**TOMATO LEAF DISEASE DETECTION USING CNN AND**

**MACHINE LEARNING**

**This is submitted in the partial fulfilment of the requirements for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**Submitted by**

TALLURI SAI TEJA REDDY :209T1A0579

CHINNAM GURU PUJITHA :209T1A0519

YEDUGUDI BHARGAVA RAMA RAJU :209T1A0587

NARADALA RAMBABU :209T1A0562 PULI UDAYA SRI :209T1A0572

UNDER THE ESTEEMED GUIDANCE OF

J VENKATA RAO, M. Tech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**VIKAS GROUP OF INSTITUTIONS**

Nunna, Vijayawada-521212, Andhra Pradesh 2020-2024

(Affiliated to JNTU Kakinada, Approved by AICTE) 2020-2024

VIKAS GROUP OF INSTITUTIONS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



**CERTIFICATE**

This is to certify that of the IV B. Tech II Sem (CSE) has satisfactorily completed the dissertation work for Major project entitled “TOMATO LEAF DISEASE DETECTION USING CNN AND MACHINE LEARNING” being Submitted by

TALLURI SAI TEJA REDDY :209T1A0579

CHINNAM GURU PUJITHA :209T1A0519

YEDUGUDI BHARGAVA RAMA RAJU :209T1A0587

NARADALA RAMBABU :209T1A0562 PULI UDAYA SRI :209T1A0572

In partial fulfilment for the award of the Degree of bachelor of technology in computer science and engineering to the Jawaharlal Nehru Technological University, Kakinada is a record if bonafied work carried out under my guidance and supervision.

Internal Guide: Head of the Department:

CH. SUNEETHA, M. Tech B. SURESH, M. Tech

Associate Professor Associate Professor & HOD

**External Examiner**

**ACKNOWLEDGEMENT**

We thank my chairperson Sri N. NARSI REDDY for providing the necessary infrastructure required for my project.

We thankful to Secretary Sri N. SATYANARAYANA REDDY for providing us excellent facilities in the college without which I would not have succeeded.

We thankful to our principal Dr.P.S SRINIVAS for fostering an excellent environment in our college and helping us to all points for achieving our task.

We grateful to B. SURESH Head of the Department of Computer Science and Engineering for his valuable guidance, which helped me to bring how this project successfully. His wise approach made me to learn the minute details of the subject. His matured and patient guidance paved a way for completing my project with sense of satisfaction and pleasure.

We very much thankful to CH. SUNEETHA for his valuable guidance, which helped me to bring out this project successfully.

Finally, We thank all the faculty of COMPUTER SCIENCE AND ENGINEERING DEPARTMENT and Library of VIKAS GROUP OF INSTITUTIONS for imparting knowledge to me throughout my course. TALLURI SAI TEJA REDDY :209T1A0579

CHINNAM GURU PUJITHA :209T1A0519

YEDUGUDI BHARGAVA RAMA RAJU :209T1A0587

NARADALA RAMBABU :209T1A0562 PULI UDAYA SRI :209T1A0572

**DECLARATION**

We hereby declare that the dissertation entitled **“TOMATO LEAF DISEASE DETECTION USING CNN AND MACHINE LEARNING”** submitted for the bachelor of technology in computer science and engineering in our original work. The dissertation and results embodied in this project report has not been submitted to any other University or Institute for the award of any Degree, Associate ship or any other similar titles**.**

TALLURI SAI TEJA REDDY :209T1A0579

CHINNAM GURU PUJITHA :209T1A0519

YEDUGUDI BHARGAVA RAMA RAJU :209T1A0587

NARADALA RAMBABU :209T1A0562 PULI UDAYA SRI :209T1A0572

**Place: Nunna**

**Date:**

**INTRODUCTION**

Early plant disease detection plays a significant role in efficient crop yield. Plant diseases like black measles, black rot, bacterial spot, etc. affect the growth, crop quality of plants and economic impacts in the agriculture industry. To avoid the impact of these diseases, expensive approaches and the use of pesticides are some solutions the farmers usually implement. The use of chemical means damages the plant and the surrounding environment. In addition, this kind of approach intensifies the cost of production and major monetary loss to farmers.

Early discovery of diseases as they occur is the most important period for efficient disease management. Manual disease detection through human experts to identify and recognize plant diseases is a usual practice in agriculture . With the improvements in technology, automatic detection of plant diseases from raw images is possible through computer vision and artificial intelligence researches . In this study, the researchers were able to investigate plant diseases and pest’s infestation that affects the leaves of the plants.

Image processing techniques are now commonly employed in agriculture and it is applied for the detection and recognition of weeds, fruit-grading, identifying and calculating disease infestations of plants ,and plant genomics. Currently, the introduction of deep learning methods turns out to be popular.Deep learning is the advanced methods of machine learning that uses neural networks that works like the human brain Traditional methods involve the use of semantic features as the classification method. A convolutional neural network (CNN) is a deep learning model that is widely used in image processing.

The work of Lee et al. presents a hybrid model to obtain characteristics of leaves using CNN and classify the extracted features of leaves. The methodology in the study involves three key stages: acquisition of data, preprocessing of data and image classification. The study utilized dataset from Plant village dataset that contains plant varieties of apple, corn, grapes, potato, sugarcane, and tomato. There are 11 types of plant diseases identified in the study including healthy images of identified plants. Image preprocessing involves resized images and enhancement before supplying it for the classification model. CONVULUTIONAL NEURAL NETWORK Deep learning is a subsection of Artificial Intelligence and machine learning that uses Artificial neural networks (ANN). Training the deep learning models divides the feature extraction and extracts its features for 3 classification. There are several applications of deep learning which include computer vision, image classification, restoration, speech, video analysis, etc. A convolutional neural network with nominal process can simply detect and categorize. It is efficient in evaluating graphical images and extracts the essential features through its multi-layered structure. As shown in Figure 1.1, the CNN involves our layers, that is: input image, convolutional layer and pooling layer, fully connected layers, and output. 

**Figure1.1.** Sample images from Plant Village dataset for types of leaf diseases.

CNN deep-learning models are popular for image-based research. They are efficient in learning low-level complex features from images. However, deep CNN layers are difficult to train as this process is computationally expensive. To solve such issues, transfer learning-based models have been proposed by various researchers. These models are trained with the ImageNet dataset, which consists of multiple classes. Such models can be used for training with any dataset as the features of the images, such as edges and contours, are common among the datasets. Hence, the transfer learning approach has been found to be the most suitable and robust model for image classification. Further, transfer learning can improve learning even when there is a smaller dataset.

**Methodology**

A block diagram presented in Figure 1.2 shows the Input Dataset, Image Acquisition, Image pre-processing and Classification.

**Figure 1.2**. Plant leaf detection and disease recognition methodology

**Image Acquisition**

Image dataset used for training the model was acquired in the Plant Village repository. A python script was used to download images of the plant diseases from the repository. The acquired dataset consists of approximately 35,000 images with 32 different classes plant varieties and diseases.

**Image Pre-processing**

Pre-processed images are reduced image size and image crop to a given input. It processes and enhances the image to its needed colour scale. The study uses coloured and resized images to 96x96 resolution for processing.

**Classification**

Classification uses a fully connected layers and for feature extraction it uses convolutional and pooling layers. The classification process classifies the plant leaf if it is infected with the disease or not, identifies the type of plant disease and recognize the plant variety. Data set Image Acquisition Classification Image Pre -Processing

**2.LITERATURE SURVEY**

**2.1 A survey of image processing techniques for agriculture**

**AUTHORS**: Lalit P. Saxena and Leisa J. Armstrong

**ABSTRACT**: Computer technologies have been shown to improve agricultural productivity in a number of ways. One technique which is emerging as a useful tool is image processing. This paper presents a short survey on using image processing techniques to assist researchers and farmers to improve agricultural practices. Image processing has been used to assist with precision agriculture practices, weed and herbicide technologies, monitoring plant growth and plant nutrition management. This paper highlights the future potential for image processing for different agricultural industry contexts.

**2.2 ImageNet classification with deep convolutional neural networks**

**AUTHORS**: A. Krizhevsky, I. Sutskever and G. E. Hinton,

**ABSTRACT**: We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used nonsaturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overriding in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

**2.3 Integrating soms and a bayesian classifier for segmenting diseased plants in uncontrolled environments**

**AUTHORS**: D. L. Hernández-Rabadán, F. Ramos-Quintana and J. Guerrero Juk **ABSTRACT**: This work presents a methodology that integrates a nonsupervised learning approach (self-organizing map (SOM)) and a supervised one (a Bayesian classifier) for segmenting diseased plants that grow in uncontrolled environments such as greenhouses, wherein the lack of control of illumination and presence of background bring about 7 serious drawbacks. During the tr aining phase two SOMs are used: one that creates color groups of images, which are classified into two groups using K -means and labeled as vegetation and nonvegetation by using rules, and a second SOM that corrects classification errors made by the first SOM. Two color histograms are generated from the two color classes and used to estimate the conditional probabilities of the Bayesian classifier. During the testing phase an input image is segmented by the Bayesian classifier and then it is converted into a binary image, wherein contours are extracted and analyzed to recover diseased areas that were incorrectly classified as nonvegetation. The experimental results using the proposed methodology showed better performance than two of the most used color index methods.

**2.4 Visible-near infrared spectroscopy for detection of Huanglongbing in citrus orchards**

**AUTHORS**: S. Sankaran, A. Mishra, J. M. Maja and R. Ehsani

**ABSTRACT**: This paper evaluates the feasibility of applying visible-near infrared spectroscopy for in-field detection of Huanglongbing (HLB) in citrus orchards. Spectral reflectance data from the wavelength range of 350–2500nm with 989 spectral features were collected from 100 healthy and 93 HLBinfected citrus trees using a visible-near infrared spectroradiometer. During data preprocessing, the spectral data were normalized and averaged every 25nm to reduce the spectral features from 989 to 86. Three datasets were generated from the preprocessed raw data: first derivatives, second derivatives, and a combined dataset (generated by integrating preprocessed raw data, first derivatives and second derivatives). The preprocessed datasets were analyzed using principal component analysis (PCA) to further reduce the number of features used as inputs in the classification algorithm. The dataset consisting of principal components were randomized and separated into training and testing datasets such that 75% of the dataset was used for training; while 25% of the dataset was used for testing the classification algorithms. The number of samples in the training and testing datasets was 145 and 48, respectively. The classification algorithms tested were: linear discriminant analysis, quadratic discriminant analysis (QDA), k-nearest neighbor, and soft independent modeling of classification analogies (SIMCA). The reported classification accuracies of the algorithms are an average of three runs. When the second derivatives dataset were analyzed, the QDAbased classification algorithm yielded the highest overall average classification 8 accuracies of about 95%, with HLB-class classification accuracies of about 98%. In the combined dataset, SIMCA-based algorithms resulted in high overall classification accuracies of about 92% with low false negatives (less than 3%).

**2.5 Rethinking the inception architecture for computer vision**

**AUTHORS**:Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna

**ABSTRACT**: Convolutional networks are at the core of most state-of-the-art computer vision solutions for a wide variety of tasks. Since 2014 very deep convolutional networks started to become mainstream, yielding substantial gains in various benchmarks. Although increased model size and computational cost tend to translate to immediate quality gains for most tasks (as long as enough labeled data is provided for training), computational efficiency and low parameter count are still enabling factors for various use cases such as mobile vision and big-data scenarios. Here we explore ways to scale up networks in ways that aim at utilizing the added computation as efficiently as possible by suitably factorized convolutions and aggressive regularization. We benchmark our methods on the ILSVRC 2012 classification challenge validation set demonstrate substantial gains over the state of the art: 21.2% top-1 and 5.6% top-5 error for single frame evaluation using a network with a computational cost of 5 billion multiply-adds per inference and with using less than 25 million parameters. With an ensemble of 4 models and multi-crop evaluation, we report 3.5% top-5 error on the validation set (3.6% error on the test set) and 17.3% top-1 error on the validation set.

**3.SYSTEM ANALYSIS**

**SYSTEM ANALYSIS**

Leaf disease detection using deep learning is a popular area of research that involves developing computer algorithms capable of detecting and classifying plant diseases from digital images of plant leaves. This technology has the potential to revolutionize agriculture by allowing farmers to quickly and accurately identify and treat diseased plants, thereby reducing crop losses and increasing yields.

**3.1Existing system**

The current approach for detecting plant disease is simple naked eye observation by plant experts, which can be used to detect and identify plant diseases. In these circumstances, the suggested technique is useful for tracking vast fields of crops. Furthermore, in some nations, farmers lack adequate facilities or are unaware that they can contact experts. As a result, consulting experts is not only more expensive but also more time consuming. In those circumstances, the suggested technique for tracking a large number of plants would be useful.

**3.2Proposed system**

This study is focused on the identification of plant diseases. The segmentation, feature extraction, and classification techniques are used to detect plant diseases. Photos of leaves from various plants are taken with a digital camera or similar unit, and the images are used to classify the affected region in the leaves. To detect plant disease, we use a Convolution neural network and a Deep neural network in the proposed framework. This paper proposes a framework that employs low-cost, open-source software to achieve the task of reliably detecting plant disease.

**4. SOFTWARE REQUIREMENTS & SPECIFICATIONS**

**4.1 PURPOSE, SCOPE:**

**PROJECT SCOPE**

The project titled as “Plant Disease Detection using CNN” is a console based application. This project helps the farmers to detect diseases of plants authorities easily in less time and makes the space constraint also less. This console application makes the whole process easier and faster. It surpasses all the previous methods of image processing and brings out a very better result than others methods.

**PURPOSE:**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

**4.2 SYSTEM REQUIREMENTS**

**HARDWARE REQUIREMENTS:**

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

• System : 64-bit operating system, x64-based processor

• Hard Disk : 512SD

• Monitor : virtual display terminal

• Mouse : HP

• Ram : 8Gb

**SOFTWARE REQUIREMENTS :**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

• Python libraries such as pandas, NumPy, Keras.

• Google Colab for implementation.

• Operating System : Windows 10 or above

**4.3 SOFTWARE DESCRIPTION**

**Install Python on Windows**

The Python programming language is an increasingly popular choice for both beginners and experienced developers. Flexible and versatile, Python has strengths in scripting, automation, data analysis, machine learning, and back-end development.

**STEP-1 DOWNLOADING THE PYTHON INSTALLER**

1. Go to the official Python download page for Windows

2. Find a stable Python 3 release. This tutorial was tested with Python version

3.10.10 3. Click the appropriate link for your system to download the executable file: Windows installer (64-bit) or Windows installer (32-bit).

**STEP-2 RUNNING THE EXECUTABLE INSTALLER**

1. After the installer is downloaded, double-click the .exe file, for example python3.10.10-amd64.exe, to run the Python installer.

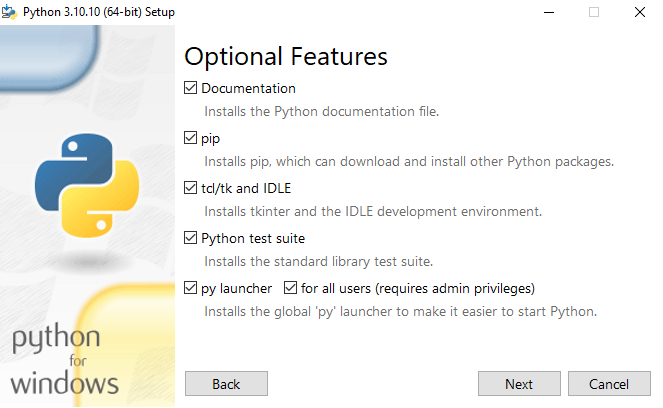
2. Select the Install launcher for all users checkbox, which enables all users of the computer to access the Python launcher application.

3. Select the Add python.exe to PATH checkbox, which enables users to launch Python from the command line.

**Figure4**.**1**. Installation of python

4. If you’re just getting started with Python and you want to install it with default features as described in the dialog, then click Install Now and go to Step 4 - Verify the Python Installation. To install other optional and advanced features, click Customize installation and continue.

5.The Optional Features include common tools and resources for Python and you can install all of them, even if you don’t plan to use them.



**Figure 4.2** Optional features of python

**Select some or all of the following options**:

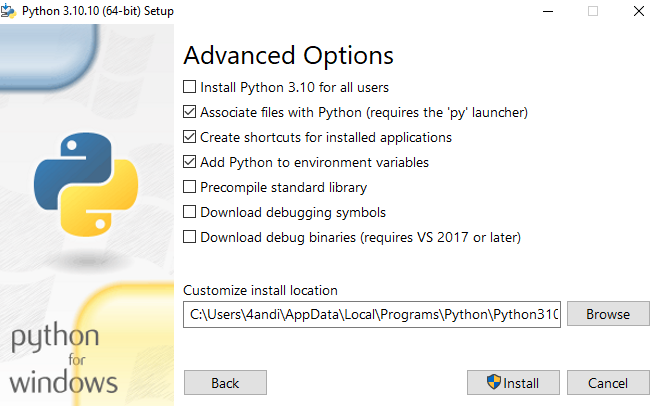
• **Documentation**: recommended o pip: recommended if you want to install other Python packages, such as NumPy or pandas

• **tcl/tk and IDLE**: recommended if you plan to use IDLE or follow tutorials that use it

• **Python test suite**: recommended for testing and learning o py launcher and for all users: recommended to enable users to launch Python from the command line.

6. **Click Next.**

7. The Advanced Options dialog displays.



**Figure 4.3** Advanced options of python

**Select the options that suit your requirements:**

• **Install for all users**: recommended if you’re not the only user on this computer **• Associate files with Python**: recommended, because this option associates all the Python file types with the launcher or editor

• **Create shortcuts for installed applications:** recommended to enable shortcuts for Python applications

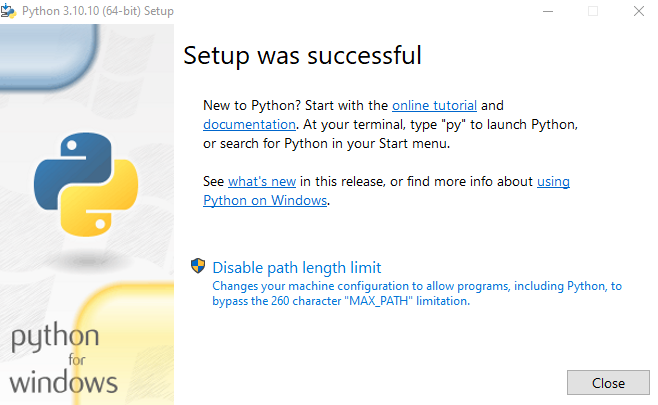
• **Add Python to environment variables**: recommended to enable launching Python

• **Precompile standard library**: not required, it might down the installation

**• Download debugging symbols and Download debug binaries:** recommended only if you plan to create C or C++ extensions Make note of the Python installation directory in case you need to reference it later.

8.Click Install to start the installation.

1. 9.After the installation is complete, a Setup was successful message displays. Figure



**.4 Successful installation of python**

**STEP-3 VERIFY THE PYTHON INSTALLATION**

You can verify whether the Python installation is successful either through the command line or through the Integrated Development Environment (IDLE) application, if you chose to install it.

python –version

Go to **Start** and enter cmd in the search bar. Click **Command Prompt**.

Enter the following command in the command prompt:

An example of the output is

You’ve installed Python on your Windows 10 computer.

Output

Python 3.10.10

**What is Python**

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

• Machine Learning

• GUI Applications (like Kivy, Tkinter, PyQt etc. )

• Web frameworks like Django (used by YouTube, Instagram, Dropbox)

• Image processing (like Opencv, Pillow)

• Web scraping (like Scrapy, BeautifulSoup, Selenium)

• Test frameworks

• Multimedia

**Advantages of Python :-**

Let’s see how Python dominates over other languages.

1. Extensive Libraries Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unittesting, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

**2. Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

**3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

**4. Improved Productivity**

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

**5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

**6. Simple and Easy**

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

**8. Object-Oriented**

This language supports both the procedural and object- oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

**9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any systemdependent features.

**11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Advantages of Python Over Other Languages

**1.Less Coding**

**2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

**3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an allrounder programming language.

**Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

1. **Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

**2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle. The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

**3.Design Restrictions**

As you know, Python is dynamically-typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

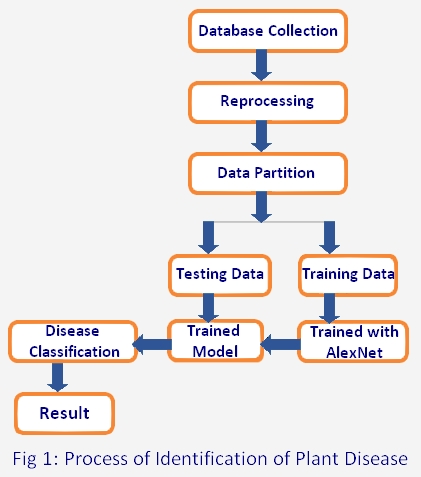
**4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

**5. Simple**

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary. This was all about the Advantages and Disadvantages of Python Programming Language.

**5. SYSTEM DESIGN**

.   
 

**Figure5**.1 System Architecture for leaf disease detection using deep learning

Image Acquisition: The first component of the system is the acquisition of leaf images. This can be done using a camera or mobile device, and the images are typically captured under natural lighting conditions.

Pre-processing: The second component of the system is pre-processing the images to enhance features and remove noise. This involves techniques such as image resizing, cropping, and filtering.

Feature Extraction: The third component of the system is feature extraction. In this step, the deep learning model extracts relevant features from the preprocessed images. This is typically done using Convolutional Neural Networks (CNNs) or Deep Convolutional Neural Networks (DCNNs).

Training: The fourth component of the system is training the deep learning model using a large dataset of labeled images. The model is trained to distinguish between healthy and diseased plants based on the extracted features.

Validation: The fifth component of the system is validating the trained model using a separate set of images. This is done to ensure that the model is accurate and can generalize well to new images.

Classification: The final component of the system is classification. Once the model has been trained and validated, it can be used to classify new images of plant leaves as healthy or diseased.

In addition to these components, the system may also include a user interface for inputting new images, a database for storing labeled images, and a reporting mechanism for communicating the results of the classification process to the user. Overall, the architecture of a leaf disease detection system using deep learning is designed to be scalable, accurate, and user-friendly, allowing farmers to quickly and easily detect and treat plant diseases.

**6. IMPLEMENTATION**

**IMPLEMENTATION**

Leaf disease detection using deep learning is a popular area of research that involves developing computer algorithms capable of detecting and classifying plant diseases from digital images of plant leaves. This technology has the potential to revolutionize agriculture by allowing farmers to quickly and accurately identify and treat diseased plants, thereby reducing crop losses and increasing yields.

Deep learning is a subset of machine learning that utilizes artificial neural networks with multiple layers to process and analyze large datasets. In the case of leaf disease detection, deep learning algorithms are trained using thousands of images of healthy and diseased plants to learn the features that distinguish healthy plants from those affected by disease. Once trained, these algorithms can be used to classify new images of plant leaves as either healthy or diseased.

There are several deep learning models that can be used for leaf disease detection, including Convolutional Neural Networks (CNNs) and Deep Convolutional Neural Networks (DCNNs). These models have been used successfully to detect a wide range of plant diseases, including tomato leaf mold, citrus greening, and apple scab.

To develop an effective leaf disease detection system, several steps must be followed. These include collecting a large dataset of plant images, preprocessing the images to remove noise and enhance features, training the deep learning model on the pre-processed images, and validating the model using a separate set of images. Once the model has been validated, it can be used to classify new images of plant leaves as either healthy or diseased.

Overall, leaf disease detection using deep learning has the potential to significantly improve agricultural practices by allowing farmers to quickly and accurately identify and treat diseased plants, thereby increasing crop yields and reducing losses

**6.1 The general steps involved in leaf disease detection**

**Data collection:**

The first step is to collect a dataset of images of leaves affected by different diseases, as well as images of healthy leaves. The dataset should be diverse, containing leaves from different plant species, different lighting conditions, angles, and backgrounds.

**Data pre-processing:**

The collected dataset needs to be pre-processed to ensure that the images are of the same size and format. Pre-processing may also involve data augmentation techniques such as flipping, rotating, and scaling the images to increase the diversity of the dataset.

**Model selection:**

The next step is to select a deep learning model suitable for image classification. Popular models for this task include Convolutional Neural Networks (CNNs) such as VGG, ResNet, and Inception.

**Model training:**

The selected model is then trained on the preprocessed dataset. The training involves feeding the model with input images and their corresponding labels (disease or healthy). The model then learns to classify the images based on the patterns and features it learns from the dataset.

**Model evaluation:**

The trained model is then evaluated on a separate dataset to assess its performance. The evaluation metrics used may include accuracy, precision, recall, and F1 score.

**Deployment:**

The final step is to deploy the trained model to a production environment where it can be used for real-time disease detection. This can be done by integrating the model into an application or web-based system that can process images of leaves and provide a diagnosis.

**Data pre-processing:**

The collected dataset needs to be pre-processed to ensure that the images are of the same size and format. Pre-processing may also involve data augmentation techniques such as flipping, rotating, and scaling the images to increase the diversity of the dataset.

Model selection:

The next step is to select a deep learning model suitable for image classification. Popular models for this task include Convolutional Neural Networks (CNNs) such as VGG, ResNet, and Inception.

**Model training:**

The selected model is then trained on the preprocessed dataset. The training involves feeding the model with input images and their corresponding labels (disease or healthy). The model then learns to classify the images based on the patterns and features it learns from the dataset.

**Model evaluation:**

The trained model is then evaluated on a separate dataset to assess its performance. The evaluation metrics used may include accuracy, precision, recall, and F1 score.

**Deployment:**

The final step is to deploy the trained model to a production environment where it can be used for real-time disease detection. This can be done by integrating the model into an application or web-based system that can process images of leaves and provide a diagnosis.

**6.2 Code Used In Python**

import pandas as pd

import cv2 as cv

import numpy as np

import matplotlib.pyplot as plt

import os

import seaborn as sns

DATASET="train"

DATASET2="valid"

CATEGORIES=["Tomato\_\_Bacterial\_spot","Tomato\_Early\_blight","Tomato\_healthy","Tomato\_Late\_blight","Tomato\_Leaf\_Mold","Tomato\_Septoria\_leaf\_spot","Tomato\_Spider\_mites Two-spotted\_spider\_mite","Tomato\_Target\_Spot","Tomato\_Tomato\_mosaic\_virus","Tomato\_\_Tomato\_Yellow\_Leaf\_Curl\_Virus"]

train\_data=[]

for category in CATEGORIES:

label=CATEGORIES.index(category)

path=os.path.join(DATASET,category)

for img\_file in os.listdir(path):

img=cv.imread(os.path.join(path,img\_file),1)

img=cv.cvtColor(img,cv.COLOR\_BGR2RGB)

img=cv.resize(img,(64,64))

train\_data.append([img,label])

test\_data=[]

for category in CATEGORIES:

label=CATEGORIES.index(category)

path=os.path.join(DATASET2,category)

for img\_file in os.listdir(path):

img=cv.imread(os.path.join(path,img\_file),1)

img=cv.cvtColor(img,cv.COLOR\_BGR2RGB)

img=cv.resize(img,(64,64))

test\_data.append([img,label])

print(len(train\_data))

print(len(test\_data))

import random

random.shuffle(train\_data)

random.shuffle(test\_data)

for lbl in train\_data[:10]:

print(lbl[1])

X\_train=[]

y\_train=[]

for features,label in train\_data:

X\_train.append(features)

y\_train.append(label)

Y=[]

for i in y\_train:

if i==0:

Y.append("BACTERIAL SPOT")

elif i==1:

Y.append("EARLY BLIGHT")

elif i==2:

Y.append("HEALTHY")

elif i==3:

Y.append("LATE BLIGHT")

elif i==4:

Y.append("LEAF MOLD")

elif i==5:

Y.append("SEPTORIA LEAF SPOT")

elif i==6:

Y.append("SPIDER MITE")

elif i==7:

Y.append("TARGET SPOT")

elif i==8:

Y.append("MOSAIC VIRUS")

else:

Y.append("YELLOW LEAF CURL VIRUS")

len(X\_train),len(y\_train)

X\_test=[]

y\_test=[]

for features,label in test\_data:

X\_test.append(features)

y\_test.append(label)

Z=[]

for i in y\_test:

if i==0:

Z.append("BACTERIAL SPOT")

elif i==1:

Z.append("EARLY BLIGHT")

elif i==2:

Z.append("HEALTHY")

elif i==3:

Z.append("LATE BLIGHT")

elif i==4:

Z.append("LEAF MOLD")

elif i==5:

Z.append("SEPTORIA LEAF SPOT")

elif i==6:

Z.append("SPIDER MITE")

elif i==7:

Z.append("TARGET SPOT")

elif i==8:

Z.append("MOSAIC VIRUS")

else:

Z.append("YELLOW LEAF CURL VIRUS")

len(X\_test),len(y\_test)

X\_train=np.array(X\_train).reshape(-1,64,64,3)

X\_train=X\_train/255.0

X\_train.shape

X\_test=np.array(X\_test).reshape(-1,64,64,3)

X\_test=X\_test/255.0

X\_test.shape

order=['BACTERIAL SPOT','EARLY BLIGHT','HEALTHY','LATE BLIGHT','LEAF MOLD','SEPTORIA LEAF SPOT','SPIDER MITE','TARGET SPOT','MOSAIC VIRUS','YELLOW LEAF CURL VIRUS']

ax=sns.countplot(Y, order=order)

ax.set\_xlabel("Leaf Diseases")

ax.set\_xticklabels(ax.get\_xticklabels(), rotation=40, ha='right')

ax.set\_ylabel("Image Count")

ax=sns.countplot(Z, order=order)

ax.set\_xlabel("Leaf Diseases")

ax.set\_xticklabels(ax.get\_xticklabels(), rotation=40, ha='right')

ax.set\_ylabel("Image Count")

from keras.utils import to\_categorical

one\_hot\_train=to\_categorical(y\_train)

one\_hot\_train

one\_hot\_test=to\_categorical(y\_test)

one\_hot\_test

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D,Dense,Flatten,MaxPooling2D,Dropout

classifier=Sequential()

classifier.add(Conv2D(32,(3,3), input\_shape=(64,64,3), activation='relu'))

classifier.add(MaxPooling2D(pool\_size=(2,2)))

classifier.add(Dropout(0.2))

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

classifier.add(MaxPooling2D(pool\_size=(2,2)))

classifier.add(Dropout(0.2))

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

classifier.add(MaxPooling2D(pool\_size=(2,2)))

classifier.add(Dropout(0.4))

classifier.add(Flatten())

classifier.add(Dense(activation='relu', units=64))

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

classifier.add(Dense(activation='relu', units=64))

classifier.add(Dense(activation='softmax', units=10))

classifier.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

classifier.summary()

hist=classifier.fit(X\_train,one\_hot\_train,epochs=75,batch\_size=128,validation\_split=0.2)

test\_loss,test\_acc=classifier.evaluate(X\_test,one\_hot\_test)

test\_loss,test\_acc

plt.plot(hist.history['loss'])

plt.plot(hist.history['val\_loss'])

plt.title('Classifier Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train','Validation'],loc='upper right')

plt.show()

plt.plot(hist.history['accuracy'])

plt.plot(hist.history['val\_accuracy'])

plt.title('Classifier Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train','Validation'],loc='upper left')

plt.show()

y\_pred=classifier.predict\_classes(X\_test)

y\_pred

y\_prob=classifier.predict\_proba(X\_test)

y\_prob

from sklearn.metrics import roc\_curve, auc

fpr = {}

tpr = {}

thresh ={}

roc\_auc={}

n\_class = 10

for i in range(n\_class):

fpr[i], tpr[i], thresh[i] = roc\_curve(y\_test, y\_prob[:,i], pos\_label=i)

roc\_auc[i] = auc(fpr[i], tpr[i])

plt.plot(fpr[0], tpr[0], color='orange',label='Bacterial Spot AUC = %0.3f' % roc\_auc[0])

plt.plot(fpr[1], tpr[1], color='green',label='Early Blight AUC = %0.3f' % roc\_auc[1])

plt.plot(fpr[2], tpr[2], color='blue',label='Healthy AUC = %0.3f' % roc\_auc[2])

plt.plot(fpr[3], tpr[3], color='red',label='Late Blight AUC = %0.3f' % roc\_auc[3])

plt.plot(fpr[4], tpr[4], color='pink',label='Leaf Mold AUC = %0.3f' % roc\_auc[4])

plt.plot(fpr[5], tpr[5], color='purple',label='Septoria Leaf Spot AUC = %0.3f' % roc\_auc[5])

plt.plot(fpr[6], tpr[6], color='brown',label='Spider Mites AUC = %0.3f' % roc\_auc[6])

plt.plot(fpr[7], tpr[7], color='cyan',label='Target Spot AUC = %0.3f' % roc\_auc[7])

plt.plot(fpr[8], tpr[8], color='yellow',label='Mosaic Virus AUC = %0.3f' % roc\_auc[8])

plt.plot(fpr[9], tpr[9], color='black',label='Yellow Leaf Curl Virus AUC = %0.3f' % roc\_auc[9])

plt.title('Tomato Leaves Diseases ROC curve')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive rate')

plt.legend(loc='best')

from sklearn.metrics import confusion\_matrix

sns.heatmap(confusion\_matrix(y\_test,y\_pred))

cm=confusion\_matrix(y\_test,y\_pred)

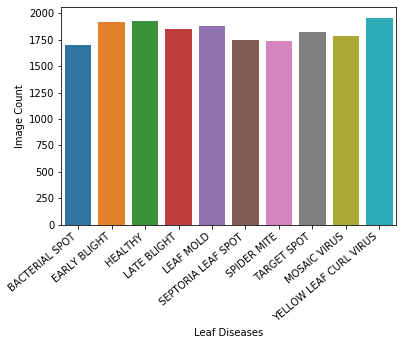
**RESULTS AND SCREEN SHOTS**

Leaf disease detection using deep learning has shown promising results in recent years. The use of deep learning models, such as Convolutional Neural Networks (CNNs), has significantly improved the accuracy of leaf disease detection systems.

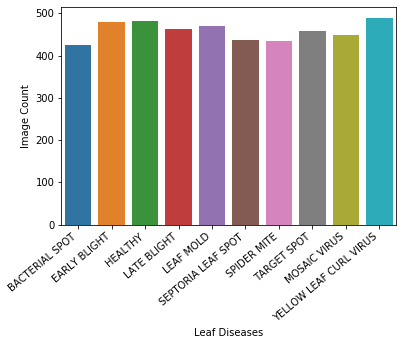
One of the most widely used datasets for leaf disease detection is the PlantVillage dataset, which contains images of healthy and diseased leaves from various plant species. Researchers have used this dataset to train deep learning models, such as CNNs, to accurately identify different types of leaf diseases.

The performance of these models varies depending on the architecture and training parameters used. However, some studies have reported very high accuracy rates for leaf disease detection. For example, a study published in the journal Computers and Electronics in Agriculture reported an accuracy rate of 98.67% for tomato leaf disease detection using a deep learning model based on the Inception-v3 architecture.

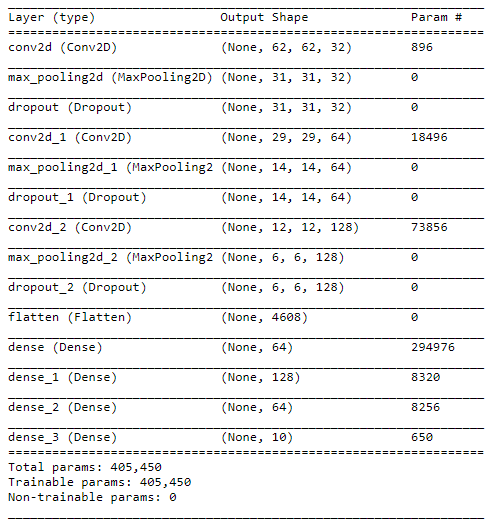
Overall, leaf disease detection using deep learning has shown great potential for improving crop yield and reducing crop loss due to diseases. With further A research and development, it is likely that we will see even more accurate and efficient systems for detecting and diagnosing leaf diseases in the future.



A total of 4585 images were taken for testing.

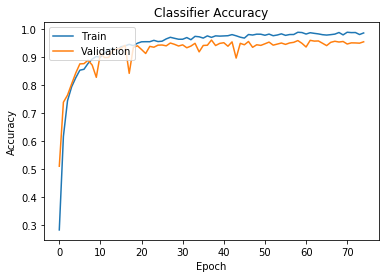
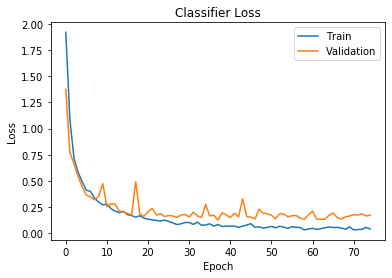


**Model Summary**



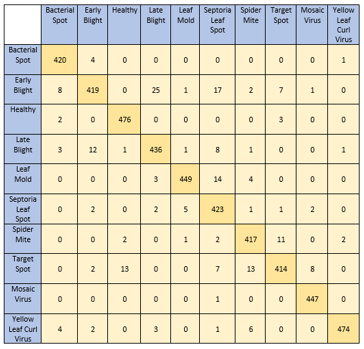
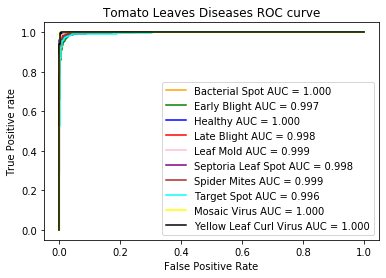
**Training**

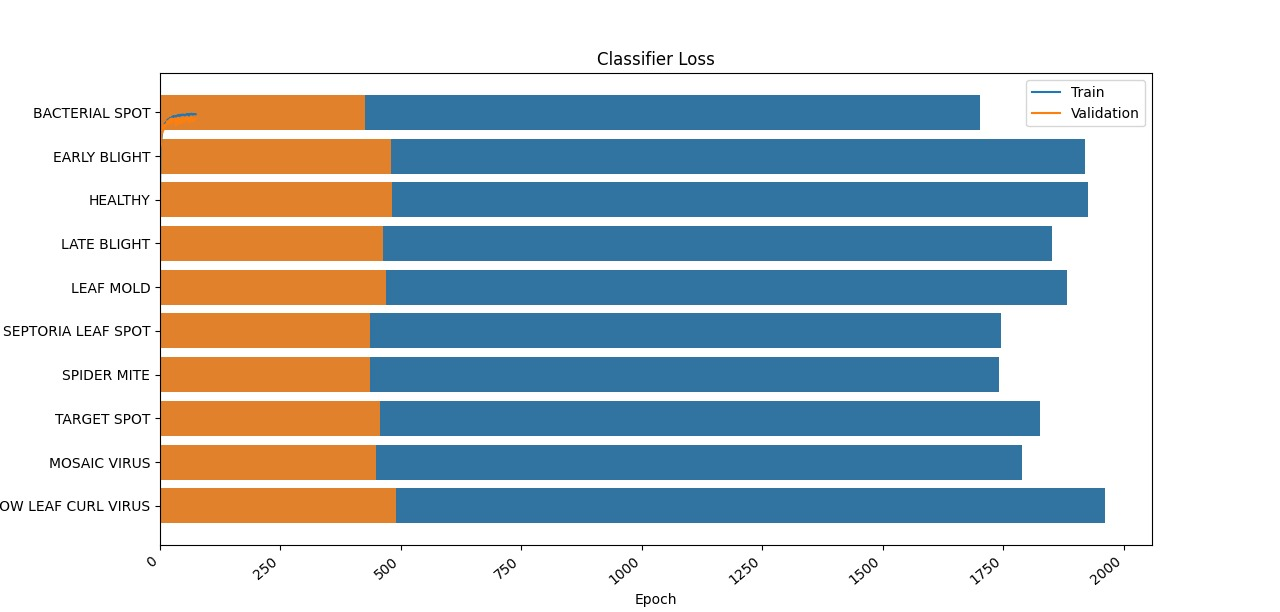
A validation split 20% was used. The model was trained on 14676 images and validated on 3669 images for 75 epochs (50 epochs will do just fine). The training accuracy was 98.63% while that of validation was 95.48%.

**Testing**

Testing of the model was done on 4585 images. The accuracy was found to be 95.42%. The ROC-AUC score was also calclulated and for each category of disease the AUC score was greater than 0.95.



**TESTING**

**SYSTEM TEST**

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

**8.1 TYPES OF TESTS**

Unit testing Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be

Functions : identified functions must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing White**

Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level**.**

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

• All field entries must work properly.

• Pages must be activated from the identified link.

• The entry screen, messages and responses must not be delayed.

**Features to be Tested**

• Verify that the entries are of the correct format

• No duplicate entries should be allowed

• All links should take the user to the correct page. **Integration Testing** Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:**

All the test cases mentioned above passed successfully. No defects encountered.

**9. CONCLUSION**

CONCLUSION

This project presents an automated, low cost and easy to use end-toend solution to one of the biggest challenges in the agricultural domain for farmers – precise, instant and early diagnosis of crop diseases and knowledge of disease outbreaks - which would be helpful in quick decision making for measures to be adopted for disease control. This proposal innovates on known prior art with the application of deep Convolutional Neural Networks (CNNs) for disease classification, introduction of social collaborative platform for progressively improved accuracy, usage of geocoded images for disease density maps and expert interface for analytics.

High performing deep CNN model “Inception” enables real time classification of diseases in the Cloud platform via a user facing mobile app. Collaborative model enables continuous improvement in disease classification accuracy by automatically growing the Cloud based training dataset with user added images for retraining the CNN model. User added images in the Cloud repository also enable rendering of disease density maps based on collective disease classification data and availability of geolocation information within the images.

Overall, the results of our experiments demonstrate that the proposal has significant potential for practical deployment due to multiple dimensions – the Cloud based infrastructure is highly scalable and the underlying algorithm works accurately even with large number of disease categories, performs better with high fidelity real-life training data, improves accuracy with increase in the training dataset, is capable of detecting early symptoms of diseases and is able to successfully differentiate between diseases of the same family.

**10.BIBLOGRAPHY**

**REFERENCES:**

1. 1.Lalit P.Saxena and Leisa J.Armstrong.”A Survey of image processing techniques for Agriculture

2. A.Krizhevsky,I.Sutskever and G.E.Hinton.”Imagenet Classification with deep convolutional neural network”.

3. D.L.Hernandez-Rabadan,F.Ramous-Quintana and J.Guerrero Juk.”Integrating soms and a bayesian classifier for segmenting diseased plants in uncontrolled environments”.

4. S.Sankaran,A.Mishra,J.M.Majaand.”Visible-near infrared spectroscopy in citrus orchards”.

5. Christian Szegedy,Vincent Vanhoucke.”Rethinking thr inception architecture for computer vision”. 6. Jeon, Wang-Su, and Sang-Yong Rhee. "Plant leaf recognition using a convolution neural network." International Journal of Fuzzy Logic and Intelligent Systems 17.1 (2017): 26- 34.

7. Amara, Jihen, Bassem Bouaziz, and Alsayed Algergawy. "A Deep Learning-based Approach for Banana Leaf Diseases Classification." BTW (Workshops). 2017.

8. Lee, Sue Han, et al. "How deep learning extracts and learns leaf features for plant classification." Pattern Recognition 71 (2017): 1-13.

9. Sladojevic, Srdjan, et al. "Deep neural networks-based recognition of plant diseases by leaf image classification." Computational intelligence and neuroscience 2016 (2016).

10. Lee, Sue Han, et al. "Plant Identification System based on a a. Convolutional Neural Network for the LifeClef 2016 Plant Classification Task." CLEF (Working Notes). 2016.

11. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference.

12. K.Padmavathi, and K.Thangadurai,“Implementation of RGB and Gray scale images in plant leaves disease detection –comparative study,”

13. Kiran R. Gavhale, and U. Gawande, “An Overview of the Research on Plant Leaves International Journal of Pure and Applied Mathematics Special Issue 882.Disease detection using Image Processing Techniques,” IOSR J. of Compu. Eng.

14. Y. Q. Xia, Y. Li, and C. Li, “Intelligent Diagnose System of Wheat Diseases Based on Android Phone,” J. of Infor. & Compu. Sci., vol. 12, pp. 6845-6852, Dec. 2015.

15. Wenjiang Huang, Qingsong Guan, JuhuaLuo, Jingcheng Zhang, Jinling Zhao, Dong Liang, Linsheng Huanand Dongyan Zhang, “New Optimized Spectral Indices for Identifying and Monitoring Winter Wheat Diseases”, IEEE journal of selected topics in applied earth observation and remote sensing,Vol. 7, No. 6, June 2014

16. Monica Jhuria, Ashwani Kumar, and RushikeshBorse, “Image Processing For Smart Farming: Detection Of Disease And Fruit Grading”, Proceedings of the 2013.

17. Zulkifli Bin Husin, Abdul Hallis Bin Abdul Aziz, Ali Yeon Bin MdShakaffRohaniBinti S Mohamed Farook, “Feasibility Study on Plant Chili Disease Detection Using Image Processing Techniques”, 2012. 18. Mrunalini R. Badnakhe, Prashant R. Deshmukh, “Infected Leaf a. Analysis and Comparison by Otsu Threshold and k-Means Clustering”,

19. H. Al-Hiary, S. Bani-Ahmad, M. Reyalat, M. Braik and Z. ALRahamneh, “Fast and Accurate Detection and Classification of Plant Diseases”,

20. Chunxia Zhang, Xiuqing Wang, Xudong Li, “Design of Monitoring and Control Plant Disease System Based on DSP&FPGA”,

21. RajneetKaur , Miss. ManjeetKaur“A Brief Review on Plant DiseaseDetection using in Image Processing”IJCSMC, Vol. 6, Issue. 2, February 2017

22. L. Saxena and L. Armstrong, “A survey of image processing techniques for agriculture,” in Proceedings of Asian Federation for Information Technology in Agriculture, 2014, pp. 401-413.

23. E. L. Stewart and B. A. McDonald, “Measuring quantitative virulence in the wheat pathogen Zymoseptoria tritici using high-throughput automated image analysis,” in Phytopathology 104 9, 2014, pp. 985– 992.

24. A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems, 2012.

25. D. P. Hughes and and M. Salathé, “An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing,” in CoRR abs/1511.08060, 2015

26. S. Raza, G. Prince, J. P. Clarkson and N. M. Rajpoot, “Automatic detection of diseased tomato plants using thermal and stereo visible light images,” in PLoS ONE, 2015. 52

27. D. L. Hernández-Rabadán, F. Ramos-Quintana and J. Guerrero Juk, “Integrating soms and a bayesian classifier for segmenting diseased plants in uncontrolled environments,” 2014, in the Scientific World Journal, 2014.

28. S. Sankaran, A. Mishra, J. M. Maja and R. Ehsani, “Visible-near infrared spectroscopy for detection of huanglongbing in citrus orchards,” in Computers and Electronics in. Agriculture 77, 2011, pp. a. 127–134.

29. C. B. Wetterich, R. Kumar, S. Sankaran, J. B. Junior, R. Ehsani and L. G. Marcassa, “A comparative study on application of computer vision and fluorescence imaging spectroscopy for detection of huanglongbing citrus disease in the USA and Brazil,” in Journal of Spectroscopy, 2013.