A Project Report
on

Plant Disease Detection Using CNN

Submitted to

Acharya Nagarjuna University

In Partial Fulfilment of the Requirement for the Award of Bachelor's Degree in Information Technology
by

K. Sri Sasank Chowdary	**Y19AIT449**
D. Chandana	**Y19AIT426**
G. Hemanth Surya Narayana	**Y19AIT433**
K. Bhanu Prakash	**Y19AIT460**

Under the guidance of
Sri P. Ravi Kumar, MTech., (Ph.D.)., Assistant Professor.

Department of Information Technology
Bapatla Engineering College
(Autonomous)
Mahatmaji Puram, Bapatla – 522102

2022-2023

Affiliated to

Acharya Nagarjuna University

A Project Report
on

Plant Disease Detection Using CNN

Submitted to

Acharya Nagarjuna University

In Partial Fulfilment of the Requirement for the Award of Bachelor's Degree in Information Technology
by

K. Sri Sasank Chowdary	**Y19AIT449**
D. Chandana	**Y19AIT426**
G. Hemanth Surya Narayana	**Y19AIT433**
K. Bhanu Prakash	**Y19AIT460**

Under the guidance of
Sri P. Ravi Kumar, MTech., (Ph.D.)., Assistant Professor.

Department of Information Technology
Bapatla Engineering College
(Autonomous)
Mahatmaji Puram, Bapatla – 522102

2022-2023

Affiliated to

Acharya Nagarjuna University

Bapatla Engineering College
(Autonomous)

Department of Information Technology

Mahatmaji Puram, Bapatla - 522102

CERTIFICATE

This is to certify that the Project entitled.

"Plant Disease Detection Using CNN"

K. Sri Sasank Chowdary	**Y19AIT449**
D. Chandana	**Y19AIT426**
G. Hemanth Surya Narayana	**Y19AIT433**
K. Bhanu Prakash	**Y19AIT460**

is a record of Bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Technology (Information Technology) at BAPATLA ENGINEERING COLLEGE, Bapatla under the Acharya Nagarjuna University. This work is done during year 2022-2023, under our guidance.

Date:

(Asst. Prof. P. Ravi Kumar)
Project Guide

(Asst. Prof. M. Praveen Kumar)
Project Coordinator

(Dr. N. Siva Ram Prasad)
H.O.D, I.T Department

Sig. of External Examiner

Acknowledgement

We are profoundly grateful to **Asst. Prof. Mr. P. Ravi Kumar** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We would like to express deepest appreciation towards **Dr. Nazeer Shaik** Principal, Bapatla Engineering College, **Dr. N. Siva Ram Prasad**, Head of Department of Information Technology and **Asst. Prof. Mr. M. Praveen Kumar**, Project Coordinator whose invaluable guidance supported us in completing this project.

At last, we must express our sincere heartfelt gratitude to all the staff members of Information Technology Department who helped us directly or indirectly during this course of work.

K. Sri Sasank Chowdary
D. Chandana
G. Hemanth Surya Narayana
K. Bhanu Prakash

ABSTRACT

Machine learning, Deep learning, and Artificial intelligence are the Future. We use these technologies in almost every field. In the Farming sector, we can also use this technology for the Preparation of soil, adding fertilizers, sowing of seed, Irrigation, weed protection, harvesting, disease prediction: When plants and crops are affected by pests it affects the agricultural production of the country. Usually, farmers or experts observe the plants with naked eye for detection and identification of disease. But this method can be time processing, expensive and inaccurate. Automatic detection using image processing techniques provide fast and accurate results. This paper is concerned with a new approach to the development of plant disease recognition model, based on leaf image classification, using deep convolutional networks. Advances in computer vision present an opportunity to expand and enhance the practice of precise plant protection and extend the market of computer vision applications in the field of precision agriculture. Novel way of training and the methodology used facilitate a quick and easy system implementation in practice. All essential steps required for implementing this disease recognition model are fully described throughout the paper, starting from gathering images in order to create a database, assessed by agricultural experts, a deep learning framework to perform the deep CNN training. This method paper is a new approach in detecting plant diseases using the deep convolutional neural network trained and fine-tuned to fit accurately to the database of a plant's leaves that was gathered independently for diverse plant diseases. The advance and novelty of the developed model lie in its simplicity; healthy leaves and background images are in line with other classes, enabling the model to distinguish between diseased leaves and healthy ones or from the environment by using CNN.

Keywords: DL (Deep Learning), CNN (Convolutional Neural Networks)

List of Figures

List of Tables

CONTENTS

Chapter 1
Introduction

1.1 Introduction of Leaf Disease Detection

The problem of efficient plant disease protection is closely related to the problems of sustainable agriculture Inexperienced pesticide usage can cause the development of long-term resistance of the pathogens, severely reducing the ability to fight back. Timely and accurate diagnosis of plant diseases is one of the pillars of precision agriculture. It is crucial to prevent unnecessary waste of financial and other resources, thus achieving healthier production in this changing environment, appropriate and timely disease identification including early prevention has never been more important. There are several ways to detect plant pathologies. Some diseases do not have any visible symptoms, or the effect becomes noticeable too late to act, and in those situations, a sophisticated analysis is obligatory. However, most diseases generate manifestation in the visible spectrum, so the naked eye examination of a trained professional is the prime technique adopted in practice for plant disease detection. To achieve accurate plant disease diagnostics a plant pathologist should possess good observation skills so that one can identify characteristic symptoms. Variations in symptoms indicated by diseased plants may lead to an improper diagnosis since amateur gardeners and hobbyists could have more difficulties determining it than a professional plant pathologist. An automated system designed to help identify plant diseases by the plant's appearance and visual symptoms could be of great help to amateurs in the gardening process and trained professionals as a verification system in disease diagnostics.

Apple, grape, corn, potato, and tomato plant leaves which are categorized total 24 types of labels apple label namely: Apple scab, Black rot, apple rust, and healthy. Grape label namely: Black rot, Esca, healthy, and Leaf blight. Corn label namely: Corn Cercospora spot gray spot, Corn rust, Corn healthy, Corn Northern Blight The dataset consists of 31,119 images of apple, grape, potato and tomato, all Images are resized into 256 x 256, that images divided into two parts training and testing dataset.

1. Apple scab 2. Grape Esca 3. Corn leaf spot 4. Potato Early Blight 5. Tomato Bacterial Spot

Fig 1.1 Sample Diseased Leaf Images

In fig.1 we can see vegetable and fruit leaves like potato, tomato, corn, apple, grape with diseased part this disease can be easily detected using deep learning. Potato label namely: Early blight, healthy, and Late blight. Tomato label namely: bacterial spot, early blight, healthy, late blight, leaf mold, septoria leaf spot, spider mite, target sport, mosaic virus. techniques

However, most diseases generate manifestation in the visible spectrum, so the naked eye examination of a trained professional is the prime technique adopted in practice for plant disease detection.

Timely and accurate diagnosis of plant diseases is one of the pillars of precision agriculture. It is crucial to prevent unnecessary waste of financial and other resources, thus achieving healthier production in this changing environment, appropriate and timely disease identification including early prevention has never been more important.
To achieve accurate plant disease diagnostics a plant pathologist should possess good observation skills so that one can identify characteristic symptoms.

1.2 Applications

- Plant leaf disease detection is useful in agriculture institute.
- Plant-disease detection system provides clear benefit in monitoring of large fields.
- Some plant leaf disease detection automatic techniques are beneficial for large work of monitoring in farm of crops disease detection.

1.3 Problem Statement

Identification of the plant diseases is the key to preventing the losses in the yield and quantity of the agricultural product. The studies of the plant diseases mean the studies of visually observable patterns seen on the plant. It is very difficult to monitor the plant diseases manually. It requires tremendous amount of work, expertise in the plant diseases, and also require the excessive processing time. Hence, image processing and Deep learning techniques are used for the detection of plant diseases.

1.4 Objective

The objective of Plant disease detection using leaf images is to design an incremental model to detect the plant diseases ignoring external features like environment, noise and background. This focuses on identifying various diseases in plants and to ease the job of farmers or to educate the farmers about the disease detected. An approach of classification using Convolutional Neural Network that has very good working efficiency produces the accurate results. The system helps to improve the performance. Maintaining the project is easy and manageable.

1. To enhance the given input image by Image acquisition and Image preprocessing.
2. Identify the affected part through texture analysis and Segmentation.
3. Classify the healthy and affected leaf part by feature extraction and classification.

4. Train the model by using testing data for accurate result.

1.5 Motivation

Identifying and recognition of leaves disease is the solution for saving the reduction of large farm in crop disease detection and profit in productivity, it is beneficial in agricultural institute and biological research.

1.6 Libraries Used

· NumPy

- NumPy stands for Numerical Python.
- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

· Matplotlib

- Matplotlib is a low-level graph plotting library in python that serves as a visualization utility.
- Matplotlib was created by John D. Hunter.
- Matplotlib is open source, and we can use it freely.
- Matplotlib is mostly written in python, a few segments are written in C, Objective-C and JavaScript for Platform compatibility.

. OS

- Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.
- The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality.
- The Python OS module lets us work with the files and directories.

· TensorFlow

- TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind. However, it proved to be very useful for deep learning development as well, and therefore Google open-sourced it.

- TensorFlow works based on data flow graphs that have nodes and edges. As the execution mechanism is in the form of graphs, it is much easier to execute TensorFlow code in a distributed manner across a cluster of computers while using GPU's.

- The next part of the What is TensorFlow tutorial focuses on why we should use TensorFlow.

1.7 Project Scope

Automated detection systems are in rapid demand with all the technological advancements happening around. In agricultural field, loss of yield mainly occurs because of the widespread disease. In general, the disease is only detected once it is in an advanced stage. By that time, we will not be able to take an action as remedy to avoid the loss. For this, we propose a technical system which can detect the disease at an early stage as soon as the spots are visible on the leaf. So, one can be able to act quick and use a remedy to prevent the loss without an expert interference up to an extent. It can provide help for an individual having less knowledge about the disease. Counting on these goals, we've to extract the features like the disease. This feature extraction will be explained clearly in later sections. By this technical involvement in plant leaf disease detection, we can have many advantages, ranging from cost efficiency to yield loss preventing.

Chapter 2
Literature Survey

Authors of [1] proposed that Smart farming system using necessary infrastructure is an innovative technology that helps improve the quality and quantity of agricultural production in the country including tomato. Since tomato plant farming take considerations from various variables such as environment, soil, and amount of sunlight, existence of diseases cannot be avoided. The current advance computer system innovation made possible by deep learning that have cover the way for camera captured tomato leaf disease. This study developed the innovative solution that provides efficient disease detection in tomato plants. A motor-controlled image capturing box was made to capture four sides of every tomato plant to detect and recognize leaf diseases. A specific breed of tomato which is Diamante Max was used as the test subject. The system was designed to identify the diseases namely Phroma Rot, Leaf Miner, and Target Spot. Dataset leaves contain diseased and healthy plant leaves are collected. Then train a deep convolution neural network to identify three diseases. The system used Convolution Neural Network to identify which of the tomato diseases is present on the monitored tomato plants. The F-RCNN trained anomaly detection model produced a confidence score of 80% while the Transfer Learning disease recognition model achieves an accuracy of 95.75%. The automated image capturing system was implemented in actual and registered 91.67% accuracy in the recognition of the tomato plant leaf diseases.But this method can be time processing we uses the CNN which would be faster.

Authors of [2] proposed that Agriculture field has a high impact on our life. Agriculture is the most important sector of our Economy. Farmers are difficult to identify the leaf disease, so they produce less production. Though, videos and images of leaves provide better view for agricultural scientists can provide a better solution. So that can solve the problem of related to crop disease. It is required to note that if the productivity of the crop is diseased then, it has high risk of providing good nutrition. Due to the improvement and development in technology where devices are smart enough to recognize and detect plant diseases. Acknowledge diseases faster treatment to lessen the negative impacts on harvest. In this paper focus on plant disease detection using image processing techniques. This paper access open dataset images that consist of 5000 images of healthy and diseased plant leaves, and there used semi supervised techniques for crop types and detect the disease of four classes. In this Method is Expensive but our Method is low-cost and can be made available for every farmer

Authors of [3] proposed that This paper contains five types of apple leaf disease that are, aria leaf spot, Brown spot, Mosaic, Grey spot, and Rust. That is affected in apple. This paper used deep learning techniques to improved convolution neural networks (CNNs) for detection in apple leaf diseases. In this paper, the apple leaf disease dataset (ALDD) is used, which consist complex images and laboratory images, and rest constructed via data augmentation and image annotation technologies to create new apple leaf disease detection model that uses deep-CNNs is by using Rainbow concatenation and Google Net Inception structure. In testing dataset used

26,377 images of apple leaves disease, the proposed INAR- model is trained and then detect five common apple leaf diseases. In the experimental results show that the INAR- SSD model realizes 78.80% detection performance, with a high-detection speed of 23.13 FPS. The results demonstrate that the novel INAR-SSD model provides a high- performance solution for the early diagnosis of apple leaf diseases that can perform real-time detection of these diseases with higher accuracy and faster detection speed than previous methods. It is faster but not accurate, so we developed accuracy more by using CNN.

Authors of [4] proposed that In this paper, plant leaf disease identification using deep learning technique in convolution neural network (CNN). The Convolutional neural network model is trained using an than 39 different classes of open dataset of plant leaves diseases, and background images. That contain six types of data augmentation methods and that are used for gamma correction, image flipping, principal component analysis (PCA) color augmentation, rotation, noise injection, and scaling. Whole are notice that using data augmentation. That can increase the performance of the model. The model was trained using different training range of epochs, batch sizes and dropouts. Then CNN is compared with transfer learning approaches, the proposed model achieves better result. When using the validation data. Though simulation proposed model achieves 96.46% classification accuracy. Accuracy of the CNN is better than the accuracy of transfer learning approaches.Accuracy is better but time expensive, so we created less time executable than this.

Authors of [5] proposed that In this paper contains vegetable, fruit, crops and flowers Agricultural Images, and that leaf disease. The agricultural product type associated disease identification. These diseases are specific to the product component which can be root, seed, and leaf. This is helpful into the provide identification of disease from remote lab. The work is here divided in two steps. In first step, the ring project-based segmentation model is defined to explore the features of leaf images. Once the features are identified then work is apply for PNN classifier to identify the existence disease. The work is about to identify the health and infected disease based on featured region identification. The work is applied on randomly collected leaf images from web for different plants.CNN used by us more accurate than this PNN

Authors of [6] proposed that In recent times, drastic climate changes and lack of immunity in crops has caused substantial increase in growth of crop diseases. This causes large scale demolition of crops, decreases cultivation and eventually leads to financial loss of farmers. Due to rapid growth in variety of diseases and adequate knowledge of farmer, identification and treatment of the disease has become a major challenge. The leaves have texture and visual similarities which attributes for identification of disease type. Hence, computer vision employed with deep learning provides the way to solve this problem. This paper proposes a deep learning-based model which is trained using public dataset containing images of healthy and diseased crop leaves. The model serves its objective by classifying images of leaves into diseased category based on the pattern of defect. It was time expensive but ours is less time executable

Authors of [7] proposed that Agriculture has become far more than simply a method to feed ever growing populations. It's important wherever in additional than seventieth population of an Asian country is depends on agriculture. Which means it feeds nice range of individuals. The foremost necessary consider less amount crop of quality because of disease. Leaf disease detection may be stopping agricultural losses. The aim of this is to develop a software system answer that mechanically find and classify disease. That consist of steps like image acquisition, preprocessing, Segmentation, extraction, and classification are involves disease detection. The leaves images are used for detecting the plant diseases. Therefore, use of image process technique to find and classify diseases in agricultural. It was accurate but cost expensive and ours is low cost and available all

Authors of [8] proposed that increasing the agricultural productivity improves the Indian economy. Keeping this as objective, in order to achieve an efficient and smart farming system, identification of unhealthy leaf using image processing techniques is contributed in this paper. For this, ladies finger plant leaves are chosen and examined to find an early stage of various diseases such as yellow mosaic vein, leaf spot, powdery mildew etc. Leaf images are captured, processed, segmented, features extracted, and classified to know if they are healthy or unhealthy. Due to practical limitations in climatic conditions and other terrain regions, noisy image data sets are also created and taken into consideration. K-Means clustering is used for segmentation and for classification, SVM and ANN are used. This work uses PCA to reduce the feature set. Results show that, the average accuracy of detect.

Chapter 3
Software Requirements Specification

The software requirement specification of our project will have the entire necessary requirement which will be a baseline of our project. The software requirement specification will incorporate functional and nonfunctional requirements, system architecture, data flow diagrams, UML diagrams, experimental setup requirements and performance metrics.

- Operating System : Windows 7 and above.
- Programing language : Python 2.7 and above.
- Platform : Jupyter Notebook
- Supporting libraries : TensorFlow, OpenCV, PIL, Tkinter, os, Sklearn etc.

Chapter 4
Requirement Analysis

4.1 Functional Requirements

Functional requirements describe the system functionality, while the non-functional requirements describe system properties and constraints.

Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks, or the functions the system is required to perform. This lays out important concepts and discusses capturing functional requirements in such a way they can drive architectural decisions and be used to validate the architecture. Features may be additional functionality or differ from basic functionality along some quality attribute. In the proposed system, concert assesses the compliance of a workflow by analyzing the five established elements required to checkfor the rule adherence in workflows: activities, data, location, resources, and time limits.

4.2 Non-Functional Requirements

4.2.1 Security

- System needs to control the user access and session.
- It needs to store the data in a secure location and stored in a secure format.
- It requires a secure communication channel for the data.

4.2.2 Concurrency and Capacity

System should be able to handle multiple computations executing simultaneously, and potentially interacting with each other.

4.2.3 Performance

Performance is generally perceived as a time expectation. This is one of the most important considerations especially when the project is in the architecture phase.

4.2.4 Reliability

It is necessary to ensure and notify about the system transactions and processingas

simple as keep a system log will increase the time and effort to get it done from the very beginning. Data should be transferred in a reliable way and using trustful protocols.

4.2.5 Maintainability

Well-done system is meant to be up and running for long time. Therefore, it will regularly need preventive and corrective maintenance. Maintenance might signify scalability to grow and improve the system features and functionalities.

4.2.6 Usability

End user satisfaction and acceptance is one of the key pillars that support a project success. Considering the user experience requirements from the project conception is a win bet, and it will especially save a lot of time at the project release, asthe user will not ask for changes or even worst misunderstandings.

4.2.7 Documentation

All projects require a minimum of documentation at different levels. In many cases the users might even need training on it, so keeping good documentation practices and standards will do this task spread along the project development; but as well this must be established since the project planning to include this task in the list.

Chapter 5
System Design

5.1 Data Flow Diagram

A data flow Diagram is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step

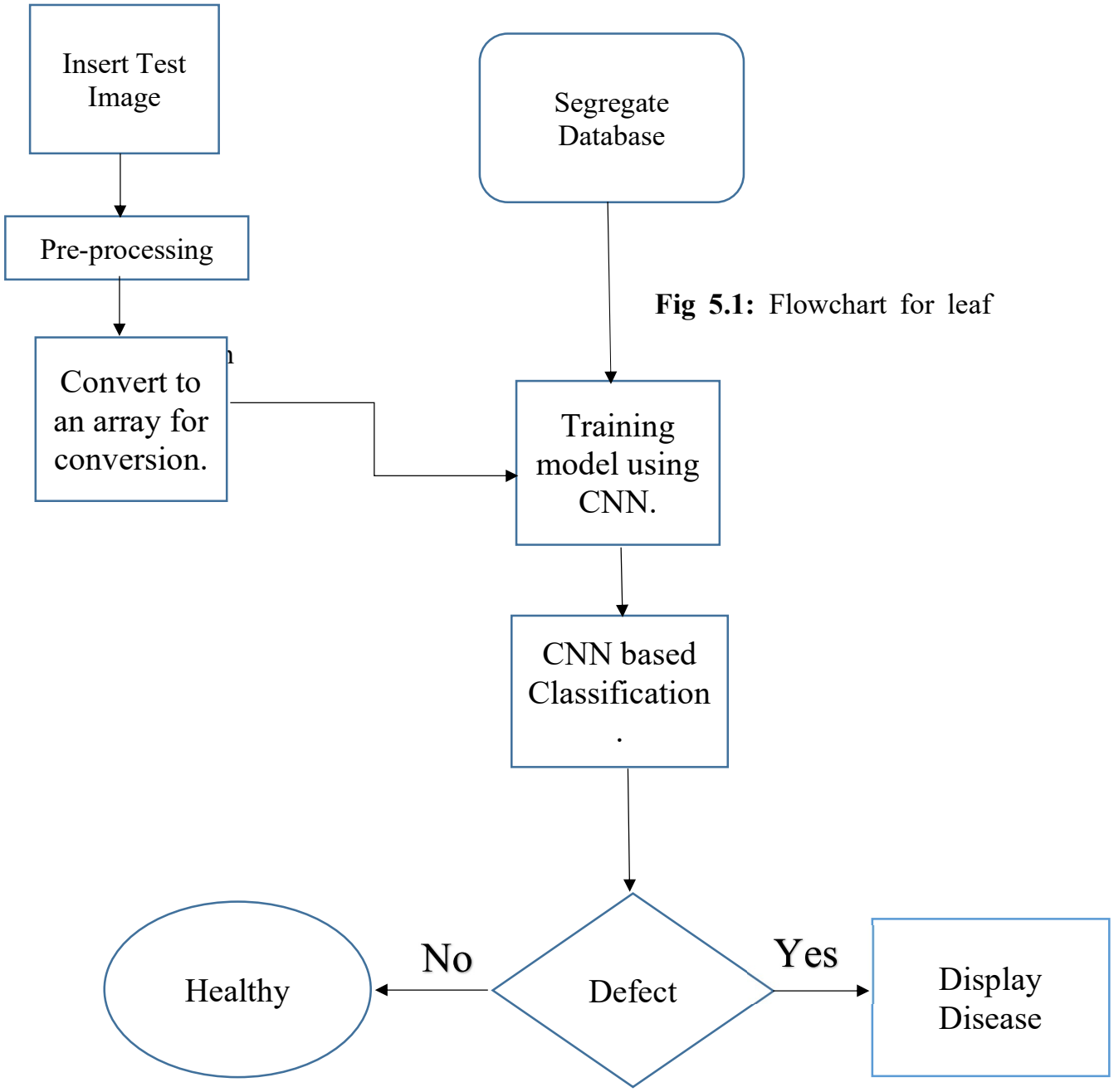

Fig 5.1: Flowchart for leaf

5.2 Use case Diagram.

Use case diagram is a graphic depiction of the interactions among the elements of asystem. Use cases will specify the expected behavior, and the exact method of makingit happen. Use cases once specified can be denoted both textual and visual representation.

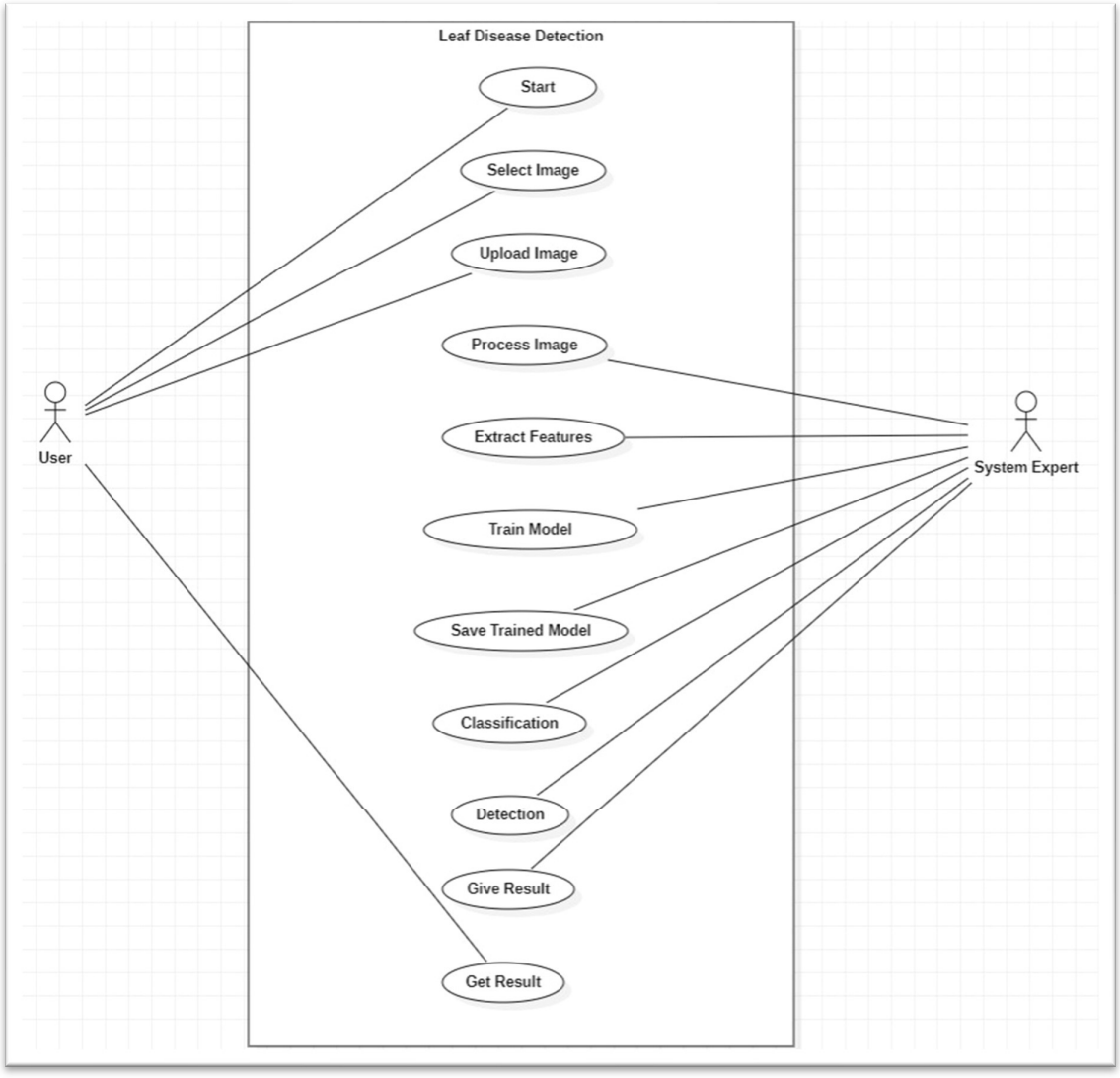

Fig 5.2: Use case diagrams

Use case diagrams are used to specify:

- Requirements (external) required usages of a system under design or analysis -to capture

what the system is supposed to do.

- The functionality offered by a subject – what the system can do.
- Requirements the specified subject poses on its environment - by defining how environment should interact with the subject so that it will be able to perform its services.

5.3 Sequence Diagram

Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows as parallel vertical lines (lifelines), different processes or objects that live simultaneously and as horizontal arrows, the messages exchanged between them in the order in which they occur.

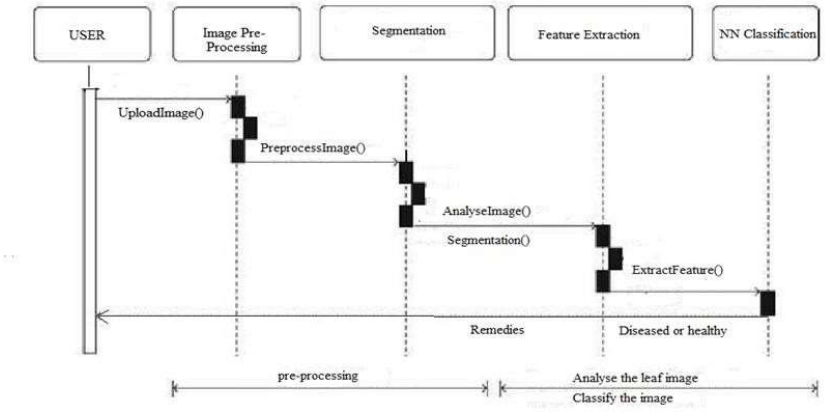

Fig 5.3 Sequence Diagram

1.Usage scenarios:

A usage scenario is a description of a potential way your system isused. The logic of a usage scenario may be part of a use case, perhaps an alternate course. It may also be one entire pass through a use case, such as the logic described bythe basic course of action or a portion of the basic course of action, plus one or more alternate scenarios. The logic of a usage scenario may also be a pass through the logic contained in several use cases.

2.The logic of methods:

Sequence diagrams can be used to explore the logic of a complex operation, function, or procedure. One way to think of sequence diagrams, particularly highly detailed diagrams, is as visual object code

3.The logic of services:

A service is effectively a high-level method, often one that can be invoked by a wide variety of clients. This includes web-services as well as business transactions implemented by a variety of technologies.

5.4 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects ofthe system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity.

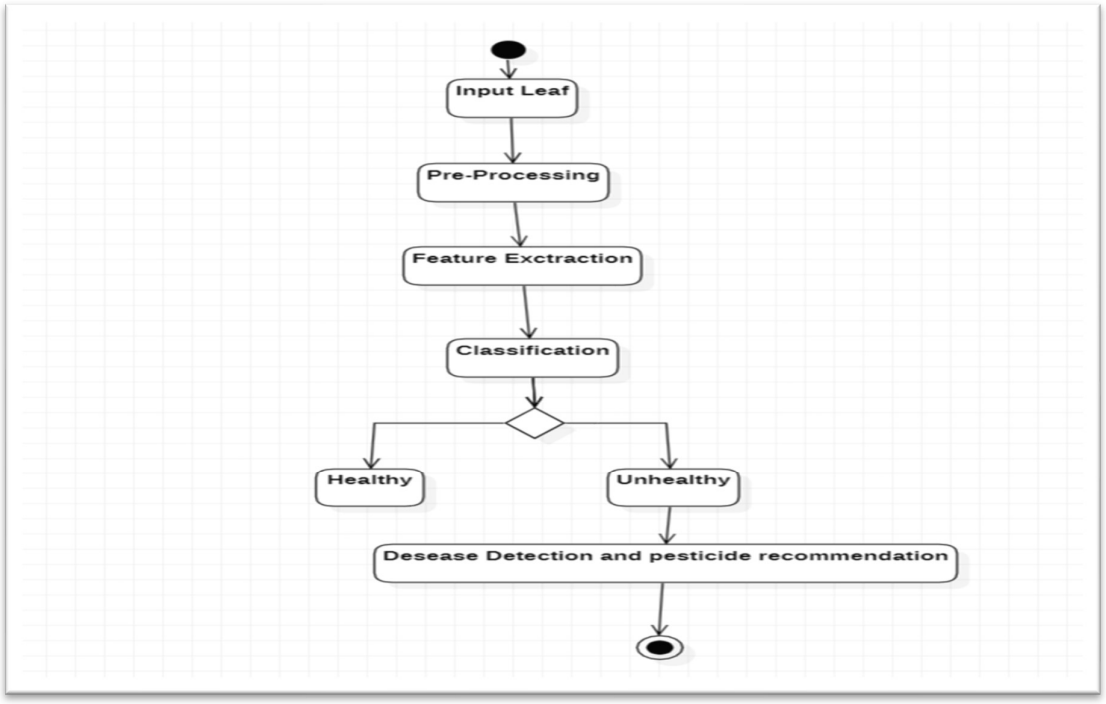

Fig 5.4 Activity Diagram

Chapter 6
System Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing canalso provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. There are various typesof testing, explained in the following sections.

6.1 Testing Objective

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yetundiscovered error.
- A successful test one that uncovers an as –yet undiscovered error.

The above objectives imply a dramatic change in viewpoint. They move counter tothe commonly held view that a successful test is one in which no errors are found. Testing cannot show the absence of detects; it can only show that software errors are present.

6.2 White Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent boxtesting, or structural testing) is a method of testing software that tests internal structuresor workings of an application, as opposed to its functionality as in Black Box Testing Using white box testing methods, the software engineer can derive test cases that:

- Guarantee that all independent paths within a module have been exercised atleast once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their validity.
- Logical errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed.
- Often believe that logical path not likely to execute when, in fact, it may be executed on regular basis.

6.3 Levels of Testing

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

- Code Testing
- Program Testing
- System Testing

6.3.1 Code Testing:

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

6.3.2 Program Testing:

Program testing is also called unit testing. The modules inthe system are integrated to perform the specific function. The modules have been testedindependently, later Assembled and tested thoroughly for integration between differentmodules.

6.3.3 System Testing:

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

The first step includes unit testing where in each module is tested to provide his correctness, validity and also determine any missing operations. Errors are noted downand corrected immediately. Unit testing is the import and major part of the project. So errors are rectified easily in particular module and program clarity is increased. In thisproject entire system is divided into several modules and is developed individually. Sounit testing is conducted to individual modules.

The second step includes integration testing. It need not be the case, the software whose modules when run individually and showing perfect results, will also show perfect results when run as a whole. The individual modules are clipped under this major module and tested again, and the results are verified.

The final step involves validation and testing which determines the software functions as the user expected. Here also there may be some modifications. In the completion of the project, it is satisfied fully by the user.

6.4 Integration test plan

Data can be lost across an interface, one module can have an adverse effort on the other sub functions, when combined, may not produce the desired major functions. Integrated testing is the systematic testing for constructing the uncover errors within the interface. The testing was done with sample data. The developed system has run successfully for this sample data. The need for integrated test is to find the overall system performance.

6.5 Unit Testing

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

Table 6.1: Unit Testing Table

Function Name	**Tests Results**
Uploading Image	Tested for uploading different types and sizes ofimages.
Detecting Disease	Tested for different images of plant leaves anddiseases Bacterial Spot, yellow leaf curl virus.

6.6 Output Testing

After performance of the validation testing, the next step is output testing. The output displayed or generated by the system under consideration is tested by asking the user about the format required by system.

Table 6.2: Output Testing Table

Functionality to be tested	**Input**	**Tests Results**
Working of Choose File option	User convenience to access images stored	Different folders where images are stored can be uploaded

Chapter 7
Project Planning

Project planning is an essential step in the successful execution of any project, and the development of a plant disease detection system using Convolutional Neural Networks (CNN) is no exception. In this project, the primary objective is to develop a machine learning model that can accurately identify the type of plant disease from an image of the plant. To achieve this objective, a comprehensive project plan must be put in place to ensure that the project is executed efficiently and effectively.

The first step in the project planning process is to define the project scope, objectives, and deliverables. This involves identifying the types of plant diseases that the system will be capable of detecting, the accuracy levels that must be achieved, and the expected deliverables at the end of the project. In this project, the primary deliverables are a trained CNN model and a user-friendly application that allows users to upload images of plants and receive information about the detected diseases.

Once the project scope, objectives, and deliverables have been defined, the next step is to develop a project timeline and allocate resources. This involves breaking down the project into smaller tasks and estimating the time required for each task. In addition, resources such as computing power, data storage, and personnel must be identified and allocated accordingly. It is essential to ensure that the project timeline and resource allocation are realistic and feasible to avoid delays and budget overruns.

In conclusion, a well-planned project is critical to the success of any project, and the development of a plant disease detection system using CNN is no exception. Defining the project scope, objectives, and deliverables, developing a project timeline, and allocating resources are essential steps in the project planning process. Proper planning can ensure that the project is executed efficiently and effectively, and the desired deliverables are achieved within the set timelines and budget constraints.

Chapter 8
Implementation

This part of the report illustrates the approach employed to classify the leaves into diseased or healthy and if the leaf is diseased, name of the disease is mentioned along with the remedies. Our methodology primarily revolves around the following five steps.

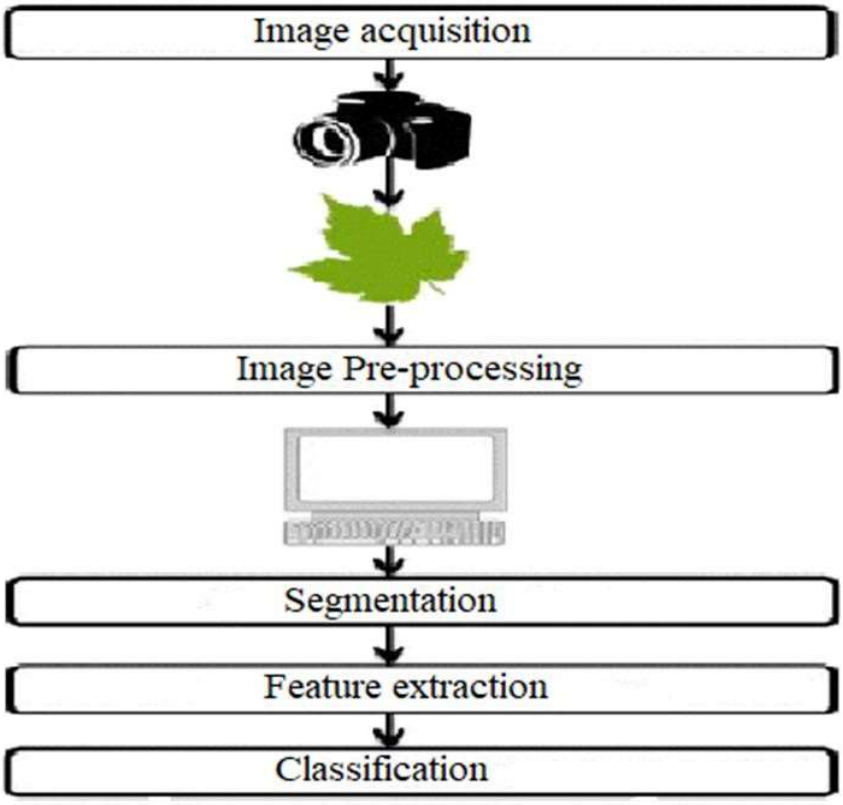

Fig 8.1: System Architecture

Algorithm written below illustrated the step-by-step approach for the proposedimage recognition and segmentation processes:

(1) Image acquisition is the very first step that requires capturing an image with the help of a digital camera.

(2) Pre-processing of input image to improve the quality of image and to remove the undesired distortion from the image. Clipping of the leaf image is performed to get theinterested image region and then image smoothing is done using the smoothing filter. To increase the contrast Image enhancement is also done.

(3) Mostly green colored pixels, in this step, are masked. In this, we computed a threshold value that is used for these pixels. Then in the following way mostly

green pixels are masked: if pixel intensity of the green component is less than the pre- computed threshold value, then zero value is assigned to the red, green and blue components of this pixel.

(4) In the infected clusters, inside the boundaries, remove the masked cells.

(5) Obtain the useful segments to classify the leaf diseases.

8.1 Image Acquisition

The initial process is to collect the data from the public repository. It takes the imageas input for further processing. We have taken most popular image domains so that wecan take any formats like .bmp, .jpg, .gif as input to our process.

The image is captured, scanned, and converted into a manageable entity. This process is known as image acquisition. During a test-phase, we acquire a series of color imagesusing a digital scanner so as to acquire a single image of leaf. The color images weredigitized to produce RGB digital color images.

8.2 Image Preprocessing

As the images are acquired from the real field it may contain dust, spores and waterspots as noise. The purpose of data pre-processing is to eliminate the noise in the image,so as to adjust the pixel values. It enhances the quality of the image.

The main aim of Pre-processing is to suppress unwanted image data and to enhancesome important image features. It includes RGB to Gray conversion, image resizing and median filtering. Here color image is converted to gray scale image to make the image device independent. The image is then resized to a size of 256*256. Then median filtering is performed on the image to remove the noise. The digital version of the rottenleaf sample consists of about 30% of leaf area and rest 70% is the background. Thus the redundant background requires high disk storage space and utilizes CPU time in thesegmentation process. In order to have efficient disk storage and achieve fast processingspeed the digital image of the leaf sample is cropped into a smaller dimension of size 16x20sq, cm. Thus the pre-processing step introduced saves about 30% of disk storagespace and increases the CPU processing 1.4 times. The cropping process does not introduce any loss in the region of interest, i.e. the selected leaf sample.

After pre- processing stage, the digital version of the sample leaf image consists about 70%

of leafarea part and rest 30% as background. For getting better results in further steps, image pre-processing is required because dust, dewdrops, insect's excrements may be presenton the plant; these things are considered as image noise. Furthermore, captured imagesmay have distortion of some water drops and shadow effect, which could create problems in the segmentation and feature extraction stages. Effect of such distortion can be weakened or removed using different noise removal filters. There may be low contrast in captured images; for such images contrast enhancement algorithms can be used. Sometimes background removal techniques may also be needed in case of regionof interest needs to be extracted. In case of noise such as salt and pepper, a median filtercan be used. Weiner filter can be used to remove a blurring effect. In case of the imagescaptured using high-definition cameras, the size of the pictures might be very large, forthat reduction of image size is required. Also, image reduction helps in reducing the computing memory power.

8.3 Image Segmentation

Image segmentation is the third step in our proposed method. The segmented images are clustered into different sectors using Otsu classifier and k-mean clustering algorithm. Before clustering the images, the RGB color model is transformed into Labcolor model. The advent of Lab color model is to easily cluster the segmented images.

8.4 Classification

Convolutional neural networks are used in the automatic detection of leaves diseases.CNN is chosen as a classification tool due to its well-known technique as a successful classifier for many real applications.

CNN architectures vary with the type of the problem at hand. The proposed model consists of three convolutional layers each followed by a max-pooling layer. The finallayer is fully connected MLP. ReLu activation function is applied to the output of everyconvolutional layer and fully connected layer.

The first convolutional layer filters the input image with 32 kernels of size 3x3. After max-pooling is applied, the output is given as an input for the second convolutional layer with 64 kernels of size 4x4. The last convolutional layer has 128 kernels of size 1x1 followed by a fully connected layer of 512 neurons. The output of this layer is givento Softmax function which produces a probability distribution of the four output classes.

Gray Level Co-occurrence Matrix is created from gray scale images and used to describe the shape feature. The Gray Level Co-occurrence Matrix is based on the repeated occurrence

of gray-level configuration in the texture. The spatial gray dependence matrix is used for texture analysis. A spatial gray dependence matrix is created based on hue, saturation, and intensity. Run Length Matrix (RLM) is another type of matrix. Same gray pixel values are the part of run and those gray values forms a two-dimensional matrix.

Example, RM- Q (x, y) x represents gray values and y represents run length.

8.5 Convolutional neural network Algorithm

1)A convolutional neural network is a class of deep neural network, most commonly applied to analyzing visual imagery.

2)A Convolutional Neural Network is a Machine Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

Input Layer

- In the figure below, we have an RGB image which has been separated by its three-color planes—Red, Green, and Blue. There are a number of such color spaces in which images exist—Grayscale, RGB, HSV, CMYK, etc.
- The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.
- This is important when we are to design an architecture which is not only goodat learning features but also is scalable to massive datasets.

Convolution layer—The kernel

- In the below demonstration, the green section resembles our 5x5x1 input image.The element involved in carrying out the convolution operation in the first partof a Convolutional Layer is called the Kernel/Filter, K, represented in the coloryellow. We have selected K as a 3x3x1 matrix {{1,0,1}, {0,1,0}, {1,0,1}}.
- The Kernel shifts 9 times because of Stride Length = 1, every time performinga matrix multiplication operation between K and the portion P of the image

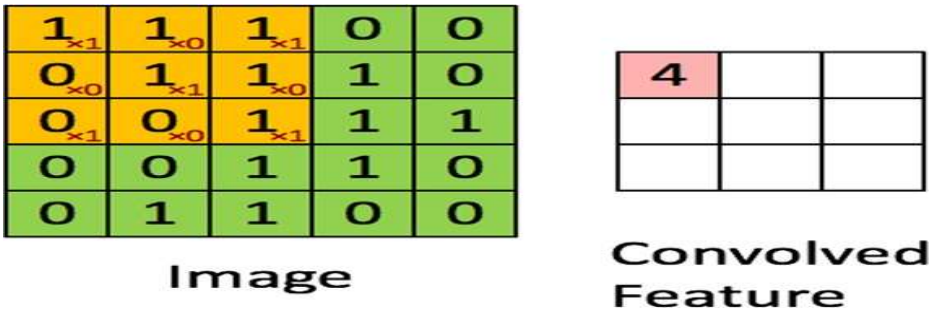

Fig 8.2 sample convolution layer image.

- The filter moves to the right with a certain Stride Value till it parses the completewidth. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

Pooling layer

- The Pooling layer is responsible for reducing the spatial size of the Convolved Feature.
- It is even useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

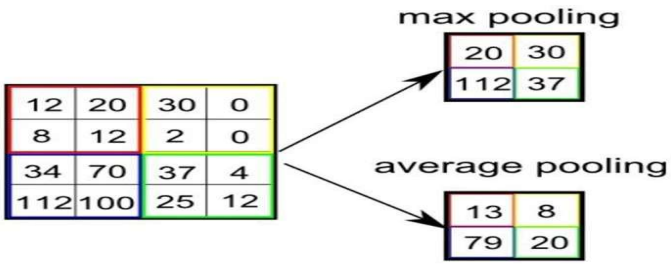

Fig 8.3 selecting maximum and average pooling matrix.

There are two types of Pooling: Max Pooling and Average Pooling.

- Max Pooling returns the maximum value from the portion of the image coveredby the Kernel.
- On the other hand, Average Pooling returns the average of all the values fromthe portion of the image covered by the Kernel.

8.6 Dataset

The dataset that was used is the Plant Village dataset taken from Kaggle. You can download the dataset through url provided [9]

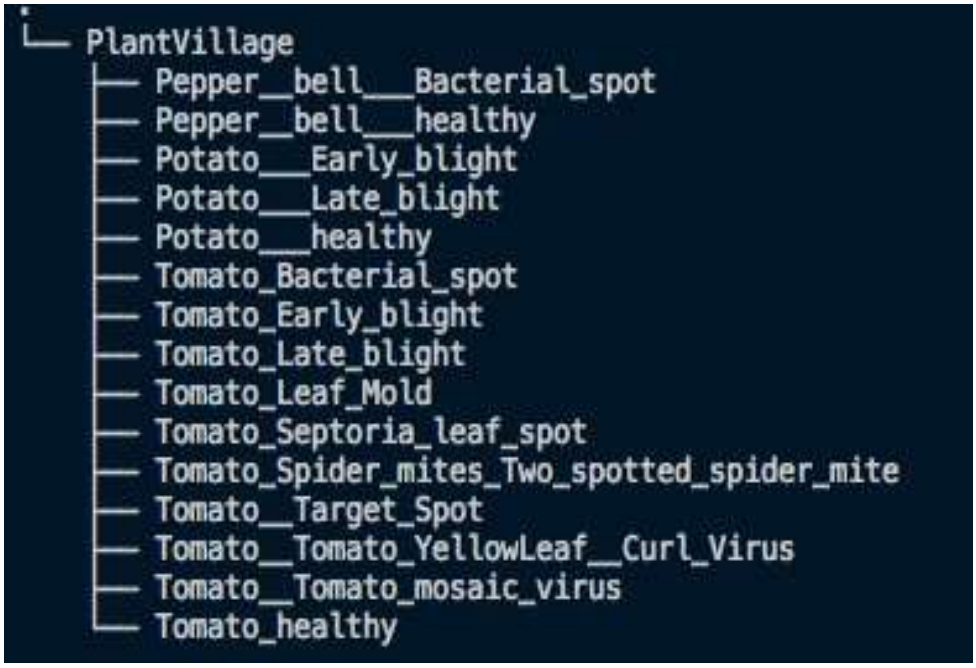

Fig 8.4 Dataset

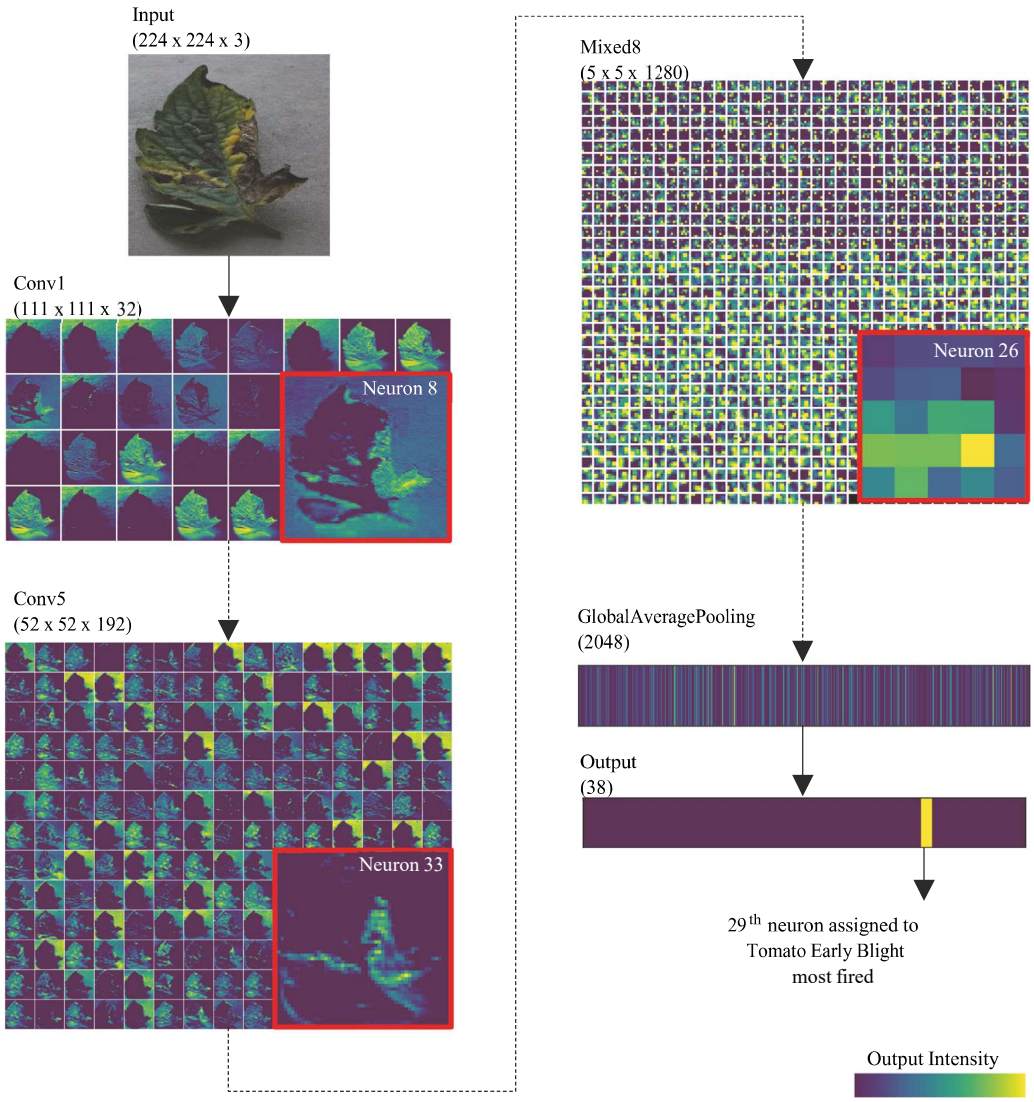

Fig :8.5 Visualization of intermediate outputs.

Visualization I:

Hidden Layer Output Visualization. Figure 3.6 visualizes the hidden layer output [6] for each layer, where an input image of tomato early blight and its generated intermediate outputs are summarized. In our trained model, some of the intermediate outputs in the shallow layers (Conv1, Conv5) highlight the yellow and brown lesions that are apparent within the image (insets with red border). However, in the deeper layer (Mixed8), owing to the convolution and pooling (i.e., down sample) layers, the image size is too small to interpret whether such extracted features have been retained. Moreover, the global average pooling layer converts images to a feature vector that eliminates the spatial information, making it highly difficult to understand how the features are handled in proceeding layers. It is difficult to distinguish whether the extracted features positively contribute to the classification of the input image to the correct disease class or are used for a reason to deny other possibilities (e.g., a furry tail

raises a possibility of an image containing a cat or a dog but certainly not a car). Hence, understanding what the CNN has learned by only exploring the intermediate output is insufficient.

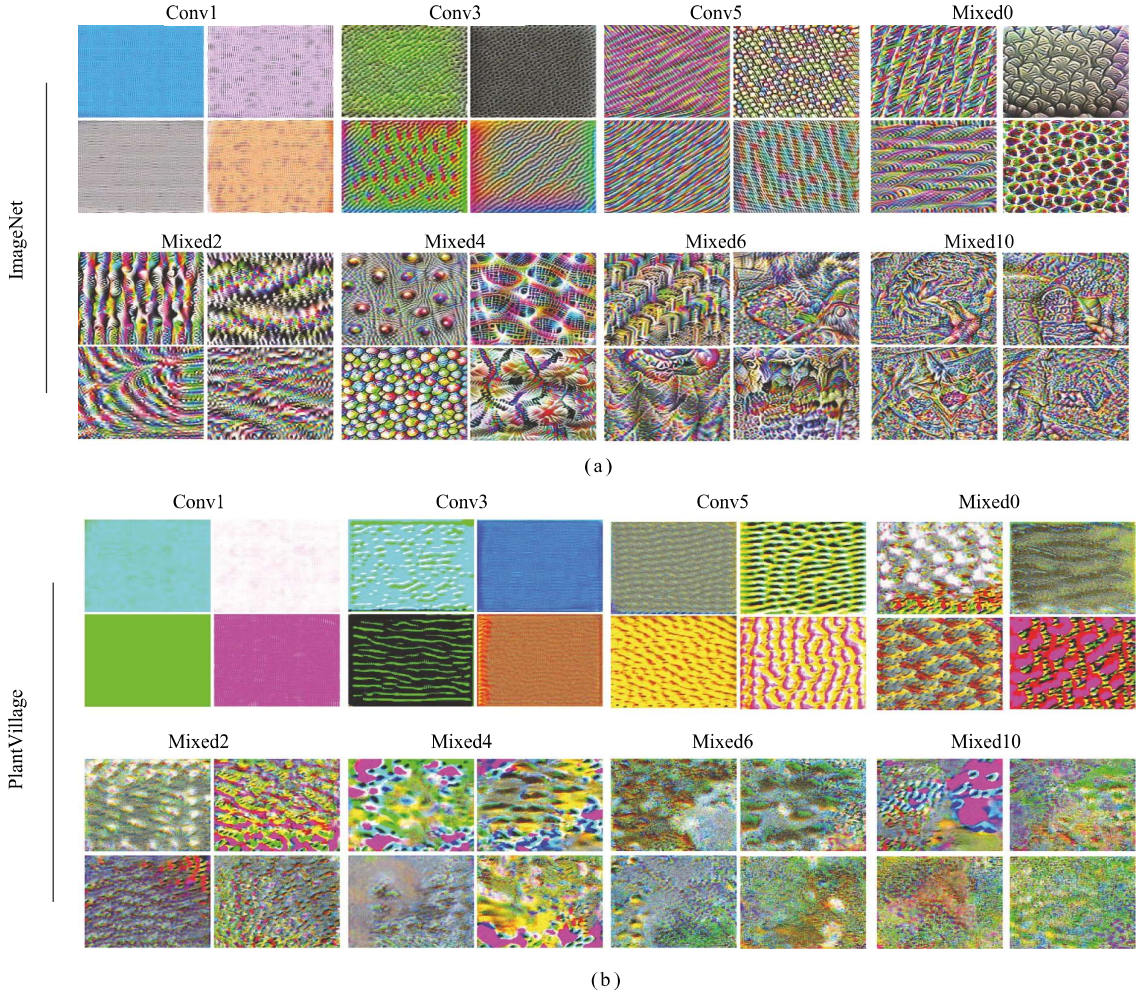

Fig 8.6 Feature visualization.

Visualization II:

Feature Visualization. Visualizations by the feature visualization method [40] applied on ImageNet and Plant Village datasets are shown in Figure 3. For the ImageNet dataset, when feature visualization was applied to neurons in the shallow layers (Conv1, Conv3, and Conv5), images containing simple patterns and textures were generated. In deeper layers (Mixed0, Mixed2, and Mixed4), both the complexity of the shape and the diversity of colors increased, forming various regular patterns or object-like appearances. In even deeper layers (Mixed 6, Mixed10), several objects were intermixed, resulting in an appearance similar to abstract paintings. The increasing Shannon-entropy values [50] of the images in proportion to the depth

of the network also suggest the increasing complexity (Figure S3). Thus, feature visualization can highlight the hierarchical features of what the CNN has learned. Since similar results were previously reported using the same method against neurons of the ImageNet-trained Google Net [41], we confirmed that the ability of the CNN is invariant from its architecture.

For the Plant Village dataset (Figure 3(a), right), similar to the ImageNet-trained network, neurons in the shallow layer generated simple textures (Conv1 to Conv5). Although the complexity of the image increased, it favored an edgeless abstract pattern comprised of a limited number of colors, in contrast to that of ImageNet-trained neurons. Since Plant Village is a dataset whose images consist of a single leaf in a uniform background, learning only the green, yellow, and brown colors may have been sufficient to describe the features of the leaves and their lesions, while pink and blue are considered background colors. Moreover, the overall edgeless and obscure images resemble the visual cues of the lesions. Foliar symptoms of lesions caused by pathogens are characterized by their colors and textures rather than shapes and sizes because the shapes and sizes are often indeterminate, and feature learning of plant diseases is possibly prioritized by the colors and textures. Collectively, feature visualization can provide an implementation of the lesion features that the neurons of the CNN have learnt. However, it is unknown if the neuron has an important role upon inference. Combination with the input data, such as semantic dictionary described below, will allow further interpretability of the network.

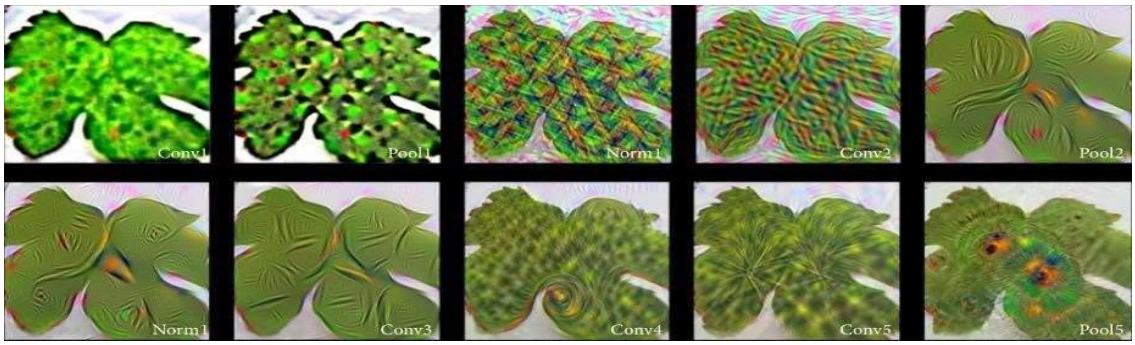

Fig 8.7 Output layer images

Chapter 9
Screenshots of Project

```
import numpy as np
import pickle
import cv2
from os import listdir
from sklearn.preprocessing import LabelBinarizer
from keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing import image
from keras_preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

```
EPOCHS = 25
INIT_LR = 1e-3
BS = 32
default_image_size = tuple((256, 256))
image_size = 0
directory_root = 'PlantVillage4'
width=256
height=256
depth=3
```

```
def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        if image is not None :
            image = cv2.resize(image, default_image_size)
            return img_to_array(image)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None
```

```
image_list, label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        # remove .DS_Store from list
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for plant_folder in root_dir :
        plant_disease_folder_list = listdir(f"{directory_root}/{plant_folder}")

        for disease_folder in plant_disease_folder_list :
            # remove .DS_Store from list
            if disease_folder == ".DS_Store" :
                plant_disease_folder_list.remove(disease_folder)

        for plant_disease_folder in plant_disease_folder_list:
            print(f"[INFO] Processing {plant_disease_folder} ...")
            plant_disease_image_list = listdir(f"{directory_root}/{plant_folder}/{plant_disease_folder}/")

            for single_plant_disease_image in plant_disease_image_list :
                if single_plant_disease_image == ".DS_Store" :
                    plant_disease_image_list.remove(single_plant_disease_image)

            for image in plant_disease_image_list[:200]:
                image_directory = f"{directory_root}/{plant_folder}/{plant_disease_folder}/{image}"
                if image_directory.endswith(".jpg") == True or image_directory.endswith(".JPG") == True:
                    image_list.append(convert_image_to_array(image_directory))
                    label_list.append(plant_disease_folder)
    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")
```

```
[INFO] Loading images ...
[INFO] Processing Apple___Apple_scab ...
[INFO] Processing Apple___Black_rot ...
[INFO] Processing Apple___Cedar_apple_rust ...
[INFO] Processing Apple___healthy ...
[INFO] Processing Blueberry___healthy ...
[INFO] Processing Cherry_(including_sour)___healthy ...
[INFO] Processing Cherry_(including_sour)___Powdery_mildew ...
[INFO] Processing Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot ...
[INFO] Processing Corn_(maize)___Common_rust_ ...
[INFO] Processing Corn_(maize)___healthy ...
[INFO] Processing Corn_(maize)___Northern_Leaf_Blight ...
[INFO] Processing Grape___Black_rot ...
[INFO] Processing Grape___Esca_(Black_Measles) ...
[INFO] Processing Grape___healthy ...
[INFO] Processing Grape___Leaf_blight_(Isariopsis_Leaf_Spot) ...
[INFO] Processing Orange___Haunglongbing_(Citrus_greening) ...
[INFO] Processing Peach___Bacterial_spot ...
[INFO] Processing Peach___healthy ...
[INFO] Processing Pepper,_bell___Bacterial_spot ...
[INFO] Processing Pepper,_bell___healthy ...
[INFO] Processing Potato___Early_blight ...
[INFO] Processing Potato___healthy ...
[INFO] Processing Potato___Late_blight ...
[INFO] Processing Raspberry___healthy ...
[INFO] Processing Soybean___healthy ...
[INFO] Processing Squash___Powdery_mildew ...
[INFO] Processing Strawberry___healthy ...
[INFO] Processing Strawberry___Leaf_scorch ...
[INFO] Processing Tomato___Bacterial_spot ...
[INFO] Processing Tomato___Early_blight ...
[INFO] Processing Tomato___healthy ...
[INFO] Processing Tomato___Late_blight ...
[INFO] Processing Tomato___Leaf_Mold ...
[INFO] Processing Tomato___Septoria_leaf_spot ...
[INFO] Processing Tomato___Spider_mites Two-spotted_spider_mite ...
[INFO] Processing Tomato___Target_Spot ...
[INFO] Processing Tomato___Tomato_mosaic_virus ...
[INFO] Processing Tomato___Tomato_Yellow_Leaf_Curl_Virus ...
[INFO] Image loading completed
```

```
image_size = len(image_list)
```

```
label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
pickle.dump(label_binarizer,open('label_transform.pkl', 'wb'))
n_classes = len(label_binarizer.classes_)
```

```
print(label_binarizer.classes_)
```

```
['Apple___Apple_scab' 'Apple___Black_rot' 'Apple___Cedar_apple_rust'
 'Apple___healthy' 'Blueberry___healthy'
 'Cherry_(including_sour)___Powdery_mildew'
 'Cherry_(including_sour)___healthy'
 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot'
 'Corn_(maize)___Common_rust_' 'Corn_(maize)___Northern_Leaf_Blight'
 'Corn_(maize)___healthy' 'Grape___Black_rot'
 'Grape___Esca_(Black_Measles)'
 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)' 'Grape___healthy'
 'Orange___Haunglongbing_(Citrus_greening)' 'Peach___Bacterial_spot'
 'Peach___healthy' 'Pepper,_bell___Bacterial_spot'
 'Pepper,_bell___healthy' 'Potato___Early_blight' 'Potato___Late_blight'
 'Potato___healthy' 'Raspberry___healthy' 'Soybean___healthy'
 'Squash___Powdery_mildew' 'Strawberry___Leaf_scorch'
 'Strawberry___healthy' 'Tomato___Bacterial_spot' 'Tomato___Early_blight'
 'Tomato___Late_blight' 'Tomato___Leaf_Mold' 'Tomato___Septoria_leaf_spot'
 'Tomato___Spider_mites Two-spotted_spider_mite' 'Tomato___Target_Spot'
 'Tomato___Tomato_Yellow_Leaf_Curl_Virus' 'Tomato___Tomato_mosaic_virus'
 'Tomato___healthy']
```

```
np_image_list = np.array(image_list, dtype=np.float16) / 225.0
```

```
print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_size=0.2, random_state = 42)
```

```
[INFO] Spliting data to train, test
```

```
aug = ImageDataGenerator(
    rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2,
    zoom_range=0.2,horizontal_flip=True,
    fill_mode="nearest")
```

```
model = Sequential()
inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (depth, height, width)
    chanDim = 1
model.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 256, 256, 32)      896

 activation_7 (Activation)   (None, 256, 256, 32)      0

 batch_normalization_6 (Batc  (None, 256, 256, 32)     128
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 85, 85, 32)       0
 2D)

 dropout_4 (Dropout)         (None, 85, 85, 32)        0

 conv2d_6 (Conv2D)           (None, 85, 85, 64)        18496

 activation_8 (Activation)   (None, 85, 85, 64)        0

 batch_normalization_7 (Batc  (None, 85, 85, 64)       256
 hNormalization)

 conv2d_7 (Conv2D)           (None, 85, 85, 64)        36928

 activation_9 (Activation)   (None, 85, 85, 64)        0

 batch_normalization_8 (Batc  (None, 85, 85, 64)       256
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 42, 42, 64)       0
 2D)

 dropout_5 (Dropout)         (None, 42, 42, 64)        0

 conv2d_8 (Conv2D)           (None, 42, 42, 128)       73856

 activation_10 (Activation)  (None, 42, 42, 128)       0
```

```
activation_9 (Activation)   (None, 85, 85, 64)        0
batch_normalization_8 (Batc  (None, 85, 85, 64)       256
hNormalization)
max_pooling2d_4 (MaxPooling  (None, 42, 42, 64)       0
2D)
dropout_5 (Dropout)         (None, 42, 42, 64)        0
conv2d_8 (Conv2D)           (None, 42, 42, 128)       73856
activation_10 (Activation)  (None, 42, 42, 128)       0
batch_normalization_9 (Batc  (None, 42, 42, 128)      512
hNormalization)
conv2d_9 (Conv2D)           (None, 42, 42, 128)       147584
activation_11 (Activation)  (None, 42, 42, 128)       0
batch_normalization_10 (Bat  (None, 42, 42, 128)      512
chNormalization)
max_pooling2d_5 (MaxPooling  (None, 21, 21, 128)      0
2D)
dropout_6 (Dropout)         (None, 21, 21, 128)       0
flatten_1 (Flatten)         (None, 56448)             0
dense_2 (Dense)             (None, 1024)              57803776
activation_12 (Activation)  (None, 1024)              0
batch_normalization_11 (Bat  (None, 1024)             4096
chNormalization)
dropout_7 (Dropout)         (None, 1024)              0
dense_3 (Dense)             (None, 38)                38950
activation_13 (Activation)  (None, 38)                0
=================================================================
Total params: 58,126,246
Trainable params: 58,123,366
Non-trainable params: 2,880
_________________________________________________________________
```

```
opt = Adam(learning_rate=INIT_LR, decay=INIT_LR / EPOCHS)
# distribution
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=["accuracy"])
# train the network
print("[INFO] training network...")
```

```
history = model.fit(
    aug.flow(x_train, y_train, batch_size=BS),
    validation_data=(x_test, y_test),
    steps_per_epoch=len(x_train) // BS,
    epochs=EPOCHS, verbose=1
    )
```

```
Epoch 1/25
188/188 [==============================] - 544s 3s/step - loss: 0.3092 - accuracy: 0.1573 - val_loss: 0.3299 - val_accuracy: 0.0549
Epoch 2/25
188/188 [==============================] - 531s 3s/step - loss: 0.0783 - accuracy: 0.4633 - val_loss: 0.2839 - val_accuracy: 0.0702
Epoch 3/25
188/188 [==============================] - 524s 3s/step - loss: 0.0639 - accuracy: 0.5841 - val_loss: 0.9973 - val_accuracy: 0.0642
Epoch 4/25
188/188 [==============================] - 4186s 22s/step - loss: 0.1104 - accuracy: 0.4107 - val_loss: 0.1454 - val_accuracy: 0.3044
Epoch 5/25
188/188 [==============================] - 641s 3s/step - loss: 0.0798 - accuracy: 0.5458 - val_loss: 0.4386 - val_accuracy: 0.1092
Epoch 6/25
188/188 [==============================] - 610s 3s/step - loss: 0.0668 - accuracy: 0.6088 - val_loss: 0.1736 - val_accuracy: 0.2535
Epoch 7/25
188/188 [==============================] - 551s 3s/step - loss: 0.0548 - accuracy: 0.6815 - val_loss: 0.0563 - val_accuracy: 0.6539
Epoch 8/25
188/188 [==============================] - 557s 3s/step - loss: 0.0523 - accuracy: 0.6975 - val_loss: 0.1031 - val_accuracy: 0.4712
Epoch 9/25
188/188 [==============================] - 555s 3s/step - loss: 0.0453 - accuracy: 0.7409 - val_loss: 0.0865 - val_accuracy: 0.5195
Epoch 10/25
188/188 [==============================] - 547s 3s/step - loss: 0.0430 - accuracy: 0.7545 - val_loss: 0.0700 - val_accuracy: 0.6062
Epoch 11/25
188/188 [==============================] - 545s 3s/step - loss: 0.0420 - accuracy: 0.7562 - val_loss: 0.0676 - val_accuracy: 0.5983
Epoch 12/25
188/188 [==============================] - 552s 3s/step - loss: 0.0363 - accuracy: 0.7877 - val_loss: 0.1147 - val_accuracy: 0.4394
Epoch 13/25
188/188 [==============================] - 553s 3s/step - loss: 0.0344 - accuracy: 0.8103 - val_loss: 0.0492 - val_accuracy: 0.7459
Epoch 14/25
188/188 [==============================] - 525s 3s/step - loss: 0.0324 - accuracy: 0.8209 - val_loss: 0.1058 - val_accuracy: 0.5295
Epoch 15/25
188/188 [==============================] - 5097s 27s/step - loss: 0.0307 - accuracy: 0.8291 - val_loss: 0.0599 - val_accuracy: 0.6737
Epoch 16/25
188/188 [==============================] - 557s 3s/step - loss: 0.0291 - accuracy: 0.8446 - val_loss: 0.0403 - val_accuracy: 0.7545
Epoch 17/25
188/188 [==============================] - 546s 3s/step - loss: 0.0280 - accuracy: 0.8484 - val_loss: 0.0465 - val_accuracy: 0.7320
Epoch 18/25
188/188 [==============================] - 554s 3s/step - loss: 0.0254 - accuracy: 0.8605 - val_loss: 0.1119 - val_accuracy: 0.4719
Epoch 19/25
188/188 [==============================] - 559s 3s/step - loss: 0.0307 - accuracy: 0.8278 - val_loss: 0.0415 - val_accuracy: 0.7882
Epoch 20/25
188/188 [==============================] - 548s 3s/step - loss: 0.0231 - accuracy: 0.8798 - val_loss: 0.0612 - val_accuracy: 0.7247
Epoch 21/25
188/188 [==============================] - 551s 3s/step - loss: 0.0229 - accuracy: 0.8830 - val_loss: 0.0302 - val_accuracy: 0.8293
Epoch 22/25
188/188 [==============================] - 549s 3s/step - loss: 0.0213 - accuracy: 0.8900 - val_loss: 0.0897 - val_accuracy: 0.5460
Epoch 23/25
188/188 [==============================] - 565s 3s/step - loss: 0.0261 - accuracy: 0.8565 - val_loss: 0.1130 - val_accuracy: 0.6281
Epoch 24/25
188/188 [==============================] - 1167s 6s/step - loss: 0.0274 - accuracy: 0.8514 - val_loss: 0.0763 - val_accuracy: 0.6056
Epoch 25/25
188/188 [==============================] - 527s 3s/step - loss: 0.0211 - accuracy: 0.8882 - val_loss: 0.0226 - val_accuracy: 0.8690
```

```
model.save("A4Model")
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convo
INFO:tensorflow:Assets written to: A4Model\assets
INFO:tensorflow:Assets written to: A4Model\assets
```

```
from tensorflow.keras import models
model1=models.load_model("A4Model")
```

```
scores=model1.evaluate(x_test,y_test)
print(f"test accuracy:{scores[1]*100}")
```

```
48/48 [==============================] - 26s 508ms/step - loss: 0.0226 - accuracy: 0.8690
test accuracy:86.8960976600647
```

```
from IPython.display import Image
Image('tomato_early_blight.jpg')
```

```
img = convert_image_to_array('tomato_early_blight.jpg')

img = np.array(img, dtype=np.float16) / 225.0

img.resize(1,256,256,3)
```

```
res = model1.predict(img)

res
```

```
1/1 [==============================] - 0s 334ms/step
array([[1.0363300e-04, 1.9756721e-09, 1.7499013e-05, 6.0898357e-05,
        8.1661959e-08, 1.7525576e-04, 2.7992891e-10, 3.7435399e-05,
        9.2543546e-09, 6.2926460e-07, 6.2718142e-09, 8.3126770e-06,
        3.4481825e-06, 6.0898792e-08, 1.4985264e-09, 3.1598276e-08,
        3.0459096e-05, 2.5371494e-07, 4.0742606e-07, 1.2857275e-06,
        1.2596735e-07, 6.1083148e-05, 2.5671449e-07, 1.2863823e-08,
        3.3296099e-06, 8.0820563e-08, 8.0156745e-08, 1.1265482e-09,
        8.4793748e-05, 9.9907839e-01, 2.6864748e-04, 4.0490562e-05,
        8.3469459e-08, 8.4574358e-06, 5.8658219e-07, 1.3671281e-05,
        1.1873181e-07, 9.5947454e-09]], dtype=float32)
```

```
print(label_binarizer.inverse_transform(res))
```

```
['Tomato___Early_blight']
```

Chapter 10
Conclusion

Plant Diseases are major food threats that should have to overcome before it leads to further loss of the entire field. But often framers unable to distinguish between similar symptoms but ace different diseases. This will mislead to wrong or overdosage of fertilizers.

In this project, an approach of using deep learning method was explored to automatically classify and detect plant diseases from leaf images.
The complete procedure consists of collecting the images used for training and validation, image pre-processing and augmentation and finally the procedure of training the deep CNN and fine-tuning. Different tests were performed to check the performance of newly created model.

10.1 Future Scope

The forecasting of diseases in early stage, so that appropriate measures can be taken to minimize the low crops
Our project has shown pretty good accuracy, it can be implemented in real time mobile applications and web services so that farmers can identify diseases simply by taking photo of suspected leaves of plants
Recommendation of chemicals and their ratio to control the further spread of diseases on the different parts of plants after the proper identification of diseases
Other than plant leaf disease identification, it can also be used for identification and classification of nutrients deficiency of plant leaves.
Creating and training a CNN model from scratch is a tedious process, this model can be used to detect and classification of other plant disease too, by simply training the model using respected datasets.

References

[1] Robert G. de Luna, Elmer P. Dadios, Argel A. Bandala, "Automated Image Capturing System for Deep Learning-based Tomato Plant Leaf Disease Detection and Recognition," International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD) 2019.

[2] Suma VR Amog Shetty, Rishab F Tated, Sunku Rohan, Triveni S Pujar, "CNN based Leaf Disease Identification and Remedy Recommendation System," IEEE conference paper 2019.

[3] Peng Jiang, Yuehan Chen, Bin Liu, Dongjian He, Chunquan Liang, "Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolution Neural Networks," IEEE ACCESS 2019.

[4] Geetha Ramani, Arun Pandian, "Identification of plant leaf diseases using a nine- layer deep convolution neural network," Computers and Electrical Engineering 76 (2019).

[5] Robert G. de Luna, Elmer P. Dadios, Argel A. Bandala, "Automated Image Capturing System for Deep Learning-based Tomato Plant Leaf Disease Detection and Recognition," Proceedings of TENCON 2018 - 2018 IEEE Region 10 Conference.

[6] Omkar Kulkarni, "Crop Disease Detection Using Deep Learning," IEEE access 2018.

[7] Abirami Devaraj, Karunya Rathan, Sarvepalli Jaahnavi and K Indira, "Identification of Plant Disease using Image Processing Technique," International Conference on Communication and Signal Processing, IEEE 2019.

[8] Velamakanni Sahithya, Brahmadevara Sai Vihari, Vellanki Krishna Vamsi, ParvathReddy Sandeep Reddy and Karthigha Balamurugan, "GUI based Detection of Unhealthy Leaves using Image Processing Techniques," International Conference on Communication and Signal Processing 2019.

[9]https://www.kaggle.com/datasets/arjuntejaswi/plantvillage/download?datasetVersionNumber=1