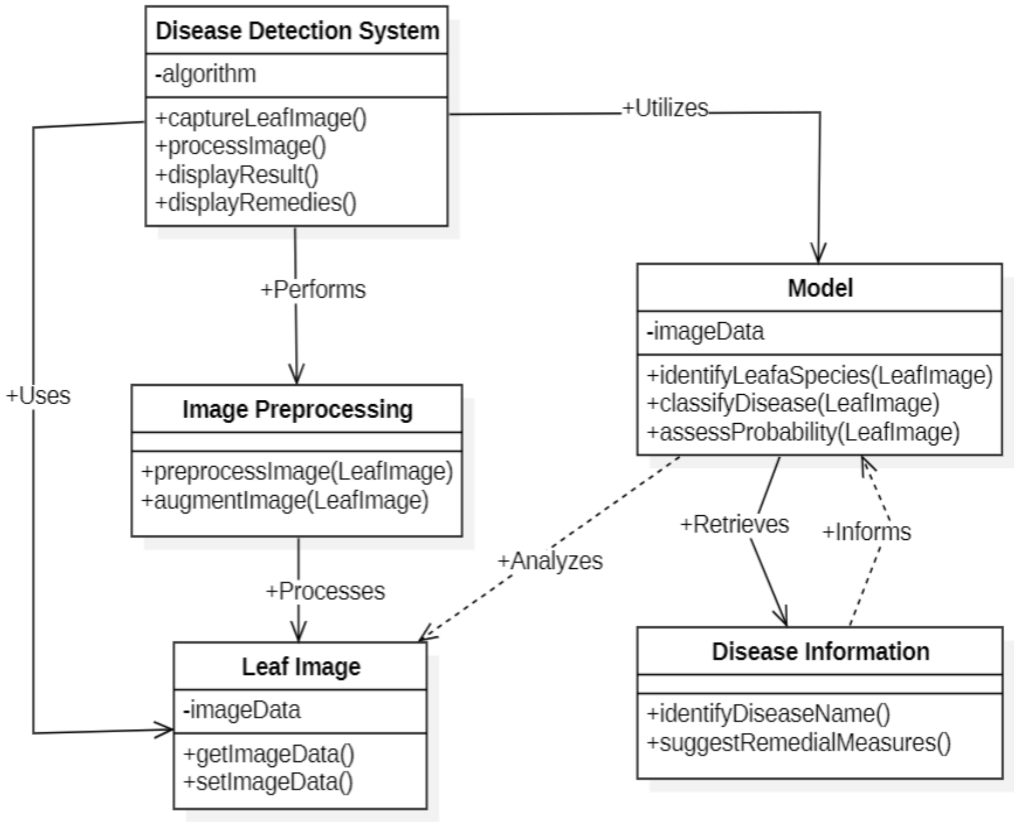

Figure 4.2 Class Diagram

4.3 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. In UML, an activity diagram provides a view of the behavior of a system by describing the sequence of actions in a process. The Figure 4.3 shows the activity diagram representation of the system.

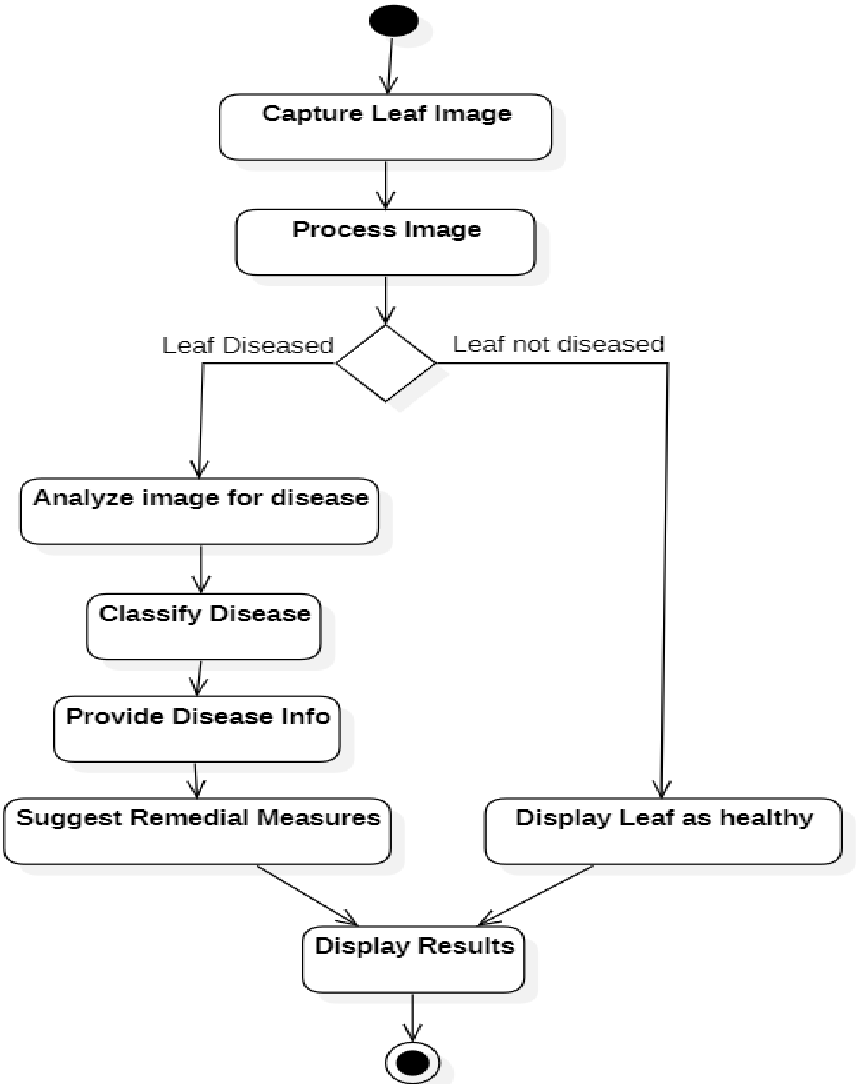

Figure 4.3 Activity Diagram

4.4 Sate Chart Diagram

A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). The Figure 4.4 shows the state chart diagram representation of the system.

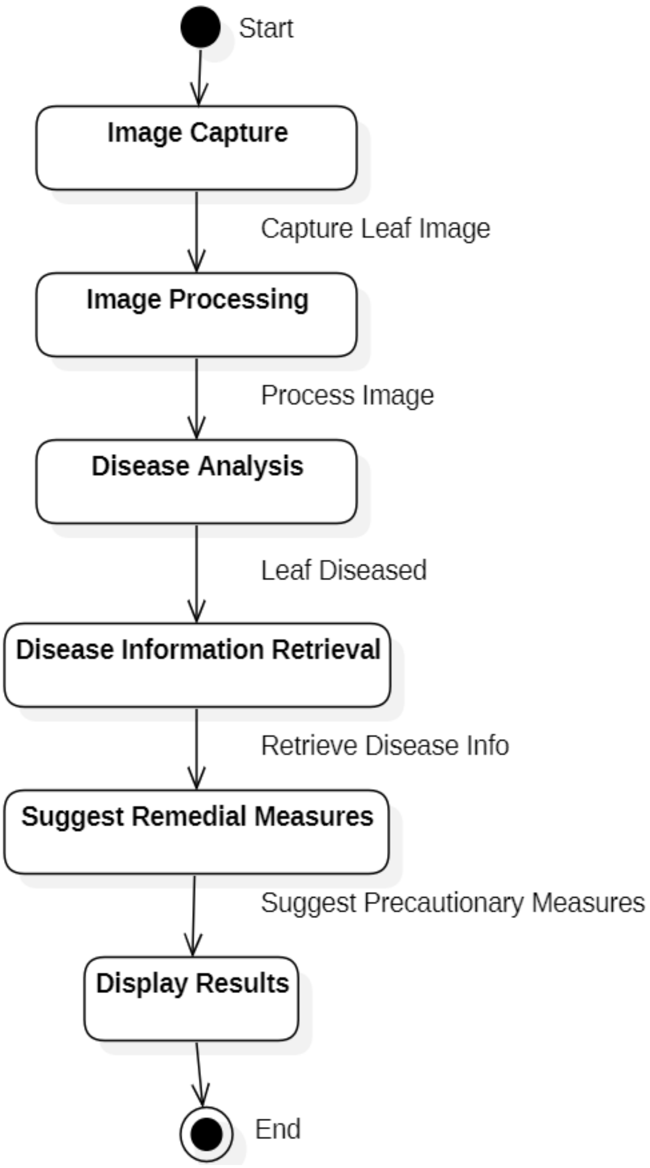

Figure 4.4 State Chart Diagram

4.5 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. The Figure 4.5 shows the sequence diagram representation of the system.

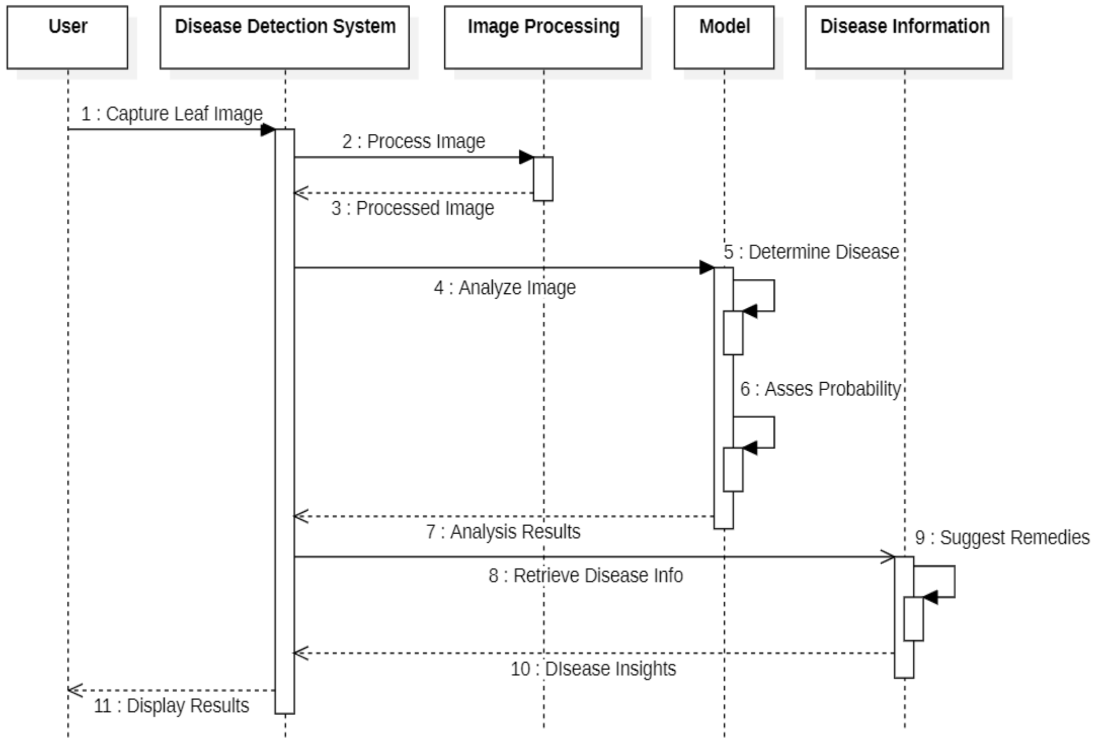

Figure 4.5 Sequence Diagram

5 Evaluation and Testing

5.1 Evaluation

To evaluate the performance of the model, you would typically use a separate dataset (e.g., a validation set or a test set) that the model hasn't seen during training. This ensures an unbiased assessment of its generalization capabilities. The most common metrics for evaluating classification models include accuracy, precision, recall, F1-score, and confusion matrix.

5.1.1 Metrics for Evaluation

Confusion Matrix: Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values. The confusion matrix provides a detailed breakdown of the model's predictions for each class, showing the number of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (TN).

- TP: Instances correctly classified as positive by the model.
- FP: Instances incorrectly classified as positive by the model.
- TN: Instances correctly classified as negative by the model.
- FN: Instances incorrectly classified as negative by the model.

Accuracy: Accuracy measures the proportion of correctly classified samples out of the total number of samples. While it's a straightforward metric, it might not be sufficient if the classes are imbalanced. The following Equation (6.1) represents the formula for calculation accuracy.

$$Accuracy = \frac{number\ of\ correct\ classifications}{total\ number\ of\ classifcations\ attempted} \quad \mathbf{5\text{-}1}$$

Precision: Precision calculates the ratio of true positive predictions to the total number of positive predictions (both true positives and false positives). It indicates how many of the predicted positive instances are actually positive.

$$Precision = \frac{True\ Postive}{True\ Postive + False\ Positive} \quad \mathbf{5\text{-}2}$$

Recall: Recall computes the ratio of true positive predictions to the total number of actual positive samples (true positives and false negatives). It indicates the model's ability to correctly identify positive instances.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad \mathbf{5\text{-}3}$$

F1-Score: F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall, making it useful when there is an imbalance between classes.

$$F1 - Score = 2.\frac{Precision *\ Recall}{Precision + Recall} \quad \mathbf{5\text{-}4}$$

The Table 5.1 represents the precision, recall and f1 – score for every class label present in the dataset.

Table 5.1 Performance Evaluation of crop diseases

Class Name	Precision	Recall	F1-Score
Apple___Apple_scab	0.99	0.97	0.98
Apple___Black_rot	0.97	0.99	0.98
Apple___Cedar_apple_rust	0.98	1.00	0.99
Apple___healthy	0.97	0.97	0.97

Blueberry___healthy	0.99	0.98	0.98
Cherry_(including_sour)___Powdery_mildew	0.98	1.00	0.99
Cherry_(including_sour)___healthy	0.97	0.99	0.98
Corn_(maize)___Cercospora_leaf_spot Gray_leaf	0.95	0.85	0.90
Corn_(maize)___Common_rust_	1.00	0.99	0.99
Corn_(maize)___healthy	1.00	0.98	0.99
Corn_(maize)___Northern_Leaf_Blight	0.87	0.97	0.92
Grape___Black_rot	0.97	1.00	0.98
Grape___Esca_(Black_Measles)	1.00	0.97	0.99
Grape___healthy	0.99	1.00	1.00
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	0.99	1.00	0.99
Orange___Haunglongbing_(Citrus_greening)	0.98	1.00	0.99
Peach___Bacterial_spot	0.97	0.98	0.97
Peach___healthy	0.99	1.00	0.99
Pepper,_bell___Bacterial_spot	0.99	0.97	0.98
Pepper,_bell___healthy	1.00	0.94	0.97
Potato___Early_blight	0.94	1.00	0.97
Potato___healthy	0.99	0.98	0.99
Potato___Late_blight	0.92	0.99	0.95
Raspberry___healthy	0.98	1.00	0.99
Soybean___healthy	0.96	0.99	0.98
Squash___Powdery_mildew	0.98	0.99	0.99
Strawberry___healthy	1.00	1.00	1.00
Strawberry___Leaf_scorch	0.99	1.00	0.99
Tomato___Bacterial_spot	0.97	0.97	0.97
Tomato___Early_blight	0.95	0.90	0.92
Tomato___healthy	0.99	1.00	0.99
Tomato___Late_blight	0.96	0.83	0.89
Tomato___Leaf_Mold	0.99	0.99	0.99
Tomato___Septoria_leaf_spot	0.96	0.89	0.93
Tomato___Spider_mites Two-spotted_spider_mite	0.98	0.96	0.97
Tomato___Target_Spot	0.93	0.97	0.95
Tomato___Tomato_mosaic_virus	0.98	1.00	0.99
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.99	0.99	0.99

5.2 Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner.

5.2.1 Levels of testing

The different levels of testing that are to be conducted are:

- Code Testing
- Program Testing
- System Testing

Code Testing: The code test has been conducted to test the logic of the program. Here, we have tested with all possible combinations of data to find out logical errors. The code testing is done thoroughly with all possible data available with library.

Program Testing: Program testing is also called unit testing. The modules in the system are integrated to perform the specific function. The modules have been tested independently, later Assembled and tested thoroughly for integration between different modules.

System Testing: System testing has been conducted to test the integration of each module in the system. It is used to find discrepancies between the system and its original objective. It is found that there is an agreement between current specifications and system documentation. Software Testing is carried out in three steps.

5.2.2 Unit Testing

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example, the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Table 5.2 Unit Testing Table

Function Name	Tests Results
Uploading Image	Tested for uploading different types and sizes of images.
Detecting Disease	Tested for different images of plant leaves and diseases Bacterial Spot, yellow leaf curl virus.
Get Remedies	Tested if the remedies are displayed successfully.

5.3 Code URL

https://github.com/VengalaRakesh/Plant_Disease_Detection

6 Results

The proposed CNN model was trained and tested using the New Plant Disease dataset. The dataset was split into training, validation and testing sets with 67K RGB images, respectively, and were labelled with 39 different classes of diseased and healthy plant leaves and background images. These images were taken from the “new plant disease dataset” that was just recently published on Kaggle and are related to plant diseases.

When recreating this dataset using offline augmentation, the original dataset, which was known as the plant village dataset, served as the point of departure. Over eighty-seven thousand RGB photographs of plant leaves, depicting both healthy and diseased conditions, are included in this dataset. These photographs have been organized into a total of 38 different classifications for your viewing pleasure. There are a total of 14 different kinds of plants that are taken into consideration within the parameters of this dataset.

The following plants are taken into consideration in this dataset: orange, grape, raspberry, tomato, peach, apple, cherry (including sour), soybean, pepper bell, corn (maize), potato, blueberry, squash, and strawberry. This particular dataset takes into account a total of 26 distinct diseases that can be found in plants.

This research demonstrates conclusively that CNNs may be used to strengthen small-scale farmers with their battle against plant disease. The intensity of crop diseases varies over time; hence, CNN models must be enhanced/modified to identify and categorize diseases throughout their whole occurrence cycle. The CNN model/architecture must be effective across a variety of light circumstances; hence the datasets must not only represent the actual environment, but also include photographs captured in various field scenarios. A detailed investigation is necessary to comprehend the aspects that influence the identification of plant illnesses, such as the categories and quantity of datasets, the learning rate, as well as the amount of light.

6.1 Screens and Reports

One of our major aims while building application was to make high-quality disease detection accessible to most crop growers. The developed system integrates advanced plant disease detection system and tracking algorithms with a user-friendly interface to analyze disease. The complete application is built using Streamlit framework.

- Figure 6.1 describes the home page of our application

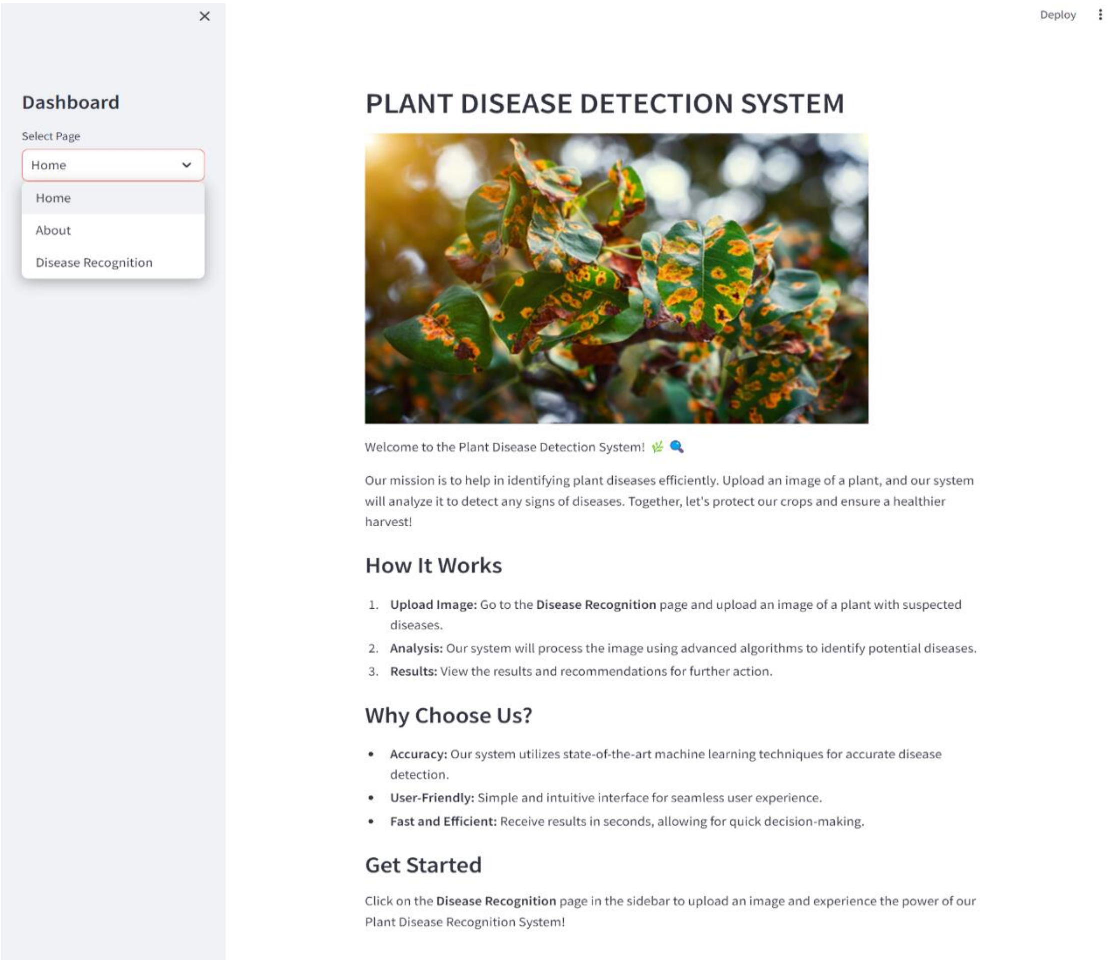

Figure 6.1 Home Page

- Figure 6.2 describes the about page of our application and this page mainly consists of the dataset that is used to train and build the system.

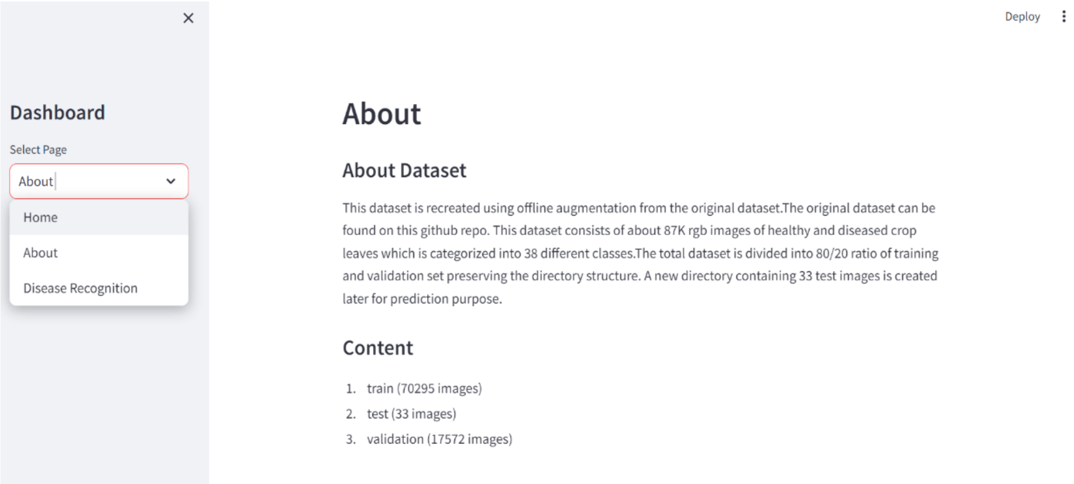

Figure 6.2 About Page

- Figure 6.3 describes the Disease Recognition page. We have a text message called "Choose an Image" so that a user can understand to choose the image. It allows the user to browse the image from the system.

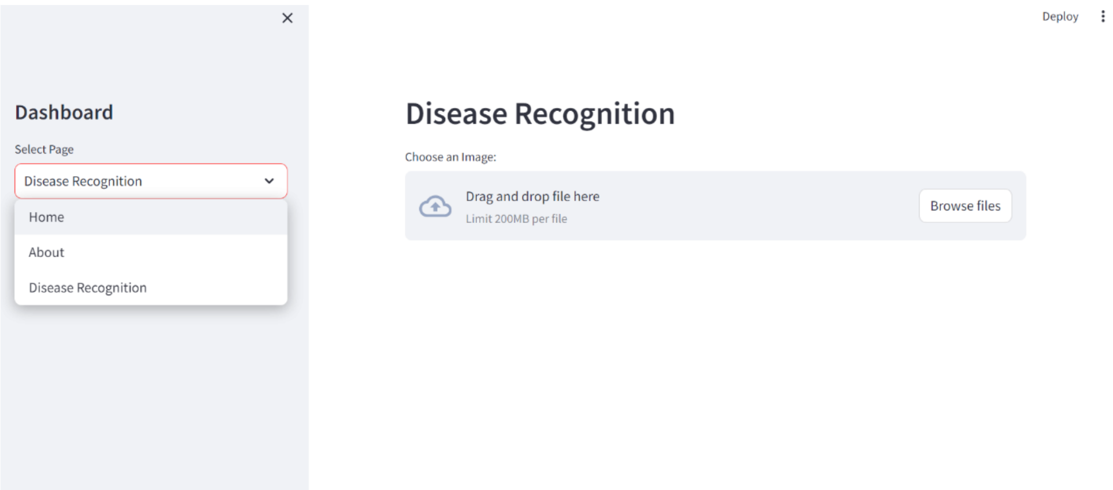

Figure 6.3 Recognition Page

- Figure 6.4 shows the popup which appears when user clicks on browse files.

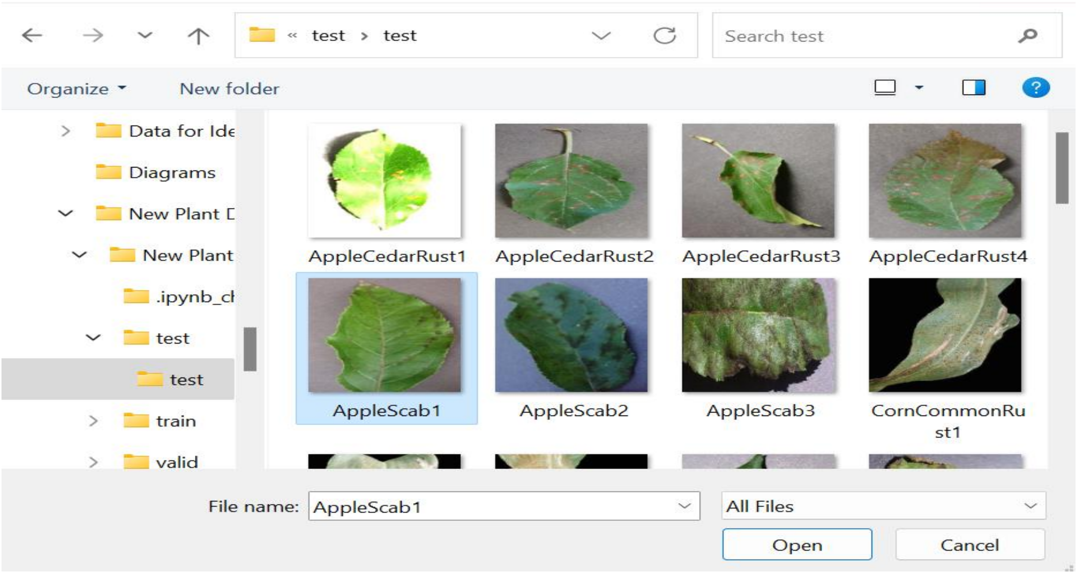

Figure 6.4 Selecting Input from the Dataset

- Figure 6.5 shows the selected image from the system directory, there will be a button provided called as 'Predict' for analysing the input image.

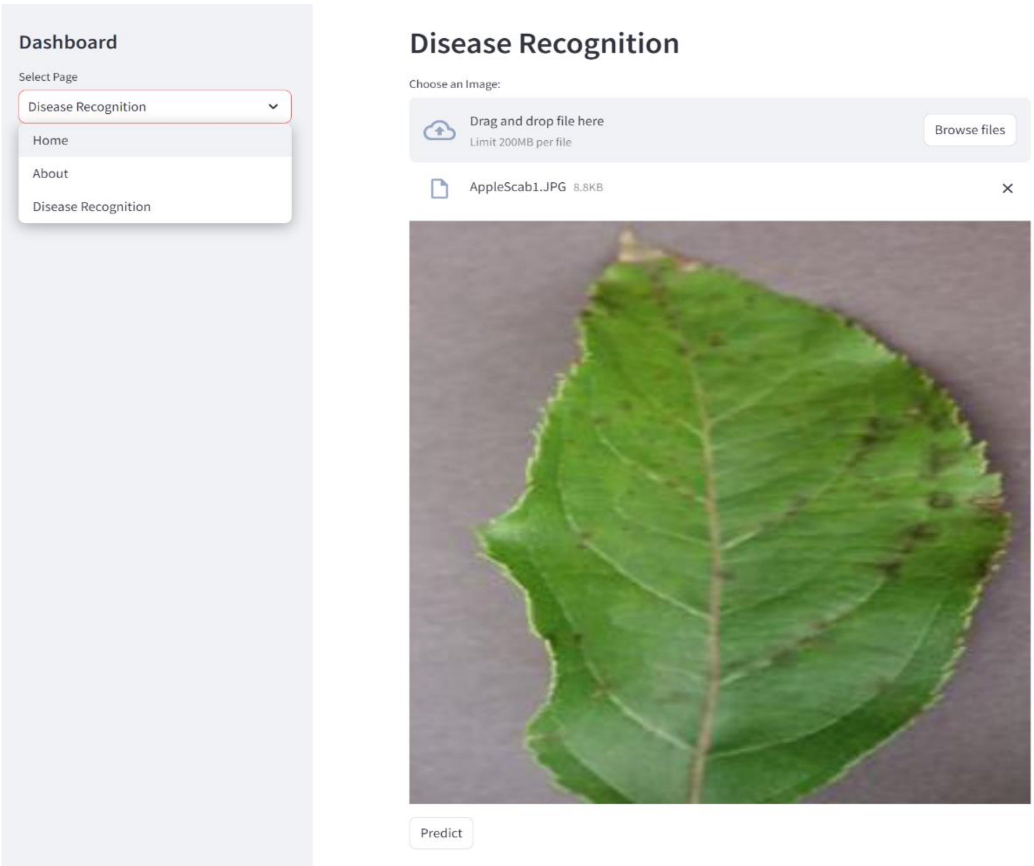

Figure 6.5 Selected Image

- Figure 6.6 displays the leaf's condition, indicating whether it is healthy or diseased, and if afflicted, it specifies the name of the disease. It also displays prevention measures and supplement names.

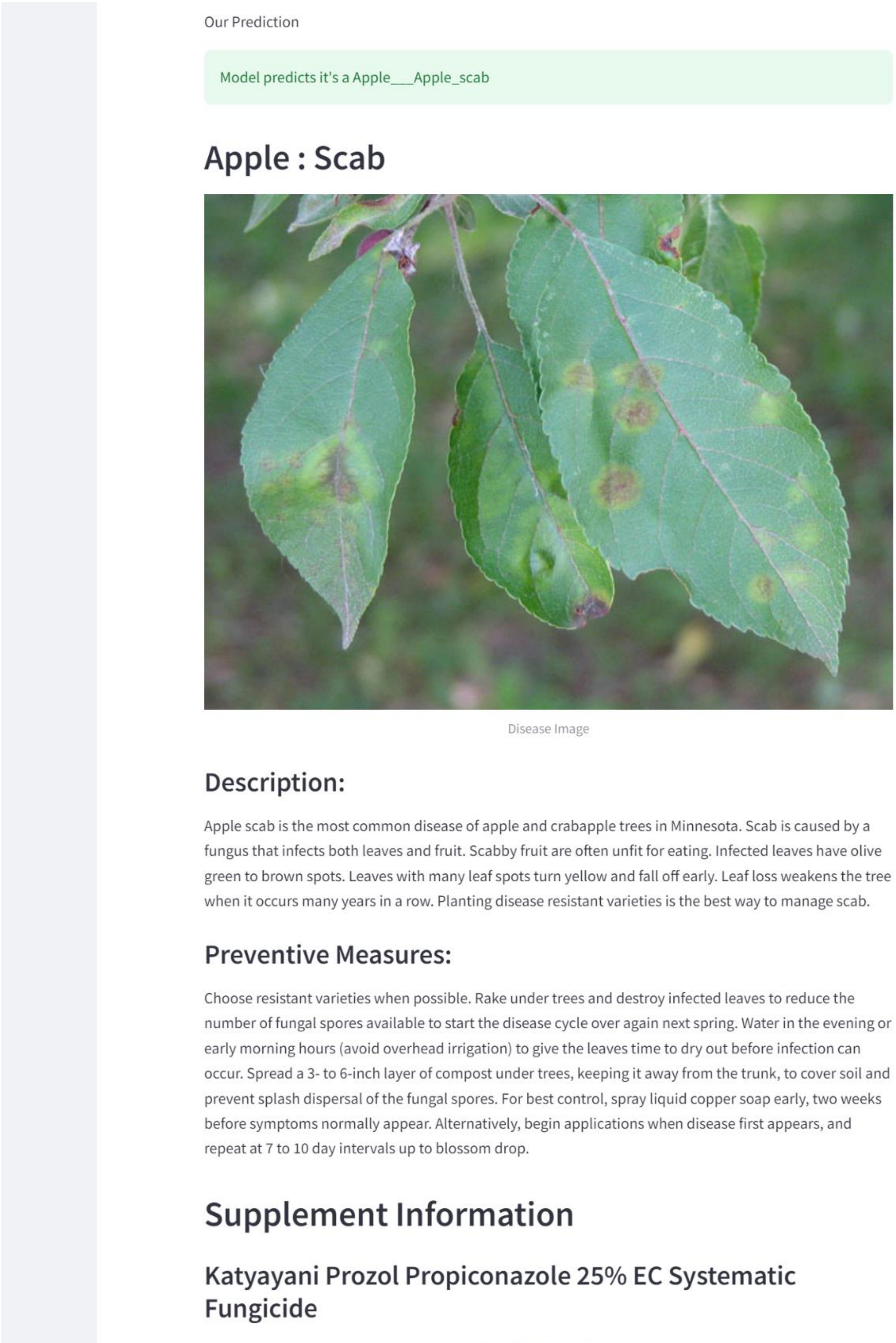

Figure 6.6 Result after prediction

7 Conclusion and Future Scope

Humans have analyzed and created plant-based food items for fiber, medicine, and other uses for generations. Crop diseases are among the numerous risks which should be addressed while farming crops. Therefore, it is essential that we improve food quality & maintain a steady agriculture sector, since this protects the food security of the country.

There has been widespread usage of Convolutional neural network techniques in the identification of plant diseases. Recognizing the disease accurately and efficiently is mainly the purpose of the proposed approach. The experimental results indicate that the proposed approach is a valuable approach, which can significantly support an accurate detection of leaf diseases in a little computational effort.

The objective of the "Plant Disease Detection Using CNN" research is to develop a neural network competent of recognizing 14 crop species and 26 prevalent illnesses. When verified in a controlled setting, an overall accuracy of 97.34% is shown.

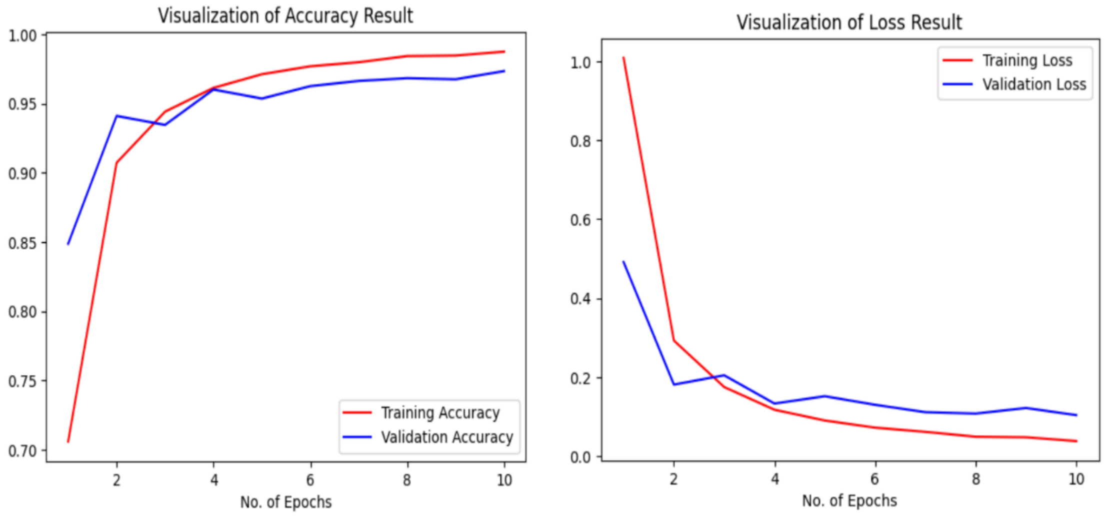

Figure 7.1 Accuracy Plots

Looking ahead, there are several avenues for further development and enhancement of this system:

Improved plant disease detection: In this project we have aimed to build and deploy a plant disease detection system that covers the most commonly facing diseases, and this dataset was so huge and it takes longer hours for the model to be trained. Also, we have attained an accuracy of 97.34%, which is remarkable and can be improved by improving the quality of the dataset. And even more types of plant diseases can be added to the dataset for better precision.

Real-Time Processing Enhancements: While the system performs efficiently, there is always room for improvement in processing speed and real-time analysis. Exploring advanced computational techniques or hardware accelerations could yield faster processing times.

User Interface Customization: Further customization options in the user interface, tailored to specific user requirements or industry standards, could enhance the system's usability.

Scalability and Cloud Integration: Enhancing the system's scalability and potentially integrating cloud-based services could facilitate handling larger datasets and enable remote access and analysis.

Integration with IoT and Sensor Data: Incorporating data from IoT devices and sensors, such as soil moisture sensors, weather stations, and crop health monitoring systems, can provide real-time data inputs to the project. This integration can enable more accurate decision-making, predictive analytics, and proactive alerts for farmers regarding irrigation, fertilization, and crop health management.

8 Bibliography

[1] R. G. de Luna, E. P. Dadios, and A. A. Bandala, “Automated Image Capturing System for Deep Learning based Tomato Plant Leaf Disease Detection and Recognition,” IEEE Region 10 Annual International Conference, Proceedings/TENCON, vol. 2018-October, pp. 1414–1419, Feb. 2019, doi:10.1109/TENCON.2018.8650088.

[2] X.-P. Fan, J.-P. Zhou, and Y. Xu, ‘‘Recognition of field maize leaf diseases based on improved regional convolutional neural network,’’ J. South China Agricult. Univ., vol. 41, no. 6, pp. 82–91, Jun. 2020.

[3] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, “Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks,” IEEE Access, vol. 7, pp. 59069–59080, 2019, doi: 10.1109/ACCESS.2019.2914929.

[4] P. Soni and R. Chahar, “A segmentation improved robust PNN model for disease identification in different leaf images,” 1st IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems, ICPEICES 2016, Feb. 2017, doi:10.1109/ICPEICES.2016.7853301.

[5] N. M. Madhav and U. Scholar, “Plant Disease Detection Using Deep Learning,” International Research Journal on Advanced Science Hub (IRJASH) Special, vol. 03, 2021, Accessed: Nov. 25, 2022. [Online]. Available: www.rspsciencehub.com.

[6] G. Geetharamani and A. P. J., “Identification of plant leaf diseases using a nine-layer deep convolutional neural network,” Computers & Electrical Engineering, vol. 76, pp. 323–338, Jun. 2019, doi:10.1016/J.COMPELECENG.2019.04.011.

[7] V. Suma, R. A. Shetty, R. F. Tated, S. Rohan, and T. S. Pujar, “CNN based Leaf Disease Identification and Remedy Recommendation System,” Proceedings of the 3rd International Conference on Electronics and Communication and Aerospace Technology, ICECA 2019, pp. 395– 399, Jun. 2019, doi:

10.1109/ICECA.2019.8821872.

[8] “New Plant Diseases Dataset | Kaggle.” https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

[9] T. Subetha, R. Khilar, M. C.-M. T. Proceedings, and undefined 2021, “A comparative analysis on plant pathology classification using deep learning architecture–Resnet and VGG19,” Elsevier.

[10] S. Vallabhajosyula, V. Sistla, V. K.-J. of P. D. and, and undefined 2022, “Transfer learningbased deep ensemble neural network for plant leaf disease detection,” Springer.