Plant Disease Prediction using Convolutional Neural Network

Thesis Submitted
In Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
Software Engineering
by

SUNIL KUMAR
(23/SWE/14)

Under the Supervision of
Dr. Abhilasha Sharma
(Associate Professor, SE, DTU)

To the
Department of Software Engineering
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

May, 2025

ACKNOWLEDGEMENT

I wish to express my sincerest gratitude to Dr. Abhilasha Sharma for her continuous guidance and mentorship that she provided me during the project. She showed me the path to achieve my targets by explaining all the tasks to be done and explained to me the importance of this project as well as its industrial relevance. She was always ready to help me and clear my doubts regarding any hurdles in this project. Without her constant support and motivation, this project would not have been successful.

Place: Delhi

Date: 27/05/25

Sunil Kumar

Sunil Kumar

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CANDIDATE DECLARATION

I SUNIL KUMAR (23/SWE/14) hereby certify that the work which is being presented in the thesis entitled **"Plant Disease Prediction using Convolutional Neural Network"** in partial fulfillment of the requirements for the award of the Degree of Master of Technology submitted in the Department of Software Engineering, Delhi Technological University in an authentic record of my work carried out during the period from August 2023 to May 2025 under the supervision of Dr. Abhilasha Sharma.

The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Sunil kumar
Candidate's Signature

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.

[illegible]
Signature of Supervisor

[illegible]
Signature of External Examiner

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CERTIFICATE

Certified that Sunil Kumar (23/SWE/14) has carried out their project work presented in this thesis entitled "Plant Disease Prediction using Convolutional Neural Network" for the award of **Master of Technology** from the Department of Software Engineering, Delhi Technological University, Delhi under my supervision. The thesis embodies results of original work, and studies are carried out by the student himself 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 from this or any other University/Institution.

[illegible]

Dr. Abhilasha Sharma
Associate Professor
Department of Software Engineering,
DTU-Delhi, India

Date: 27/05/25

ABSTRACT

Identification of plant diseases at an early stage is critical to forever maintaining agricultural productivity as well as ensuring food security. Often times relying on a manual inspection, traditional methods of diagnostics can be slow and inaccurate as well. This research proposes a new automated method with the use of Convolution Neural Networks (CNNs) to diagnose and classify plant diseases with the aid of their leaves images. This model uses a vast dataset consisting of different crops as well as different diseases. The model that is being proposed, CNN, is capable of feature extraction making it easier for the model to recognize the disease. After sufficient training and validation, the model can distinguish and tell with great precision whether the plant is healthy or infested with some form of disease. This technology can easily be integrated in mobile systems so that it can be made available to farmers and this can help them in real time diagnosis which enables them to make timely decisions and prevent losses. This study shows how Deep Learning can be evolutionary for agriculture with available and effective measures for managing the health of plants.

TABLE OF CONTENT

LIST OF TABLE(S)

LIST OF FIGURE(S)

LIST OF ABBREVIATION(S)

PCA	Principal Component Analysis
CNN	Convolutional Neural Network
CV	Computer Vision
NLP	Natural Language Processing
AI	Artificial Intelligence
ML	Machine Learning
DNN	Deep Neural Network
ReLU	Rectified Linear Unit
GPU	Graphics Processing Unit
IP	Image Processing
SVM	Support Vector Machine
DL	Deep Learning

CHAPTER - 1

INTRODUCTION

1.1 Background

Plant diseases become a really big problem to the world nowadays. Caused by a wide variety of pathogens—including nematodes, bacteria, viruses, and phytoplasmas—these diseases can have adverse effect to crop yield and production, quality, and profitability[17]. The increasing demand for food due to population growth has raised the need for effective disease management strategies because crop losses caused by various plant diseases can have adverse effects on the economy and society. Plant diseases have had catastrophic consequences.

The degree of complexity in plant-pathogen interactions is affected by different environmental aspects, how susceptible plants are and how viruses have changed over time. Doing monoculture and intense farming on a large scale usually decreases the variety of genes among plants and provides conditions for pathogens to thrive.

The strategies employed to combat plant diseases include chemical treatments, biological control, integrated pest management, and the development of crop that are disease- resistant of various varieties through conventional breeding and biotechnological methods. Because pathogen evolution is dynamic, plant pathologists must continuously innovate and conduct research to develop effective and long-lasting disease management strategies.

Understanding the processes underlying the emergence and spread of plant diseases is essential to creating resilient agricultural systems. This thesis aims to further this understanding and have an impact on future methods of managing and preventing plant diseases.

1.2 Problem Statement

Plant diseases pose a serious threat to global agricultural production, food security, and economic stability. Numerous pathogens, such as nematodes, bacteria, viruses, and fungi, can cause these diseases, which can lead to large yield losses and reduced crop quality. The challenge is exacerbated by monoculture practices, climate change, and the increasing resistance of infections to conventional chemical treatments[18]. Owing to long and demanding processes, as well as the need for trained professionals, it remains difficult to spot plant diseases as early and accurately as needed. Although new technologies in agriculture are being used, it is still necessary to find better ways to diagnose and treat illnesses. No solution to this problem means crops would be less capable of growing and strong agricultural practices would be less effective.

1.3 Objectives

Tree Disease Classification is focused on developing an automated method for diagnosing and categorizing plant diseases by means of CNNs. To reach this aim, deep learning methods are used to make early disease diagnoses so crop losses are minimized. To enhance plant health monitoring in farming, the model will be trained and tested using information shared publicly.

1.4 Motivation

Many economies depend on agriculture which is important for supplying the global cereals requirement. Alternatively, one of the biggest problems for farmers and agronomists is the presence of plant diseases[3]. These diseases may produce reduced quality of crops, create financial concerns or cause major losses to farmers. Using traditional techniques to discover diseases is a tiring process because tollens st insects is often required and can produce errors. As a result of these concerns, more automatic, reliable and fast systems for plant disease prediction are needed. New advances in machine learning and computer vision and remote sensing can provide us with the solution to this problem. These technologies allow for the scanning of huge amounts of agriculture data and offer related information to farmers and disease diagnosis at a quick pace. These systems, supplemented by them, use fewer chemicals in agriculture because personalized actions can be undertaken. It is this lag between traditional agriculture practices and current technology that this study seeks to bridge. A successful plant disease model that predicts disease will be able to assist farmers, increase harvests and help secure global food safety as population sizes grow and the climate continues to change both the size of people and the development of climate change.

CHAPTER - 2

LITERATURE REVIEW

In the past decade, Convolutional Neural Networks (CNNs) have been a potential approach towards detection of plant diseases from image analysis with potential accuracy and automation improvement. Mohanty et al.'s [1] pioneering research showed how deep CNN architectures can be learned for detection of various crop diseases from publicly available sets of leaf images, a breakthrough in agricultural diagnostics. To this end, Lu et al. [2] designed an automatic diagnosis system for wheat diseases that could be implemented in the field directly, demonstrating the feasibility of applying CNN models in the field.

There has been increasing interest in the creation of lean and effective CNN models that may be deployed on devices with limited power. Hassan et al. [4] and Ali et al. [5] introduced shallow CNNs that minimize computational overhead but are also effective in disease classification. Furthermore, Zhang et al. [6] effectively transferred deep CNNs into apple leaf disease classification, showing the flexibility of CNNs for different crops. To supplement this work, Shruthi et al. [8] performed their tests on low-resolution leaf images and proved that CNNs can retain strong disease detection ability even when given poor-quality input data.

In order to preserve spatial and temporal information of plant disease, hybrid CNN-LSTM models have been proposed. Kumar et al. [7] and Pan [13] proved that the combination of CNNs and LSTMs enhances model robustness through summarization of sequential data, which is appropriate for time-evolving diseases. Furthermore, residual connections and attention mechanisms, as discussed by Tang et al. [3] and Thakur et al. [9], improve feature extraction and contribute to improved classification outcomes and increased model interpretability.

Explainable AI has also been an area of study for plant disease detection. Prince et al. [14] integrated visualization methods with CNN-SVM hybrid models, incorporating transparency into the process of decision making and enabling trust by users in automated systems. Transfer learning approaches, such as those of Singh et al. [12], use pre-trained models to enhance performance on small, specialized plant disease datasets with less training time and data needs. Efforts to develop smartphone-compatible solutions for farmers in rural regions have also been a research area. Gupta et al. [10] and Bobde et al. [11] used CNN models on smartphones for real-time disease diagnosis on widely available phones. Patel et al. [15] also enhanced classification performance with multi-scale CNN structures, which learn features across multiple granularities for improved accuracy in distinguishing diseases.

Lastly, the extensive review of Sankhe and Ambhaikar [16] including advancements in detecting plant diseases discusses issues and challenges such as model scaling and different datasets.

Authors	**Model/Technique**	**Dataset/Application**	**Key Contribution/Findings**
Mohanty et al. [1]	Deep CNN	Public crop leaf images	High-precision multi-class plant disease classification
Lu et al. [2]	CNN-based system	In-field wheat images	Mechanized wheat disease diagnosis in real field
Tang et al. [3]	Depthwise CNN + SE + Residual	Leaf images	Enhanced feature extraction with attention mechanisms
Hassan et al. [4]	Shallow CNN	Plant leaf images	Lightweight, efficient plant disease identification
Ali et al. [5]	CNN	Various plant diseases	CNN model for classification of plant diseases
Zhang et al. [6]	Deep CNN	Apple leaf images	Special attention paid to apple leaf disease classification
Kumar et al. [7]	CNN-LSTM hybrid	Real-time plant disease data	Combines spatial and temporal features for prediction
Shruthi et al. [8]	Deep CNN	Low-resolution leaf images	Robust disease detection on low-quality images
Thakur et al. [9]	Vision Transformer + CNN	Leaf disease datasets	Explainable AI integration for model transparency
Gupta et al. [10]	Optimized CNN for smartphones	Smartphone leaf images	Mobile real-time disease detection
Bobde et al. [11]	CNN	Various crop images	Real-time disease detection models
Singh et al. [12]	Transfer Learning	Limited plant datasets	Effective use of

	CNN		pretrained models for disease ID
Pan [13]	CNN-LSTM hybrid	Leaf disease images	Improved classification with combined CNN-LSTM
Prince et al. [14]	CNN-SVM + Explainable AI	Crop disease images	Explainability with AI visualization methods
Patel et al. [15]	Multi-scale CNN	Plant disease images	Multi-scale feature extraction for better accuracy
Sankhe & Ambhaikar [16]	Literature review	Various	Comprehensive overview of plant disease detection

These works mark an interesting progression from simple object classification based on CNNs, to more complex, interpretable models which are ready to meet the demanding needs of modern agriculture.

CHAPTER - 3

METHODOLOGY

This chapter outlines the systematic procedure for developing and evaluating a predictive model for plant disease detection. Among the most important steps in the methodology are data collection, preprocessing, model building, training and validation, and performance assessment. Every step is carefully planned to ensure the accuracy, reliability, and generalizability of the prediction model.

3.1 Methodology Flow Chart

The figure[17] given below tells us the workflow, which elaborates how to identify plant leaf diseases.

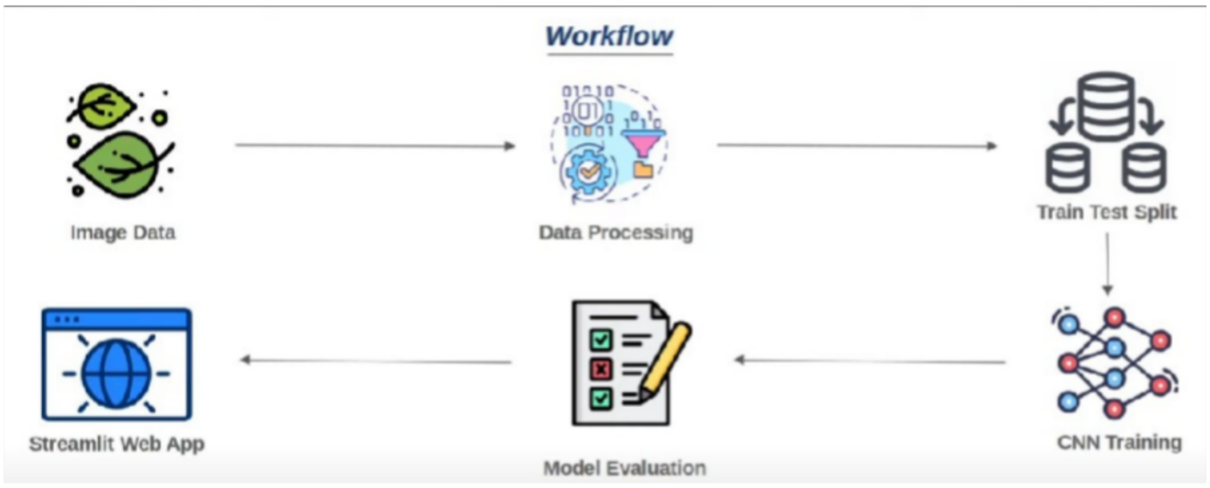


Fig 3.1 : Workflow Chart for Plant Disease Prediction[17]

3.2 Data Collection

Computational models are trained on data to identify patterns that differentiate healthy plants from diseased ones[4]. Large volumes of labeled data are necessary for these models, especially those that use deep learning and artificial intelligence, to achieve high accuracy in real-world situations. Prediction systems may experience biases, overfitting, or poor generalizability in the absence of adequate or representative data.

3.3 Data Preprocessing

Data preprocessing is a vital phase in any ML pipeline, especially when employing CNNs for image-based operation such as plant disease detection. The structure and quality of the data getting input directly influence the generalization, performance and accuracy and capabilities of the model getting trained[5]. In this context, preprocessing involves preparing raw leaf images to be compatible with the CNN architecture while enhancing relevant features and minimizing noise.

3.3.1 Image Acquisition and Annotation

The very first step in data preprocessing involves collecting images that are high-quality and contains plant leaves affected by various diseases, as well as healthy specimens. These images are either captured under controlled conditions or sourced from publicly available datasets such as PlantVillage. Each image must be correctly annotated with its corresponding label, indicating the type of disease or its absence[15]. Proper annotation ensures that the CNN learns meaningful patterns during supervised training.

3.3.2 Image Resizing and Normalization

CNNs need images with a specific dimension as input. As a result, every image is resized to a standard resolution, such as 256 x 256 pixels or 224 x 224 pixels. The dataset's consistency and compatibility with the selected CNN architecture—such as VGGNet, ResNet, or EfficientNet—are guaranteed by this resizing[2]. Pixel values are normalized after resizing; this is usually done by scaling them to a range of 0 to 1 or by standardizing them using the mean and standard deviation of the dataset. This normalization avoids numerical instability and speeds up convergence during training.

3.3.3 Data Augmentation

Model robustness and chances of overfitting decrease when techniques for data augmentation are applied. They involve moving, rotating, zipping, brightening, making darker and random flipping. To achieve diversity, the existing data can be manipulated with augmentation; this means no further manual data collection is needed. Plus, this training allows the model to better adapt to various lighting, environments and angles found in everyday images.

3.3.4 Noise Removal and Background Filtering

Some image datasets have leaves that looks distorted, but these images do not correspond to disease classification[8]. These tools make it easier to separate the leaf from the rest of the picture. At this point, the CNN pays attention to lesions, spots or any discoloration in the leaf to make more accurate predictions.

3.4 Dataset Split

The way data is collected and organized impacts CNN models used for predicting plant diseases. It is important to divide a dataset into test and training sets[15]. You should divide the data into separate parts for the model's suitable training, validation and testing. In most cases, data is sorted into three sets: training, validation and testing.

Usually, 70–80% of data is used for training, 10%–15% used for validation and the rest 10%–15% is used for testing. The training set trains the CNN and backpropagation updates the CNN's internal weights. During training, the validation set allows reviewing the model's capabilities and changing the hyperparameters to stop the model from overfitting. The purpose of testing is to judge if the model will be able to deal with new information, as seen in the testing frame.

CHAPTER - 4

ARCHITECTURE AND TRAINING OF CNN MODEL

Convolutional Neural Network (CNN) Architecture

Several key factors can be attributed to the emergence of CNNs in the world of image classification, especially as it concerns the identification of plant diseases. To begin with, CNNs are essentially a type of CNN that is engineered to process images in a way that emulates the manner in which the visual system of a human (or other animals) works[5]. These networks are then layered upon one another in such a way that the first few layers capture very simple components of an image (for example, edges and colors) and the deeper layers capture very complex, high- level combinations of those simple components (for example, when recognizing a whole object, like a plant, or a whole scene, like a field with many plants).

In addition, convolutional neural networks employ convolutional layers to automatically derive features from images, which eliminate the necessity for manual feature engineering. This automation not only accelerates the training process but also boosts the model's power to generalize across various datasets. Consequently, CNNs have outperformed classical ML methods across diverse plant disease classification tasks[9]. Additionally, with large annotated data sets becoming available and the advance in computing power, the adoption here of CNNs has come a long way. We can now train sophisticated models on large data sets and attain an unprecedented level of accuracy and reliability for disease detection systems previously unimaginable a few years ago.

Summing it all up, the success of CNNs in detecting plant disease hinges on their ability to learn abstract visual patterns, eliminate the need for hand-crafted feature extraction, and benefit from enormous databases, making them a treasure trove in agri-tech.

4.1 CNN MODEL ARCHITECTURE

In crop disease prediction, CNNs have served as an effective DL framework for image-based classification problems. Leaf image analysis is particularly successful using CNN models to detect the symptoms of various plant diseases since they learn and extract hierarchical features from raw image data on their own[6]. A standard CNN model for plant disease prediction consists of a number of convolutional layers, followed by activation functions (i.e., ReLU) and another layer named pooling layers utilized to minimize spatial dimensions while maintaining important information. The layers assist the model in identifying patterns such as dots, color shift, and texture shift that represent some plant problems. The linked (dense) layers that take up this information then perform higher-order thinking and classification.

So that the model should perform better on new data and should not overlearn the training set, individuals tend to utilize such stunts as dropout and batch normalization[12]. The final output layer employs a softmax activation function to score the probability of various types of diseases.Having trained on a large dataset of labeled plant images makes this architecture very accurate and robust when deployed in real-world agricultural settings, and it is therefore an indispensable tool for sustainable crop care and early disease detection.

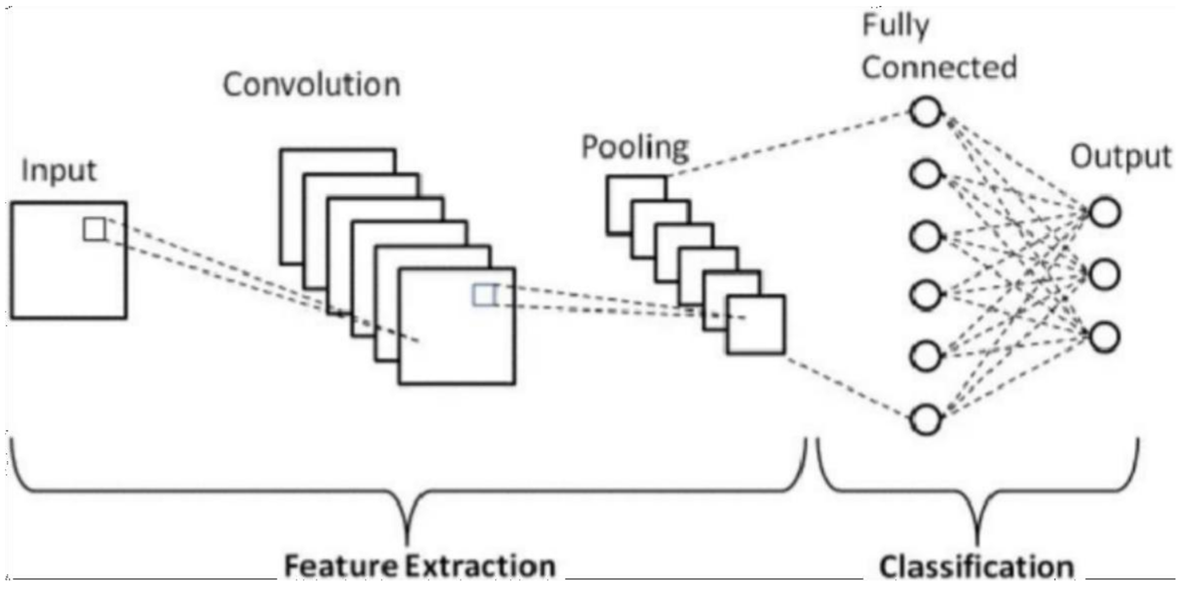


Fig 4.1 : Convolutional Neural Networks Architecture[18]

4.1.1 Input Layer

CNNs play a very crucial role in predicting plant disease in DL due to their tremendous image processing potential[9]. At the center of any CNN is the input layer, and it serves as the first where images are fed in and it serves as a bridge between the neural network model and the raw image data.

The input layer needs to be where visual data of crops or plant leaves that are typically RGB photographs. The photographs often have notable characteristics that describe the condition of disease, such as color change, texture, spots, or lesions.

In a CNN technique rightly used in vision processing, CNNs tend to be resized into the same dimensions(e.g., 224x224x3). The first two digits are height and width of image, and the third digit is the number of color channels(Red, Green, and Blue).This standardization not only ensures computational efficiency but also helps maintain consistency across the dataset, which is essential for creating trustworthy models[13]. At this stage, pixel values are often standardized to a range of 0 to 1 so that the convergence get improve and model performance.Image condition variations such as lighting, angle, and cluttered background should be regulated by the input layer in order to forecast plant diseases. Preprocessing methods such as contrast and noise reduction enhancement, and removal of background, are most common prior to the image presentation at the input layer in order to facilitate feature extraction in later layers.

4.1.2 Convolutional Layer

CNNs have become a vital part of various image-based classification applications, including the vital field of plant disease prediction.
The convolutional layer, which is central to the design of a CNN, is quite simply the most important component when it comes to extracting vital characteristics from input images of crops or plant leaves. In essence, the job of the convolutional layer is to convolve the input image with several different filters. Optimally, these filters should be learned during the training process.

The primary work of convolutional layer is to apply a set of learnable filters, sometimes called kernels, that move spatially over the input image[6]. Each filter performs convolution operations, creating feature maps that capture regional patterns such as edges, textures, and shapes. Each filter is doing convolution operations and generating feature maps that detect local features like edges, texture, and shapes.

The regional features are responsible for separating healthy and diseased plants, which are prone communicate themselves as small changes in color, spots, or deformats of the plant.manifest as slight changes in the color, spots, or deformities of the plant.

Operation in mathematics[18]:

- For an input image I and a filter F, the convolution output at position (i,j) is:

$$O(i,j) = \sum_{m} \sum_{n} I(i+m, j+n) \cdot F(m,n)$$

Fig 4.2 : Mathematical Representation of the 2D Convolution Operation[18]

4.1.3 Activation Function (ReLU)

Activation functions are simply the type of mathematical functions which is applied to output layer of each neuron to decide whether the following neuron should be activated or not[21]. It enables CNNs to perform non-linear transformations if activation functions are not there then CNNs would not able to non linear transformations and it would not able to find or learn complex patterns and relationships among the data.

ReLU is a type of activation functions which is widely used due to its simplicity and efficiency. ReLU is a piecewise linear function that output 0 in case of negative values and in case of positive values it gives the exact values as it is.

Mathematically ReLU functions[20] can defined as:

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

This means:

- If $x > 0$, then $\mathrm{ReLU}(x) = x$
- If $x \leq 0$, then $\mathrm{ReLU}(x) = 0$

Fig 4.3 : Formula for ReLU function[20]

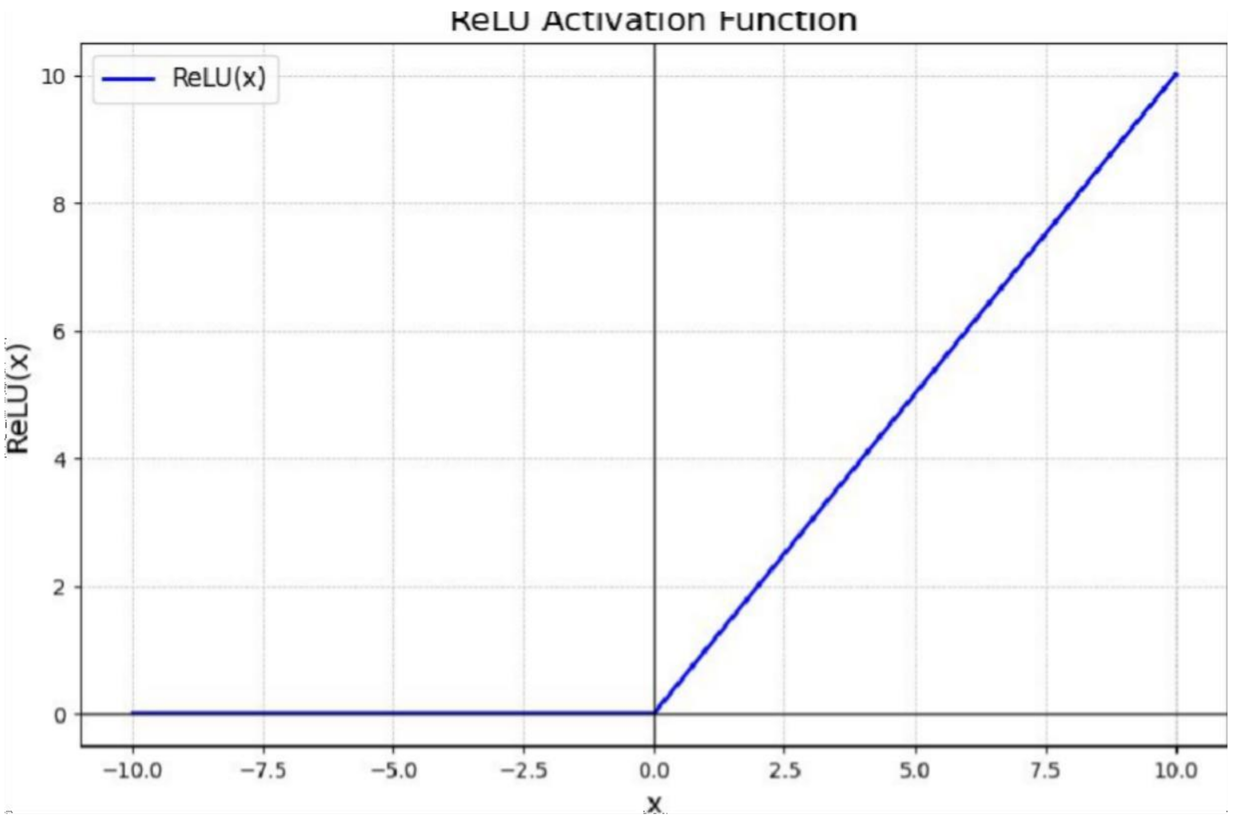


Fig 4.4 : ReLU function Graphical Representation[20]

4.1.4 Pooling Layer

Pooling layer is a type of layer used in CNNs which is introduced in CNNs to decrease the dimensions of the incoming feature maps while retaining the vital attributes of the feature maps. It helps us to reducing the computation and controlling the overfitting. Because overfitting is one of the main issue in CNNs models[10].

Pooling can be of two types:

Max pooling[21] is a type of pooling which selects the maximum value from the given region. Therefore, the output of max pooling layer would contains the most vital info of the previous feature map.

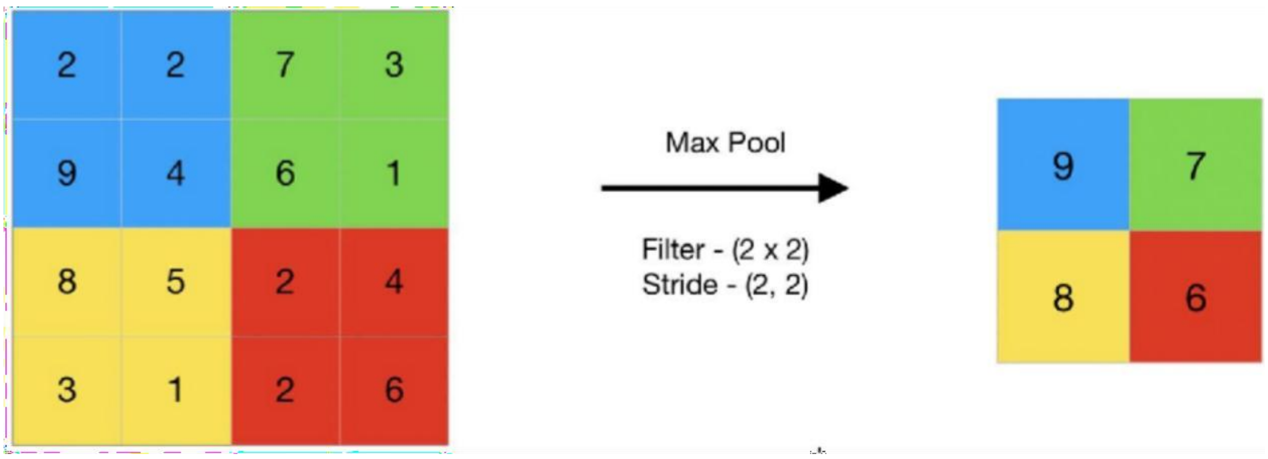


Fig 4.5 : Max Pooling Layer[21]

While a average pooling[21] would selects the average value from the region.
Therefore, the average pooling gives us the average features of the map and it is mostly used when we want to preserve the overall context of the feature map.

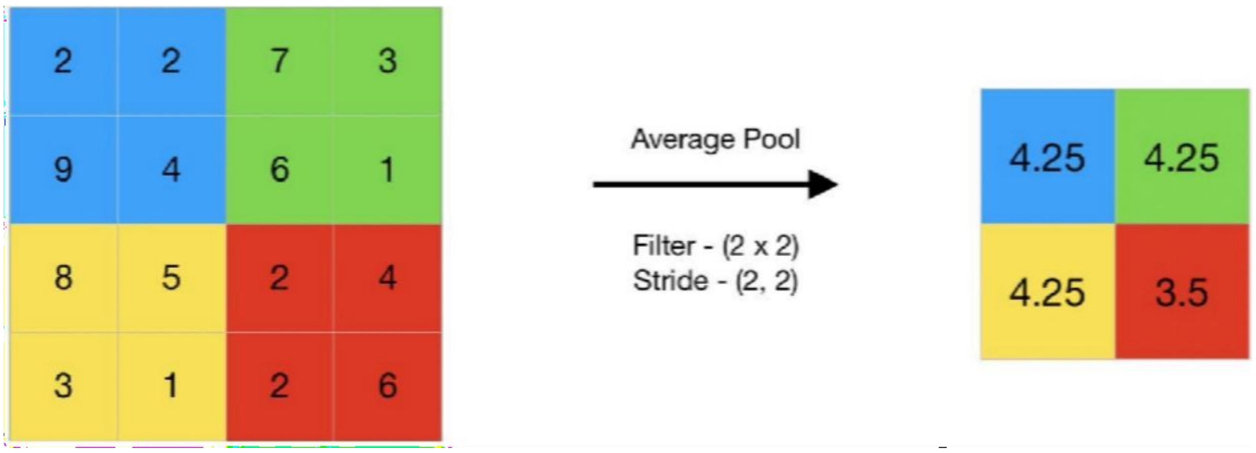


Fig 4.6 : Average Pooling Layer[21]

4.1.5 Dropout Layer (Regularization)

Regularization is a set of techniques used to improve the generalization ability of a CNN. When a CNN becomes too complex or is trained for too long, it can start to memorize training data rather than learn general patterns[11]. This can result in overfitting — a situation where the model performs well on training data but on unseen data ir performs poorly . Regularization helps to reduce this risk.

Dropout is a regularization technique introduced to prevent overfitting in neural networks. The primary idea of dropout is to randomly deactivate a subset of neurons during the training process.

Specifically, at each training step, a fraction of the neurons is "dropped" (i.e., their outputs are set to zero) according to a predefined dropout rate (commonly between 0.2 and 0.5)[9]. This introduces redundancy in the network and forces the remaining neurons to learn more robust and generalized features.

4.1.6 Flattening Layer

In a CNN, the flattening layer serves as a crucial bridge between the convolutional/pooling layers and the fully connected (dense) layers that follow.

What is Flattening?

Flattening is the process of tranforming a multi-dimensional tensor (usually the output of convolutionlayer and pooling layers) into a one-dimensional tensor. This transformation is required because fully connected layers—like those in traditional artificial neural networks—expect input in the form of a 1D array[20].

How It Works

Suppose you have an output from the final pooling layer shaped as (height, width, channels)—for example, (4, 4, 64). The flattening layer will convert this into a 1D array of size $4 \times 4 \times 64 = 1024$. This flattened vector is then fed into the dense layer(s) for further processing and final classification or regression.

Why It’s Important

- **Transition Layer**: It connects feature extraction layers (like Conv and Pooling) to the classification or prediction part of the network.
- **Preserves Learned Features**: It retains all the spatial features learned in the earlier layers and prepares them for decision-making.
- **Essential for Classification**: Many tasks like image classification, object detection, and facial recognition require this step to process the extracted features into output labels.

4.1.7 Fully Connected (Dense) Layers

In Convolutional Neural Networks (CNNs), fully connected layers[18]—also called as dense layers— which are are typically found close to the end of the architecture. These layers play a crucial role in making final predictions after the spatial features have been extracted by convolutional and pooling layers.

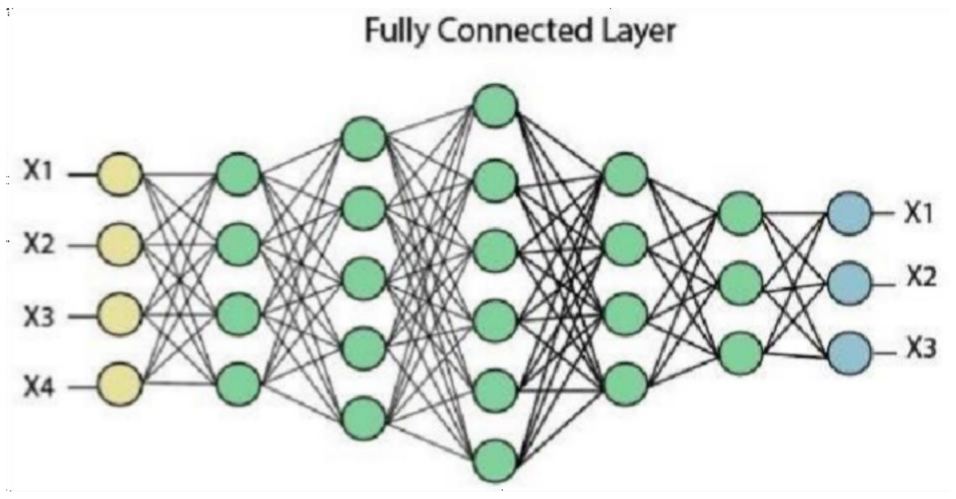


Fig 4.7 : Fully Connected (Dense) Layers[18]

4.1.8 Output Layer

The layer contains as many neurons as the number of classes (types of plant diseases + healthy class). For example, if there are 5 plant diseases and 1 healthy class, the output layer will have 6 neurons.

Each neuron gives the probability to which input image belongs to that class.

Softmax Function

It is mathematical tool commonly used in the output layer of a Convolutional Neural Network (CNN), especially when solving classification problems[1]. Its primary role is to convert the raw output scores (also called logits) from the network into probabilities that sum up to 1. This makes it easier to interpret the model’s prediction as a probability distribution over different classes.

Formula[19]:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Fig 4.8 : Formula for Softmax function[19]

4.1.9 Loss Function and Optimizer

A loss function is a type of functions that quantifies the difference between the predicted output of the CNN and the actual label[12]. It serves as the objective the model tries to minimize during training. A lower loss means better predictions.

An optimizer can be defined as algorithm that updates the parameters (weights and biases) of a neural network based on the gradients calculated during backpropagation.

Adam Optimizer

The Adam optimizer is widely is used to train CNNs due to efficiency and using it become easy. It has the power of two other extensions of stochastic gradient descent[15]. AdaGrad and RMSProp. Here's how the Adam optimizer works:

- Every parameter has two moving averages maintained:
  - The initial moment (mean) of the gradients (m).
  - The second instant (uncentered variance) (v) of the gradients.
- During training, these moving averages are updated using exponential decay rates.
- Bias correction is applied to account for initialization at zero.
- These corrected averages are then used to update the parameters, changing the learning rate for each one.

4.2 Training Process

Step 1: Data Collection and Preprocessing

The very first step in training a CNN for plant disease prediction is to collect a sizable collection of plant leaf images that show both healthy and unhealthy leaves[6]. These images are often taken from publicly available datasets or samples collected in the field. It is important that data should be preprocess so that the data is uniform and enhance model training and performance. All images are resized to a standard dimension that is suitable for the CNN input. Normalization of pixel values is done to a specified scale. Occasionally, the data is improved with transformations such as flips, rotations, or color changes. This is done to augment the dataset and introduce diversity into the training set.

Step 2: Model Building

Next after data pre-processing comes creation of the CNN model for plant disease classification. The architecture consists of a large number of convolutional layers, which each serve as a type of spatial data extractor in the manner of borders, textures and color patterns that indicate the presence of possible diseases[2]. After each of the above layers is a pooling layer that reduces the dimension further retaining the features that can help in improving the efficiency of computation.. The images are then finally clustered into relevant categories though the cropped image.. A model’s capability of recognizing sophisticated patterns is impacted by amount of layers, filter sizes, and activation functions selected.

Step 3: Model Compiling

As any other model training, a CNN model for predicting plant diseases needs appropriate loss functions, optimizers, evaluation metrics, and classifying methods to ensure proper learning and classification accuracy[8]. For disease classification in particular, loss functions of categorical cross- entropy are used where Adam optimizers are preferred due to their flexible learning abilities.

Step 4: Model Training

Both sick and healthy plants are annotated with images and these are passed through several stages of convolution and pooling in order to train a CNN model for feature detection. Based on the relevant spatial properties defined by these layers, fully linked layers classify the input images into several categories of diseases.

The model can be optimized by minimizing a loss function through techniques such as gradient descent and backpropagation. Through iterative training on large, varied datasets, the CNN improves its accuracy in identifying a range of plant diseases, making it a useful tool for early diagnosis and control in agriculture.

CHAPTER - 5

SOFTWARE IMPLEMENTATION

In this part, we elaborate the algorithmic process and related code to implement the application.

5.1 Algorithm Process

Step. 1 – Importing the required libraries.

Step. 2 – Setting various input parameters.

Step. 3 – Importing the dataset from Kaggle and preprocess it

Step. 4 – Building the CNN model architecture and compiling it.

Step. 5 - Training the built model at different epochs.

Step. 6 – Evaluation of the model on various aspects

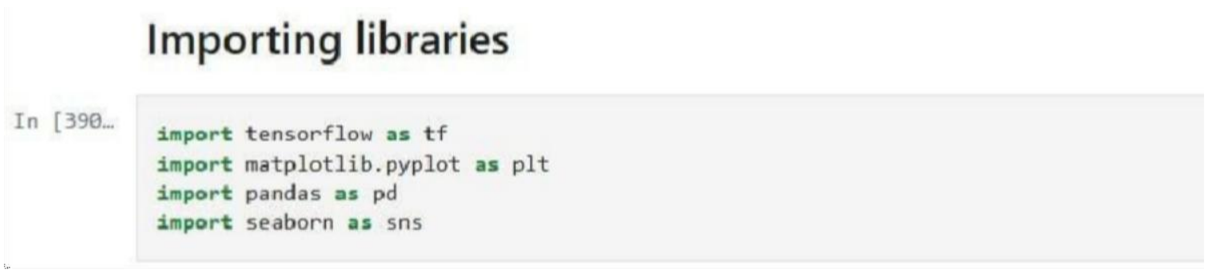
Step. 7 – Test the model for various model.

5.2 Code Implementation:

5.2.1 Importing the necessary libraries like TensorFlow, pandas

TensorFlow is a framework which is developed by Google Brain and it is a open- source meaning everyone can use to develop applications. It is used to train machine learning especially deep learning models. It supports various type of applications like NLP, CV, etc.

Pandas is a library developed by python which is used widely for data processing in the field of data science, machine learning, it provides us with various data structures to make things smooth.



Importing libraries

In [390…

```
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Fig 5.1 Importing Libraries

5.2.2 Setting input parameters

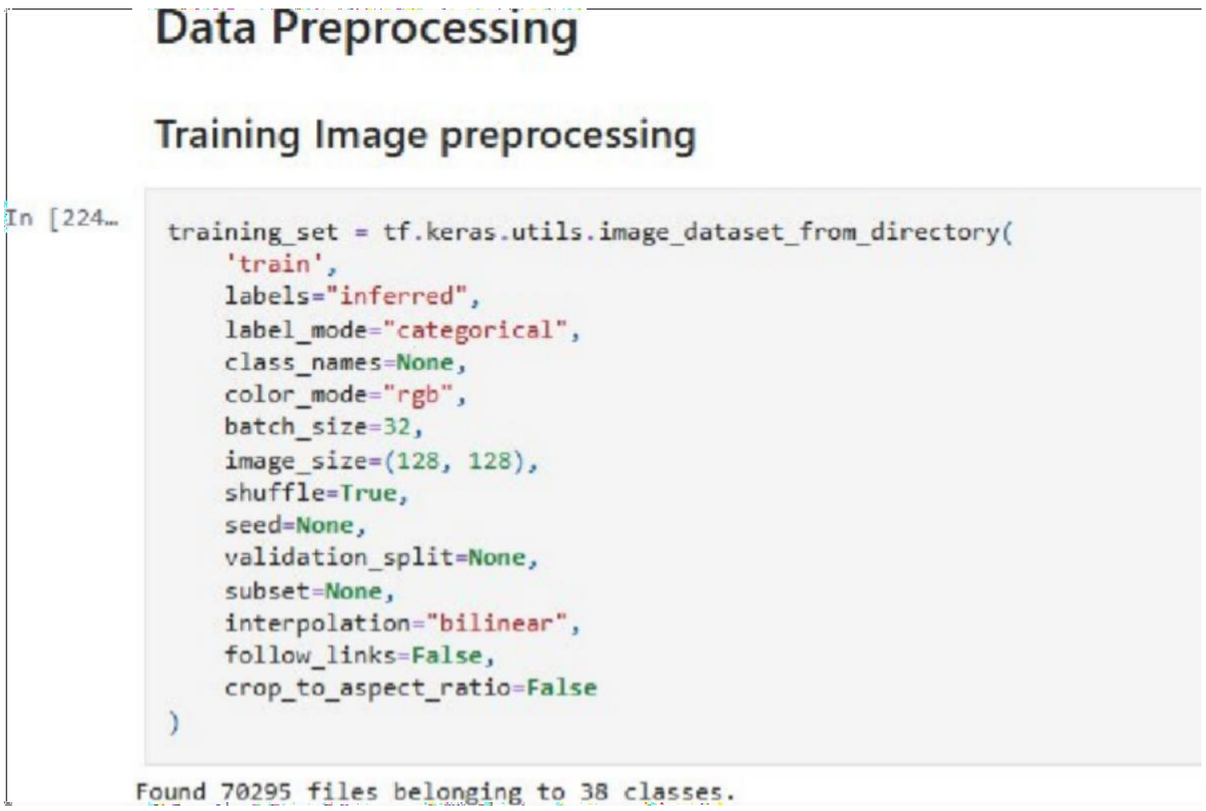
Image Resizing can be defined as the resizing of the image as per the requirements so that the CNN model could be trained more accurately[18].

Batch size is parameter which tells us about the numbers of training records would goes in one iteration of backward and forward pass of a Neural Network.

Number of Classes can be defined as the number of groups which is available for classification in a dataset.

5.2.3 Loading and preprocessing dataset

The dataset that we are using is from Kaggle. This dataset contains about 87K images of various plants leaf, it contains various types of images like greyscale, rgb, etc. It is augmented using offline augmentation. It contains 38 different classes and the data split is 80/20 ratio[15].



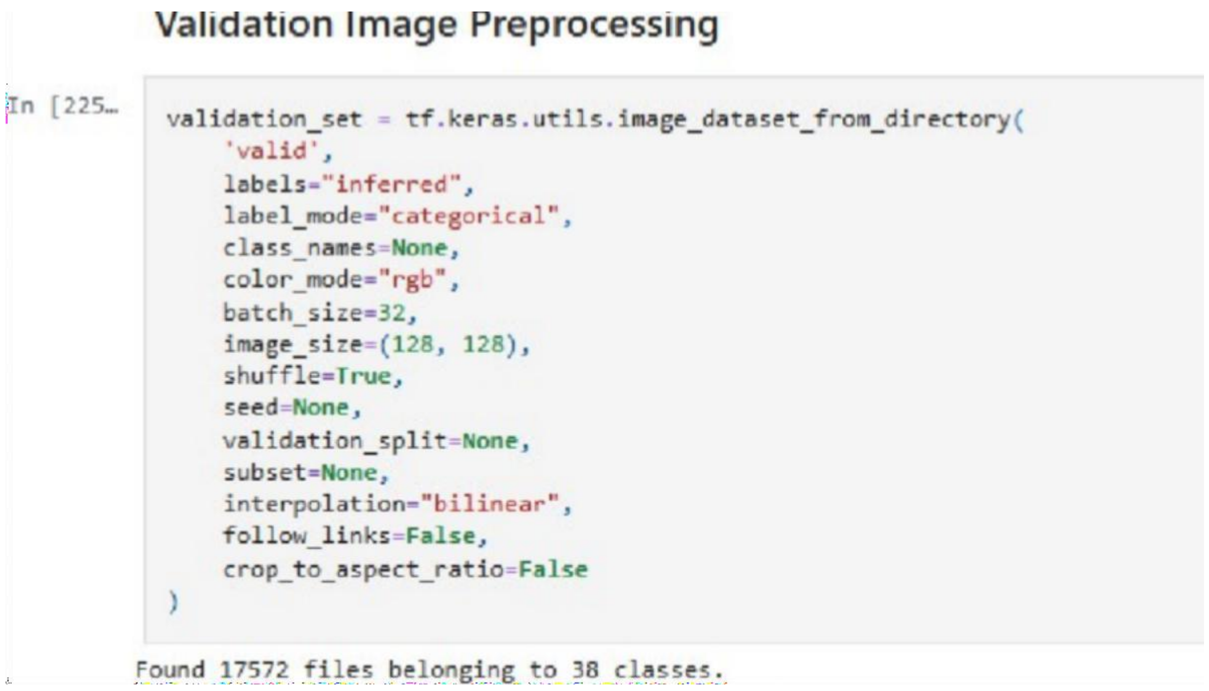
Data Preprocessing

Training Image preprocessing

```
In [224…
training_set = tf.keras.utils.image_dataset_from_directory(
    'train',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
Found 70295 files belonging to 38 classes.
```

Fig 5.2 : Training Image Preprocessing


Validation Image Preprocessing

In [225…

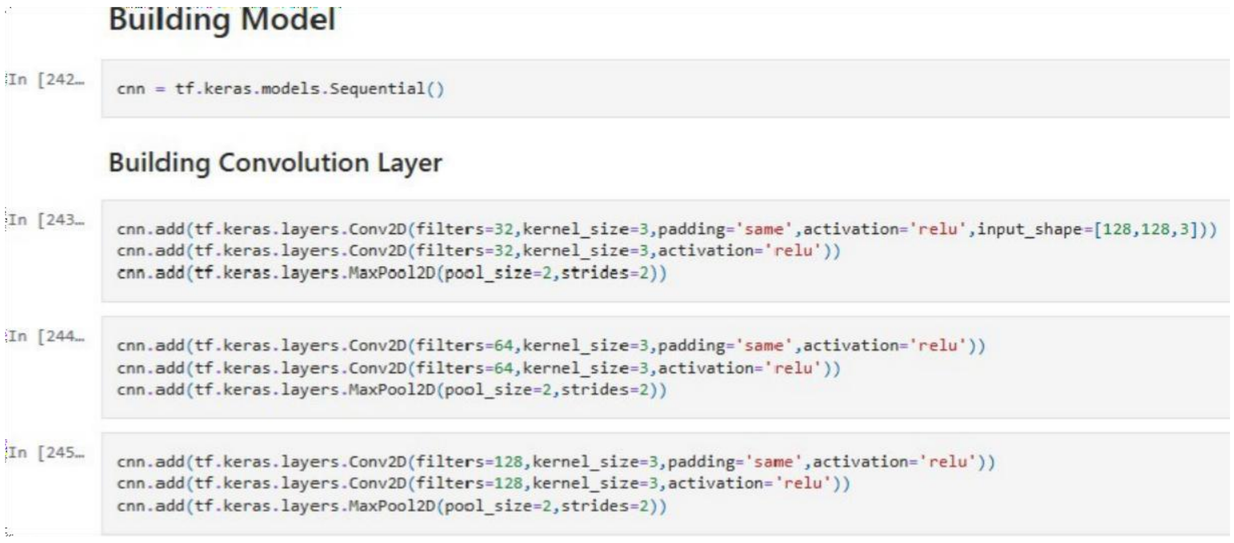
```
validation_set = tf.keras.utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
Found 17572 files belonging to 38 classes.
```

Fig 5.3 : Validation Image Preprocessing

5.2.4 Define the CNN model architecture.

CNN is essentially a kind of DNN that have been engineered to process images in a way that emulates the manner in which the visual system of a human (or other animals) works[19]. These networks are then layered upon one another in such a way that the first few layers capture very simple components of an image (for example, edges and colors) and the deeper layers capture very complex, high- level combinations of those simple components (for example, when recognizing a whole object, like a plant, or a whole scene, like a field with many plants).


Building Model

```
In [242…
cnn = tf.keras.models.Sequential()
```

Building Convolution Layer

```
In [243…
cnn.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,padding='same',activation='relu',input_shape=[128,128,3]))
cnn.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
```

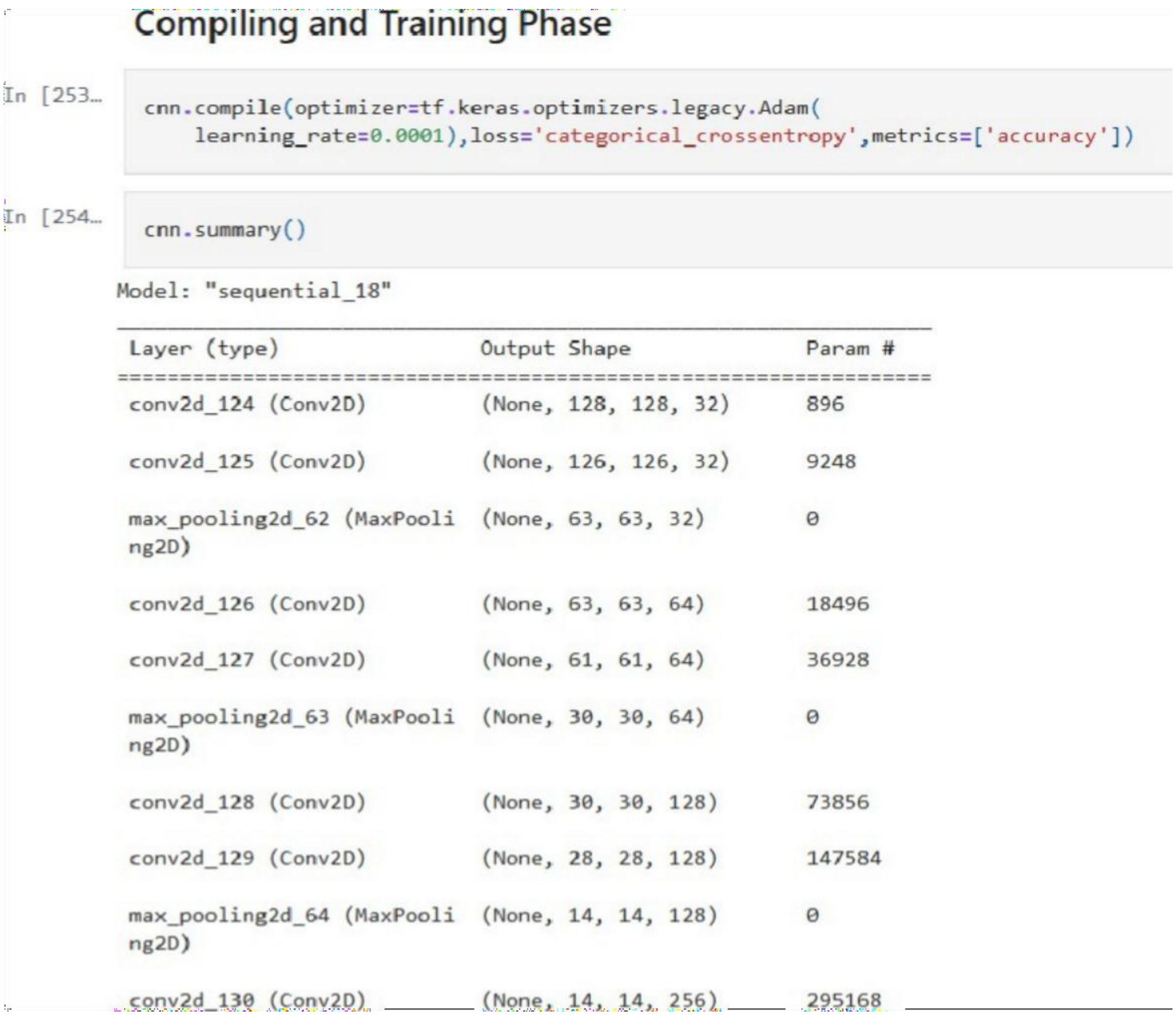
```
In [244…
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding='same',activation='relu'))
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
```

```
In [245…
cnn.add(tf.keras.layers.Conv2D(filters=128,kernel_size=3,padding='same',activation='relu'))
cnn.add(tf.keras.layers.Conv2D(filters=128,kernel_size=3,activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
```

Fig 5.4 : Define the CNN Model Architecture

5.2.5 Training the CNN model at different epochs.

Training a CNN model plays a important role in prediction of images. So, in this step we train the model for various epochs.



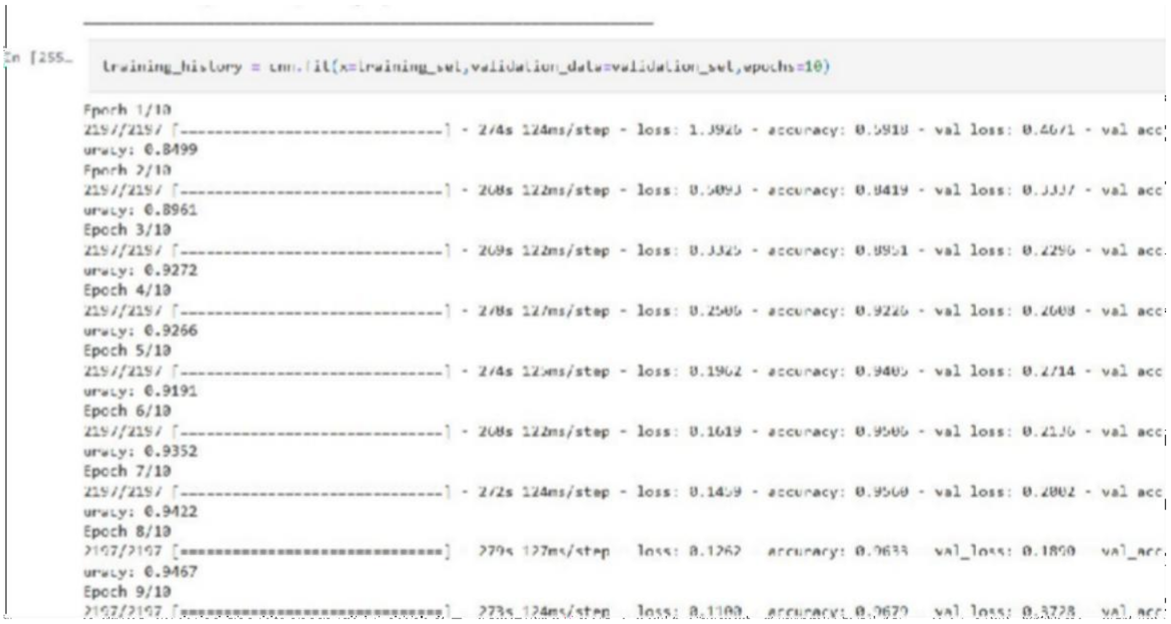
Compiling and Training Phase

```
In [253…  cnn.compile(optimizer=tf.keras.optimizers.legacy.Adam(
              learning_rate=0.0001),loss='categorical_crossentropy',metrics=['accuracy'])
```

```
In [254…  cnn.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_124 (Conv2D)         (None, 128, 128, 32)      896

 conv2d_125 (Conv2D)         (None, 126, 126, 32)      9248

 max_pooling2d_62 (MaxPooli  (None, 63, 63, 32)        0
 ng2D)

 conv2d_126 (Conv2D)         (None, 63, 63, 64)        18496

 conv2d_127 (Conv2D)         (None, 61, 61, 64)        36928

 max_pooling2d_63 (MaxPooli  (None, 30, 30, 64)        0
 ng2D)

 conv2d_128 (Conv2D)         (None, 30, 30, 128)       73856

 conv2d_129 (Conv2D)         (None, 28, 28, 128)       147584

 max_pooling2d_64 (MaxPooli  (None, 14, 14, 128)       0
 ng2D)

 conv2d_130 (Conv2D)         (None, 14, 14, 256)       295168
```

Fig 5.5 : Compiling the CNN Model

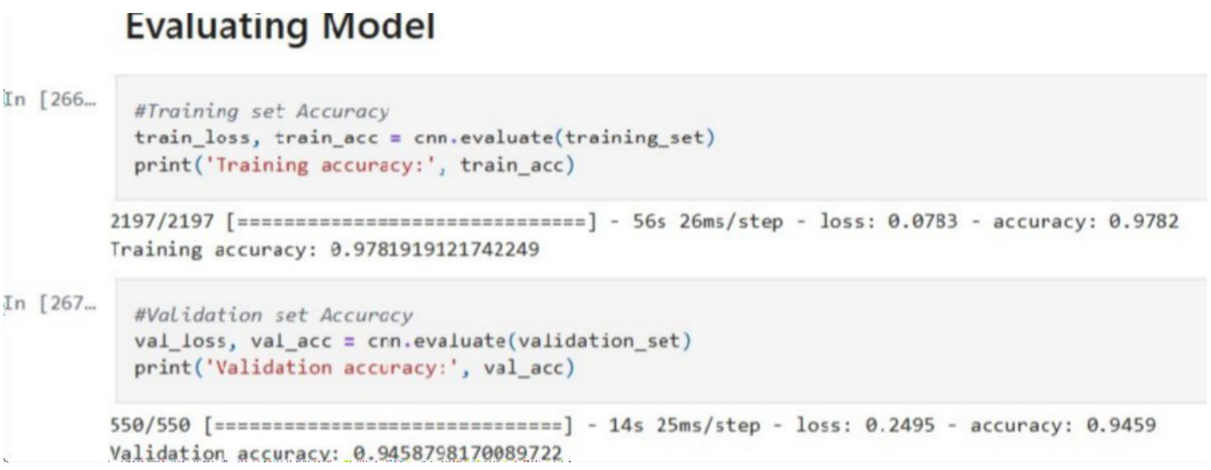


```
In [255…  training_history = cnn.fit(x=training_set,validation_data=validation_set,epochs=10)

Epoch 1/10
2197/2197 [==============================] - 274s 124ms/step - loss: 1.3926 - accuracy: 0.5918 - val_loss: 0.4671 - val_acc
uracy: 0.8499
Epoch 2/10
2197/2197 [==============================] - 268s 122ms/step - loss: 0.5093 - accuracy: 0.8419 - val_loss: 0.3337 - val_acc
uracy: 0.8961
Epoch 3/10
2197/2197 [==============================] - 269s 122ms/step - loss: 0.3325 - accuracy: 0.8951 - val_loss: 0.2296 - val_acc
uracy: 0.9272
Epoch 4/10
2197/2197 [==============================] - 278s 127ms/step - loss: 0.2506 - accuracy: 0.9226 - val_loss: 0.2608 - val_acc
uracy: 0.9266
Epoch 5/10
2197/2197 [==============================] - 274s 125ms/step - loss: 0.1962 - accuracy: 0.9405 - val_loss: 0.2714 - val_acc
uracy: 0.9191
Epoch 6/10
2197/2197 [==============================] - 268s 122ms/step - loss: 0.1619 - accuracy: 0.9506 - val_loss: 0.2136 - val_acc
uracy: 0.9352
Epoch 7/10
2197/2197 [==============================] - 272s 124ms/step - loss: 0.1459 - accuracy: 0.9560 - val_loss: 0.2002 - val_acc
uracy: 0.9422
Epoch 8/10
2197/2197 [==============================] - 279s 127ms/step - loss: 0.1262 - accuracy: 0.9633 - val_loss: 0.1890 - val_acc
uracy: 0.9467
Epoch 9/10
2197/2197 [==============================] - 273s 124ms/step - loss: 0.1100 - accuracy: 0.9679 - val_loss: 0.3728 - val_acc
```

Fig 5.6 : Training Model for Different Epochs

5.2.6 Evaluation of the model on various aspects

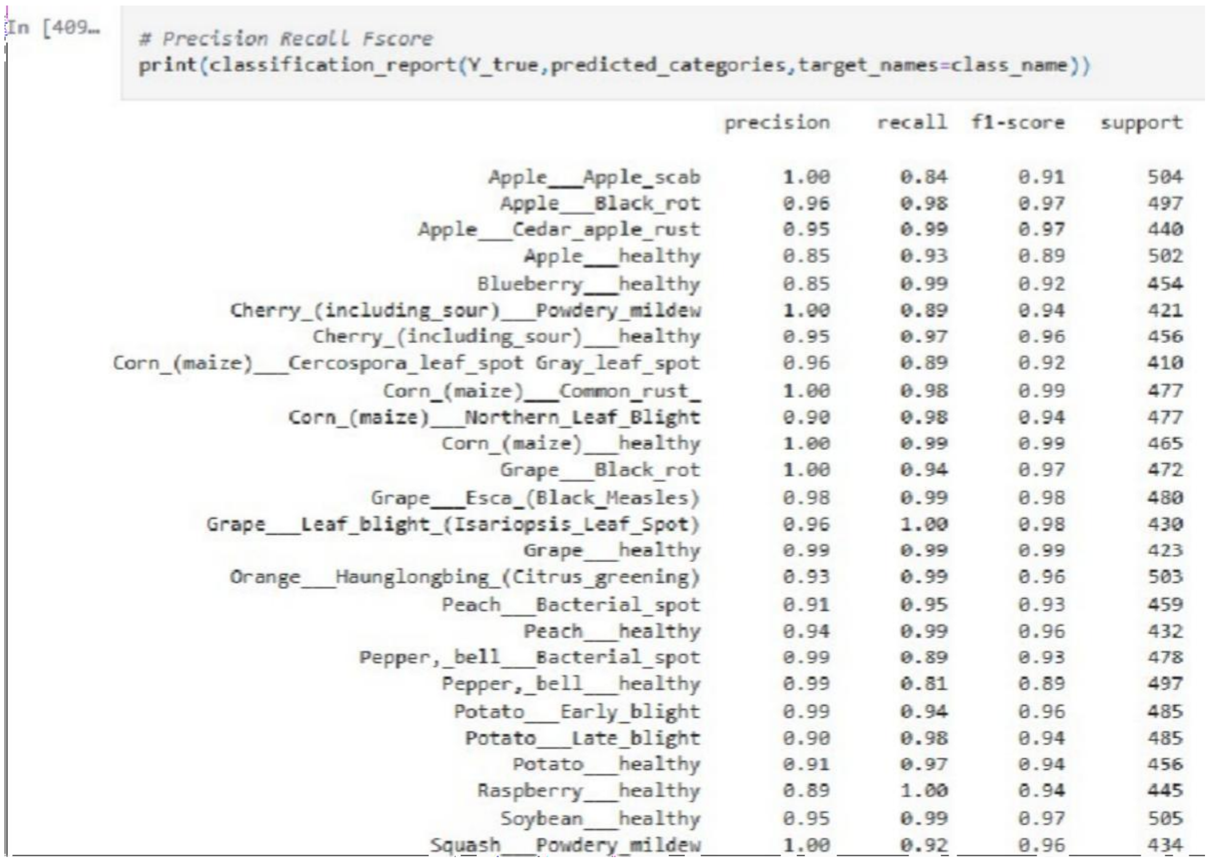
In this step evaluation of the model on various aspects is performed



Evaluating Model

```
In [266…
#Training set Accuracy
train_loss, train_acc = cnn.evaluate(training_set)
print('Training accuracy:', train_acc)

2197/2197 [==============================] - 56s 26ms/step - loss: 0.0783 - accuracy: 0.9782
Training accuracy: 0.9781919121742249

In [267…
#Validation set Accuracy
val_loss, val_acc = cnn.evaluate(validation_set)
print('Validation accuracy:', val_acc)

550/550 [==============================] - 14s 25ms/step - loss: 0.2495 - accuracy: 0.9459
Validation accuracy: 0.9458798170089722
```

Fig 5.7 : Evaluating model for different accuracy

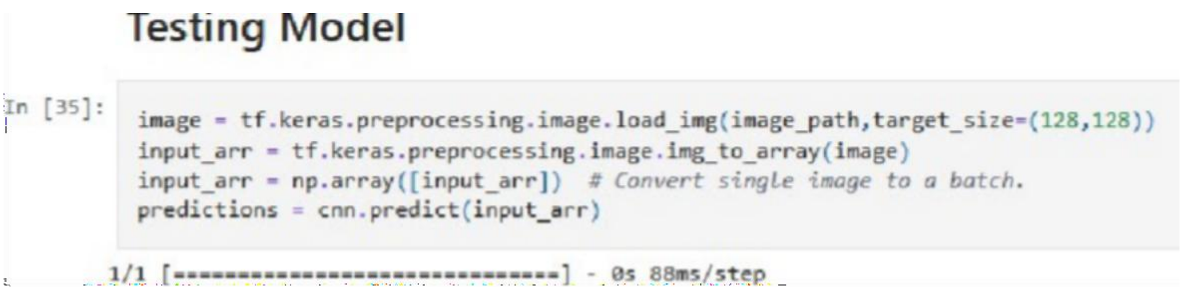


```
In [409…
# Precision Recall Fscore
print(classification_report(Y_true,predicted_categories,target_names=class_name))

                                                      precision    recall  f1-score   support

                                Apple___Apple_scab         1.00      0.84      0.91       504
                                 Apple___Black_rot         0.96      0.98      0.97       497
                          Apple___Cedar_apple_rust         0.95      0.99      0.97       440
                                   Apple___healthy         0.85      0.93      0.89       502
                               Blueberry___healthy         0.85      0.99      0.92       454
          Cherry_(including_sour)___Powdery_mildew         1.00      0.89      0.94       421
                 Cherry_(including_sour)___healthy         0.95      0.97      0.96       456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot         0.96      0.89      0.92       410
                       Corn_(maize)___Common_rust_         1.00      0.98      0.99       477
               Corn_(maize)___Northern_Leaf_Blight         0.90      0.98      0.94       477
                            Corn_(maize)___healthy         1.00      0.99      0.99       465
                                 Grape___Black_rot         1.00      0.94      0.97       472
                      Grape___Esca_(Black_Measles)         0.98      0.99      0.98       480
        Grape___Leaf_blight_(Isariopsis_Leaf_Spot)         0.96      1.00      0.98       430
                                   Grape___healthy         0.99      0.99      0.99       423
          Orange___Haunglongbing_(Citrus_greening)         0.93      0.99      0.96       503
                            Peach___Bacterial_spot         0.91      0.95      0.93       459
                                   Peach___healthy         0.94      0.99      0.96       432
                     Pepper,_bell___Bacterial_spot         0.99      0.89      0.93       478
                            Pepper,_bell___healthy         0.99      0.81      0.89       497
                             Potato___Early_blight         0.99      0.94      0.96       485
                              Potato___Late_blight         0.90      0.98      0.94       485
                                  Potato___healthy         0.91      0.97      0.94       456
                               Raspberry___healthy         0.89      1.00      0.94       445
                                 Soybean___healthy         0.95      0.99      0.97       505
                           Squash___Powdery_mildew         1.00      0.92      0.96       434
```

Fig 5.8 : Precision, Recall, Fscore

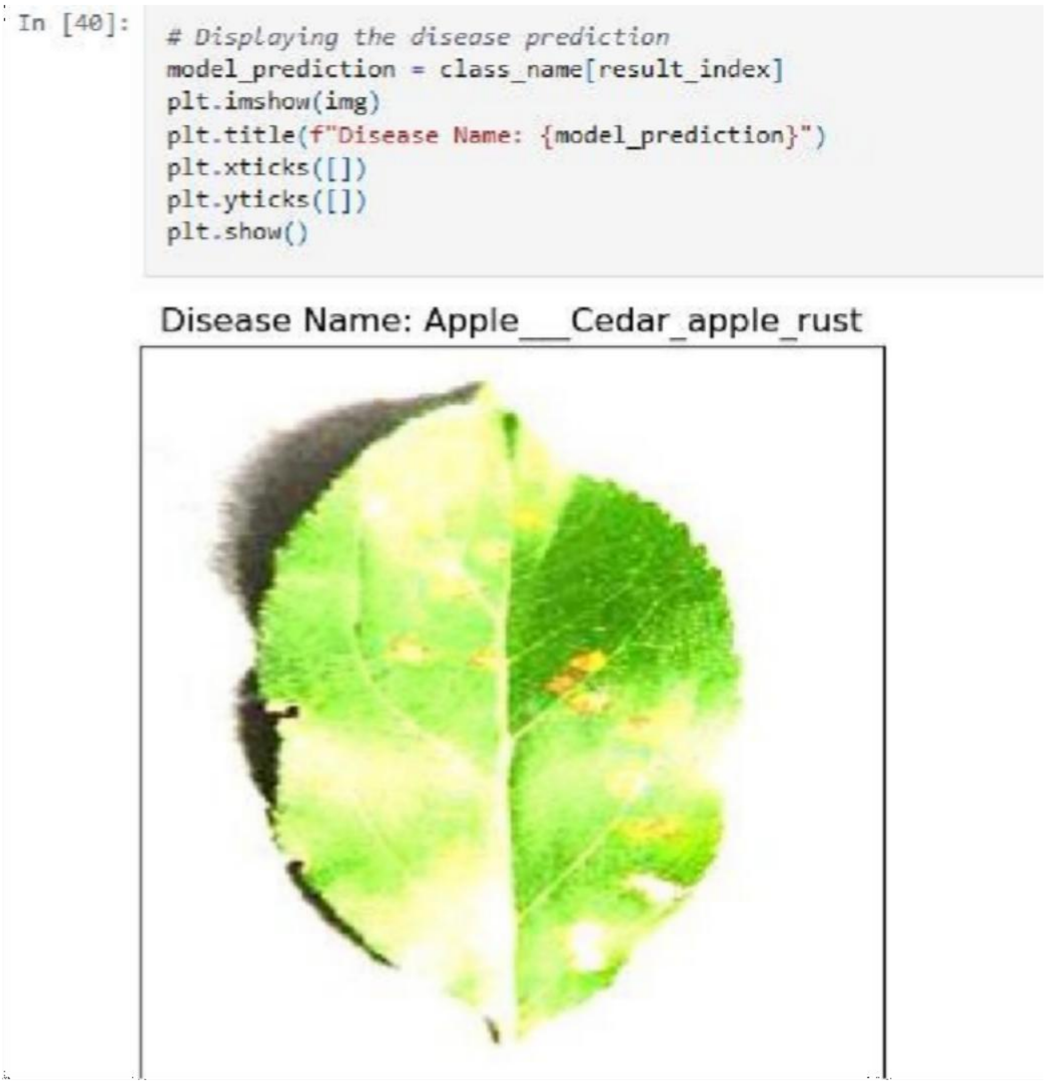
5.2.7 Testing the image for a sample prediction

In this step we give sample image to the model and test whether the model is predicting the result right or not



Testing Model

```
In [35]:
image = tf.keras.preprocessing.image.load_img(image_path,target_size=(128,128))
input_arr = tf.keras.preprocessing.image.img_to_array(image)
input_arr = np.array([input_arr])  # Convert single image to a batch.
predictions = cnn.predict(input_arr)

1/1 [==============================] - 0s 88ms/step
```



```
In [40]:
# Displaying the disease prediction
model_prediction = class_name[result_index]
plt.imshow(img)
plt.title(f"Disease Name: {model_prediction}")
plt.xticks([])
plt.yticks([])
plt.show()
```

Fig 5.9 Testing the Model[20]

CHAPTER - 6

VISUALIZATION

Visualization is one of most vital aspects of model training and predictive analysis. So, in this chapter we are going to visualize the results.

6.1 Visualization

The primary objective of this study to detect various plant disease by analyzing the leaves of the plants. The data that we have used is collected by various means and it contains various types of plants leaves images. It contains 38 types of different classes[21]. Every image is labeled with healthy and unhealthy leaves along with the diseases.

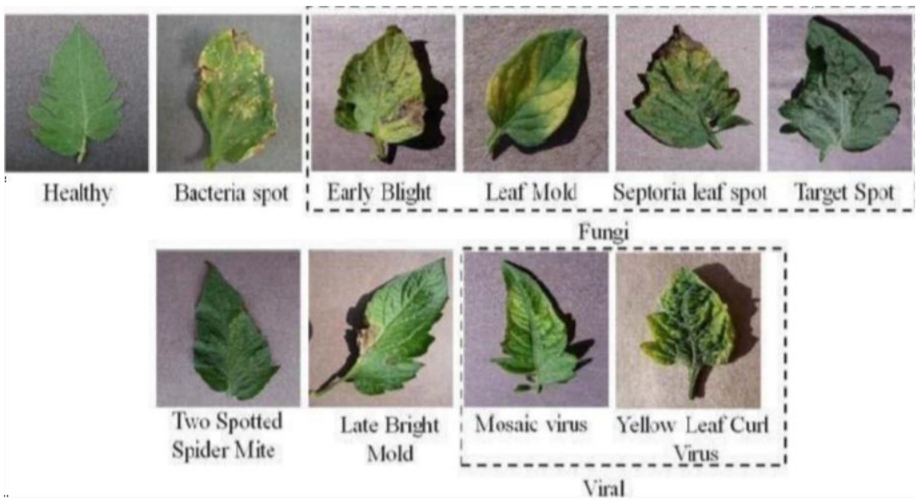


Fig 6.1 : Types of Diseases in Plant[21]

6.2 Training Results

In this section we are going to check the results of our model that how accurately the images get predicted along with confidence level.

It includes the calculation of train loss and train accuracy metrics during training phase of the model[16]. The loss metric is the measure how much accurately the model is performing on the training data. It is the difference between the actual output and predicted output. Our goal is to minimize the loss function.

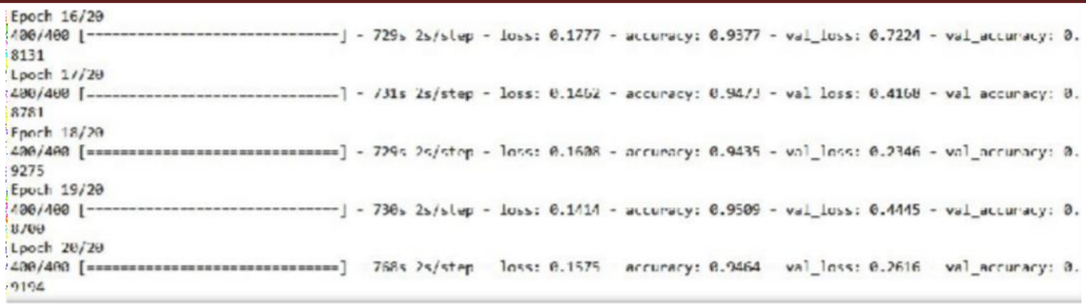
CNN model is trained for various epochs like 10, 20, 50. Its result are given in the below table as we increase the epochs, we get the loss getting decreasing and accuracy getting increased.

The table given below tells us about loss and accuracy for train and valid at various epochs

S. No	**No. of epochs**	**Train Accuracy**	**Train Loss**	**Valid Accuracy**	**Valid Loss**
1	10	0.65	0.35	0.62	0.46
2	20	0.94	0.16	0.92	0.27
3	50	0.96	0.09	0.96	0.15

Table 6.1: Valid Accuracy and Valid Loss for Different Epochs

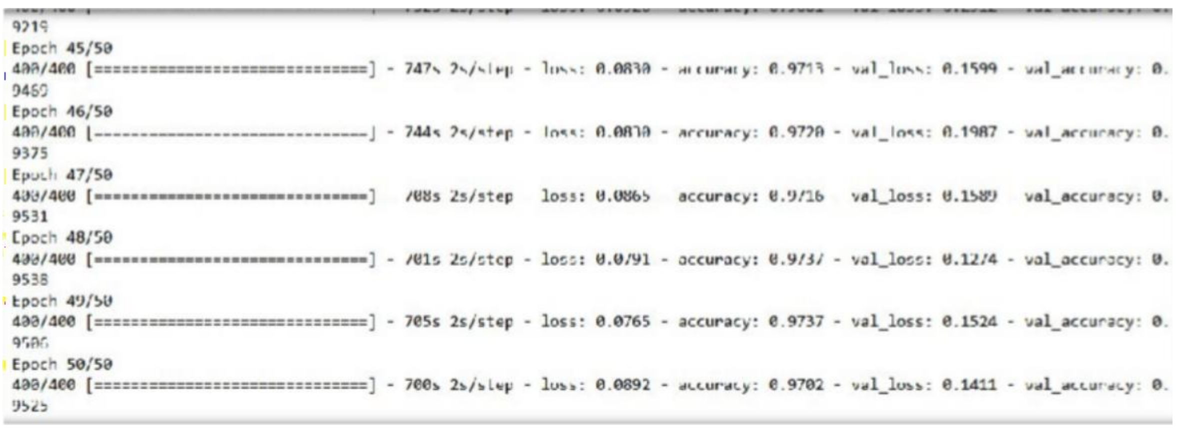
The following fig shows the various accuracy metrics such as accuracy and loss for the model at 20 epochs



```
Epoch 16/20
400/400 [==============================] - 729s 2s/step - loss: 0.1777 - accuracy: 0.9377 - val_loss: 0.7224 - val_accuracy: 0.
8131
Epoch 17/20
400/400 [==============================] - 731s 2s/step - loss: 0.1462 - accuracy: 0.9473 - val loss: 0.4160 - val accuracy: 0.
8781
Epoch 18/20
400/400 [==============================] - 729s 2s/step - loss: 0.1608 - accuracy: 0.9435 - val_loss: 0.2346 - val_accuracy: 0.
9275
Epoch 19/20
400/400 [==============================] - 730s 2s/step - loss: 0.1414 - accuracy: 0.9509 - val_loss: 0.4445 - val_accuracy: 0.
8700
Epoch 20/20
400/400 [==============================]   768s 2s/step   loss: 0.1575   accuracy: 0.9464   val_loss: 0.2616   val_accuracy: 0.
9194
```

Fig 6.2 : Accuracy and Loss for Model at 20 Epochs

The following figure shows the various accuracy metrics such as accuracy and loss for the model at 50 epochs



```
9219
Epoch 45/50
400/400 [==============================] - 747s 2s/step - loss: 0.0830 - accuracy: 0.9713 - val_loss: 0.1599 - val_accuracy: 0.
9469
Epoch 46/50
400/400 [==============================] - 744s 2s/step - loss: 0.0830 - accuracy: 0.9720 - val_loss: 0.1987 - val_accuracy: 0.
9375
Epoch 47/50
400/400 [==============================]   708s 2s/step   loss: 0.0865   accuracy: 0.9716   val_loss: 0.1589   val_accuracy: 0.
9531
Epoch 48/50
400/400 [==============================] - 701s 2s/step - loss: 0.0791 - accuracy: 0.9737 - val_loss: 0.1274 - val_accuracy: 0.
9538
Epoch 49/50
400/400 [==============================] - 705s 2s/step - loss: 0.0765 - accuracy: 0.9737 - val_loss: 0.1524 - val_accuracy: 0.
9506
Epoch 50/50
400/400 [==============================] - 700s 2s/step - loss: 0.0892 - accuracy: 0.9702 - val_loss: 0.1411 - val_accuracy: 0.
9525
```

Fig 6.3 : Accuracy and Loss for Model at 50 Epochs

Table shows accuracy and loss for different epochs

S.No	**No. of Epochs**	**Train Accuracy**	**Loss Accuracy**
1	10	0.61	0.37
2	20	0.92	0.26
3	50	0.95	0.11

Table 6.2 : Train Accuracy and Train Loss for Different Epochs

Graph plotted between training and validation accuracy and training and validation loss for different epochs

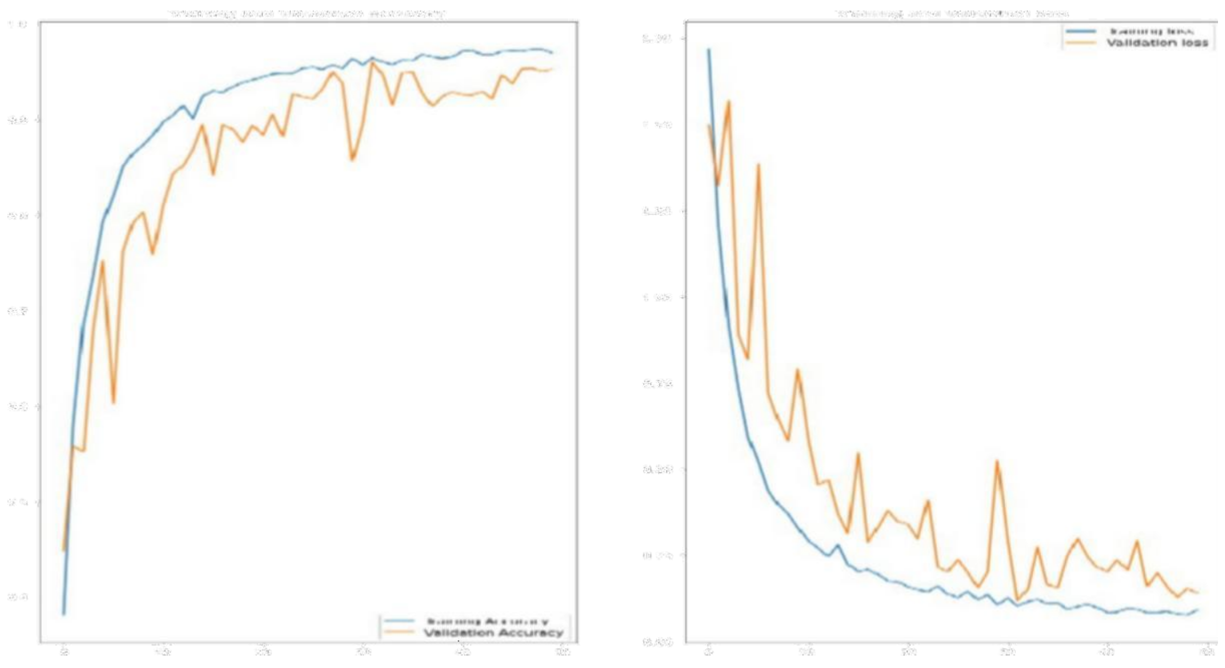


Fig 6.4 : Graph between Training and Validation Accuracy

The following given graphs shows us the variations of accuracy and loss for both trained and validation

The above graphs show with increase in the training accuracy there is increase in the validation accuracy.

And from the training and loss validation graph shows that with decrease in training loss there is decrease in validation loss.

6.3 Predicted Images

As our model is trained with taking care of various performance aspects of the model. Now we can save the model and use it for prediction for any particular images .

So, we create a streamlit web app to predict the diseases for any plant by using the images of the plants leaf. In the web app we use dynamic path to get the image from the system and predict the conditions of the leaf[18]. By using web app we make the life easier for the farmer they just have to upload the image and they will get the result whether the plant is healthy or not.

This web app use the saved model for prediction and run all the commands which are written in jupyter notebooks.A sample prediction using web app is given in below figure.

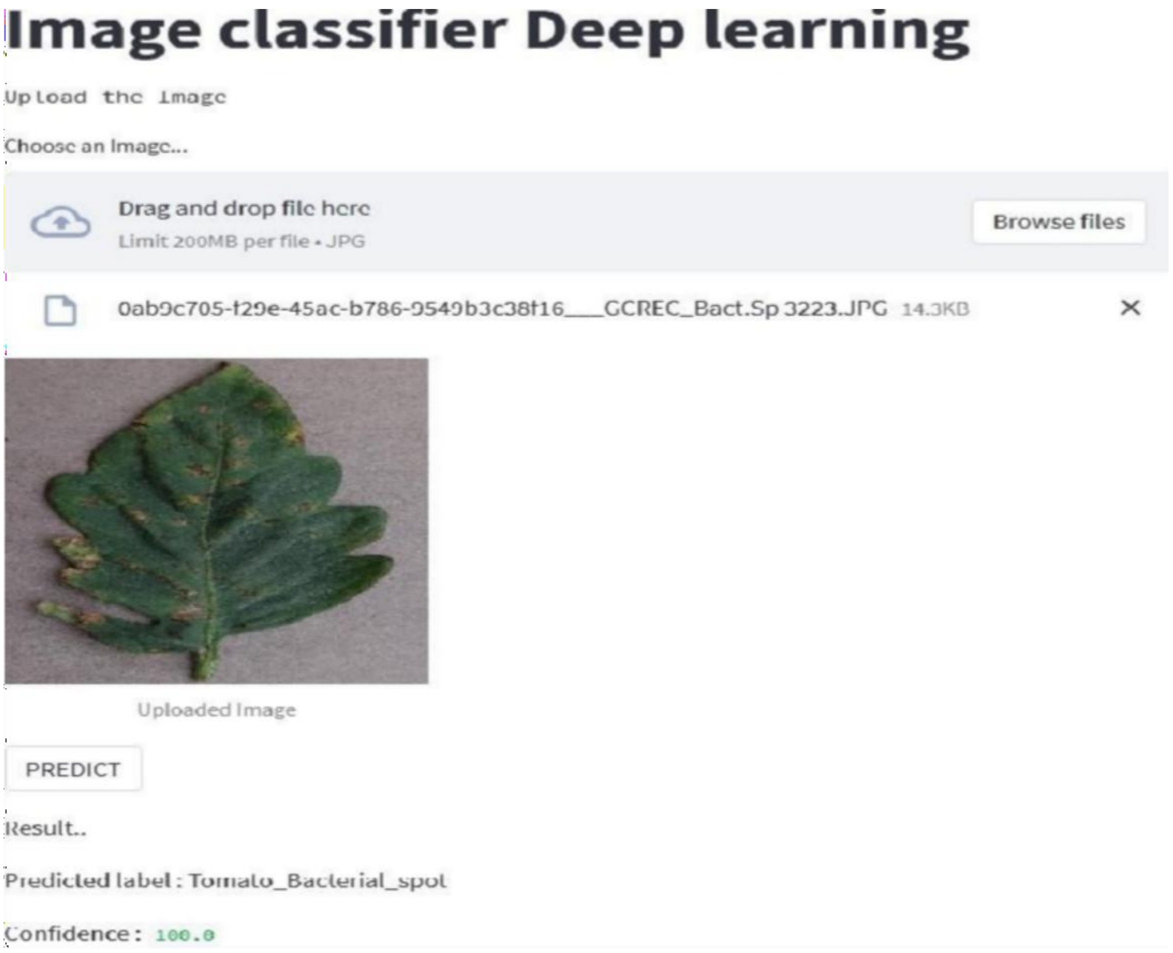


Image classifier Deep learning

Upload the image

Choose an Image...

Drag and drop file here
Limit 200MB per file • JPG

Browse files

0ab9c705-f29e-45ac-b786-9549b3c38f16___GCREC_Bact.Sp 3223.JPG 14.3KB

Uploaded Image

PREDICT

Result..

Predicted label : Tomato_Bacterial_spot

Confidence : 100.0

Fig 6.5 : Prediction done by streamlit Web App[20]

Different types of plants leaves predicted by using the above trained model is given below

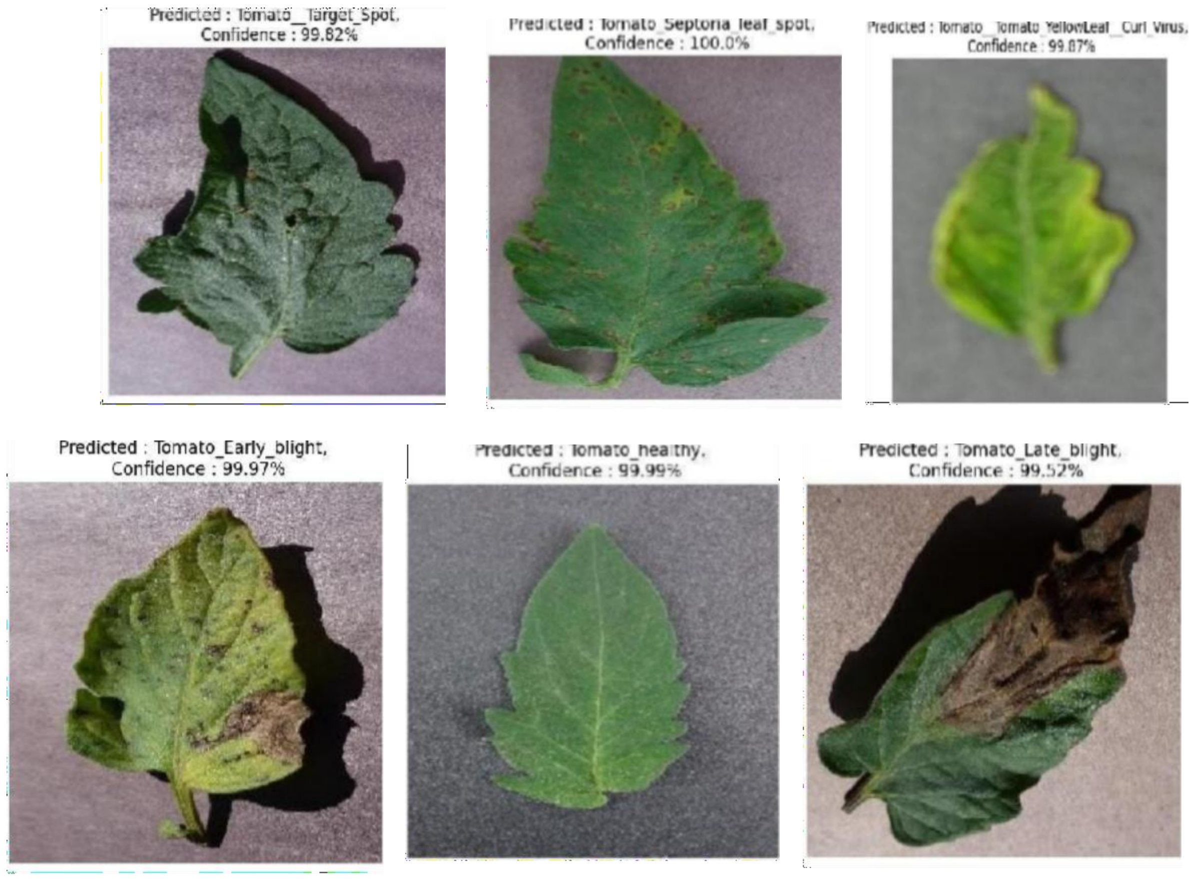

Predicted : Tomato__Target_Spot,
Confidence : 99.82%
Predicted : Tomato_Septoria_leaf_spot,
Confidence : 100.0%
Predicted : Tomato__Tomato_YellowLeaf__Curl_Virus,
Confidence : 99.87%
Predicted : Tomato_Early_blight,
Confidence : 99.97%
Predicted : Tomato_healthy,
Confidence : 99.99%
Predicted : Tomato_Late_blight,
Confidence : 99.52%

Fig 6.6 : Different types Diseases Prediction[21]

CHAPTER – 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

We have introduced a new approach towards the diagnosis of diseases in plant leaves in this project, based on the capability of convolutional neural networks (CNNs). Our system is built around a deep learning-based model that is trained on a huge dataset of images of plant leaves[15]. These photos were taken from various publicly accessible and reliable sources to achieve diversity and quality. After extensive training and testing, the model developed was extremely accurate in classifying 38 plant diseases. It has many diseases on numerous plants.

Our system is designed to provide a friendly, swift, and effective system for plant leaf disease diagnosis. To do this, we incorporated the model into an internet-based platform in a way that the end-users like farmers, farmworkers, and researchers can upload images of plant leaves[14]. The moment one uploads an image, there is an instantaneous feedback given by the system by predicting whether or not there is a disease in real-time. Besides, the solution is also architectured to process and deal with large amounts of image inputs at high speed and thus very well suited for large-scale deployment in agriculture where efficient and speedy disease detection is beneficial.

7.2 Future Scope

Use of Convolutional Neural Networks (CNN) for the identification of plant leaf diseases can revolutionize agricultural technology. With the system in place, several fields of vast potential now lie ready for further development and application of the technology to applications in real-life, everyday usage:

Real-Time Disease Identification

In this study, diagnosis of diseases on plant leaves was performed using the images taken earlier.. Nevertheless, as a future enhancement, there might be an ability to construct a real-time disease detecting system. This new system could include a camera placed on a robotic arm that can drive over plant farm fields and capture live photos of the leaves. Real-time processing of images would assist in detecting the diseases early at initial levels[7]. Early detection would enable early treatment, which could inhibit harm to the crops, minimize loss in agriculture, and substantially boost overall yield.

Incorporation of Transfer Learning Techniques

While the present project utilized a standard CNN model trained on the tomato leaf dataset, future research and development could leverage the power of transfer learning to enhance model performance[16]. This technique involves taking a pre-trained deep learning model—trained on a massive and diverse dataset— and adapting or fine- tuning it specifically for tomato leaf disease classification. By doing so, the model can achieve higher accuracy, better generalization, and improved robustness, even with a relatively smaller dataset specific to tomato plants.

Deployment on Portable and Mobile Platforms

Implementation of the system under consideration was done on a computer or desktop platform, which would restrict its portability and applicability in-field[15]. One can go in the direction of porting and optimizing the system for portable devices such as mobile phones, tabs, or hand scanners.

This way, farmers and farm workers would have access to the technology on the field level, enabling on-site disease diagnosis at the site of occurrence and instant decision-making on treatment. This kind of on-the-move integration would significantly improve the system's utility and usability, particularly in rural and resource-poor agriculture environments.

REFERENCES

[1] Mohanty, S.P., Hughes, D.P., Salathé, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection.

[2] Lu, J., Hu, J., Zhao, G., Mei, F., Zhang, C. (2017). An In-field Automatic Wheat Disease Diagnosis System.

[3] Tang, Y., Chen, L., Wang, J., Zhang, X., Li, H., Liu, Y. (2020). Enhancing Plant Disease Detection through Deep Learning: A Depthwise CNN with Squeeze and Excitation Integration and Residual Skip Connections.

[4] Hassan, S.M., Jasinski, M., Leonowicz, Z., Jasinska, E., Maji, A.K. (2021). Plant Disease Identification Using Shallow Convolutional Neural Network.

[5] Ali, A.A., Chramcov, B., Jasek, R., Katta, R., Krayem, S. (2021). Classification of Plant Diseases Using Convolutional Neural Networks.

[6] Zhang, D., Yang, H., Cao, J. (2021). Identify Apple Leaf Diseases Using Deep Learning Algorithm.

[7] Kumar, A., Singh, R., Gupta, A., Sharma, A., Yadav, S. (2021). Hybrid CNN-LSTM Model for Real-Time Plant Disease Prediction.

[8] Shruthi, A., Nitin, D., Vijay, S. (2021). Deep CNN for Detection of Plant Diseases on Low-Resolution Leaf Images.

[9] Thakur, P.S., Khanna, P., Sheorey, T., Ojha, A. (2022). Explainable Vision Transformer Enabled Convolutional Neural Network for Plant Disease Identification: PlantXViT.

[10] Gupta, A., Kumar, A., Sharma, A., Yadav, S., Singh, R. (2022). CNN-Based Plant Disease Detection Using Smartphone Images.

[11] Bobde, S., Kalambe, K., Tripathi, A., Deoda, K., Haridas, V. (2023). Plant Disease Detection using CNN Models.

[12] Singh, P., Kumar, S., Gupta, R., Sharma, P., Singh, A. (2023). Transfer Learning-Based CNN for Plant Disease Detection.

[13] Pan, Y. (2024). Leaf Disease Detection and Classification with a CNN-LSTM Hybrid Model.

[14] Prince, R.H., Mamun, A.A., Peyal, H.I., Miraz, S., Khandakar, A., Ayari, M.A. (2024). CSXAI: A Lightweight 2D CNN-SVM Model for Detection and Classification of Various Crop Diseases with Explainable AI Visualization.

[15] Patel, V., Mehta, S., Shah, D. (2024). Multi-scale CNN for Enhanced Plant Disease Classification.

[16] Sankhe, S.R., Ambhaikar, A. (2025). Plant Disease Detection and Classification Techniques: A Review.

[17] https://www.scribd.com/presentation/550939859/mini-project-review-ppt-1

[18] https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[19] https://www.geeksforgeeks.org/the-role-of-softmax-in-neural-networks-detailed-explanation-and-applications/

[20] https://www.geeksforgeeks.org/relu-activation-function-in-deep-learning/

[21] https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

PLAGIARISM VERIFICATION

Title of the Thesis plant Disease prediction using convolutional Neural Network

Total Pages 47 Name of the Scholar SUNIL KUMAR

Supervisor (s)

(1) Dr. Abhilasha Sharma

(2) ____

(3) ____

Department Software Engineering

This is to report that the above thesis was scanned for similarity detection. Process and outcome is given below:

Software used: Turnitin Similarity Index: 8%, Total Word Count: 6116

Date: 27/05/25

Sunil Kumar
Candidate's Signature

[illegible]
Signature of Supervisor(s)

Thesis sunil final.docx

Delhi Technological University

Document Details

Submission ID

trn:oid:::27535:97353344

Submission Date

May 23, 2025, 1:09 PM GMT+5:30

Download Date

May 23, 2025, 1:11 PM GMT+5:30

File Name

Thesis sunil final.docx

File Size

8.5 MB

34 Pages

6,116 Words

33,042 Characters

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 8 words)

Match Groups

- 44 Not Cited or Quoted 8%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

4%	Internet sources
4%	Publications
4%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

44 Not Cited or Quoted 8%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

4%	Internet sources
4%	Publications
4%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	www.rgmcet.edu.in	3%
2	Publication	Thangaprakash Sengodan, Sanjay Misra, M Murugappan. "Advances in Electrical ...	<1%
3	Publication	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelli...	<1%
4	Publication	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelli...	<1%
5	Publication	Rajabi, Hosein. "Utilizing Machine Learning Techniques for Dispersion Measure Es...	<1%
6	Submitted works	University of Greenwich on 2023-11-28	<1%
7	Submitted works	National Institute of Business Management Sri Lanka on 2023-09-20	<1%
8	Publication	Poonam Nandal, Mamta Dahiya, Meeta Singh, Arvind Dagur, Brijesh Kumar. "Pro...	<1%
9	Submitted works	APJ Abdul Kalam Technological University, Thiruvananthapuram on 2025-03-03	<1%
10	Submitted works	University of Bradford on 2024-09-12	<1%

11	Submitted works	
Southampton Solent University on 2021-03-25		<1%
12	Submitted works	
University of Kent at Canterbury on 2021-04-01		<1%
13	Submitted works	
Hellenic Open University on 2023-02-24		<1%
14	Internet	
www.frontiersin.org		<1%
15	Submitted works	
Berlin School of Business and Innovation on 2025-01-27		<1%
16	Publication	
Vivek Sharma, Ashish Kumar Tripathi, Himanshu Mittal. "DLMC-Net: Deeper light...		<1%
17	Submitted works	
Berlin School of Business and Innovation on 2025-01-13		<1%
18	Submitted works	
University of Brighton on 2018-05-21		<1%
19	Submitted works	
University of West London on 2023-01-29		<1%

Thesis sunil final.docx

Delhi Technological University

Document Details

Submission ID

trn:oid:::27535:97353344

Submission Date

May 23, 2025, 1:09 PM GMT+5:30

Download Date

May 23, 2025, 1:11 PM GMT+5:30

File Name

Thesis sunil final.docx

File Size

8.5 MB

34 Pages

6,116 Words

33,042 Characters

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

0 AI-generated only 0%
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate [illegible] writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used [illegible] adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.