DR. PLANT
LEAF DISEASE DETECTION SYSTEM

A
Major Project Report Submitted
In partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
Information Technology

by

A. Ayushendra Prasad	**19N31A1210**
B. Tharun	**20N35A1201**
A. Varun Sai Reddy	**19N31A1202**

Under the esteemed guidance of

Mr. Uma Maheswara Rao
Assoc. Professor

Department of Information Technology

Malla Reddy College of Engineering & Technology
(Autonomous Institution- UGC, Govt. of India)
(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A'Grade)
Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100
website: www.mrcet.ac.in

2019-2023

Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A'Grade)

Maisammaguda, Kompally, Dhulapally,Secunderabad – 500100

website: www.mrcet.ac.in

CERTIFICATE

This is to certify that this is the bonafide record of the project entitled **"Dr. Plant leaf disease detection system"**, submitted by **A. Ayushendra Prasad(19N31A1210)**, **B. Tharun(20N35A1201)** and **A. Varun Sai Reddy (19N31A1202)** of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Information Technology, Department of IT during the year 2022-2023. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide	**Head of the Department**
Mr. Uma Maheswara Rao	**Dr. G.Sharada**
Assoc. Professor	**Professor**

External Examiner

DECLARATION

We hereby declare that the project titled "Project Title" submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Information Technology is a result of original research carried-out in this report. It is further declared that the project report or any partthereof has not been previously submitted to any University or Institute for the award of degree or diploma.

A. Ayushendra Prasad- 19N31A1210
B. Tharun - 20N35A1201
A. Varun Sai Reddy - 19N31A1202

<u>ACKNOWLEDGEMENT</u>

We feel honored to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous) for giving us an opportunity to do this Project as part of our B.Tech Program. We are ever grateful to our Director **Dr. VSK Reddy** and Principal **Dr.S.Srinivas Rao** who enabled us to have experience in engineering and gain profound technical knowledge.

We express our heartiest thanks to our HOD, **Dr. G. Sharada** for encouraging us in every aspect of our course and helping us realize our full potential.

We would like to thank our Project Guide **Mr. Uma Maheswara Rao** for his regular guidance, suggestions and constant encouragement. We are extremely grateful to our Project Coordinator **Ms. N. Prameela** for her continuous monitoring and unflinching co-operation throughout project work.

We would like to thank our Class In charge **Ms. N.Prameela** who in spite of being busy with her academic duties took time to guide and keep us on the correct path.

We would also like to thank all the faculty members and supporting staff of the Department of IT and all other departments who have been helpful directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for thecompletion of our project that gave us strength to do our project.

With regards and gratitude

A. Ayushendra Prasad	**- 19N31A1210**
B. Tharun	**- 20N35A1201**
A. Varun Sai Reddy	**- 19N31A1202**

ABSTRACT

Crop production plays a significant role in the agricultural sector. The loss of food is primarily attributed, in particular, to contaminated crops, which reflexively decreases the rate of development. The detection of plant disease within the field of agriculture is extremely difficult.

When identification is incorrect then the assembly of the product and the market's economic value will suffer a significant loss.

This research involves a new approach to model identification of plant diseasesgrowth through the use of large, convolutional networks, based on the classification ofthe leaf image. Novel approach and technique used to promote the easy and simple implementation of the program in observance.

The developed model is capable of identifying thirteen completely different kinds of plant diseases from healthy leaves, with the flexibility to tell apart from the surrounding plant leaves. This technique for the identification of diseases was projected for the first time according to our knowledge.

All necessary steps were taken by agricultural consultants to incorporate this disease recognition model area unit pictured during the project, beginning with the collection of photographs to make details. We prefer to use Python & PyCham to do the deep CNN process.

TABLE OF CONTENTS

1.INTRODUCTION

1.1 PURPOSE AND OBJECTIVES

Dr. Plant is an innovative application that performs quantitative assessments of plant diseases on plant organs such as leaves. Users can collect or submit photographs of diseased plant organs and calculate the percentage of diseased tissue. The application employs user-specified values for up to eight colors of healthy tissues in the photograph. The color of each pixel is then evaluated for its distance from the healthy colors and assigned a status of either healthy or diseased. The threshold bar allows users to slide until they are satisfied that diseased tissues are accurately represented before the percentage calculation. The assessment data and photographs may be sent by e-mail to any recipients, making it easy to share the information with others. Instructions for using the application are provided on the About screen, making it easy for users to get started.

Dr. Plant allows users to identify and better understand all kinds of plants living in nature, including flowering plants, trees, grasses, conifers, ferns, vines, wild salads, or cacti. The application can also identify a large number of cultivated plants in parks and gardens, but this is not its primary purpose. Users can observe plants in nature or in the middle of their vegetable gardens or on sidewalks in the cities.

The more visual information a plant has, the more accurate the identification will be. Many plants look alike from afar, and it is sometimes small details that distinguish two species of the same genus. Flowers, fruits, and leaves are the most characteristic organs of a species, and users should photograph them first. But any other detail can be useful, such as thorns, buds, or hair on the stem. A photograph of the whole plant or tree, if it is one, is also very useful information, but it is often not sufficient to allow a reliable identification.

Dr. Plant's approach to assessing plant diseases is innovative and effective. The application employs advanced algorithms to evaluate the color of each pixel in a photograph and assign a status of healthy or diseased. This approach is effective because it takes into account the range of healthy colors, and it allows users to adjust the threshold until they are satisfied that the results are accurate. The ability to share the assessment data and photographs by e-mail is also a valuable feature, as it allows users to collaborate with others and get feedback on their findings.

Dr. Plant's ability to identify plants in nature is also a valuable feature. The application can identify a wide range of plant species, including those that grow on the sidewalks of our cities or in the middle of vegetable gardens. This feature is particularly useful for botanists, horticulturists, and anyone interested in learning more about the plants in their surroundings. The ability to identify plants accurately is essential for understanding their characteristics, uses, and potential hazards.

In conclusion, Dr. Plant is an innovative application that performs quantitative assessments of plant diseases and identifies all kinds of plants living in nature. The application's advanced algorithms, user-friendly interface, and ability to share data make it a valuable tool for botanists, horticulturists, and anyone interested in plants. The more visual information a plant has, the more accurate the identification will be, and users should photograph the flowers, fruits, and leaves of a plant first. The ability to identify plants accurately is essential for understanding their characteristics, uses, and potential hazards, making Dr. Plant an essential tool for plant enthusiasts.

1.2 EXISTING AND PROPOSED SYSTEM

EXISTING SYSTEM:

The problem of preventing economic diseases is tightly tied to issues of sustainable forestry, and climate change. Recent findings suggest that weather change will alter stages and levels of production of infectious agents; host tolerance may also be affected, resulting in physiological control of host interactions.

Additionally, the situation is exacerbated by the very actuality that more (easily) than ever, nowadays, diseases are being transmitted globally. New diseases will occur where they were previously unidentified and inevitably, and where there is no local expertise to combat them. Inexperienced use of pesticides can cause the pathogens to develop semi- resistance, significantly reducing their ability to fight back. One of the foundations of exactness in agriculture is the prompt and accurate diagnosis of plant diseases.

Forestalling needless waste of economic and different resources is vital, so achieving sustainable growth, addressing the downside of the long- creation of infectious agent resistance and reducing the negative effects ofclimate change.
In this complex climate, appropriate and timely identification of the disease, including early bars, was never additionally vital.

There are several methods of identifying pathologies in plants. Many illnesses have no obvious effects, or the effect is evident too late to respond, so complicated diagnosis is mandatory in certain cases.

Nonetheless, most diseases produce some form of manifestation in the spectrum, and a qualified professional's Oculus test is the first technique adopted for disease detection. A plant specialist should possess sensitive observation skills in order to achieve correct disease medicine and one can evaluate characteristic symptoms.

PROPOSED SYSTEM:

We designed an application to encourage plant health through the look of the plant, and visual symptoms can be useful for amateurs in the agriculture system aswell as trained practitioners as a verification tool in disease medicine. Advances in laptop vision offer an opportunity to extend and develop accurate plant safety monitoring and to broaden the demand for computer vision applications in the field ofprecision farming. A popular digital image process technique such as color analyzes and thresholds [9] were used to detect and classify plant diseases. There are actually various approaches for occupational detective diseases, and most styles are artificial neural networks (ANNs) [10] and support vector machines (SVMs). They are combined with slightly different ways of pre-processing the image in terms of extracting higher features. The brain is comprised of a large number of highly interconnected neurons working together to solve particular problems. A human- generated nerve cell can be a part of a process with multiple inputs and one output.With each input, the nerve cell often has weights that are related to a general bias.

Using offline raise this dataset is recreated from the original data collection. The original dataset can be found at this github folder. This dataset consists of roughly 87 Kb images of healthy and diseased crop leaves that are divided into 38 categories.The overall dataset is divided into an 80/20 ratio collection of training and validation which retains the directory structure. A new directory, containing 33 test images, is later generated for predictive purposes. Convolutional Neural networks or covnets are neural networks with common parameters.

Types of layers:

1. Input Layer
2. Convolution Layer
3. Activation Function Layer
4. Pool Layer
5. Fully-Connected Layer

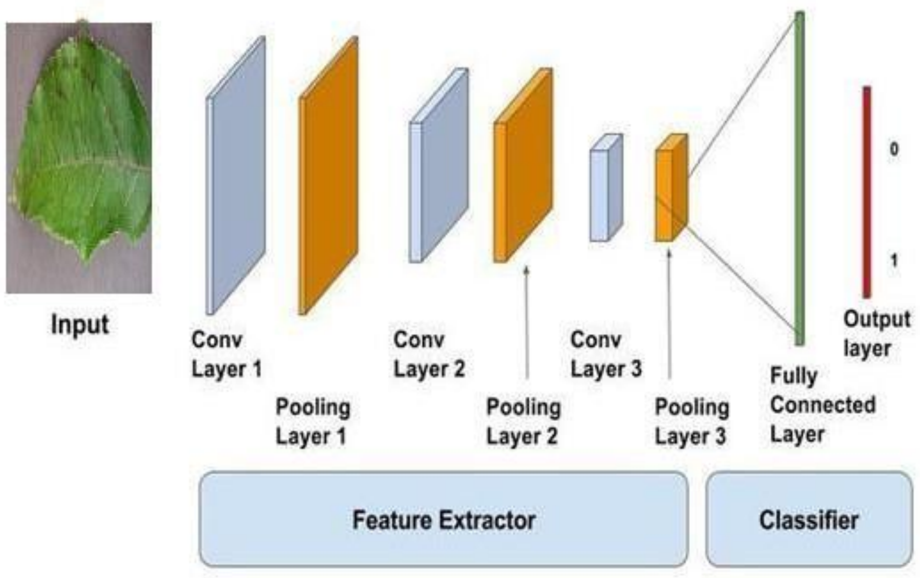


Fig.1.1 Diagram that shows flow of our model

Input Layer: This layer contains the image's raw input to width 32, height 32 and depth 3.
Convolution Layer: This layer calculates the volume of the output by calculating the dot product between all filters and patching the image. Suppose we use a total of 12 filters to get output volume of dimension 32 x 32 x 12 for this sheet.
Activation Function Layer: This layer applies element wise activation function to convolution layer performance. Some a activation functions include RELU: max (0, x), Sigmoid: 1/(1+e^-x), Tanh, Leaky RELU, etc.

1.3 SCOPE OF PROJECT

Plant diseases in the world's agricultural industry are causing significant growth and economic losses. Health monitoring and the identification of disease in plants and trees is critical for sustainable agriculture. To the best of our knowledge there is no commercially available sensor for real-time assessment of health conditions in trees. Scouting is probably the most widely used method of measuring stress in trees, and is a costly, labor-intensive, and time consuming process. Convolutional neural networks.

(CNNs) are used for the identification of plant diseases that require extensive sampling and processing. Awareness of early crop health and disease detection may promote disease control through successful management techniques such as vector control using pesticides, fungicide applications, and disease-specific chemical applications; and can increase productivity. We offer a method for detecting diseases in the plant leaf and test it.

2. LITERATURE SURVEY

Once the author’s review of their work and results had been provided, it was entirely up to us to use the image processing disease recognition methodamong various methods [1] widely used in plant disease medicine, such as double-stranded RNA review, organic compound samples, and results.

Different unit of measurement techniques currently in use for computer vision identification of plant diseases. One is the identification of sickness by extracting color features as provided by authors. In the study disease spots and noise from entirely different sources like camera flash were successfully detected [2 - 4].

The identification of plant diseases can also be accomplished by removing the technique of type choices. Patil and Bodhe applied detection of this disease method to sugarcane leaves everywhere they wanted to use for achieving the typical accuracy of ninetyeight.60 percent in the final experiments.

In addition, the extraction of texture characteristics may be used in plantdisease detection [5]. Patil and Kumar have proposed a model for victimizationtexture detection of plant disease selections for example, the inertia, homogeneity and correlation obtained by calculating the gray level matrix co- occurring on the image.

They experimented with the identification of diseases on maize leaves inconjunction with color extraction. A combination of these decisions offers a strong collection of image enhancement functionality and higher classification . In, the writers conferred a survey of well-known commonplace ways of extracting functions. Thanks to the rapid advancement of AI (AI) research, adding this paper is especially based on the application of these methodologiesand techniques.

Several ways of combining the extraction function and the Neural Network Ensemble (NNE) for unwellness detection, too. By coaching a particular form of neural networks and putting their tests together immediately, the north- northeast provides a much stronger generalization of humor. Only to identify herbal diseases with a final test accuracy of ninety-one was such a technique applied.

In our work we tend to use deep learning approaches to diagnose unwellness; inspired by the development of deep learning techniques and their practical application.

Several ways of combining the extraction function and the Neural Network Ensemble (NNE) for unwellness detection, too. By coaching a particular form of neural networks and putting their tests together immediately, the north- northeast provides a much stronger generalization of humor. Only to identify herbal diseases with a final test accuracy of ninety-one was such a technique applied .

The identification of plant diseases can also be accomplished by removing the technique of type choices. Patil and Bodhe applied detection of this disease method to sugarcane leaves everywhere they wanted to use for achieving the typical accuracy of ninetyeight.60 percent in the final experiments.

3. SYSTEM ANALYSIS

3.1 HARDWARE AND SOFTWARE REQUIREMENT

The hardware requirement specifies each interface of the software elements andthe hardware elements of the system. These hardware requirements include configuration characteristics.

- **Processor** : Intel i3 and any updated processor
- **Ram** : Min 4 GB
- **Hard Disk** : Min 100 GB

The software requirements specify the use of all required software products likedata management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

- **Operating System** : Windows 10
- **Technology** : Python 3

3.2 SOFTWARE REQUIREMENT SPECIFICATION:

Requirement Specification provides a high secure storage to the web server efficiently. Software requirements deal with software and hardware resources that need to be installed on a serve which provides optimal functioning for the application.

These software and hardware requirements need to be installed before the packages are installed. These are the most common set of requirements defined by any operation system. These software and hardware requirements provide a compatible support to the operation system in developing an application

4. SYSTEM DESIGN

4.1 DESCRIPTION

What is Waterfall Model?

Waterfall Model is a sequential model which divides the software development into different phases. Each stage is designed to perform specific operation during SDLC process. It was first made public in 1970 by Winston Royce.

Requirements:

The first step is to understand what needs to be prepared and what the role, meaning,and so on are. The parameters for input and output, or the end result, are evaluated and labelled here.

System Design:

The requirements from the first phase are reviewed during this phase, and the design of the device is planned. System Architecture helps to decide requirements for hardware and software, and also helps to describe the overall design of the system. Now the software code to be written in the next level is created.

Implementation:

The system is first developed in small programs called units, with system design inputbeing integrated into the next phase. Every unit is designed and tested for its functionality known as Unit Testing.

Integration and Testing:

Once each unit is tested, all the units built during the implementation process are incorporated into a system. The built-in software needs to go through continuous software testing to determine whether there are bugs or errors. Testing is carried outso that the system faces no problems during software delivery.

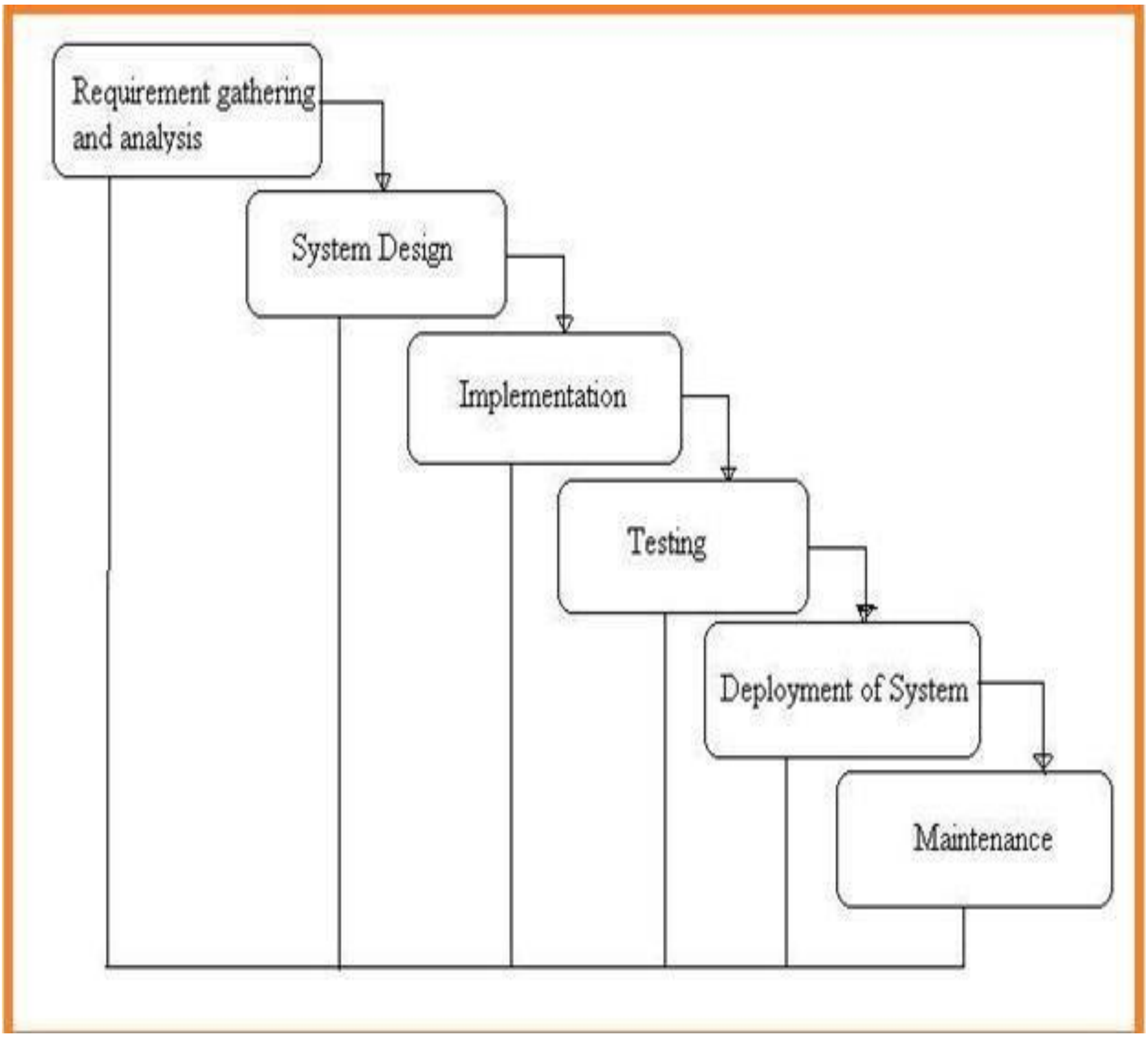
Deployment of system:

When the functional and non-functional testing is done, the product in the customer environment is either launched or released into the market.

Maintenance:

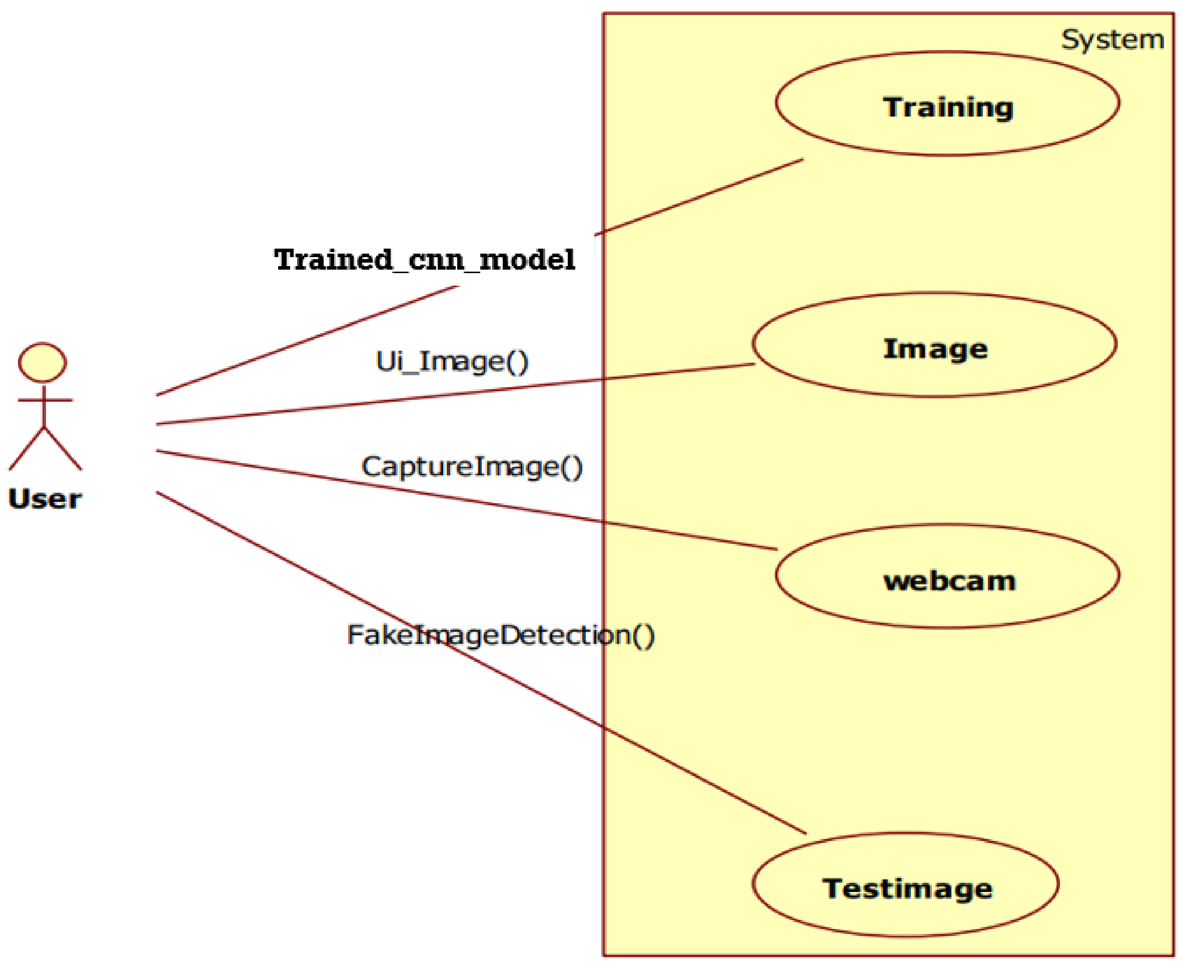
After installation, this process includes making system or individual component changes to alter attributes, or enhance performance. These improvements occur either from customer-initiated improvement requests, or from vulnerabilities discovered during live use of the program.

4.2 ARCHITECTURE:



4.3 UML DIAGRAMS

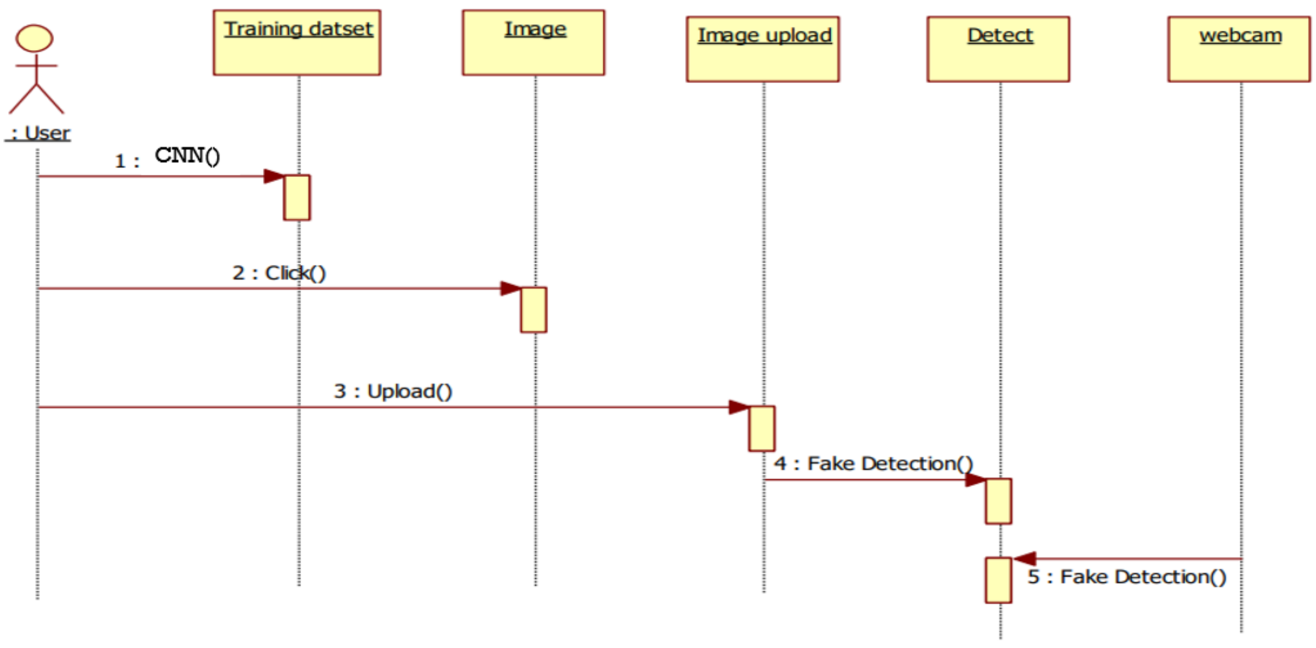
USE CASE DIAGRAM:



Description:

A use case diagram describes the interactions between interface components in agraphic form. A use case is a method used in the system analysis to describe, explain, and organize system specifications. The relationships between the actorsand use cases, and between them.

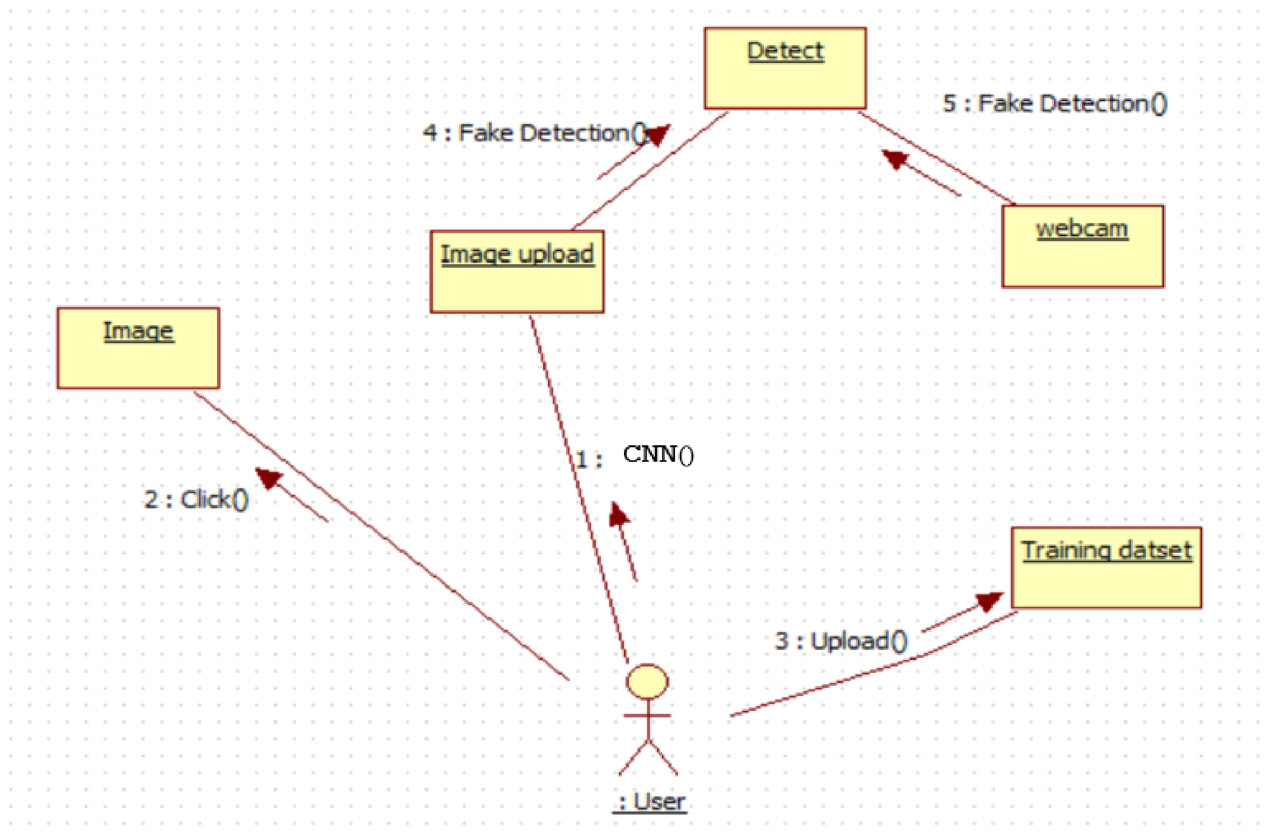
SEQUENCE DIAGRAM:



Description:

Sequence Diagrams are interaction diagrams which explain how operations are performed. They capture the interaction among objects within the context of collaboration. Phase Diagrams are time-based and display the order of the interaction visually, using the vertical axis of the diagram to indicate when messages are sent.

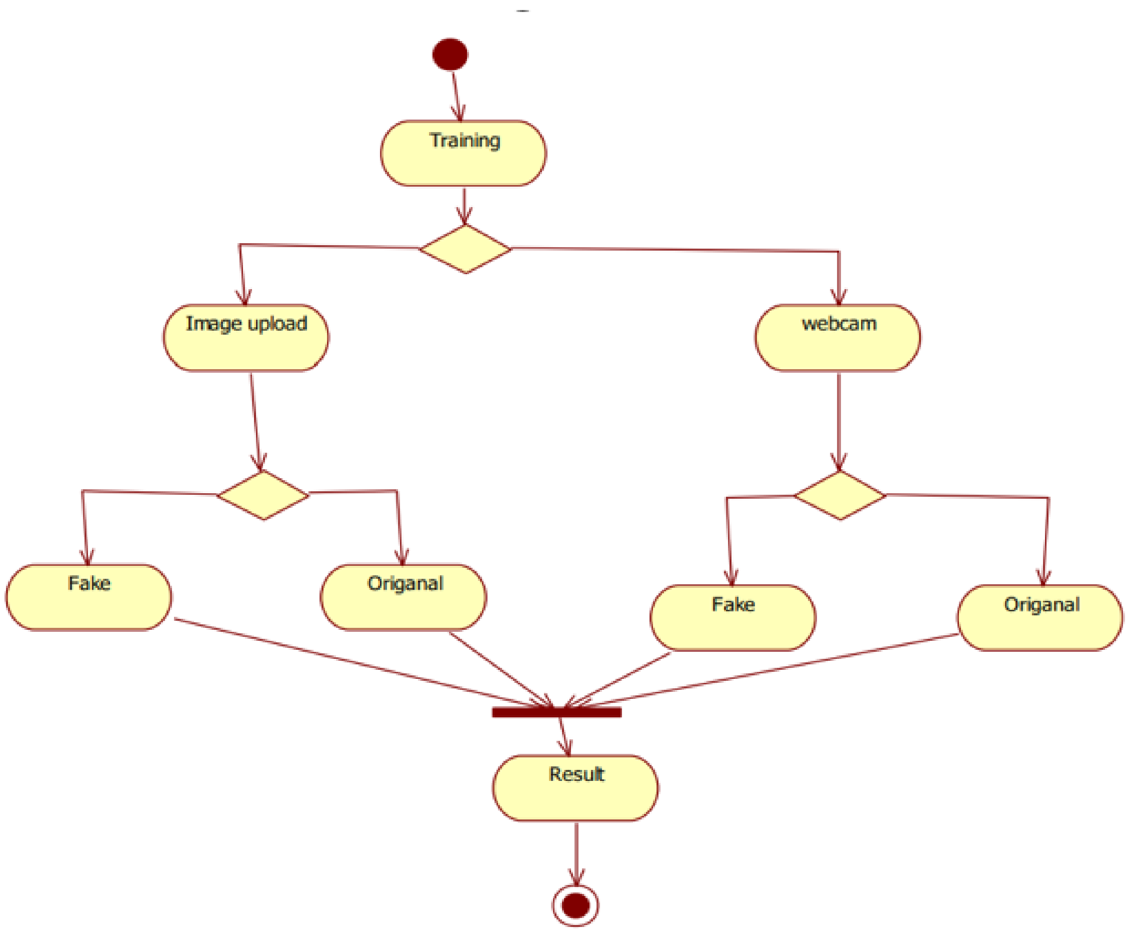
COLLABORATION DIAGRAM:



Description:

A collaboration diagram, also known as a touch diagram, is an example of the connections and interactions between software objects in the Unified Modeling Language (UML). These diagrams can be used to represent the complex behavior of a specific use case, and to define each entity's position

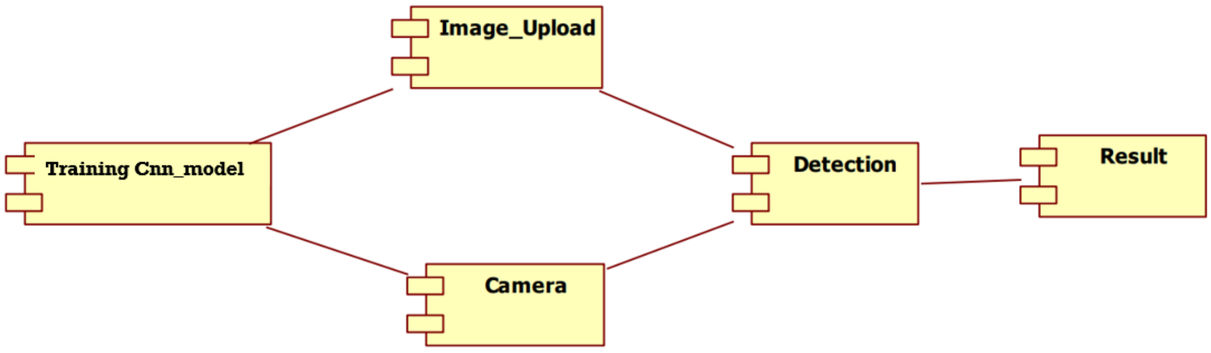
ACTIVITY DIAGRAM:



Description:

An event or activity diagram is a behavior diagram, that is, it describes an action of a machine. An action diagram displays the control flow from start to finish stage, illustrating the various decision paths that occur when performing the function.

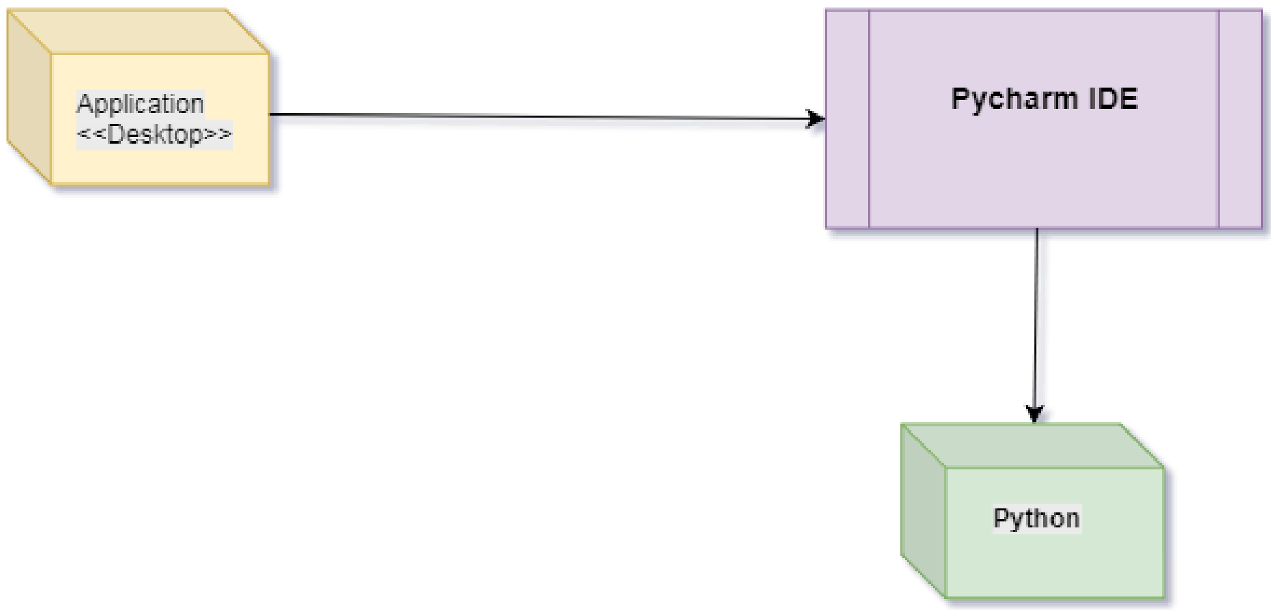
COMPONENT DIAGRAM:



Description:

A system diagram, also known as UML system diagram, describes how physical components are organized and cabled within a device. Part diagrams are also drawn to help model implementation details and double-check that the expected design covers every element of the required functions of the system.

DEPLOYMENT DIAGRAM:



Description:

A deployment diagram is a type of UML diagram showing the execution architecture of a system including nodes like hardware or software execution environments and the middleware connecting them. Usually deployment diagrams are used to represent the system and the actual hardware.

5. METHODOLOGY

5.1 TECHNOLOGIES USED:

PYTHON:

Python is a widely-used high-level programming language, that is known for its simplicity, readability, and ease of use. It was first released in 1991 and is continuously evolving with new features and libraries. Python's design philosophy emphasizes code readability, which means that the language is designed to be easy to read and write, reducing the time and effort needed to create and maintain code. This makes Python an excellent choice for beginners and professionals alike.

One of the key features of Python is its use of significant indentation to structure code blocks. This approach makes Python code more readable and reduces the need for braces and other syntax elements. Python's syntax is also simple and easy to learn, making it an excellent choice for beginners who want to learn programming.

Python is dynamically typed, which means that the data type of a variable is determined at runtime, rather than being specified in advance. This makes it easier to write code quickly and reduces the chances of type-related errors. Additionally, Python is garbage-collected, which means that memory management is taken care of automatically, freeing developers from the need to manage memory manually.

ANACONDA:

Anaconda is a distribution of the Python and R programming languages for scientific computing that is widely used by data scientists, researchers, and developers. It provides a comprehensive set of tools for scientific computing and data analysis and aims to simplify package management and deployment.

The Anaconda distribution includes a large number of pre-installed packages and tools that are commonly used in scientific computing, including NumPy, SciPy, Pandas, Matplotlib, and Scikit-learn. These packages are optimized to work together, providing a streamlined workflow for data analysis and modeling.

Anaconda's package management system is one of its key features. It includes the Anaconda package manager, which simplifies the process of installing, updating, and managing packages for Python and R. The Anaconda package manager allows users to create virtual environments, which can be used to isolate packages and dependencies, making it easy to manage different versions of packages and avoid conflicts between them.

The Anaconda distribution is available for Windows, Linux, and macOS, making it a versatile tool for scientific computing and data analysis. It also includes a number of development tools and editors, such as Jupyter Notebook, Spyder, and Visual Studio Code, which provide a convenient environment for developing and testing code.

Overall, Anaconda is a powerful and flexible tool for scientific computing and data analysis, providing a comprehensive set of tools and packages, and a streamlined workflow for package management and deployment.

5.2 MODULE DESCRIPTION

1. **Detection with image:** If we already have a picture of the leaf i.e., complete information about the leaf is already available, the leaf is trained and tested thoroughly by convolutionary neural networks (CNNs). Now the picture is taken and assumptions are made about that leaf, prediction.py runs in the backend. Now the module allows the detection and the results are returned. As stated earlier, this module is only used when the user already has the picture of the leaf.

2. **Detection with camera**: When we do not have the leaf image then the camera module catches the leaf image. In the Cnn_predict.py script leaf information is applied. CNN will train and check data accordingly. As this is newly collected leaf picture, the leaf image will be built in the build module. It is same as detection with image but we used a live detection so that it will be handier than plucking a leaf. The script prediction.py runs in the backend. Now the module allows the detection and the results are returned.

5.3 PROCESS/ALGORITHMS:

The process/algorithm of a leaf disease detection system typically involves the following steps:

Data Collection:

Collecting a dataset of healthy and diseased leaves for the target plant species. This can be done by taking high-quality images of the leaves from various angles and under different lighting conditions.

Data Preprocessing:

The collected images need to be preprocessed to remove noise and unwanted features. Preprocessing techniques such as normalization, resizing, and color correction can be used to make the images suitable for analysis.

Feature Extraction:

Extracting meaningful features from the preprocessed images is critical for accurate classification. Commonly used features include color histograms, texture features, and shape descriptors.

Classification:

A classification algorithm is trained on the extracted features to differentiate between healthy and diseased leaves. Popular machine learning algorithms such as support vector machines (SVM), k-nearest neighbors (KNN), and convolutional neural networks (CNN) can be used for classification.

Validation and Testing:

The accuracy of the classification algorithm is evaluated using validation techniques such as k-fold cross-validation or holdout validation. Once the algorithm's accuracy is satisfactory, it can be tested on new, unseen images to evaluate its performance.

Deployment:

Finally, the leaf disease detection system can be deployed in a real-world setting, such as an agricultural field, to detect and diagnose plant diseases in real-time.

Overall, a leaf disease detection system involves collecting and preprocessing data, extracting features, training a classification algorithm, testing the system's performance, and deploying it for practical use.

6. IMPLEMENTATION

6.1 SAMPLE CODE

AdminHOME.Py :

```
from PyQt5 import QtCore, QtGui, QtWidgets


from CNN import build

from Prediction import Ui_Prediction

from Detection import Ui_Detection

import sys

class Ui_AdminHome(object):


    def cnnbuild(self):

        try:

            build();

            self.showMessageBox("Message", "CNN build successfully.")


        except Exception as e:

            print("Error=" + e.args[0])

            tb = sys.exc_info()[2]

            print(tb.tb_lineno)
```

```
def predict(self):


    try:
        self.admn = QtWidgets.QDialog()
        self.ui = Ui_Prediction()
        self.ui.setupUi(self.admn)
        self.admn.show()
    except Exception as e:
        print(e.args[0])
        tb = sys.exc_info()[2]
        print(tb.tb_lineno)


def detection(self):


    try:
        self.admn = QtWidgets.QDialog()
        self.ui = Ui_Detection()
        self.ui.setupUi(self.admn)
        self.admn.show()
    except Exception as e:
        print(e.args[0])
        tb = sys.exc_info()[2]
        print(tb.tb_lineno)
    def showMessageBox(self, title, message):
        msgBox = QtWidgets.QMessageBox()
        msgBox.setIcon(QtWidgets.QMessageBox.Information)
        msgBox.setWindowTitle(title)
        msgBox.setText(message)
        msgBox.setStandardButtons(QtWidgets.QMessageBox.Ok)
        msgBox.exec_()


    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(700, 454)
        Dialog.setStyleSheet("")
        self.label = QtWidgets.QLabel(Dialog)
        self.label.setGeometry(QtCore.QRect(-70, 0, 841, 471))
        self.label.setStyleSheet("image: url(bg6.jpg);")
        self.label.setObjectName("label")
        self.pushButton = QtWidgets.QPushButton(Dialog)
        self.pushButton.setGeometry(QtCore.QRect(230, 80, 241, 51))
        self.pushButton.setStyleSheet("background-color: rgb(34,139,34);\n"
"font: 12pt \"Franklin Gothic Heavy\";")
        self.pushButton.setObjectName("pushButton")
        self.pushButton.clicked.connect(self.cnnbuild)
        self.pushButton_2 = QtWidgets.QPushButton(Dialog)
        self.pushButton_2.setGeometry(QtCore.QRect(230, 180, 241, 51))
        self.pushButton_2.setStyleSheet("background-color: rgb(34,139,34);\n"
"font: 12pt \"Franklin Gothic Heavy\";")
```

```
        self.pushButton_2.setObjectName("pushButton_2")
        self.pushButton_2.clicked.connect(self.predict)
        self.pushButton_3 = QtWidgets.QPushButton(Dialog)
        self.pushButton_3.setGeometry(QtCore.QRect(230, 280, 241, 51))
        self.pushButton_3.setStyleSheet("background-color: rgb(34,139,34);\n"
"font: 12pt \"Franklin Gothic Heavy\";")
        self.pushButton_3.setObjectName("pushButton_3")
        self.pushButton_3.clicked.connect(self.detection)
        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Leaf Disease Detection"))
        self.label.setText(_translate("Dialog", "TextLabel"))
        self.pushButton.setText(_translate("Dialog", "Build CNN Model"))
        self.pushButton_2.setText(_translate("Dialog", "Detection with Image"))
        self.pushButton_3.setText(_translate("Dialog", "Detection with Camera"))




if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Dialog = QtWidgets.QDialog()
    ui = Ui_AdminHome()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())
```

Prediction.Py :

```
from PyQt5 import QtCore, QtGui, QtWidgets
import os
import sys


from Cnn_predict import predict
class Ui_Prediction(object):


    def browse_file(self):
        fileName, _ = QtWidgets.QFileDialog.getOpenFileName(None, "Select Photo")
        print(fileName)
        self.lineEdit.setText(fileName)


    def detection_img(self):
        try:
            image = self.lineEdit.text()


            if image == "" or image == "null":
                self.showMessageBox("Information", "Please Select Image")
            else:
```

```
            result = predict(image)

            print("res=",result)


            self.label_3.setText("Result : "+result)


        except Exception as e:

          print(e.args[0])

          tb = sys.exc_info()[2]

          print(tb.tb_lineno)


    def setupUi(self, Dialog):

        Dialog.setObjectName("Dialog")

        Dialog.resize(558, 409)

        Dialog.setStyleSheet("background-color: rgb(113, 75, 56);")

        self.label = QtWidgets.QLabel(Dialog)

        self.label.setGeometry(QtCore.QRect(190, 60, 301, 71))

        self.label.setStyleSheet("color: rgb(255, 255, 255);\n"

"font: 16pt \"Georgia\";")

        self.label.setObjectName("label")

        self.label_2 = QtWidgets.QLabel(Dialog)

        self.label_2.setGeometry(QtCore.QRect(110, 150, 101, 20))

        self.label_2.setStyleSheet("color: rgb(255, 255, 255);\n"

"font: 12pt \"Georgia\";")

        self.label_2.setObjectName("label_2")

        self.lineEdit = QtWidgets.QLineEdit(Dialog)

        self.lineEdit.setGeometry(QtCore.QRect(110, 170, 291, 31))
        self.lineEdit.setStyleSheet("font: 75 10pt \"Verdana\";")

        self.lineEdit.setText("")

        self.lineEdit.setObjectName("lineEdit")

        self.pushButton = QtWidgets.QPushButton(Dialog)

        self.pushButton.setGeometry(QtCore.QRect(420, 170, 91, 31))

        self.pushButton.setStyleSheet("color: rgb(255, 255, 255);\n"

"font: 10pt \"Georgia\";\n"

"background-color: rgb(57, 115, 172);")

        self.pushButton.setObjectName("pushButton")

        self.pushButton.clicked.connect(self.browse_file)

        self.pushButton_3 = QtWidgets.QPushButton(Dialog)

        self.pushButton_3.setGeometry(QtCore.QRect(190, 230, 121, 31))

        self.pushButton_3.setStyleSheet("color: rgb(255, 255, 255);\n"

"font: 14pt \"Georgia\";\n"

"background-color: rgb(57, 115, 172);")

        self.pushButton_3.setObjectName("pushButton_3")

        self.pushButton_3.clicked.connect(self.detection_img)

        self.label_3 = QtWidgets.QLabel(Dialog)

        self.label_3.setGeometry(QtCore.QRect(120, 300, 411, 51))

        self.label_3.setStyleSheet("color: rgb(255, 255, 255);\n"

"font: 16pt \"Georgia\";")

        self.label_3.setObjectName("label_3")



        self.retranslateUi(Dialog)

        QtCore.QMetaObject.connectSlotsByName(Dialog)
```

```
    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Image"))
        self.label.setText(_translate("Dialog", "Leaf Disease Detection"))
        self.label_2.setText(_translate("Dialog", "Select Image"))
        self.pushButton.setText(_translate("Dialog", "Browse"))
        self.pushButton_3.setText(_translate("Dialog", "Detect"))
        self.label_3.setText(_translate("Dialog", "Result :"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())
```

Cnn_prediction.py:

```
import numpy as np

import operator

from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array

from keras.models import Sequential, load_model

from keras.preprocessing import image


def predict(img):

    # load model

    img_width, img_height = 128, 128

    model_path = 'cnn.h5'

    model_weights_path = 'weights_cnn.h5'

    model = load_model(model_path)

    model.load_weights(model_weights_path)


    # Prediction on a new picture

    from keras.preprocessing import image as image_utils


    from PIL import Image, ImageTk

    import requests

    class_labels = ['Apple___Apple_scab', 'Apple___Black_rot', 'Apple___Cedar_apple_rust',
'Apple___healthy','Blueberry___healthy','Cherry_(including_sour)___healthy','Cherry_(including
_sour)___Powdery_mildew','Corn_(maize)___Cercospora_leaf_spot
Gray_leaf_spot','Corn_(maize)___Common_rust_','Corn_(maize)___healthy','Corn_(maize)___N
orthern_Leaf_Blight','Grape___Black_rot','Grape___Esca_(Black_Measles)','Grape___healthy','G
rape___Leaf_blight_(Isariopsis_Leaf_Spot)','Orange___Haunglongbing_(Citrus_greening)','Peac
h___Bacterial_spot','Peach___healthy','Pepper_bell___Bacterial_spot','Pepper_bell___healthy','P
otato___Early_blight','Potato___healthy','Potato___Late_blight','Raspberry___healthy','Soybean_
__healthy','Squash___Powdery_mildew','Strawberry___healthy','Strawberry___Leaf_scorch','To
mato___Bacterial_spot','Tomato___Early_blight','Tomato___healthy','Tomato___Late_blight','To
mato___Leaf_Mold','Tomato___Septoria_leaf_spot','Tomato___Spider_mites Two-
spotted_spider_mite','Tomato___Target_Spot','Tomato___Tomato_mosaic_virus','Tomato___To
mato_Yellow_Leaf_Curl_Virus']
    test_image = image.load_img(img, target_size=(128, 128))
    test_image = image.img_to_array(test_image)
    test_image = np.expand_dims(test_image, axis=0)

    test_image /= 255
    result = model.predict(test_image)

    decoded_predictions = dict(zip(class_labels, result[0]))
    decoded_predictions = sorted(decoded_predictions.items(), key=operator.itemgetter(1),
reverse=True)
    print(decoded_predictions[0][0])

    count = 1
    for key, value in decoded_predictions[:5]:
        print("{}. {}: {:8f}%".format(count, key, value * 100))
        count += 1

    return decoded_predictions[0][0]
```

Camera.py:

```
import cv2


def CaptureImage():
    cam = cv2.VideoCapture(0)
    cam.set(1, 40)
    cam.set(1, 80)
    while True:
        ret, img = cam.read()
        cv2.imshow('Capture', img)
        cv2.imwrite(filename="cameraimg.jpg", img=img)
        k = cv2.waitKey(10) & 0xff
        if k == 27:
            break
    print("\n close camera")
    cam.release()
    cv2.destroyAllWindows()


#CaptureImage()
```

CNN.py:

```
from keras.models import Sequential

from keras.layers import Conv2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

from keras.preprocessing.image import ImageDataGenerator


def build():

    print("building")

    # Initialize the CNN

    classifier = Sequential()

    # Convolution and Max pooling

    classifier.add(Conv2D(32, (3, 3), input_shape=(128, 128, 3), activation='relu'))

    classifier.add(MaxPooling2D(pool_size=(2, 2)))

    classifier.add(Conv2D(64, (3, 3), activation='relu'))

    classifier.add(MaxPooling2D(pool_size=(2, 2)))

    classifier.add(Conv2D(128, (3, 3), activation='relu'))

    classifier.add(MaxPooling2D(pool_size=(2, 2)))

    # Flatten

    classifier.add(Flatten())


    # Full connection

    classifier.add(Dense(128, activation='relu'))
    classifier.add(Dense(38, activation='softmax'))


    # Compile classifier
    classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])


    # Fitting CNN to the images
    train_datagen = ImageDataGenerator(rescale=1. / 255, shear_range=0.2, zoom_range=0.2,
horizontal_flip=True)
    test_datagen = ImageDataGenerator(rescale=1. / 255)
    training_set = train_datagen.flow_from_directory('./dataset/train', target_size=(128, 128),
batch_size=32,
                                                     class_mode='categorical')
    test_set = test_datagen.flow_from_directory('./dataset/test', target_size=(128, 128),
batch_size=32,
                                                class_mode='categorical')
    classifier.fit_generator(training_set, steps_per_epoch=25, epochs=50, validation_data=test_set,
                             validation_steps=200 / 32)


    # save model
    classifier.save('cnn.h5')
    classifier.save_weights('weights_cnn.h5')
```

Detection.py:

```
from PyQt5 import QtCore, QtGui, QtWidgets

from Cnn_predict import predict

from Camera import CaptureImage

class Ui_Detection(object):


    def getCameraImage(self, event):

        try:

            CaptureImage()

            self.showMessageBox("Information", "Picture Captured..!")

        except Exception as e:

            print(e.args[0])

            tb = sys.exc_info()[2]

            print(tb.tb_lineno)

        event.accept()


    def showMessageBox(self, title, message):

        msgBox = QtWidgets.QMessageBox()

        msgBox.setIcon(QtWidgets.QMessageBox.Information)

        msgBox.setWindowTitle(title)

        msgBox.setText(message)

        msgBox.setStandardButtons(QtWidgets.QMessageBox.Ok)

        msgBox.exec_()
```

```
 def detect(self):
        result = predict("cameraimg.jpg")
        print("res=", result)


        self.label_2.setText("Result :  "+result)


    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(631, 493)
        Dialog.setStyleSheet("background-color: rgb(170, 85, 127);")
        self.label = QtWidgets.QLabel(Dialog)
        self.camera.setGeometry(QtCore.QRect(230, 120, 161, 121))
        self.camera.setStyleSheet("image: url(camera.png);")
        self.camera.setText("")
        self.camera.setObjectName("camera")
        self.camera.mousePressEvent = self.getCameraImage
        self.label_5 = QtWidgets.QLabel(Dialog)
        self.label_5.setGeometry(QtCore.QRect(250, 250, 181, 41))
        self.label_5.setStyleSheet("font: 75 12pt \"Vani\";")
        self.label_5.setObjectName("label_5")
        self.pushButton = QtWidgets.QPushButton(Dialog)
        self.pushButton.setGeometry(QtCore.QRect(240, 310, 121, 41))
        self.pushButton.setStyleSheet("font: 75 12pt \"Vani\";\n"
"background-color: rgb(0, 85, 127);\n"
"color: rgb(255, 255, 255);")
        self.pushButton.setObjectName("pushButton")
        self.pushButton.clicked.connect(self.detect)

        self.label_2 = QtWidgets.QLabel(Dialog)

        self.label_2.setGeometry(QtCore.QRect(140, 400, 401, 61))

        self.label_2.setStyleSheet("font: 75 16pt \"Vani\";")

        self.label_2.setObjectName("label_2")


        self.retranslateUi(Dialog)

        QtCore.QMetaObject.connectSlotsByName(Dialog)


    def retranslateUi(self, Dialog):

        _translate = QtCore.QCoreApplication.translate

        Dialog.setWindowTitle(_translate("Dialog", "Camera"))

        self.label.setText(_translate("Dialog", "Leaf Disease Detection"))

        self.label_5.setText(_translate("Dialog", "Click on Camera"))

        self.pushButton.setText(_translate("Dialog", "Detect"))

        self.label_2.setText(_translate("Dialog", "Result :"))



if __name__ == "__main__":

    import sys

    app = QtWidgets.QApplication(sys.argv)

    Dialog = QtWidgets.QDialog()

    ui = Ui_Dialog()

    ui.setupUi(Dialog)

    Dialog.show()

    sys.exit(app.exec_())
```

6.2 OUTPUT SCREENS

Detection of Leaf Diseases was established with proper planning and guidance. As shown in Fig 6.1 we have an Admin Page. We have three options here, i.e., CNN model design, image detection; camera detection. The user should proceed as per his / her needs. Unless the consumer does not have an image of the leaf then it may also be possible to detect disease by recording the leaf using a camera as shown in fig 6.2. If the user already has a picture of the leaf then the detection of disease is done by selecting "image detection" on the admin page as shown in fig. 6.1, and the detection process is done as shown in fig.6.3. Results are shown in figure.

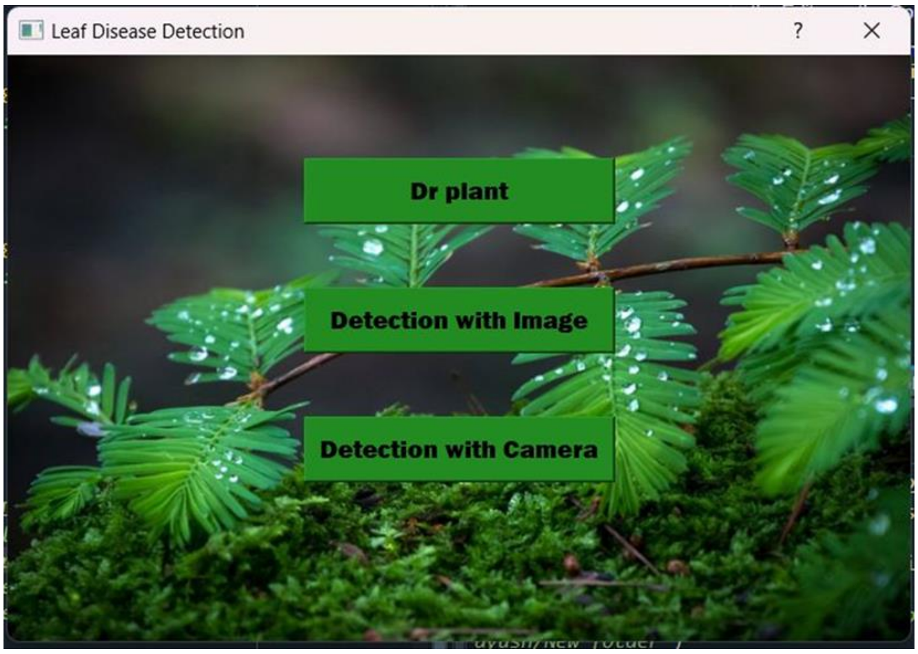


Fig:6.1 Home Page

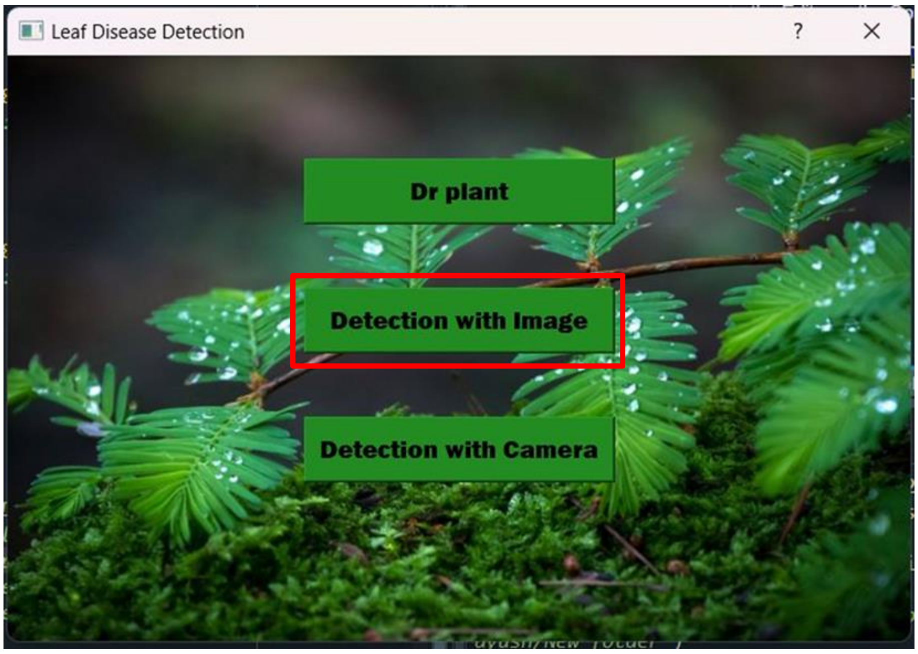


Fig:6.2Detection using Image

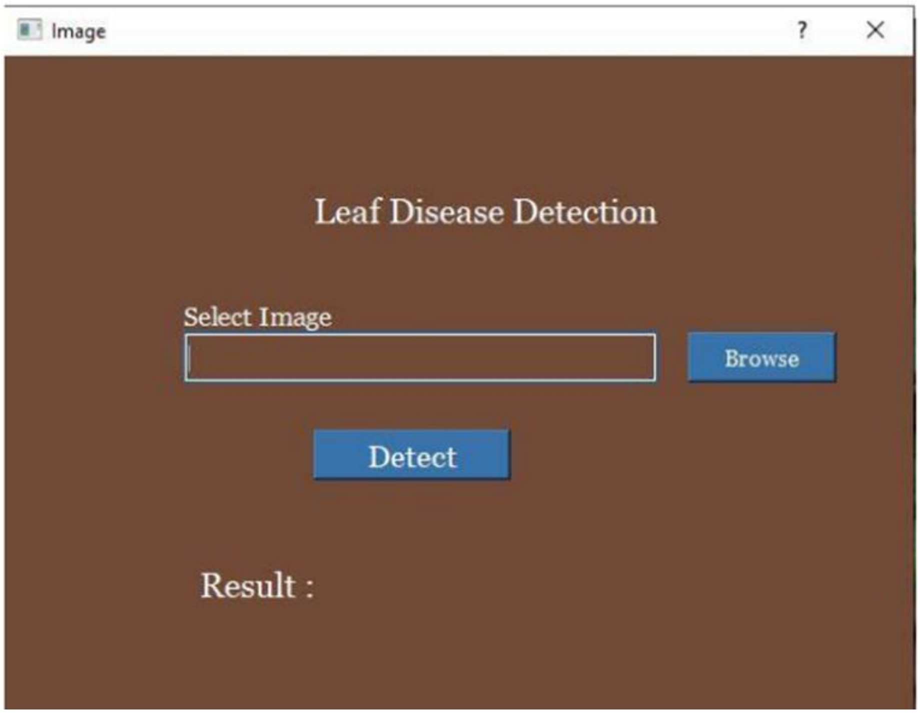


Fig:6.3Browsing an Image

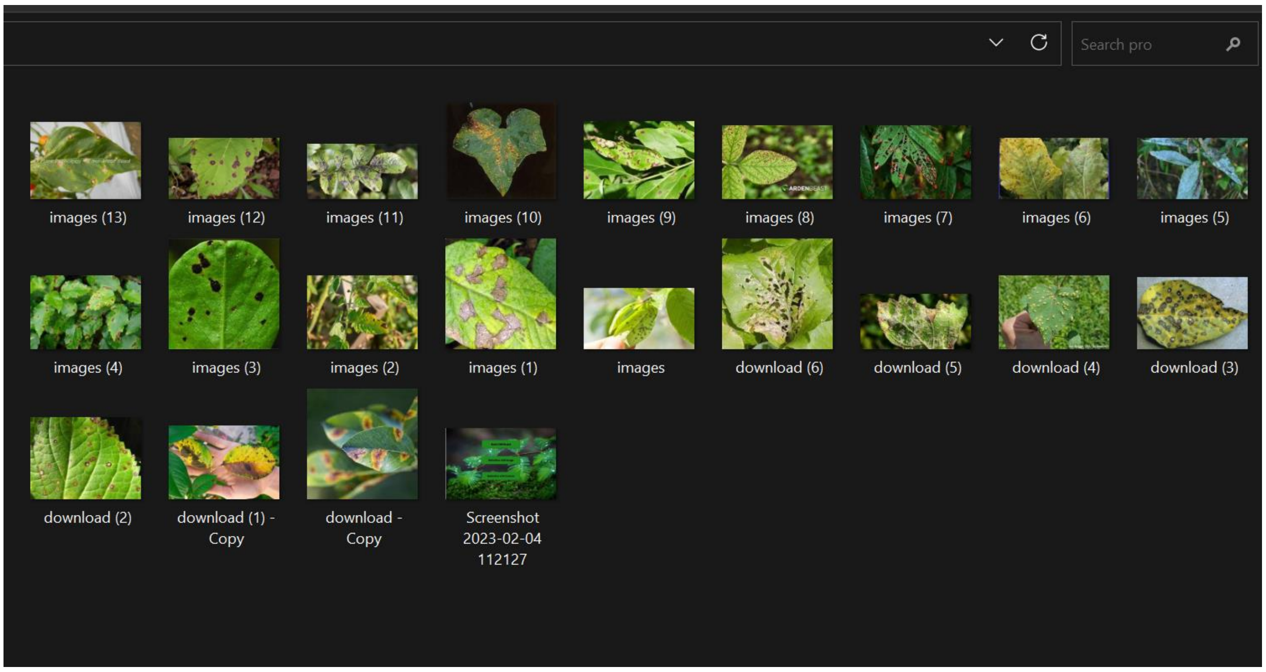
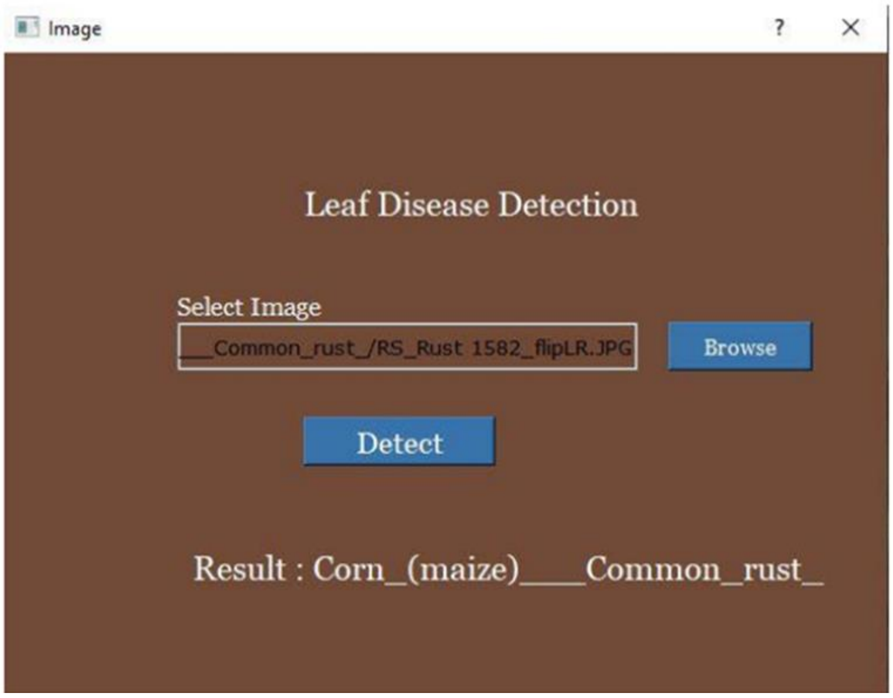


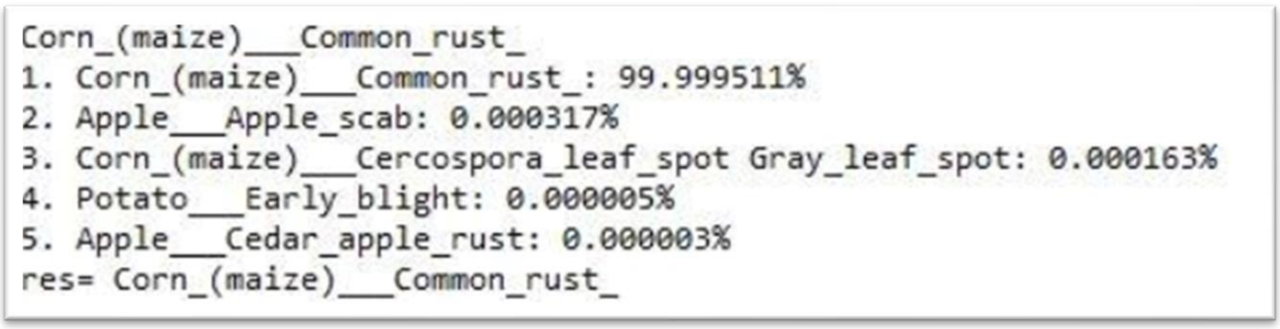
Fig:6.4Different Types Of Leaf Diseases



__Common_rust_/RS_Rust 1582_flipLR.JPG

Result : Corn_(maize)___Common_rust_

Fig:6.5Selection of image



```
Corn_(maize)___Common_rust_
1. Corn_(maize)___Common_rust_: 99.999511%
2. Apple___Apple_scab: 0.000317%
3. Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot: 0.000163%
4. Potato___Early_blight: 0.000005%
5. Apple___Cedar_apple_rust: 0.000003%
res= Corn_(maize)___Common_rust_
```

Fig:6.6Result 1

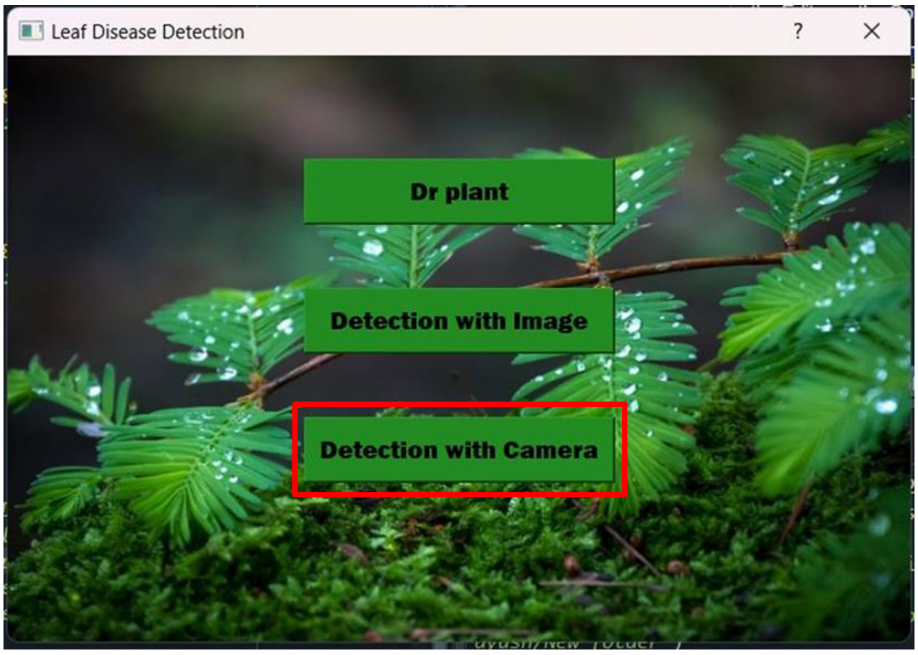


Fig:6.7Detection using Camera

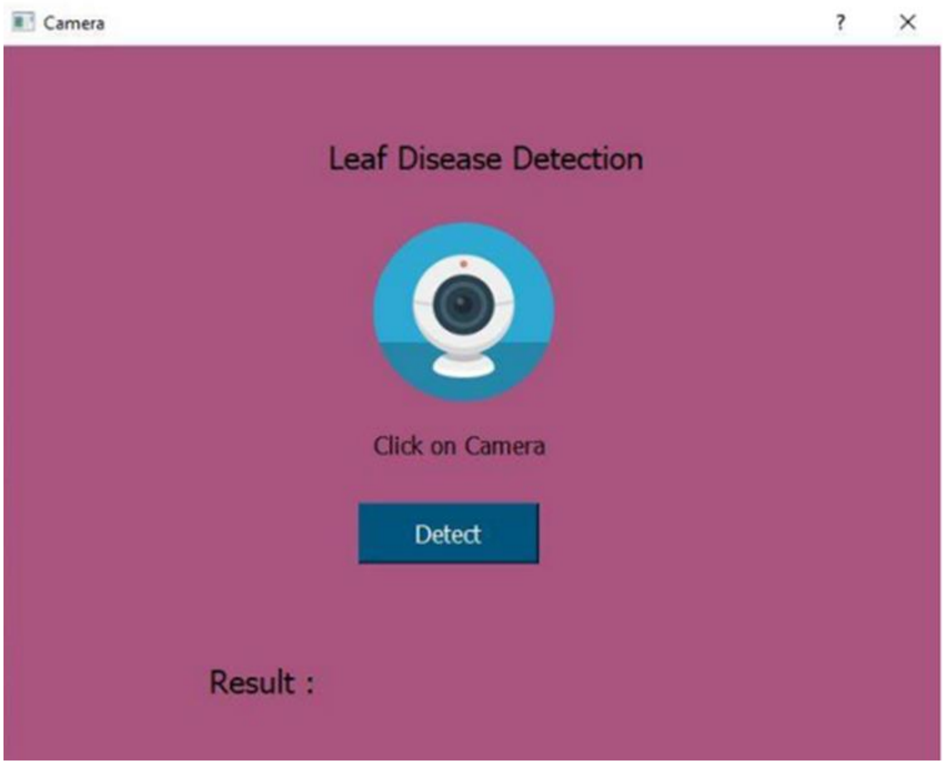
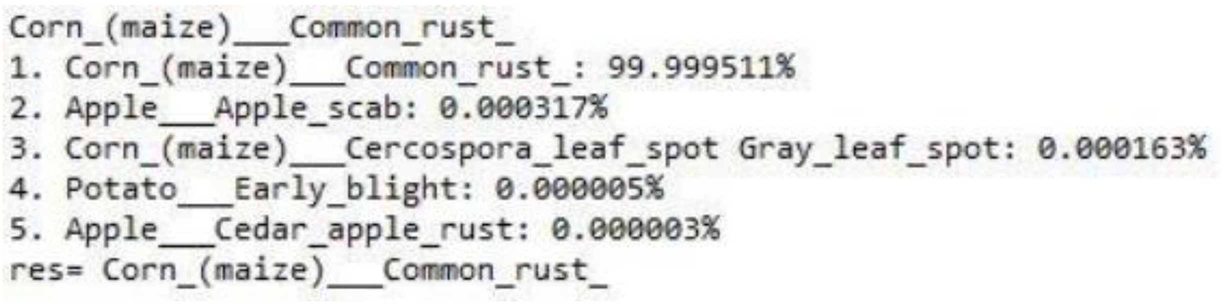


Fig:6.8Click on Camera icon



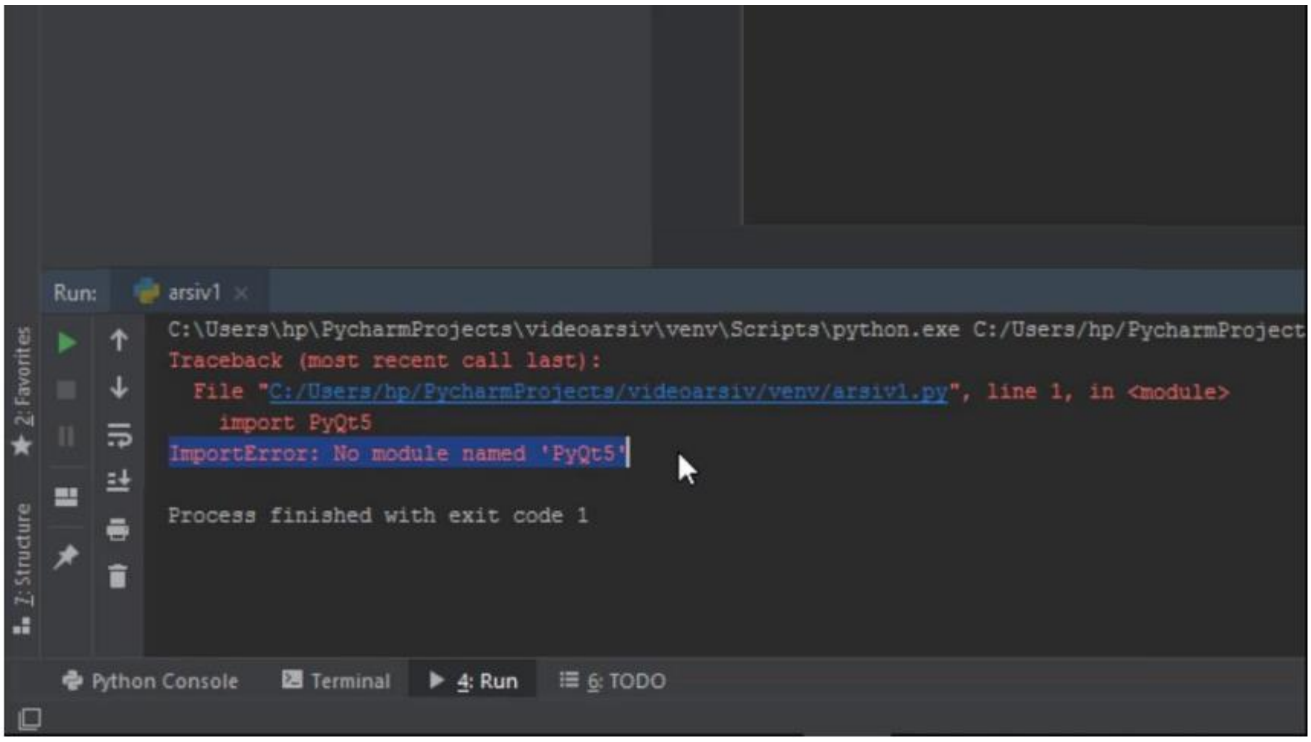
```
Corn_(maize)___Common_rust_
1. Corn_(maize)___Common_rust_: 99.999511%
2. Apple___Apple_scab: 0.000317%
3. Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot: 0.000163%
4. Potato___Early_blight: 0.000005%
5. Apple___Cedar_apple_rust: 0.000003%
res= Corn_(maize)___Common_rust_
```

Fig:6.9Result2

6.3 TEST CASES

6.3.1 System Testing

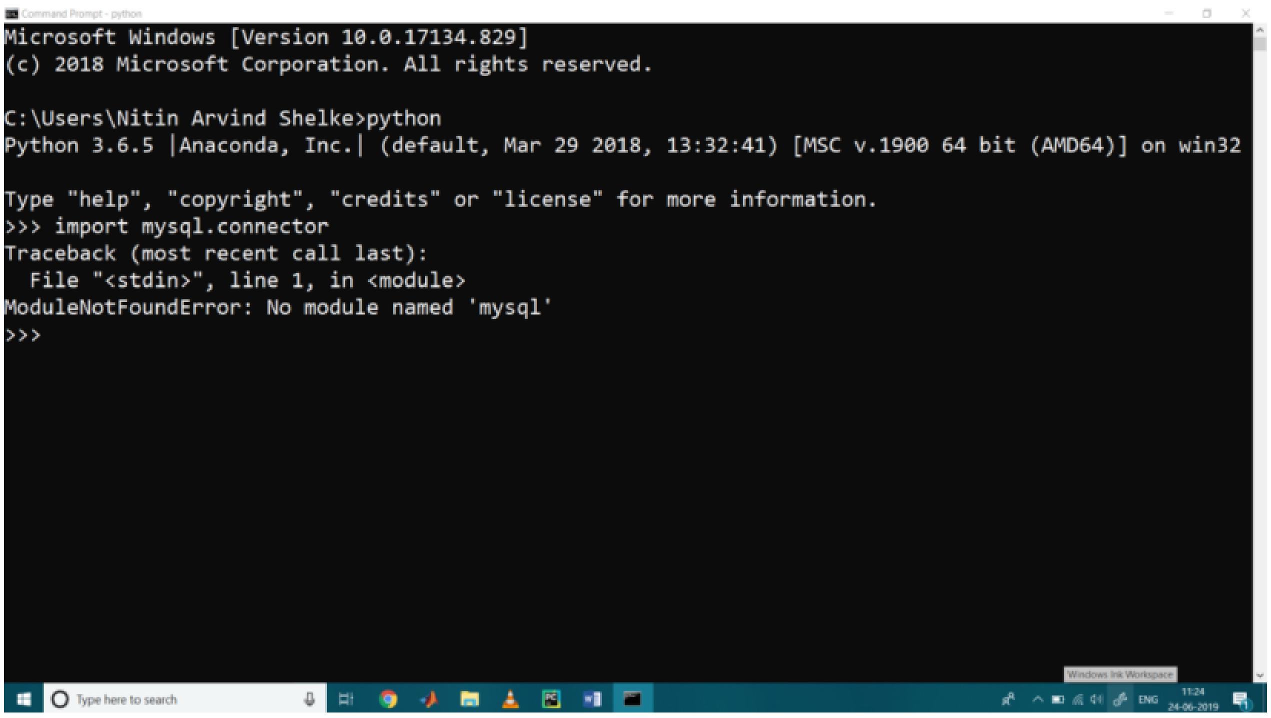
Unit testing:



```
C:\Users\hp\PycharmProjects\videoarsiv\venv\Scripts\python.exe C:/Users/hp/PycharmProject
Traceback (most recent call last):
  File "C:/Users/hp/PycharmProjects/videoarsiv/venv/arsiv1.py", line 1, in <module>
    import PyQt5
ImportError: No module named 'PyQt5'

Process finished with exit code 1
```

Fig 6.3.1 Pyqt5 module not found error

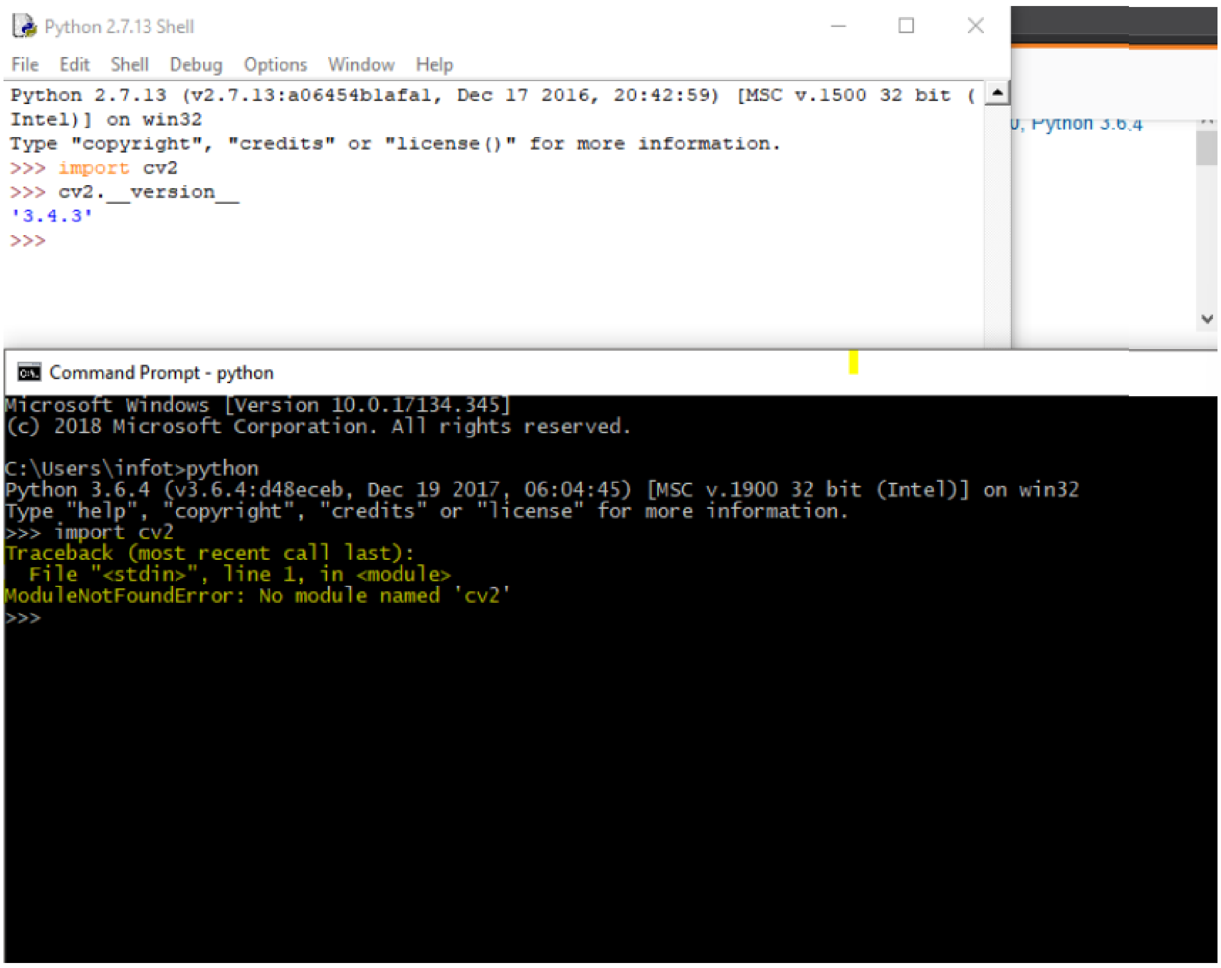


```
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Nitin Arvind Shelke>python
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.
>>> import mysql.connector
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'mysql'
>>>
```

Fig 6.3.2 mysql module not found the error



```
Python 2.7.13 (v2.7.13:a06454blafal, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (
Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import cv2
>>> cv2.__version__
'3.4.3'
>>>
```

```
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\infot>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'cv2'
>>>
```

Fig 6.3.3 cv2 module not found error

7. CONCLUSION

There are several avenues in the identification and diagnosis technique of machine-driven or portable computer vision disease, but even this area of research is lacking. In fact, there are still no business approaches on the market, except that photographs of the leaf helped the management of plant species identification. A brand-new approach to victimization deep learning technique was discussed in this paper to automatically identify and spot plant disease fromphotographs of the leaf. The model developed was able to detect the presence ofa virus and distinguish between healthy leaves and 13 completely different diseases which can be visually diagnosed.

The whole procedure was to delineate, seriously, from the grouping of the images used for training and testing to the preprocessing and augmentation of the image and, finally, the deep CNN and fine-tuning procedure of jobs. To get the efficiency of the freshly generated model, entirely different tests were conducted.

The qualified model's final overall accuracy was 96.3%. Fine-tuning hasn't made many big improvements in precision, but the technique of augmentation has had a stronger impact on producing acceptable results. As a result of the given technique, as we seem to all understand, there was no overlap with the related findings in the field of disease detection, victimization of the particular technique.

8. LIMITATIONS AND FUTURE ENHANCEMENTS

Limitations:

Disease detection can be achieved on only few plants.
Only a few diseases can be detected by our project, we need to train it more so that it recognizes more diseases.

Limited scope:
Many plant disease detection systems are designed to identify only a specific set of diseases, and may not be effective in detecting newly emerging diseases or diseases that are not well-known.

Data collection and analysis:
The accuracy of plant disease detection systems depends on the quality and quantity of data used to train them. Collecting and analyzing large datasets can be time-consuming and resource-intensive.

Environmental factors:

Environmental factors such as lighting, humidity, and temperature can affect the accuracy of plant disease detection systems, and it may be challenging to control for these factors in real-world settings.

Overall, while plant disease detection systems hold promise for improving crop health and increasing yields, it is important to be aware of their limitations and challenges to ensure that they are used effectively and appropriately.

Future Enhancements:

There are several potential future enhancements for leaf disease detection systems, including:

Improved accuracy:
One of the main areas for improvement is increasing the accuracy of the system. This could involve incorporating new machine learning algorithms or improving existing ones, as well as collecting more data to improve training sets.

Real-time detection:
Another area for improvement is real-time detection, which would enable farmers to identify and respond to disease outbreaks more quickly.

Detection of multiple diseases:
Most current systems are only capable of detecting a single disease. Future enhancements could enable the system to detect multiple diseases simultaneously.

Integration with other systems:
Leaf disease detection systems could be integrated with other systems such as irrigation and fertilization systems to create a more comprehensive farming solution.

Mobile applications:
Developing a mobile application that could be used by farmers to monitor the health of their crops in real-time would be a valuable enhancement to the current systems.

Affordable and accessible:
It would be desirable to make the system more affordable and accessible for small-scale farmers who may not have access to high-end technology or resources.

Overall, the future of leaf disease detection systems is promising, and with ongoing advancements in technology, we can expect to see more sophisticated and effective systems in the coming years.

We will try to add to our project a feature such as robotic or drone access which can make our production more productive.

9. REFERENCES

K. A. Garrett, S. P. Dendy, E. E. Frank, M. N. Rouse, and S. E. Travers, “Climatechange effects on plant disease: genomes to ecosystems,” Annual Review of Phytopathology, vol. 44, pp. 489–509, 2016.

S. M. Coakley, H. Scherm, and S. Chakraborty, “Climate change and plant diseasemanagement,” Annual Review of Phytopathology, vol. 37, no. 1, pp. 399–426, 2019.

S. Chakraborty, A. V. Tiedemann, and P. S. Teng, “Climate change: potential impacton plant diseases,” Environmental Pollution, vol. 108, no. 3, pp. 317–326, 2000.

A. J. Tatem, D. J. Rogers, and S. I. Hay, “Global transport networks and infectiousdisease spread,” Advances in Parasitology, vol. 62, pp. 293–343, 2016.

J. R. Rohr, T. R. Raffel, J. M. Romansic, H. McCallum, and P. J. Hudson,
“Evaluating the links between climate, disease spread, and amphibian declines,” Proceedingsof the National Academy of Sciences of the United States of America, vol. 105, no. 45, pp. 17436–17441, 2008. [6]

T. Van der Zwet, “Present worldwide distribution of fire blight,” in Proceedings of the 9th International Workshop on Fire Blight, vol. 590, Napier, New Zealand, October 2021.

S. A. Miller, F. D. Beed, and C. L. Harmon, “Plant disease diagnostic capabilities andnetworks,” Annual Review of Phytopathology, vol. 47, pp. 15–38, 2009.

M. B. Riley, M. R. Williamson, and O. Maloy, “Plant disease diagnosis. The Plant Health Instructor,” 2012.

J. G. Arnal Barbedo, “Digital image processing techniques for detecting, quantifyingand classifying plant diseases,” SpringerPlus, vol. 2, article 660, pp. 1–12, 2015. [10] H. Cartwright, Ed., Artificial Neural Networks, Humana Press, 2015.