PhytoGuard

(A Plant Disease Detection System)

A Project Report

submitted for the partial fulfillment for the award of the degree of

Bachelor of Technology

Submitted by

Anushree Ranjeet - 2116120
Himanshi Khaitan – 2116169
Jahnvi Tiwari – 2116179
Priyanshi Singh - 2116806

Under the supervision of

Dr. Swati Nigam

Department of Computer Science
Banasthali Vidyapith

Session: 2023-24

<u>Certificate</u>

Certified that Anushree Ranjeet, Himanshi Khaitan, Jahnvi Tiwari and Priyanshi Singh have carried out the project work titled **“PhytoGuard”** from _______ to _______ for the award of the Bachelors of Technology from Banasthali Vidyapith under my supervision. The thesis embodies result of original work and studies carried out by students themselves and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.

Dr. Swati Nigam

Designation:

Place:

Date:

Abstract

Plant diseases pose significant threats to agricultural productivity and food security worldwide. Rapid and accurate detection of plant diseases is crucial for timely intervention and effective management. In this project, a novel approach is presented for plant disease detection using deep learning techniques. The methodology utilizes a Sequential Convolutional Neural Network (CNN) model, trained on a comprehensive dataset comprising images of various plant diseases and healthy leaves. The trained model is integrated into the PhytoGuard website, providing a user-friendly interface for farmers and agricultural professionals to upload leaf images and obtain real-time diagnosis.

The backend of the system employs the Sequential CNN architecture, leveraging its ability to effectively capture spatial hierarchies and patterns in images. The model is trained on a diverse dataset encompassing 38 classes of plant diseases, enabling robust detection across a broad spectrum of plant ailments. The frontend interface of the PhytoGuard website enables users to conveniently upload leaf images, which are then processed by the CNN model to determine the presence of disease and identify the specific ailment. Additionally, the system accurately identifies healthy leaves, aiding in proactive disease management strategies.

The experimental results demonstrate the effectiveness of the proposed approach, achieving high accuracy in disease classification and providing rapid diagnosis to support timely agricultural interventions. The integration of deep learning techniques with web-based platforms holds promise for scalable and accessible solutions in plant pathology, facilitating informed decision-making and contributing to sustainable agricultural practices.

Acknowledgement

We extend our sincere gratitude to all those who have contributed to the successful completion of this project on Plant Disease Detection. First and foremost, we express our heartfelt appreciation to our project supervisor, Dr. Swati Nigam, whose guidance, expertise, and unwavering support have been invaluable throughout this endeavour.

We are grateful to our institution Banasthali Vidyapith for providing the necessary resources and infrastructure for the successful completion of this project. The support of the teachers has been indispensable, as they willingly assisted us in resolving any queries or concerns that arose during the course of the project.

We also acknowledge the contributors of the PlantVillage dataset on Kaggle, particularly SP Mohanty, for providing the valuable dataset that formed the foundation for training the CNN model in the project.

Also, we express our deepest appreciation to the farmers who are a motivation for us to work on this project.

Table of Contents:

1. Introduction

The agriculture sector in India holds the record for second-largest agricultural land in the world generating employment for about half of the country's population. Thus, farmers become an integral part of the sector to provide us with a means of sustenance. The current state of agriculture faces a significant challenge in the form of plant diseases, which can lead to substantial crop losses and economic consequences for farmers and food security. Various crops are susceptible to a range of leaf diseases that can significantly impact yield. Potatoes and bell peppers are commonly affected by fungal diseases like late blight and early blight, while strawberries can be plagued by fungal issues like gray mold and leaf scorch. Apples and cherries can suffer from fungal infections like scab and leaf spot, while corn (maize) is vulnerable to rust and blight. Grapes contend with powdery mildew, and peach trees are susceptible to peach leaf curl.

Tomato crops are highly affected by diseases like bacterial spot, early blight, late blight, and leaf mold. The blight is the most prevalent disease among others. The tomato crop is highly susceptible to a wide range of disease at each stage of its growth. This is due to different factors based on climatic conditions and environmental parameters. By identifying these diseases, tremendous loss in the yield can be alleviated. Also, the final agricultural product obtained in terms of quality and quantity can be improved [5].

Plant diseases represent a significant challenge to global agriculture, causing substantial losses in crop yield and quality. Timely detection and effective management of these diseases are critical for ensuring food security and sustainable agricultural practices. Traditional methods of disease diagnosis often rely on visual inspection of plant diseases, which is an exceedingly laborious and time-consuming task, necessitating not only a substantial workforce but also expertise in plant pathology. The advent of deep learning technologies, particularly Convolutional Neural Networks (CNNs) emerges as a promising solution for the swift and accurate detection of plant diseases.

In this project, a comprehensive approach for the detection of plant diseases using deep learning techniques is presented. The methodology centers around the development of a Sequential CNN model trained on a diverse dataset encompassing images of diseased and healthy leaves of various plants. The dataset consists of 38 classes of plant diseases, covering a wide range of plant ailments and symptoms commonly observed in agricultural settings such as "Late Blight", "Early Blight", "Gray Mold", etc.

The core component of the system is the integration of the trained CNN model into the PhytoGuard website, providing a user-friendly interface for the user to upload leaf images and obtain rapid diagnosis. Leveraging the power of deep learning, the system enables real-time detection of plant diseases, allowing for timely intervention and informed decision-making.

In this report, a detailed overview of the methodology integrated in this project is provided, including the dataset collection and preprocessing steps, the architecture and training of the Sequential CNN model, and the implementation of the PhytoGuard website. The experimental results demonstrating the performance of the system in terms of accuracy, speed, and usability are also presented. Furthermore, the potential impact of the approach on agricultural practices and future directions for research in the field of plant disease detection using deep learning is discussed.

2. Literature Review

[1] Rapid identification of crop diseases is crucial for global food security, but limited infrastructure in many regions hinders this process. S. P. Mohanty et al.,2016 have explored the potential of smartphone-assisted disease diagnosis using deep learning and the increasing prevalence of smartphones worldwide. They leveraged a publicly available dataset of 54,306 images of diseased and healthy leaves from 14 crop species to train a deep convolutional neural network for disease identification (including healthy states). The trained model achieves an impressive accuracy of 99.35% on a separate test set, demonstrating the feasibility of this approach. This study highlights the promise of deep learning on large, publicly available datasets for developing smartphone-based tools for large-scale crop disease diagnosis. This technology has the potential to revolutionize disease identification in resource-limited areas, promoting global food security.

[2] Sanika Singh et al., proposed a deep learning-based approach for the automated detection of plant leaf diseases. A dataset of leaf of diseased and healthy plants was collected and used to train a convolutional neural network (CNN) model. The model was trained on several architectures and the best performing model was selected based on its accuracy on the validation dataset. The final model was tested on a test dataset and achieved an accuracy of 99.2%.

[3] Tomato is one of the most produced crops all around the world. Images of tomato leaves were taken from PlantVillage dataset. Ten different classes including healthy images are used. Two different deep learning network architectures were tested first AlexNet and then SqueezeNet by Halil Durmus et al.,2017. For both of these deep learning networks training and validation were done on the Nvidia Jetson TX1. The accuracy of AlexNet was 0.9565 and SqueezeNet was 0.943 on the test set. It is shown that SqueezeNet is good candidate for the mobile deep learning classification due to its lightweight and low computational needs. Another advantage of using smaller network is the updating model. When the mobile application is updated through the mobile communication it will cost lower data and updating speed will be higher.

[4] Artificial intelligence (AI) acts as a bridge between information and communication technology (ICT) and various industry applications. AI algorithms drive decision-making, with machine learning (ML) and deep learning (DL) being the leading techniques. DL mimics the human brain's structure using layers and optimizers, enabling the creation of highly accurate and reliable models. In this research, Suja Radha et al., 2021 explored both ML and DL approaches, finding that DL yielded superior results. They evaluated the models using metrics like precision, F1 score, accuracy, and area under the curve (AUC). Our findings showed that random forest (RF) achieved a classification accuracy (CA) of 76.8%, followed by SGD (86.5%), SVM (87%), VGG-19 (87.4%), Inception-v3 (89%), and VGG-16 (89.5%). Stakeholders regularly capture and submit images to the system, empowering farmers to make informed decisions about pesticides and prevent crop damage.

[5] The main contribution of this work by Karthik R. et al., 2019 is the integration of attention mechanism on top of the Residual network for effective feature learning. It helps to selectively weigh the features different layers at the inception of a single layer. Hence, the receptive field at a layer is extended to look at feature maps from different levels of the processing hierarchy. The current layer can now process its input with more contextual information. Learning at the layers preceding the current layer is now aided by the perception of the features at the current layer. This is due to back propagation of the tensors along the skip connections.

[6] Shalya Saxena et al., 2023 have proposed an ensemble model combining a deep learning approach (convolutional layers from a Convolutional Neural Network) with a Random Forest classifier. To evaluate its effectiveness, the model was tested on 380,000 images from standard datasets, achieving an impressive 99.89% accuracy during validation. The performance of the ensemble model was compared to standalone Random Forest and CNN models for classifying 46 different disease and healthy plant categories. While both standalone models achieved good results (92% for Random Forest and 93% for CNN), the ensemble approach significantly outperformed them. In

essence, the combination of convolutional layers for feature extraction and Random Forest for classification proved highly effective in detecting and classifying plant leaf diseases.

[7] In this paper, an adaptive approach for the identification of fruit diseases is proposed by Dubey S.R. et al., 2013 and experimentally validated. The image processing based proposed approach is composed of the following main steps; in the first step K-Means clustering technique is used for the defect segmentation, in the second step some state of the art features are extracted from the segmented image, and finally images are classified into one of the classes by using a Multi-class Support Vector Machine. Diseases of apple as a test case was considered and the approach was evaluated for three types of apple diseases namely apple scab, apple blotch and apple rot. The experimental results express that the proposed solution can significantly support accurate detection and automatic identification of fruit diseases. The classification accuracy for the proposed solution is achieved up to 93%.

[8] Wang B. et al., 2021 have highlighted the potential of deep learning (DL) for accurate plant disease detection. Various DL architectures like AlexNet and ResNet effectively classify diseases, and visualization techniques aid in understanding their decision-making process. Hyperspectral imaging (HSI) offers promising results for early disease identification, with DL models efficiently analyzing HSI data. This review suggests DL as a powerful tool for plant disease detection with ongoing advancements in model robustness.

[9] Barbedo (2018) in Biosystems Engineering explores the factors impacting the application of deep learning (DL) for plant disease recognition. The paper highlights the limitations of traditional machine learning and the surge in DL research within this field since 2015. It explores the effectiveness of various DL architectures and emphasizes the role of visualization techniques in understanding these models' decision-making processes. The potential of hyperspectral imaging (HSI) for early disease detection is discussed, along with the ability of DL models to analyze HSI data. This review underlines the promise of DL for plant disease recognition while acknowledging the need for advancements in model robustness.

[10] Kamilaris and Prenafeta-Boldú (2018) survey the applications of deep learning (DL) in agriculture in Computers and Electronics in Agriculture. They explore the recent rise of DL in agriculture, highlighting its potential to address complex agricultural challenges. The review examines 40 research efforts utilizing DL for tasks like weed detection, yield prediction, and disease identification. It analyzes the specific DL models employed, data sources, and performance metrics used within these studies. The findings suggest that DL achieves high accuracy, potentially surpassing traditional image processing techniques in agricultural applications.

[11] This review paper by M.H. Saleem et al., 2019 explained DL approaches for the detection of plant diseases. Moreover, many visualization techniques/mappings were summarized to recognize the symptoms of diseases. Some research gaps were pointed out. Hyperspectral/multispectral imaging is an emerging technology. it should be used with the efficient DL architectures to detect the plants' diseases even before their symptoms are clearly apparent.

A more efficient way of visualizing the spots of disease in plants should be introduced as it will save costs by avoiding the unnecessary application of fungicide/pesticide/herbicide.

The severity of plant diseases changes with the passage of time, therefore, DL models should be improved/modified to enable them to detect and classify diseases during their complete cycle of occurrence.

DL model/architecture should be efficient for many illumination conditions, so the datasets should not only indicate the real environment but also contain images taken in different field scenarios.

A comprehensive study is required to understand the factors affecting the detection of plant diseases, like the classes and size of datasets, learning rate, illumination, and the like.

[12] Anshul Tripathi et al., 2022 have achieved an accuracy of 98.18% using a CNN trained on a dataset of 20,639 images from the Plant Village dataset on Kaggle. The study focused on differentiating between three crops (tomato, pepper, and potato) and diagnosing twelve different diseases. The high accuracy achieved highlights the capability of CNNs to extract key features

from plant images for disease classification within a neural network environment. Furthermore, the study implemented data augmentation techniques to address the limitations of a small dataset, demonstrating its effectiveness in improving model performance. Additionally, they utilized dropout with a value of 0.25 to successfully mitigate overfitting, signifying the importance of regularization techniques in achieving optimal CNN performance. The study reinforces the notion that CNNs can effectively classify healthy and diseased leaves within a limited dataset scenario, even with a minimum number of parameters (in this case, 58K).

[13] Ghosh H et al., 2023 compared three deep learning models (VGG19, DenseNet121, ResNet50) for potato leaf disease classification. A large dataset with healthy and diseased potato leaf images was augmented and used for training and evaluation. All models achieved high accuracy, with DenseNet121 excelling at 97.92% on the validation set. The research highlights the potential of deep learning for accurate and efficient potato leaf disease detection, paving the way for improved crop management and yield.

[14] This research by Sucharitha et al. (2023) explores the application of deep learning models for potato leaf disease classification. The study investigates the performance of three prominent deep learning architectures: VGG19, DenseNet121, and ResNet50. These models were trained on a large dataset of potato leaf images encompassing healthy and diseased samples. To improve model robustness and generalization capabilities, data augmentation techniques were employed. Their findings revealed that all three models achieved high classification accuracy for various potato leaf diseases. However, DenseNet121 emerged as the most effective model, achieving an outstanding accuracy of 97.92% on the validation dataset. This study reinforces the potential of deep learning for accurate and efficient plant disease detection, paving the way for improved disease management strategies and enhanced crop productivity.

[15] Garima Shrestha et al., 2020 deployed the convolutional neural network to detect the plant disease. Authors have successfully classified 12 plant diseases with 88.80% accuracy. The dataset of 3000 high resolution RGB images were used for experimentation. The network has 3 blocks of

convolution and pooling layers. This makes the network computationally expensive. Also, the F1 score of the model is 0.12 which is very low because of a higher number of false negative predictions.

[16] A CNN model was investigated for its effectiveness in early detection of Alphonso mango leaf diseases with the goal of improving crop production. Jolly Masih et al., 2023 achieved a data accuracy of 82%, falling within the anticipated range of 85-90% through the applied method. This promising outcome paves the way for further research in other fruit crops. The approach has the potential to significantly improve identification and classification of bacterial, viral, and fungal diseases in plant leaves. Early disease detection facilitated by this method could lead to reduced pesticide use and contribute to healthier crops.

[17] Recent research proposes a new method for plant disease detection using a combination of Convolutional Autoencoders (CAE) and Convolutional Neural Networks (CNN). This method addresses the limitations of high training parameters or low accuracy in existing techniques. Punam Bedi and Pushkar Gole, 2021 proposed a model that utilizes CAE for dimensionality reduction of leaf images, leading to a significant decrease in training parameters compared to conventional approaches. This reduction translates to faster training and disease identification times. The study achieved high accuracy (99.35% training, 98.38% testing) using only 9,914 training parameters. The proposed hybrid model has achieved 98.0% Precision. Its Recall is 98.72%, and its F1-measure is 98.36%.

[18] Konstantinos P. Ferentinos, 2018 highlighted the efficacy of Convolutional Neural Networks (CNNs) for plant disease detection using leaf images. Researchers developed specialized CNN architectures trained on a public dataset of 87,848 images encompassing 25 plant species and 58 disease categories. The most successful VGG-based model achieved 99.53% accuracy on a unseen testing set of 17,548 images. This emphasizes the suitability of CNNs for automated disease diagnosis. Notably, the inclusion of real-world field images significantly boosted performance, suggesting its importance for future model development. Furthermore, the low computational cost (2ms per image on a single GPU) paves the way for mobile applications on smartphones

and agricultural drones for real-time disease monitoring. It could lead to the development of automated pesticide prescription systems, requiring confirmation from the disease diagnosis system, which could drastically reduce pesticide overuse and environmental impact.

[19] Srdjan Sladojevic et al., 2016 have investigated an ensemble approach using pre-trained deep learning models for apple leaf disease detection. The ensemble consisted of DenseNet121, EfficientNetB7, and EfficientNet NoisyStudent models and achieved an accuracy of 96.25% on a dataset with four classes: healthy, apple scab, apple cedar rust, and multiple diseases. The authors highlight that their work focused on a limited set of classes and diseases. However, their ensemble approach outperformed existing models trained on the same dataset. Additionally, they explored individual model performance using various metrics before combining predictions through averaging. Finally, they emphasized on the potential for this method to be used by farmers through a web application for automated disease classification, reducing reliance on experts and saving time and resources.

3. Methodology

To develop a practical and effective solution for detecting plant diseases, the proposed methodology of this project includes several key steps such as image processing, deep learning CNN model development, validation, performance evaluation and testing. The following section provides a detailed explanation of the general methodology as well as the specific methodology used for developing the deep learning Sequential CNN model.

3.1 Dataset

The project uses images from the publicly available dataset, PlantVillage curated by Sharada P. Mohanty et al. [1], for its processing, model training, validation and testing. The dataset consists of around 87000 RGB images of healthy and unhealthy plant leaves. There are 38 classes available in the dataset for experimentation of our algorithm. These classes are shown in
Table 1. The images are segregated for training, validation and testing. The number of images used from the dataset for:
Training: 43444, Validation: 10860, Testing: 30.

Table 1: Dataset Specifications

Plant	**Disease name**	**No. of images**
Apple	Healthy	1645
	Diseased: Apple scab	630
	Diseased: Black rot	621
	Diseased: Cedar apple rust	275
Blueberry	Healthy	1502
Cherry	Healthy	854
	Diseased: Powdery mildew	1052

Corn (maize)	Healthy	1162
	Diseased: Cercospora leaf spot (gray leaf spot)	513
	Diseased: Common rust	1192
	Diseased: Northern leaf blight	985
Grape	Healthy	423
	Diseased: Black rot	1180
	Diseased: Esca (Black measles)	1383
	Diseased: Leaf Blight (Isariopsis leaf spot)	1076
Orange	Diseased: Huanglongbing (Citrus greening)	5507
Peach	Healthy	360
	Diseased: Bacterial spot	2297
Bell Pepper	Healthy	1478
	Diseased: Bacterial spot	997
Potato	Healthy	152
	Diseased: Early blight	1000
	Diseased: Late blight	1000
Raspberry	Healthy	371
Soybean	Healthy	5090

Squash	Powdery mildew	1835
Strawberry	Healthy	456
	Diseased: Leaf scorch	1109
Tomato	Healthy	1591
	Diseased: Target spot	1404
	Diseased: Tomato mosaic virus	373
	Diseased: Tomato yellow leaf curl virus	3209
	Diseased: Bacterial spot	2127
	Diseased: Early blight	1000
	Diseased: Late blight	1909
	Diseased: Leaf Mold	952
	Diseased: Septoria leaf spot	1771
	Diseased: two spotted spider mite	1676

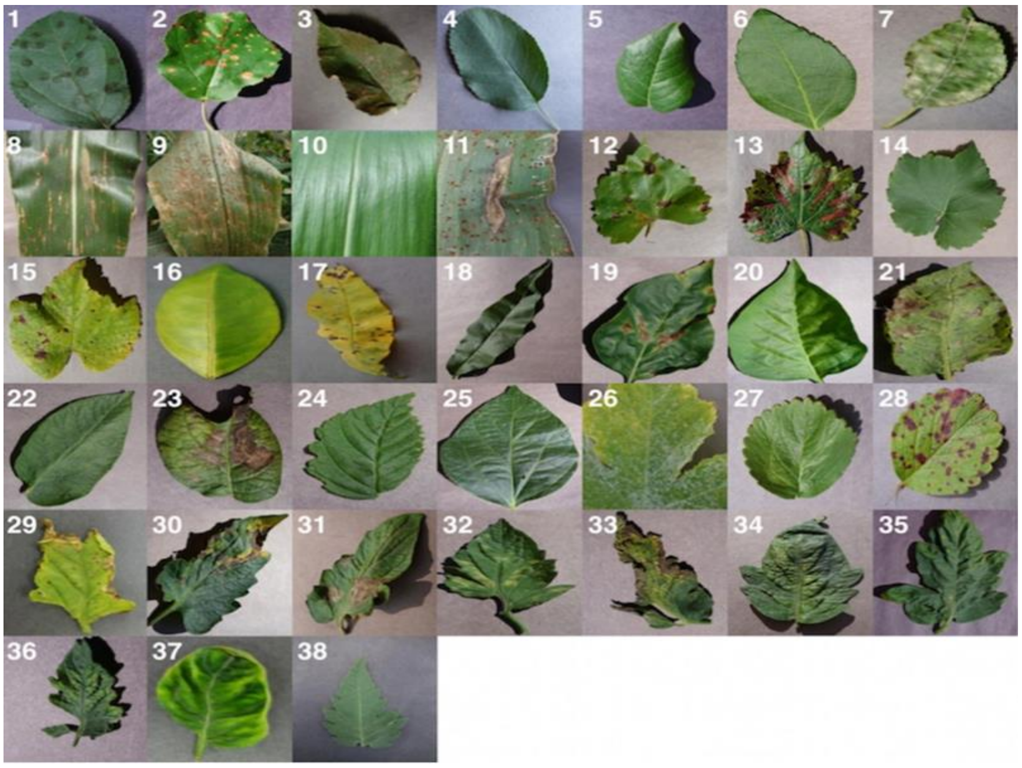


Fig 1: Sample Images in dataset used

3.2 Importing Libraries

- TensorFlow (import tensorflow as tf): This is the core deep learning library used to build, train, and deploy CNN models. It provides the building blocks for neural networks and handles numerical computations.
- Pandas (import pandas as pd): Pandas is used for data manipulation and analysis. In plant disease detection, it helps with tasks like loading image data, creating dataframes for storing labels, and preprocessing the data for the CNN model.
- (from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense): This line imports essential layers used to construct the Sequential CNN model.
  - Conv2D: Creates convolutional layers that extract features from the input images.
  - MaxPool2D: Performs down sampling to reduce data size and control overfitting.
  - Flatten: Converts the high-dimensional feature maps into a one-dimensional vector for feeding into fully-connected layers.

    - Dense: Represents fully-connected layers that combine features and make the final disease classification.

- (from tensorflow.keras.models import Sequential): This line imports the Sequential API from tensorflow.keras, allowing us to build the CNN model by stacking layers sequentially in a linear fashion.
- matplotlib.pyplot (import matplotlib.pyplot as plt): This library is used for creating visualizations like charts and graphs. It is helpful for visualizing training progress (e.g., accuracy curves) and exploring the image data (e.g., displaying sample images). This is required in model validation and testing.
- Seaborn (import seaborn as sns): Built on top of Matplotlib, Seaborn provides a higher-level interface for creating statistical graphics. It can be used to create more visually appealing and informative visualizations compared to raw Matplotlib plots. While not essential for the core CNN model, it can be useful for exploratory data analysis related to plant disease detection. This is required in model validation.
- NumPy (import numpy as np): This library provides functions for working with multi-dimensional arrays. In this context, it's used to convert uploaded image data (likely in formats like JPEG or PNG) into NumPy arrays, which are a fundamental data structure for machine learning models like CNNs.
- cv2 (import cv2): The OpenCV library offers functions for image processing tasks, such as loading the image from the test set.
- ImageDataGenerator (from keras.preprocessing.image import ImageDataGenerator): This module from Keras (a deep learning framework) is used for image data augmentation. During testing, it can be used to create variations of the uploaded image (e.g., rotations, flips) to improve the model's robustness and generalizability. This helps the model perform better on unseen images that might have slightly different orientations or lighting conditions.

3.3 Data Preprocessing

Images were organized into "train" and "valid" directories, representing training and validation sets, respectively. 29.5% of the total dataset was used for model development.
TensorFlow's "tf.keras.utils.image_dataset_from_directory" function was employed for loading and preprocessing the images. The Keras preprocessing layers allows to build Keras-native input processing pipelines, which can be used as independent preprocessing code in non-Keras workflows, combined directly with Keras models, and exported as part of a Keras saved Model.

This function offers several features crucial for image-based machine learning tasks:

- Directory Structure: It automatically infers image labels from their respective directory names, eliminating manual labelling.
- Batch Processing: Images are loaded in batches of 32, a common practice to optimize memory usage and accelerate gradient calculations during model training.
- Image Resizing: All images are resized to 128 x 128 pixels to ensure consistency in input dimensions for the CNN model. This standardization is essential for neural networks to effectively process the data.
- Color Mode: The 'rgb' color mode is specified, indicating full-color images with three channels (red, green, and blue) to capture color information relevant for disease identification.
- Label Encoding: The 'categorical' label mode is used, as the task involves classifying images into multiple disease categories (38 classes in this case). This mode represents each class with a one-hot encoded vector, a format well-suited for multi-class classification problems.
- Image Shuffling: The dataset is shuffled for each epoch of training to prevent potential model bias and ensure a wider range of image combinations during learning.
- Interpolation: The 'bilinear' interpolation method is applied for resizing images, helping to minimize distortions that could arise from changes in image dimensions.

The training set has each data point in the form of a pair. The first element is a preprocessed image (variable batch size, 128x128 pixels, 3 color channels) stored as a tensor with floating-point numbers. The second element is a one-hot encoded label (variable batch size, 38 elements) also as a tensor with floating-point numbers, likely representing 38 different disease classes.

3.4 Model Building

3.4.1 Deep Learning

Deep learning, a subfield of artificial intelligence (AI) is used in this project development. By mimicking the structure and function of the human brain, deep learning models excel at extracting complex patterns from large datasets of images, making them ideal for plant disease detection tasks.

At the core of deep learning lie artificial neural networks (ANNs). These intricate structures are composed of interconnected nodes (artificial neurons) arranged in multiple layers. Inspired by biological neurons, these artificial neurons process information and transmit signals to subsequent layers. The first layers typically focus on extracting low-level features from the input data (images in this project), such as edges and textures. As information progresses through the network, these layers progressively combine and refine the extracted features, ultimately leading to a high-level understanding of the input. This hierarchical feature extraction allows deep learning models to identify subtle patterns and relationships within the data, a capability particularly valuable for disease detection in plants.

Traditional machine learning (ML) models often rely on handcrafted features engineered by human experts. This feature engineering process can be time-consuming, labor-intensive, and highly dependent on domain expertise. In contrast, deep learning models excel at automatically learning these features directly from the data. By analyzing vast amounts of training images depicting healthy and diseased plant tissues, deep learning models can effectively identify the most discriminative features for disease classification. This data-driven approach not only reduces reliance on expert knowledge but also allows the model to potentially discover features that humans might overlook.

Furthermore, the inherent ability of deep learning models to handle complex, high-dimensional data makes them well-suited for the challenges of plant disease detection. Plant diseases often manifest in subtle visual variations like changes in color, texture, or leaf shape. Deep learning models, with their sophisticated architectures and powerful learning algorithms, can effectively capture these subtle variations and distinguish between healthy and diseased plants with high accuracy.

Deep learning offers several advantages over traditional ML models for plant disease detection. Firstly, it eliminates the need for manual feature engineering, saving time and effort. Secondly, its data-driven approach allows for the discovery of potentially unknown but significant features for disease classification. Finally, deep learning models excel at handling complex, high-dimensional image data, enabling them to capture subtle variations indicative of plant diseases.

3.4.2 Convolutional Neural Networks

Unlike other deep learning models that treat images as a flat collection of pixels, Convolutional Neural Networks (CNNs) exploit the spatial relationships between pixels which allows them to directly learn discriminative features from the image data, eliminating the need for complex pre-processing steps. This is achieved through convolutional layers, the building blocks of CNNs. These layers employ filters that slide across the image, analyzing small regions and detecting patterns like edges, textures, and color variations. By stacking multiple convolutional layers, CNNs can progressively extract increasingly complex features relevant to disease identification.

3.4.3 Sequential CNN Model

The core of our plant disease detection system lies in a Convolutional Neural Network (CNN) model built using TensorFlow.keras Sequential API. This API offers a modular approach, allowing us to construct the model by stacking layers sequentially. Here's a breakdown of the architecture and its suitability for plant disease detection:

- **Sequential Design:**
  The Sequential API simplifies model creation by enabling the addition of layers one after another. This linear stacking allows for straightforward experimentation with different layer configurations to optimize model performance.
- **Convolutional Layers (Conv2D):**
  The core building blocks of the model are convolutional layers (Conv2D). These layers are designed to extract features from the input images. They employ filters that slide across the image, detecting patterns and features like edges, textures, and color variations. Multiple convolutional layers can be stacked to progressively learn more complex and hierarchical features relevant to disease identification.
- **Activation Functions:**
  After each convolutional layer, non-linear activation functions (e.g., ReLU) are typically applied. These functions introduce non-linearity into the network, allowing it to model complex relationships between features and ultimately improve classification accuracy.
- **Pooling Layers (Optional):**
  Pooling layers (e.g., MaxPooling2D) can be strategically inserted between convolutional layers to reduce the dimensionality of the data. This reduces computational cost and helps control overfitting, where the model memorizes training data instead of learning generalizable patterns.
- **Flatten Layer:**
  After feature extraction through convolutional layers, a flatten layer transforms the high-dimensional feature maps into a one-dimensional vector suitable for feeding into fully-connected layers.
- **Fully-Connected Layers (Dense):**
  These layers perform traditional neural network computations, combining the extracted features from the convolutional layers to

make the final disease classification. The number of neurons in the final dense layer corresponds to the number of disease classes the model is trained to identify.

The modular design of the Sequential API allows for customization of the model architecture (number and type of layers) to adapt to the specific characteristics of the plant disease detection problem and the available dataset size.

3.4.4 Model Building Implementation

A sequential model architecture is initiated using Sequential() as a foundation for building the CNN layer by layer.

- **Constructing Convolutional Layers:**
  - **First Convolutional Layer:** - Filters: 32 3x3 filters are applied, scanning the image to extract features such as edges, textures, and color patterns. - Padding: 'same' padding ensures the output feature maps have the same dimensions as the input image, preserving spatial information. - Activation: ReLU (Rectified Linear Unit) introduces non-linearity, allowing the model to learn complex relationships between features. It's defined as: relu(x) = max(0, x), where x is the input value. - Input Shape: [128, 128, 3] specifies the expected input image dimensions—128x128 pixels with 3 color channels (RGB).
  - **Second Convolutional Layer:** This layer mirrors the first, intensifying feature extraction with 32 filters, 'same' padding, and ReLU activation.
- **Pooling for Dimensionality Reduction:**
  - **Max Pooling:** - Pool Size: 2x2 filters downsample the feature maps by selecting maximum values within 2x2 regions. - Strides: 2 determines the step size between pooling operations, halving the spatial dimensions and reducing computational cost.
- **Transitioning to Fully Connected Layers:**

    - **Flattening:** The high-dimensional feature maps are flattened into a one-dimensional vector, creating a feature vector for classification.
    - **Fully Connected Layers for Classification:**
        - **First Dense Layer:** - Units: 1024 neurons act as a dense intermediary layer, analyzing and refining the extracted features. - Activation: ReLU continues to introduce non-linearity, enabling complex feature combinations.
        - **Output Layer:** - Units: 38 neurons correspond to the 38 disease classes for multi-class classification. - Activation: Softmax converts raw scores into a probability distribution, assigning likelihoods to each class. Its formula is:
          softmax(x_i) = exp(x_i) / sum(exp(x_j)),
          where x_i is a specific class score and x_j represents all scores.
- **ReLU Activation Function:** - Rectifies input values at zero, allowing positive values to pass through unchanged while setting negative values to zero. - Combats the vanishing gradient problem, facilitating model training.
- **Softmax Activation Function:** - Normalizes output scores into a probability distribution across classes. - Essential for multi-class classification tasks.
- **Padding:** - Controls feature map dimensions during convolution. - 'Same' padding maintains dimensions. - 'Valid' padding yields smaller output maps.

3.5 Model Compilation and Training

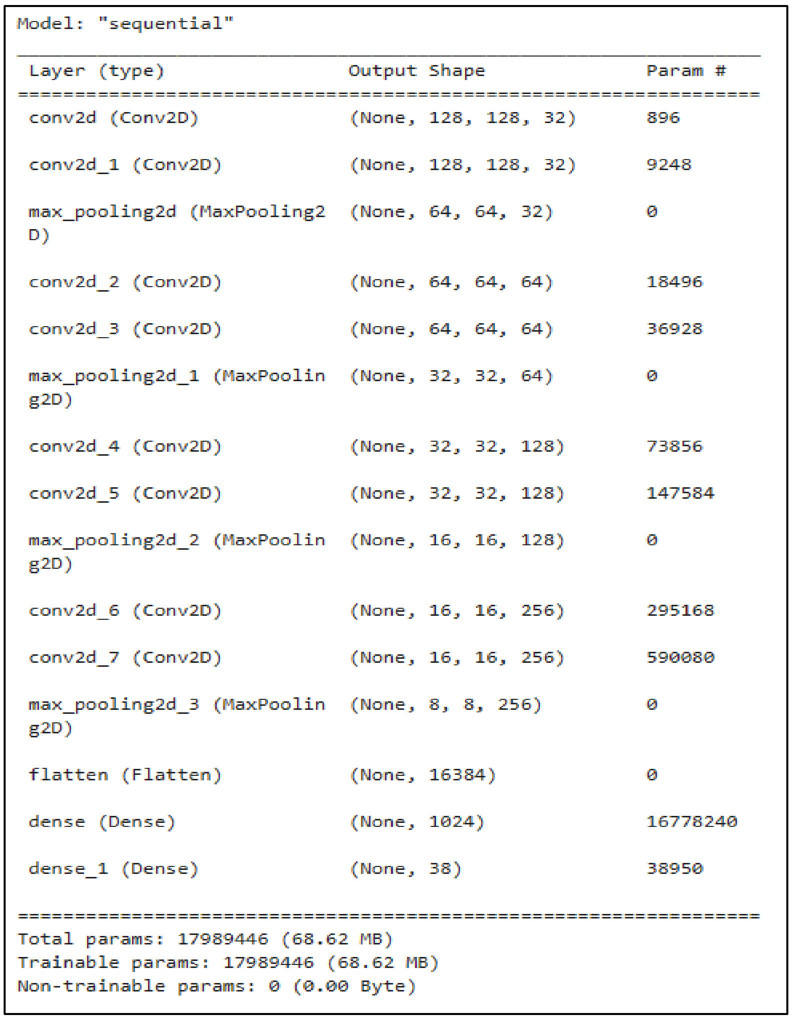
This is the core learning phase where the CNN model ingests a large dataset of pre-processed images containing healthy and diseased plant tissues. Each image is typically paired with a corresponding label indicating the specific disease (or healthy state).

The model processes these images in batches, passing them through its convolutional layers. These layers extract features like edges, textures, and color variations from the images.

After feature extraction, fully-connected layers analyze the extracted features and learn the relationships between them. Based on the provided labels, the model calculates a loss (error) between its predictions and the actual labels.

An optimizer algorithm (e.g., Adam optimizer) then uses this loss to adjust the internal weights and biases of the CNN's layers. By iteratively processing batches of images, calculating loss, and adjusting weights, the model gradually learns to differentiate between healthy and diseased patterns in the images.

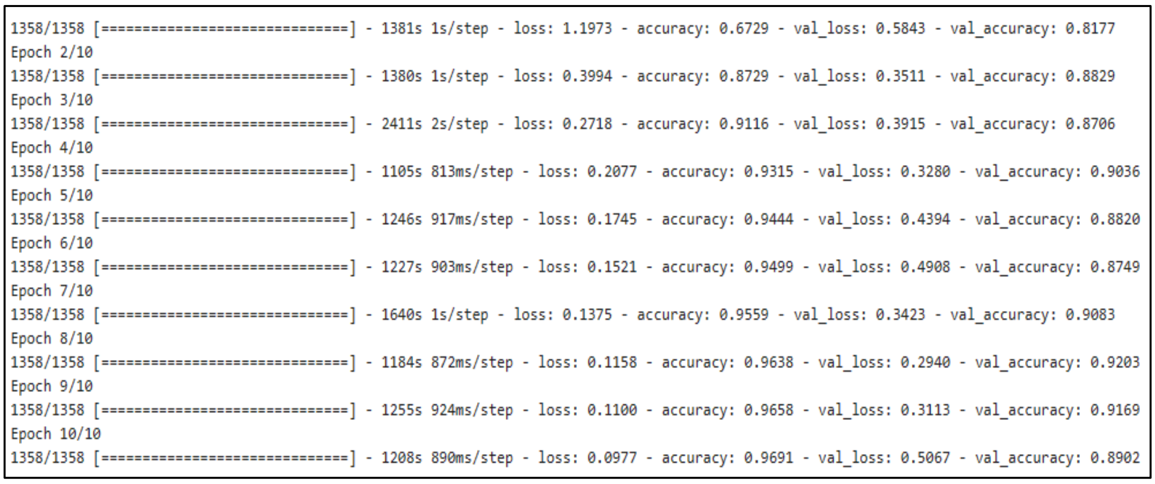
- **Adam Optimizer**(Adaptive Moment Estimation)**:** Adam is a popular optimizer known for its efficiency and effectiveness in various deep learning tasks, including plant disease detection. It combines the advantages of two other optimizers (RMSprop and AdaGrad) to address their shortcomings. Adam dynamically adjusts the learning rate for each parameter, considering past gradients and preventing the learning rate from getting stuck in local minima (poor performing areas) during the training process. This allows the model to converge on optimal solutions faster and achieve better performance.
- **Loss Function:** This function calculates the difference between the model's predicted probability distribution for each class and the actual correct labels.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      896

 conv2d_1 (Conv2D)           (None, 128, 128, 32)      9248

 max_pooling2d (MaxPooling2  (None, 64, 64, 32)        0
 D)

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

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

 max_pooling2d_1 (MaxPoolin  (None, 32, 32, 64)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 32, 32, 128)       73856

 conv2d_5 (Conv2D)           (None, 32, 32, 128)       147584

 max_pooling2d_2 (MaxPoolin  (None, 16, 16, 128)       0
 g2D)

 conv2d_6 (Conv2D)           (None, 16, 16, 256)       295168

 conv2d_7 (Conv2D)           (None, 16, 16, 256)       590080

 max_pooling2d_3 (MaxPoolin  (None, 8, 8, 256)         0
 g2D)

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 1024)              16778240

 dense_1 (Dense)             (None, 38)                38950

=================================================================
Total params: 17989446 (68.62 MB)
Trainable params: 17989446 (68.62 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 1: Model Summary

Epochs represent complete cycles of training the model on the entire training set. Since the number of epochs while training the model are taken as 10, hence the model will go through the training data 10 times, iteratively updating its internal parameters (weights and biases) to improve its ability to distinguish between healthy and diseased patterns in the images. Below is the training history in Figure 2.



```
1358/1358 [==============================] - 1381s 1s/step - loss: 1.1973 - accuracy: 0.6729 - val_loss: 0.5843 - val_accuracy: 0.8177
Epoch 2/10
1358/1358 [==============================] - 1380s 1s/step - loss: 0.3994 - accuracy: 0.8729 - val_loss: 0.3511 - val_accuracy: 0.8829
Epoch 3/10
1358/1358 [==============================] - 2411s 2s/step - loss: 0.2718 - accuracy: 0.9116 - val_loss: 0.3915 - val_accuracy: 0.8706
Epoch 4/10
1358/1358 [==============================] - 1105s 813ms/step - loss: 0.2077 - accuracy: 0.9315 - val_loss: 0.3280 - val_accuracy: 0.9036
Epoch 5/10
1358/1358 [==============================] - 1246s 917ms/step - loss: 0.1745 - accuracy: 0.9444 - val_loss: 0.4394 - val_accuracy: 0.8820
Epoch 6/10
1358/1358 [==============================] - 1227s 903ms/step - loss: 0.1521 - accuracy: 0.9499 - val_loss: 0.4908 - val_accuracy: 0.8749
Epoch 7/10
1358/1358 [==============================] - 1640s 1s/step - loss: 0.1375 - accuracy: 0.9559 - val_loss: 0.3423 - val_accuracy: 0.9083
Epoch 8/10
1358/1358 [==============================] - 1184s 872ms/step - loss: 0.1158 - accuracy: 0.9638 - val_loss: 0.2940 - val_accuracy: 0.9203
Epoch 9/10
1358/1358 [==============================] - 1255s 924ms/step - loss: 0.1100 - accuracy: 0.9658 - val_loss: 0.3113 - val_accuracy: 0.9169
Epoch 10/10
1358/1358 [==============================] - 1208s 890ms/step - loss: 0.0977 - accuracy: 0.9691 - val_loss: 0.5067 - val_accuracy: 0.8902
```

Figure 2: Model Training History

3.6 Model Evaluation

Performance metrics quantify the model's ability to make accurate predictions on unseen data. Common metrics include accuracy (percentage of correct predictions), precision (ratio of true positives to all predicted positives), recall (ratio of true positives to all actual positives), and F1-score (harmonic mean of precision and recall).

Accuracy is a metric used to gauge the model's performance at each stage: training, validation, and testing. Training accuracy refers to the percentage of images the model correctly classifies during the training process. As the model iterates through the training data (epochs), the training accuracy typically increases. This indicates the model is progressively learning to recognize patterns and features associated with different disease classes. Validation accuracy reflects how well the model performs on unseen data that resembles the actual problem it will encounter (classifying new plant disease images).

Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)

Confusion Matrix visually summarizes the model's performance by presenting the number of correct and incorrect predictions for each class. The key elements within the confusion matrix:

- **True Positives (TP):** These represent instances where the model correctly predicted a specific disease class. For example, if the image truly depicts rust disease and the model identifies it as rust disease, this falls under TP.

- **True Negatives (TN):** These represent instances where the model correctly predicted a healthy plant image (absence of disease).
- **False Positives (FP):** These represent errors where the model incorrectly classified an image. For example, if the image shows a healthy plant but the model predicts rust disease, this is a FP.
- **False Negatives (FN):** These represent errors where the model missed a diseased plant. For example, if the image has blight disease but the model predicts a healthy plant, this is a FN.
- **High values along the diagonal (TP and TN):** This indicates good overall performance with the model correctly classifying most images.
- **High values off the diagonal (FP and FN):** This signifies potential issues. Many FPs might suggest the model is overfitting to a particular disease class, while high FN values indicate the model struggles to identify certain diseases.

3.7 Model Validation

A separate validation set, also consisting of pre-processed images and labels, is used to monitor the model's performance during training. This set is crucial to prevent overfitting. Overfitting occurs when the model memorizes the training data too well, losing its ability to generalize to unseen data. The validation set helps identify this issue.

After each training epoch (a complete pass through the entire training dataset), the model is evaluated on the validation set. The loss and accuracy on the validation set provide insights into how well the model is learning without being influenced by the training data itself.

If the validation accuracy starts to plateau or decrease while the training accuracy continues to improve, it might indicate overfitting. In such cases, adjustments to the model architecture, training parameters, or data augmentation techniques might be necessary.

3.8 Model Testing

Once training is complete, a final, unseen test set is used to assess the model's generalizability and real-world performance. This set is entirely independent of the training and validation data.

The model is evaluated on the test set, and metrics like accuracy, precision, recall, and F1-score are calculated to gauge its ability to correctly classify unseen plant disease images. The performance on the test set provides a realistic estimate of how well the model would perform when deployed in a practical setting for plant disease detection.

The trained model is first loaded. The image is acquired from a designated location (represented by the path). The stored BGR format of image is converted into RGB format. To do prediction, we first have to do some preprocessing. To perform preprocessing, we have to convert the image into array format because our model is trained for the same. The model is trained for size (128,128).

The data is fed in the form of batches in the model and for this, the image is converted into numPy array. Then, when the image array is passed into the predict function, it gives the n classes probability where each column has the probability of the image belonging to each class. If the index of the class having the max probability is retrieved, then it can be said that the image belongs to this class.

Hence, the model is tested correct for 22 of the 30 test images.

4. Results

The training history shows that the training loss is 9.77% and the training accuracy is 96.90%. Whereas the validation loss is 50.67% and the validation accuracy is 89.01%.

The figure shows the visualization of training and validation accuracy.

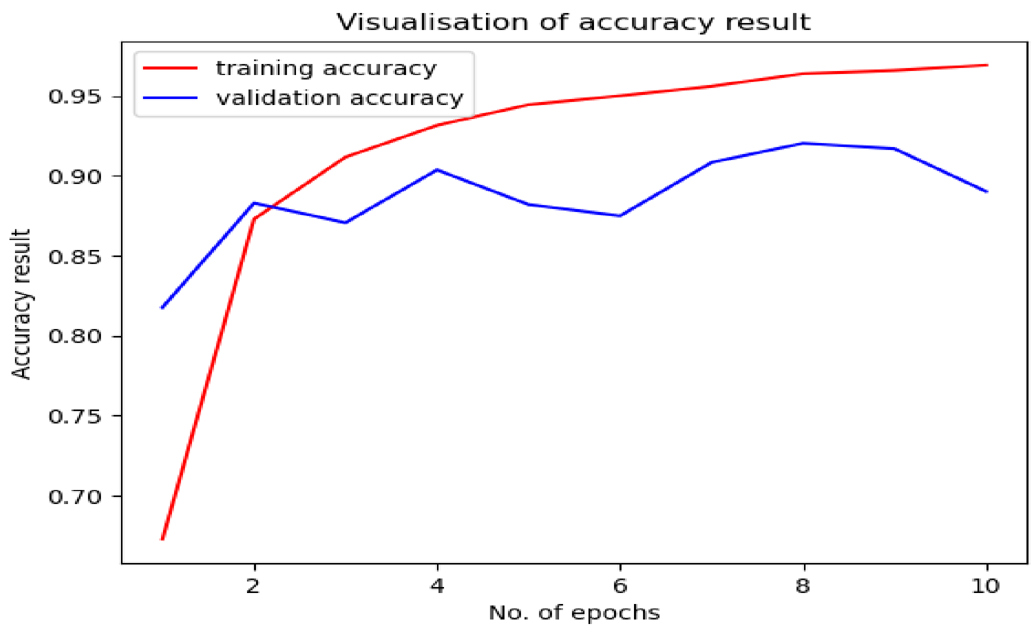


Figure 3

The confusion matrix is shown in the figure having predicted classes on the x-axis and actual classes on the y-axis. The shape of the confusion matrix is (38,38).

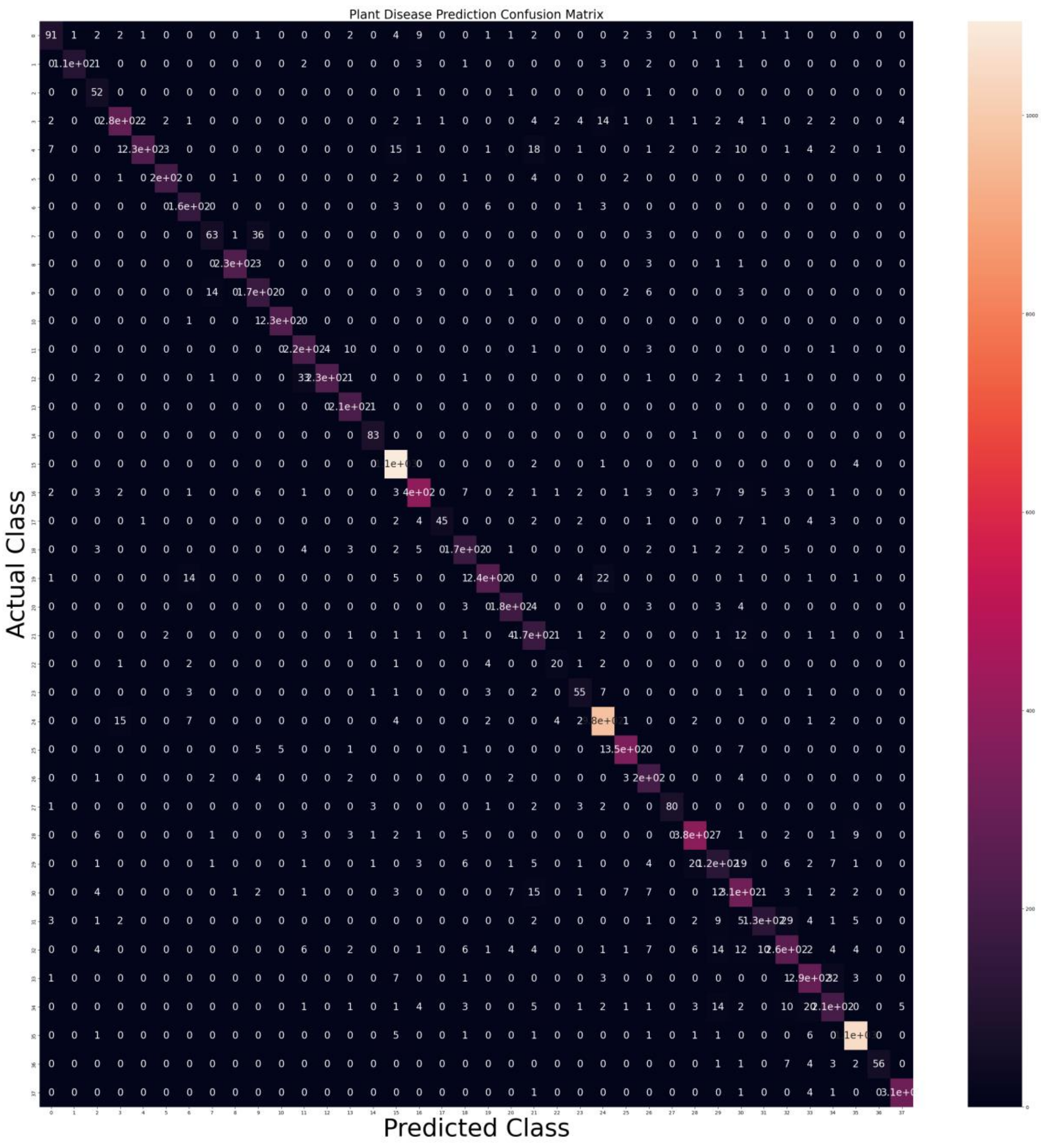

Plant Disease Prediction Confusion Matrix
Actual Class
Predicted Class

Figure 4

5. PhytoGuard Website Development

Here, we have outlined the comprehensive methodology followed for developing the PhytoGuard website, a platform designed to empower farmers with plant disease identification capabilities.

The website link is provided in the references.[20]

5.1 Project Planning and Requirements Gathering

- Defining Project Goals:
    - Conduct brainstorming sessions to establish clear and measurable project objectives.
    - The primary goal was to develop a user-friendly website that allows farmers to upload images of their crops and receive insights into potential plant diseases.
    - Secondary goals may include establishing a user database for future service enhancements or integrating with external disease diagnosis APIs.

5.2 Target Audience Identification:

- Conduct market research to identify the primary user base.
- The target audience is farmers and growers with varying levels of technical expertise.
- The website's design should cater to users who may not be familiar with advanced technological concepts.

5.3 Technical Requirements:

- Researched and selected appropriate front-end and back-end (if applicable) technologies considering factors like scalability, maintainability, and target user device compatibility.
- In this project, the front-end utilizes HTML, CSS, and Javascript to deliver an interactive user experience.

5.4 Design and Development

- User Interface (UI) Design:
  - Developed wireframes and mockups to visualize the website's layout and user interaction flow.
  - Prioritized a clean and intuitive UI with clear navigation menus and sections for easy access to information.
  - Employed visual design principles to create an aesthetically pleasing and user-friendly interface.
  - Used color palettes and fonts that evoke feelings of trust, reliability, and growth, aligning with the agricultural domain.

5.5 Information Architecture (IA):

- Designed a sitemap outlining the website's content structure and organization.
- Content is categorized logically and intuitively for users to find information effortlessly.
- Create dedicated sections for:
  - Home: Introduce PhytoGuard's purpose and key features.
  - About Us: Provide background information on the founding team and the motivation behind the project.
  - Services: Clearly explain the website's functionalities, including image upload, disease analysis, and result display.
  - Contact Us: Offer multiple contact channels for user inquiries and feedback.
  - Owner: Showcase the founding team and their vision for PhytoGuard.

5.6 Development Process:

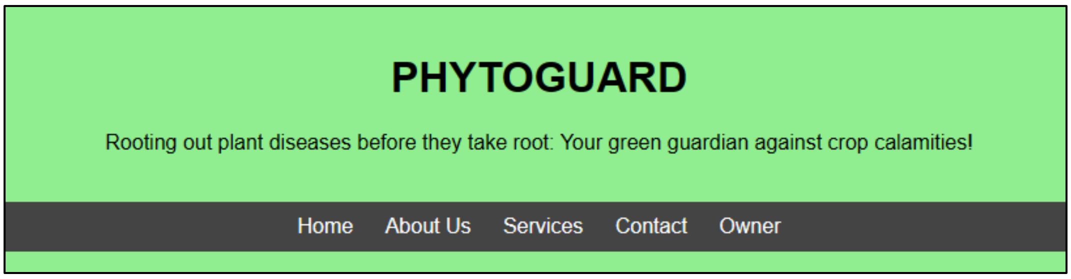
- Front-End Development:
  - Develop the website's front-end using HTML to structure the content and layout.

    - Implement CSS to style the visual elements, ensuring a consistent look and feel across all pages.
    - Utilize Javascript to add interactivity and dynamic functionality. This may include:
    - Image upload functionality using HTML file input elements and Javascript event handling.
    - "toggleDetails" function using Javascript to display additional service details upon user interaction.

- Back-End Development:
    - Flask is used in the backend of the website.

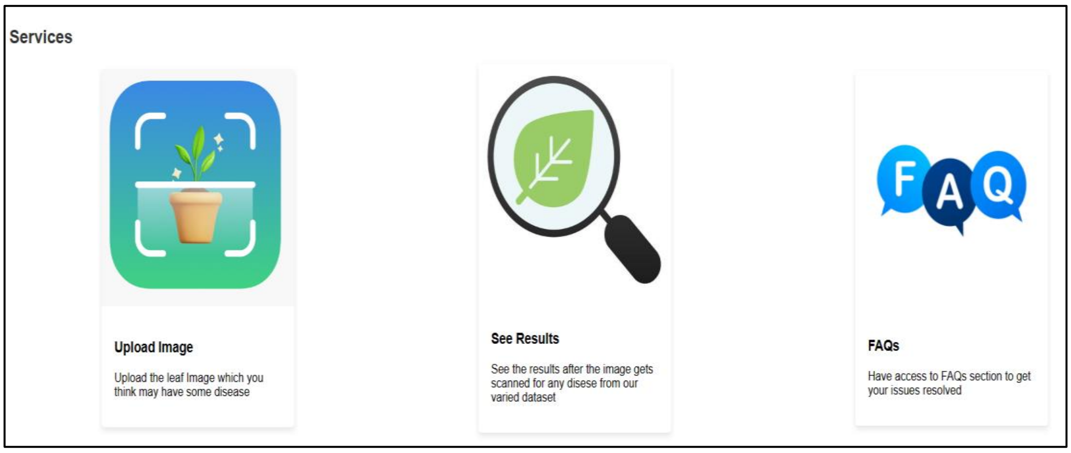
5.7 Testing and Deployment

- Responsiveness Testing:
    - Tested the website's responsiveness across various devices (desktops, tablets, and smartphones) to guarantee optimal user experience on all screen sizes.
    - Ensured proper layout adjustments and font size scaling for different screen resolutions.
- Deployment:
    - Deployed the website on a web hosting platform to make it accessible online.
    - Configured server settings and ensure all website functionalities operate correctly in the live environment. And it does!



PHYTOGUARD

Rooting out plant diseases before they take root: Your green guardian against crop calamities!



Services

Upload Image

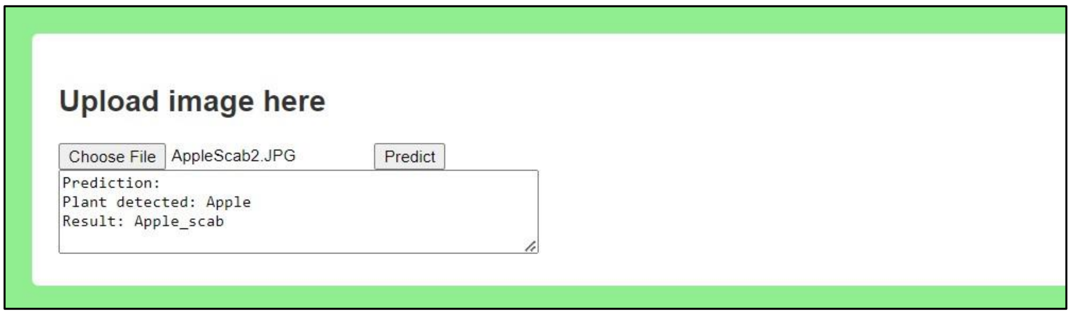
Upload the leaf Image which you think may have some disease

See Results

See the results after the image gets scanned for any disese from our varied dataset

FAQs

Have access to FAQs section to get your issues resolved



Upload image here

```
Prediction:
Plant detected: Apple
Result: Apple_scab
```

These are the snapshots of the deployed website.

5.8 Ongoing Maintenance and Updates

- Regular maintenance:
    - Schedule periodic maintenance checks to identify and address any bugs or performance issues that may occur.

6. Conclusion

The successful completion of this project has demonstrated the effectiveness of utilizing deep learning techniques for plant disease detection. The PhytoGuard website has a simple and user-friendly interface for easy navigation. Though, for now it’s only usable facility is the detection whether the leaf whose image is uploaded by the user is diseased (name of disease) or healthy.

However, there remains ample room for improvement and expansion. The addition of more data into the dataset could enhance the model's accuracy and generalization capabilities. Furthermore, ongoing advancements in algorithms offer opportunities for refining the used model and incorporating new techniques as they emerge.

In terms of the web application, enhancing its interactivity by providing comprehensive knowledge about plant diseases, preventive measures, and resources could greatly benefit users. Incorporating information about cropping patterns tailored to specific locations, soil properties, and weather conditions could further enhance its utility for farmers.

Moreover, making the project accessible to a broader audience could foster collaboration and innovation, allowing others to build upon the work done and contribute to the advancement of agricultural technologies. By continuously updating and refining the model and web application, the team can contribute to the development of sustainable agricultural practices and address the challenges posed by plant diseases effectively.

7. References

1. Mohanty SP, Hughes DP and Salathé M (2016) Using Deep Learning for Image-Based Plant Disease Detection. Front. Plant Sci. 7:1419. doi: 10.3389/fpls.2016.01419
2. Sanika Singh and Saurabh Mukherjee (January 2024) Automated Detection of Plant Leaf Diseases using Image Processing Techniques DOI:10.21203/rs.3.rs-3859987/v1
3. Halil Durmuş; Ece Olcay Güneş; Mürvet Kırcı, " Disease detection on the leaves of the tomato plants by using deep learning", 2017 6th International Conference on Agro-Geoinformatics, 2017, doi:10.1109/Agro-Geoinformatics.2017.8047016.
4. Suja Radha, Chatterjee J.M., Jhanjhi N.Z., Brohi Sarfraz, "Performance of deep learning vs machine learning in plant leaf disease detection". Microprocessors and Microsystems 80(6): 103615. February, 2021. DOI:10.1016/j.micpro.2020.103615
5. Karthik R., Hariharan M., Anand S., Mathikshara P., Johnson A., Menaka R. , "Attention embedded residual CNN for disease detection in tomato leaves". Appl. Soft Comput. 2019, 86, 105933.
6. Shalya Saxena and Sandeep Rathor, "An Ensemble-Based Model of Detecting Plant disease using CNN and Random Forest", 2023 6th

International Conference on Information Systems and Computer Networks, 2023, doi: 10.1109/ISCON57294.2023.10112023.

7. Dubey S.R., Jalal A.S., "Adapted Approach for Fruit Disease Identification using Images. In Image Processing: Concepts, Methodologies, Tools, and Applications"; IGI Global: Hershey, PA, USA, 2013; pp. 1395–1409.
8. Li L., Zhang S., Wang B., "Plant Disease Detection and Classification by Deep Learning—A Review". IEEE Access 2021, 9, 56683–56698.
9. J. G. A. Barbedo, "Factors influencing the use of deep learning for plant disease recognition", Biosyst. Eng., vol. 172, pp. 84-91, Aug. 2018.
10. A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey", Comput. Electron. Agricult., vol. 147, pp. 70-90, Apr. 2018.
11. M. H. Saleem, J. Potgieter and K. M. Arif, "Plant disease detection and classification by deep learning", Plants, vol. 8, no. 11, pp. 468-489, Oct. 2019.
12. Anshul Tripathi, Uday Chourasia, Priyanka Dixit, Victor Chang. (2022, January). " Plant Disease Detection Using Sequential Convolutional Neural Network". International Journal of Distributed Systems and Technologies. vol. 13, issue 1.
13. Ghosh H, Rahat IS, Shaik K, Khasim S, Yesubabu M. Potato Leaf Disease Recognition and Prediction using Convolutional Neural Networks. EAI Endorsed Scal Inf Syst [Internet]. 2023 Sep. 21 https://doi.org/10.4108/eetsis.3937.
14. G Sucharitha, M Sirisha, K Pravalika, K. Navya Gayathri. A Study on the Performance of Deep Learning Models for Leaf Disease Detection. EAI Endorsed Transactions on Internet of Things [Internet]. 2023, December DOI:10.4108/eetiot.4592
15. G. Shrestha, Deepsikha, M. Das and N. Dey, "Plant Disease Detection Using CNN," 2020 IEEE Applied Signal Processing Conference (ASPCON), 2020, pp. 109-113, doi: 10.1109/ASPCON49795.2020.9276722.
16. Jolly Masih, Rajasekaran Rajkumar, Abhijit Bhagawat, Kapil Rokade, Dr R Senthamil Selvan, "Using Deep Convolutional Neural Network for plant

leaf disease detection”, October 2023, DOI: 10.14704/nq.2022.20.8.NQ44884

17. Punam Bedi, Pushkar Gole, “Plant Disease Detection Using Hybrid Model based on convolutional autoencoder and convolutional neural network”, Artificial Intelligence in Agriculture, vol. 5, pp 90-101, 2021
18. Konstantinos P. Ferentinos, “Deep learning models for plant disease detection and diagnosis”, Computers and Electronics in Agriculture, vol 145, pp 311-318, February 2018
19. Srdjan Sladojevic et al., "Deep neural networks based recognition of plant diseases by leaf image classification", Computational intelligence and neuroscience, 2016.