**A**

**Mini Project Report**

**On**

**“LEAF DISEASE DETECTION USING DEEP LEARNING TECHNIQUES”**

Submitted in partial fulfillment of the Requirements for the award of the degree of

**Bachelor of Technology**

**In**

**Information Technology**

**By**

**K. NAVEEN BABU -20R21A1229**

**K. HARSHA VARDHAN - 20R21A1232**

**P. NANDU - 20R21A1241**

**J. PREM CHANDRA - 20R21A1225**

Under the guidance of

**Mrs. Jeethu Philip**

**Assistant Professor**



**2022**

**Department of Information Technology**



**Department of Information Technology**

**CERTIFICATE**

This is to certify that the project entitled **“Leaf Disease Detection Using Deep Learning Techniques”** has been submitted by K. Naveen Babu**(20R21A1229), K. Harsha Vardhan(20R21A1232), J. Prem Chandra(20R21A1225)** and **P. Nandu(20R21A1241)** in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Information Technology from Jawaharlal Nehru Technological University, Hyderabad. The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

**Internal Guide Head of the Department**

Mrs. Jeethu Philip, Dr. NV. Rajsekhar Reddy

Assistant Professor Professor

**External Examiner**



**Department of Information Technology**

**DECLARATION**

We hereby declare that the project entitled **“Leaf Disease Detection Using Deep Learning Techniques”** is the work done during the period from **August 2022 to December 2022** and is submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Jawaharlal Nehru Technology University, Hyderabad. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

**K. Naveen Babu-20R21A1229**

**K. Harsha Vardhan-20R21A1232**

**J. Prem Chandra-20R21A1225**

**P. Nandu-20R21A1241**



**Department of Information Technology**

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we now have the opportunity to express our guidance for all of them.

First of all, we would like to express our deep gratitude towards our internal guide **Mrs. Jeethu Philip, Assistant Professor, Department of IT** for his support in the completion of our dissertation. We wish to express our sincere thanks to **Dr.NV. Rajsekhar Reddy , HOD, Dept. of IT** and also to principal **Dr. K. Srinivas Rao** for providing the facilities to complete the dissertation.

We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

**K. Naveen Babu-20R21A1229**

**K. Harsha Vardhan-20R21A1232**

**J. Prem Chandra-20R21A1225**

**P. Nandu-20R21A1241**



**Department of Information Technology**

**ABSTRACT**

India is a country with a population of approximately 1.38 billion as of April 2020. Estimates put the total number of farmers in India somewhere between 95.8 million. It must be noted that 18% of India’s GDP is produced from the agricultural sector.

Agriculture is the mainstay of the Indian economy. Immense commercialization of an agriculture has creates a very negative effect on our environment. In agricultural field the disease in plants is more common and the detection of disease in plants has become more feasible.

Plant leaf diseases and destructive insects are a major challenge in the agriculture sector. Faster and an accurate prediction of leaf diseases in crops could help to develop an early treatment technique while considerably reducing economic losses. Modern advanced developments in Deep Learning have allowed researchers to extremely improve the performance and accuracy. In this paper, we proposed a deep-learning-based approach to detect leaf diseases in many different plants using images of plant leaves. Our goal is to find and develop the more suitable deep-learning methodologies for our task. The proposed system can effectively identified different types of diseases with the ability to deal with complex scenarios from a plant’s area.

**INDEX**

|  |  |
| --- | --- |
| **Certificate** | ⅱ |
| **Declaration** | ⅲ |
| **Acknowledgement** | ⅳ |
| **Abstract** | ⅴ |
| **Chapter 1** |  |
| **Introduction** | 1 |
| 1.1 Overview | 1 |
| 1.2 Purpose of the project | 1 |
| 1.3 Motivation | 2 |
| **Chapter 2** |  |
| **Literature Survey** | 3 |
| **Chapter 3** |  |
| **Proposed System** |  |
| 3.1 Existing System | 5 |
| 3.2 Limitations Of Existing System | 6 |
| 3.3 Proposed System | 7 |
| 3.3.1 Proposed System Architecture | 8 |
| 3.4 Objectives Of Proposed System | 8 |
| **Chapter 4** |  |
| **System Requirements** | 9 |
| 4.1 Software Requirements | 9 |
| 4.2 Hardware Requirements | 9 |
| 4.3 Software Description | 10 |
| 4.4 Module Diagram | 15 |
|  |  |
|  |  |

**Chapter 5 16**

**System Design**

5.1 System Architecture

5.2 Data Flow Diagram.

5.3 UML Diagram

5.3.1 Goals Of UML

5.3.2 Use Case Diagram

5.3.3 Class Diagram

5.3.4 Sequence Diagram

5.3.5 Activity Diagram

**Chapter 6**

**System Implementation**

6.1 Source Code

6.2 Modules

**Chapter 7**

**System Testing**

**Chapter 8**

**Results**

**Chapter 9**

**Conclusion**

**Chapter 10**

**References**  19

**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

Economy contributes the most for the productivity of the agriculture. In agricultural field, the disease in plants is more common and the detection of disease in plants has become more feasible due to the above reason. These days plant disease detection has acquired enlarging scrutiny in surveilling crops of large and various fields. Farmers undergo significant hassles in chop and changing from one disease administer principle to a different one. We can identify or spotting the tomato leaf diseases for detection for surveillance and monitoring experts is the standard approach for detection. The plants get seriously affected if the proper control hasn't been taken and this represents the quality of the pants the production of the plants will be affected. Detection of disease through some mechanized technique and methodology is efficient and constructive because it decreases an outsized toil of surveilling in the large cultivation. In the premature phase we can detect the symptoms of the plant diseases since their first appearance on their leaves of the plants. By using this paper we can identify the algorithm which is used for image segmentation and for automated classification used for the detection of diseases of leaves in the plants. It also covers distinct disease classification methods of working which is used for the detection of diseases in plants.

**1.2 PURPOSE OF THE PROJECT**

The objective of this research is to develop a deep learning model that can recognize the illness that affects every leaf. This model will categorize the input picture and show whether the leaf is infected or not. The user may easily utilise this method to determine if a leaf is infected or not at any moment.

**1.3 MOTIVATION**

Farming has a very wide range of issues, including those related to classifying plant diseases. Plants are being destroyed as a result of farmers' tardy detection of the disease, which makes it impossible for them to find a cure. Although there are various methods for diagnosing illnesses, most are slow and have limitations. All of these circumstances motivated us to create a ground-breaking application using deep learning technology to address this problem.

**CHAPTER 2**

**LITERATURE SURVEY**

We reviewed the current methods for identifying leaf diseases as part of a thorough literature assessment. In order to create this survey, references to research papers, journals, and publications were also made.

1. **Jiang, Peng, et al. "Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks.**

A methodology for early and accurately plant diseases detection, using artificial neural network (ANN) and diverse image processing techniques. As the proposed approach is based on ANN classifier for classification and Gabor filter for feature extraction, it gives better results with a recognition rate of up to 91%. Image identification has become feasible with the advent of Convolutional Neural Networks. But designing a CNN that identifies objects and classifies them into distinct classes is a complex task. By making use of transfer learning it can be simplified. In transfer learning we have trained our model that has been trained on Plant Village dataset using 12GB TESLA k80 GPU. Also Transfer learning significantly reduces training time and gives much better performance for relatively small dataset.

**ADVANTAGES**

* It searches from a large sampling of the cost surface.
* Large number of variables can be processed at the same time.
* It can optimize variables with highly complex cost surfaces.

**DIADVANTAGES**

* Large complexity of network structure.
* Require long training time.
* Difficult to understand the learned function.

**[2] Wang, Qimei, et al. "Identification of Tomato Disease Types and Detection of Infected Areas Based on Deep Convolutional Neural Networks and Object Detection Techniques."**

Presents the technique to classify and identify the different disease through which plants are affected. In Indian Economy a Machine learning based recognition system will prove to be very useful as it saves efforts, money and time too. The approach given in this for feature set extraction is the color co-occurrence method. For automatic detection of diseases in leaves, [neural networks](https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/neural-networks) are used. The approach proposed can significantly support an accurate detection of leaf, and seems to be important approach, in case of steam, and [root diseases](https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/root-diseases), putting fewer efforts in computation. In plants, disease appears on leaf therefore the histogram matching is done on the basis of edge detection technique and color feature. Layers separation technique is used for the training process which includes the training of these samples which separate the layers of RGB image into red, green, and blue layers and edge detection technique which detecting edges of the layered images. Spatial Gray-level Dependence Matrices are used for developing the color co-occurrence texture analysis method.

**ADVANTAGES**

* Prediction accuracy is high.
* Having robust working when training example have error in them.
* It is fast, simple and easy to program.

**DISADVANTAGES**

* Large complexity of network structure.
* Some methods are not completely established.
* Tend to overfit.

**CHAPTER 3**

**3.1 EXISTING SYSTEM**

H.Al-Hiary et al, proposed a plant disease detection method, using the k-means clustering algorithm with Neural Network. Both the detection and classification of plant diseases can be identified by this trained model. It provides a precise accuracy between 83% and 94%. Merits include precise disease detection with less computational effort. Demerits are recognition rate is found to be declined.[1]

Dheeb Al Bashish et al, proposed K-means clustering for segmentation and ANN for disease detection and classification of leaf disease. This proposed model recognized the diseases with 93% accuracy. Merits include, very effective in recognizing diseases. Demerits include, finer segmentation and feature extraction is required.[2]

Anand.H.kulkarni et al, proposed the ANN classifier for classification and recognition of diseased leaves. Gabor filter is used for filtration and segmentation of input images. The proposed model obtains 91% of accuracy. Merits include, good classification and recognition are developed. Demerits include better classifiers that can be used for improved recognition rates.[3]

S.Arivazhagan et al, proposed a Support Vector Machine(SVM) classifier. This proposed model gives 94% of accuracy. Merits include automatic detection and classification of leaf disease. Demerits include NN classifiers that can be used to attain higher performance.[4]

Usama Mokhtar et al, proposed SVM with different kernel functions to detect the tomato leaves diseases with an excellent annotation of 99.5%. Merits include effective and reliable results. Demerits include a large scale of inputs that leads to a decrease in performance.[5]

Sharada.P.Mohanty et al, proposed a deep learning technique with AlexNet and GoogleNet architectures that attains a better disease diagnosis. This trained model (GoogleNet) achieves 99.35% of accuracy. Merits include, classification is very fast in proposed DNN. Demerits include, multiple hours of training is required.[6]

Srdjan Sladojevic et al, proposed a deep Convolutional Neural Network technique. This trained model provides 96.3% of accuracy. Merits include, the proposed methodology is beneficial to get a more accurate classification. Demerits include, fine-tuning and augmentation is required to improve accuracy. [7]

H.Sabrol proposed a classification tree model based on supervised learning techniques. The extracted features from the segmented image are fed to this classification tree, which classifies tomato plant disease with 97.3% of accuracy. Merits include classification tree results with good accuracy. Demerits include the latest classification techniques that can be used.[8]

Mohammed Brahimi et al, proposed Convolutional Neural Network for improved classification. The significance of CNN is an automatic feature extraction from raw input images. Thus the trained model achieves 99.18% of accuracy. Merits include high performance. Demerits include computation and the size of deep NN can be reduced.[9]

Alvaro Fuentes et al, proposed an object detection method with Faster r-CNN, R-FCN, and SSD algorithms merged with the latest feature extractors of deep CNN. This proposed model, R-CNN with VGG-16 shown better recognition results. The merit is false positives are reduced in the training phase. The demerit is slag in performance.[10]

Halil Durmas et al, proposed Deep Learning-based network architectures namely sqeezeNet and AlexNet for tomato leaves disease detection. AlexNet provided better classification accuracy of 95.65%. The merit includes, squeezeNet is light in weight requires low computational needs. Demerits are long training time and small batch size.[11]

**3.2 LIMITATIONS OF EXISTING SYSTEM**

Concisely summarizing the disadvantages of the above implementations:

* Currently, available solutions with DL methods for plant disease detection have somewhat been successful, however, there is still large room for improvement.
* There are several current limitations in this research field.
* One of them is that currently available datasets do not contain images gathered and labeled from real-life situations. Therefore, training is conducted with images taken in a controlled environment.
* Another limitation is that currently, the proposed methods cannot detect multiple diseases in one image or cannot detect multiple occurrences of the same diseases in one image.
* Several different experiments were conducted in order to emphasize the current method’s limitations for practical usage and to try to overcome those issues by proposing novel strategies.

**PROPOSED SYSTEM**

**3.3 PROPOSED SYSTEM**

The suggested strategy is to create a new Deep Learning method where the user may provide photos of diseased leaves as input. With the aid of a deep learning classification model created using the Faster Region-based Convolutional Neural Network (RCNN) method, the image is categorized. A data set is gathered, to which data science techniques and data pre-processing techniques are employed for the purpose of extraction, in order to train the classification model.

**COMPONENTS OR USERS IN THE PROPOSED SYSTEM**

**Admin**

The user processes the dataset once it has been collected and extracts the necessary characteristics. A classification technique known as the Faster Region-based Convolutional Neural Network is used in association with the preprocessed dataset by the admin to train the model or classifier (RCNN).

**DL Model/Classifier**

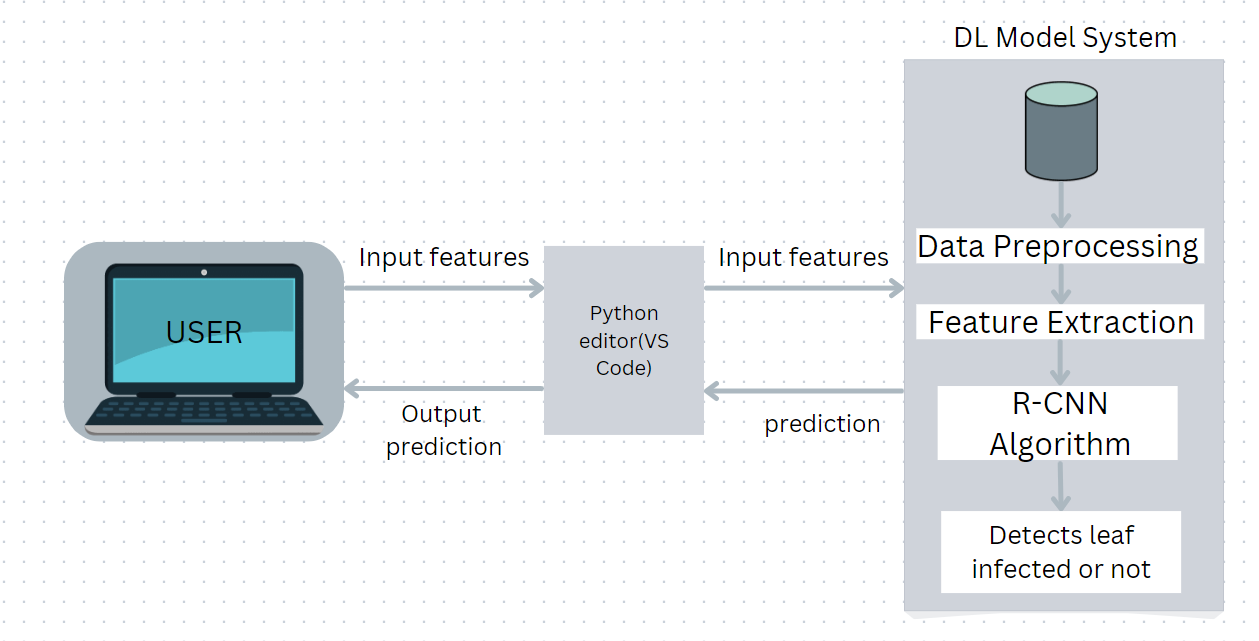
Using the dataset and the RCNN method, the admin designs the classifier. By categorizing the leaf as infected, the trained model is in charge of processing the end user's input data and providing the end user with the final result.

**End user**

The one who needs to know whether or not a leaf is infected is the end user. Sending an image of a diseased leaf to the dataset that the admin uses will allow the end user to confirm the leaf illness. The final result is sent to the user when the classifier has finished all necessary processing.

**3.3.1 PROPOSED SYSTEM ARCHITECTURE**

An architectural diagram outlines the system’s components, their relationships, and system functionality. A huge dataset where the data is pre-processed, then it is classified using RCNN algorithm and finally the model is trained. Users input the leaf image into the dataset, which evaluates the leaf based on its features and determines whether the leaf is infected or not. The final output is then sent to the user.



**3.4 OBJECTIVES OF PROPOSED SYSTEM**

The objectives of the proposed system include the following:

* To determine if the leaf is diseased or not.
* To provide a quicker categorization model and enhance accuracy.
* To obtain a dataset of numerous photos of diseased leaves and to preprocess the dataset before training the model.
* To classify the images using classification algorithm –Faster Region-based Convolutional Neural Network (RCNN).

**CHAPTER 4**

**SYSTEM REQUIREMENTS**

**4.1 SOFTWARE REQUIREMENTS**

Below are the software requirements for the application development:

1. The required language is python
2. Editor Python - PyCharm or VSCode
3. Python Libraries for Model Building
4. Google Chrome, Firefox, Microsoft Edge or Brave Browser with Extension Support

**4.2 HARDWARE REQUIREMENTS**

Below are the hardware requirements for the application development:

1. Operating System : windows 10
2. Processor : intel i5(min)
3. Ram : 8 GB(min)
4. Hard Disk : 128 GB(min)

**4.3 SOFTWARE DESCRIPTION**

**Python:**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

**Getting Python**

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python [https://www.python.org](https://www.python.org/).

Windows Installation

Here are the steps to install Python on Windows machine.

* Open a Web browser and go to <https://www.python.org/downloads/>.
* Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
* To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
* Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

**First Python Program**

Let us execute programs in different modes of programming.

**Interactive Mode Programming**

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

$ python

Python2.4.3(#1,Nov112010,13:34:43)

[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2

Type"help","copyright","credits"or"license"for more information.

>>>

Type the following text at the Python prompt and press the Enter −

>>>print"Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**. However in Python version 2.4.3, this produces the following result −

Hello, Python!

**Script Mode Programming**

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file −

print"Hello, Python!"

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

$ python test.py

This produces the following result −

Hello, Python!

**Flask Framework:**

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods −

|  |  |
| --- | --- |
| **Sr.No** | **Methods & Description** |
| 1 | **GET**  Sends data in unencrypted form to the server. Most common method. |
| 2 | **HEAD**  Same as GET, but without response body |
| 3 | **POST**  Used to send HTML form data to server. Data received by POST method is not cached by server. |
| 4 | **PUT**  Replaces all current representations of the target resource with the uploaded content. |
| 5 | **DELETE**  Removes all current representations of the target resource given by a URL |

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

<html>

<body>

<formaction="http://localhost:5000/login"method="post">

<p>Enter Name:</p>

<p><inputtype="text"name="nm"/></p>

<p><inputtype="submit"value="submit"/></p>

</form>

</body>

</html>

Now enter the following script in Python shell.

from flask importFlask, redirect,url\_for, request

app=Flask(\_\_name\_\_)

@app.route('/success/<name>')

def success(name):

return'welcome %s'% name

@app.route('/login',methods=['POST','GET'])

def login():

ifrequest.method=='POST':

user=request.form['nm']

return redirect(url\_for('success',name= user))

else:

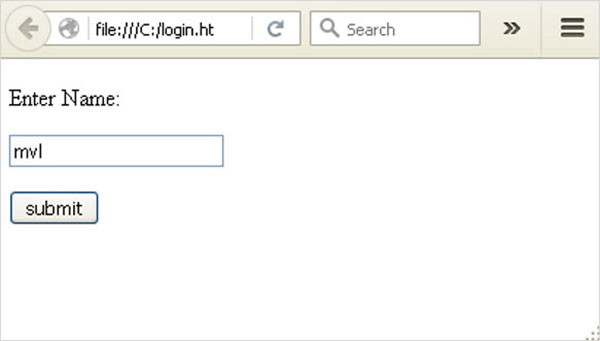
user=request.args.get('nm')

return redirect(url\_for('success',name= user))

if \_\_name\_\_ =='\_\_main\_\_':

app.run(debug =True)

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.

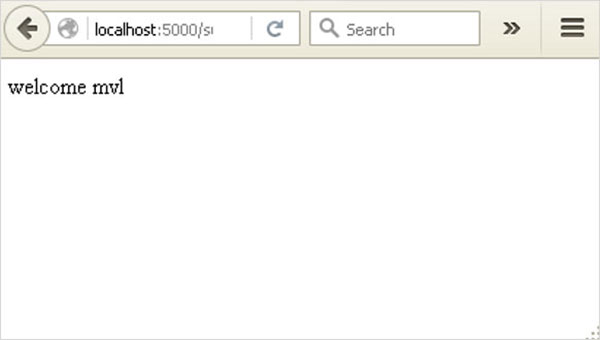


Form data is POSTed to the URL in action clause of form tag.

**http://localhost/login** is mapped to the **login()** function. Since the server has received data by **POST** method, value of ‘nm’ parameter obtained from the form data is obtained by −

user = request.form['nm']

It is passed to **‘/success’** URL as variable part. The browser displays a **welcome** message in the window.



Change the method parameter to **‘GET’** in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of ‘nm’ parameter is now obtained by −

User = request.args.get(‘nm’)

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to ‘nm’ parameter is passed on to ‘/success’ URL as before.

**What is Python?**

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

**What can Python do?**

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

**Why Python?**

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

* Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

## Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\Your Name>python --version

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

python --version

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

## Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

C:\Users\Your Name>python helloworld.py

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

print("Hello, World!")

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

C:\Users\Your Name>python helloworld.py

The output should read:

Hello, World!

Congratulations, you have written and executed your first Python program.

## The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

C:\Users\Your Name>python

From there you can write any python, including our hello world example from earlier in the tutorial:

C:\Users\Your Name>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.  
>>> print("Hello, World!")

Which will write "Hello, World!" in the command line:

C:\Users\Your Name>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.  
>>> print("Hello, World!")  
Hello, World!

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

exit()

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

>>> print("Hello, World!")  
Hello, World!

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

C:\Users\*Your Name*>python myfile.py

Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

Example

if 5 > 2:  
  print("Five is greater than two!")

Python will give you an error if you skip the indentation:

Example

if 5 > 2:  
print("Five is greater than two!")

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

Example

Comments in Python:

#This is a comment.  
print("Hello, World!")

Docstrings

Python also has extended documentation capability, called docstrings.

Docstrings can be one line, or multiline.

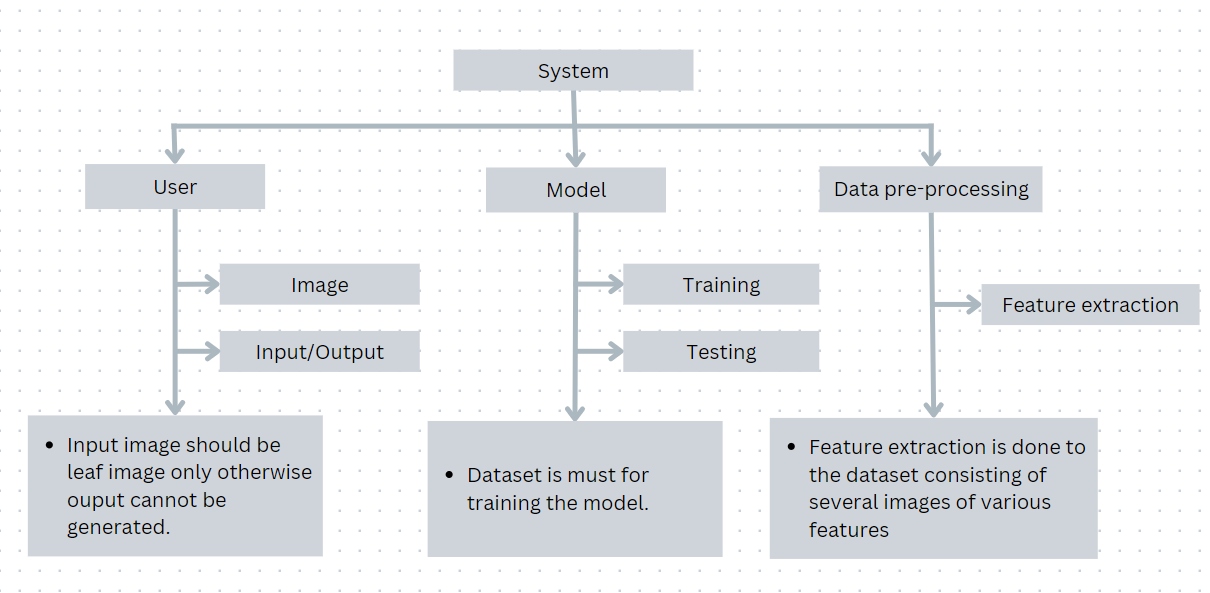
Python uses triple quotes at the beginning and end of the docstring:

Example

Docstrings are also comments:

"""This is a   
multiline docstring."""  
print("Hello, World!")

**4.4 MODULE DIAGRAM**



The proposed system consists of three modules as mentioned in the above diagram.

* User Interface
* ML Model
* Data Pre-Processing

**USER INTERFACE**

The functions in the user interface module includes users leaf image before doing any operations. Once the user selects the image, the image will be stored in the database. This module allows only the leaf images as input and output can only be generated if the input or image sent by the user is valid. The output determines the class of the disease.

**DATA PRE-PROCESSING**

Data normalization technique is used on the dataset, which removes the sophisticated noise, before analyzing them. Feature extraction is the major functionality of the data pre-processing module where various features are extracted from the dataset consisting of attributes of diseased leaf images that is used to train the model.

**ML MODEL**

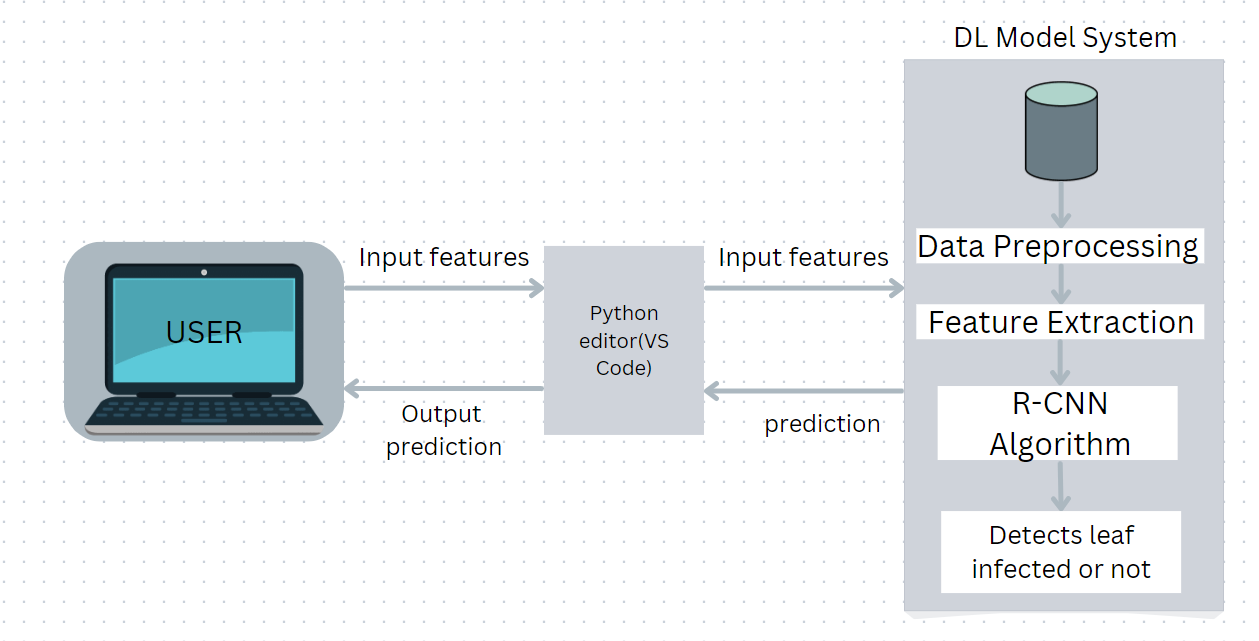
The model uses the dataset after feature extraction for the purpose of training. After the model is trained, the input image from the user is taken as test data to determine the class of the disease.

**CHAPTER 5**

**SYSTEM DESIGN**

**5.1 SYSTEM ARCHITECTURE:**

An architectural diagram outlines the system’s components, their relationships, and system functionality. A huge dataset where the data is pre-processed, then it is classified using RCNN algorithm and finally the model is trained. Users input the leaf image into the dataset, which evaluates the leaf based on its features and determines whether the leaf is infected or not. The final output is then sent to the user.



**Figure 5.1 System Architecture**

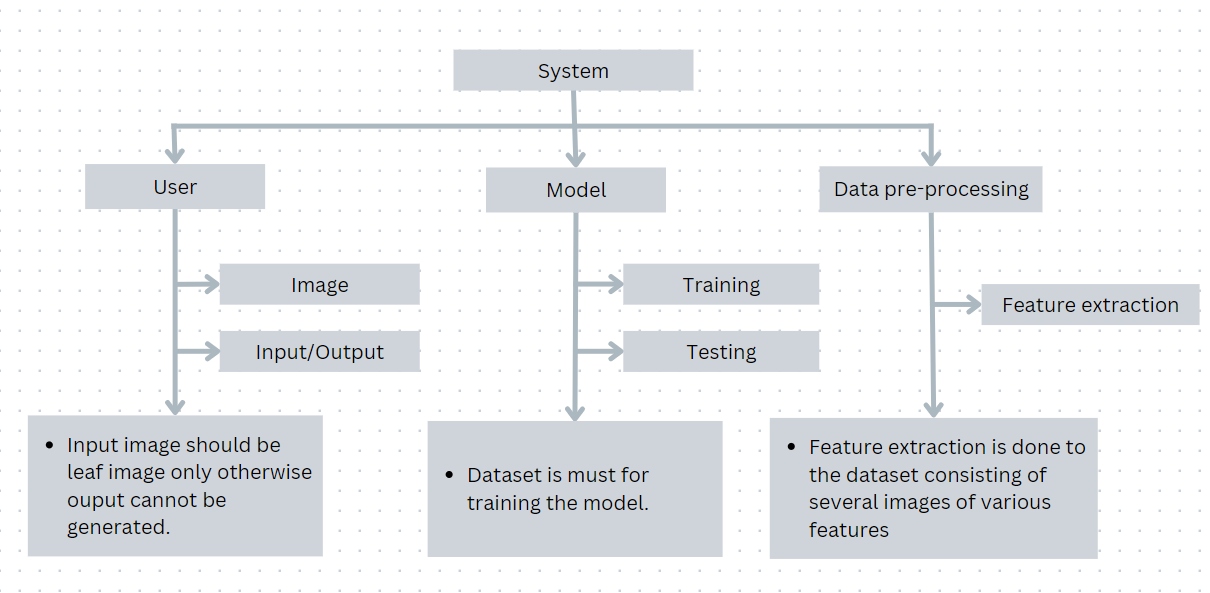
**5.2 DATA FLOW DIAGRAM:**

❖ The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

❖ The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

❖ DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

❖ DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing



**Figure 5.2 Data Flow Diagram**

**5.3 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purposemodeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Metamodel and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for businessmodeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects

**5.3.1 GOALS OF UML:**

The Primary goals in the design of the UML are as follows:

❖ Provide users a ready-to-use, expressive visual modeling Language so that they candevelop and ❖ exchange meaning models

❖ Provide extendibility and specialization mechanisms to extend the core concepts.

❖ Be independent of particular programming languages and development process. ❖ Provide a formal basis for understanding the modeling language.

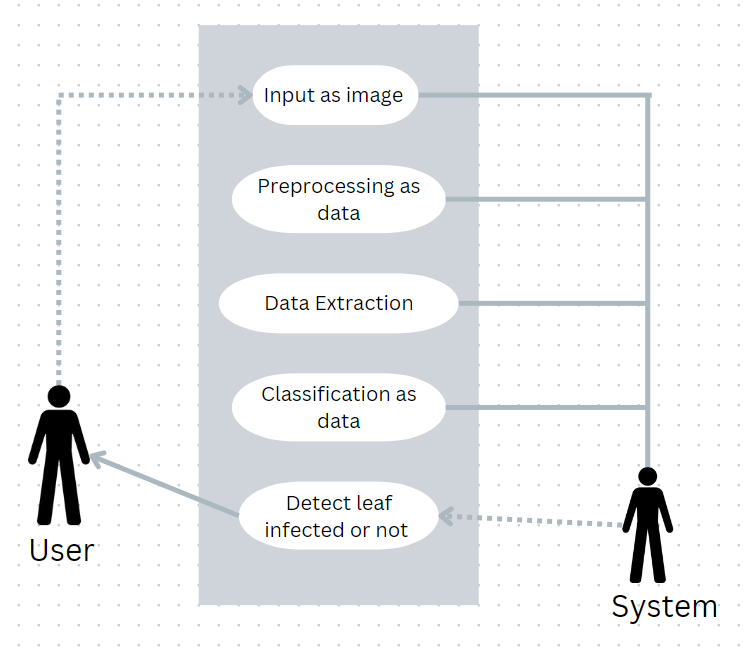
❖ Encourage the growth of OO tools market.

❖ Support higher level development concepts such as collaborations, frameworks, patterns and Components.

❖ Integrate best practices

**5.3.2 USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**Figure 5.3.2 Use Case Diagram**

**5.3.3 CLASS DIAGRAM:**

|  |
| --- |
| Output |
| Features extraction classification |

|  |
| --- |
| Input |
| Input data |
| Preprocessing() |

**Figure 5.3.3 Class Diagram**

**5.3.4 SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

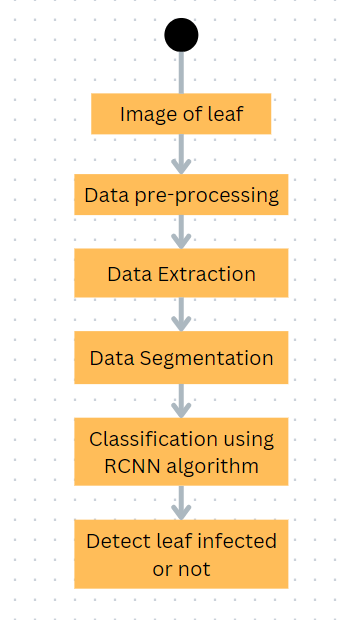
The sequence diagram depicts the processes involved and the sequence of messages exchange between the processes needed to carry out the functionalities.

**The sequence of the application is as follows:**

* + - 1. The initial step in our project includes collection of dataset and applying deep learning techniques to perform pre-processing of the data.
      2. The pre-processed data is passed to the classification algorithm which is Region based Convolutional Neutral Network(RCNN) in order to build and train a classifier.Now the model is ready to accepts input from the user.
      3. The Input that is pre-processed is passed to the trained model which classifies and gives the class of the disease.
      4. The model evaluates the input leaf image based on various attributes and classifies it as a infected or not. This classified result is sent to the user as final output.

**5.3.5 ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities andactions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step- by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**Figure 5.3.5 Activity Diagram**

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**6.1 Source Code**

import cv2

from tkinter import \*

import numpy as np

from tkinter import filedialog

root = Tk()

root.geometry('400x400') # Set Window size

root.configure(background='black') # Set background to black

def cal\_MaxX\_MaxY(width, height, mask):

max\_x = 0

max\_y = 0

for i in range(0, height): # Detect length of whites on X-axis

temp = 0

for j in range(0, width):

if mask[i, j] == 255:

temp += 1

if temp > max\_x:

max\_x = temp

for i in range(0, width): # Detect length of whites on X-axis

temp = 0

for j in range(0, height):

if mask[j, i] == 255:

temp += 1

if temp > max\_y:

max\_y = temp

return max\_x, max\_y

def main(img\_rec):

diseases = ['brown\_spots', 'paddy blast', 'bacterial leaf', 'Normal leaf']

img = img\_rec

img = cv2.resize(img, (400, 400))

hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

lower\_range = np.array([21, 8, 63], dtype=np.uint8) # Works for BACTERIAL LEAF (100% Accuracy)

upper\_range = np.array([30, 255, 255], dtype=np.uint8)

Ylower\_range = np.array([17, 100, 100], dtype=np.uint8) # For detecting yellow in paddy blast

Yupper\_range = np.array([23, 255, 255], dtype=np.uint8)

Blower\_range = np.array([0, 80, 40], dtype=np.uint8) # Works for separating brown spots and bacterial leaf

Bupper\_range = np.array([20, 255, 255], dtype=np.uint8)

maskLB = cv2.inRange(hsv, lower\_range, upper\_range)

maskY = cv2.inRange(hsv, Ylower\_range, Yupper\_range)

maskB = cv2.inRange(hsv, Blower\_range, Bupper\_range)

tempY = maskY - maskB

tempB = maskB - maskY

heightLB, widthLB = maskLB.shape[:2]

heightY, widthY = maskY.shape[:2]

heightB, widthB = maskB.shape[:2]

heightTY, widthTY = tempY.shape[:2]

heightTB, widthTB = tempB.shape[:2]

print ("heightLB =",heightB)

LBmax\_x, LBmax\_y = cal\_MaxX\_MaxY(widthLB, heightLB, maskLB)

Ymax\_x, Ymax\_y = cal\_MaxX\_MaxY(widthY, heightY, maskY)

Bmax\_x, Bmax\_y = cal\_MaxX\_MaxY(widthB, heightB, maskB)

TYmax\_x, TYmax\_y = cal\_MaxX\_MaxY(widthTY, heightTY, tempY)

TBmax\_x, TBmax\_y = cal\_MaxX\_MaxY(widthTB, heightTB, tempB)

print ("light brown LBmax\_x =",LBmax\_x)

print ("light brown LBmax\_y =",LBmax\_y)

print ("yellow to light brown Ymax\_x =",Ymax\_x)

print ("yellow to light brown Ymax\_y =",Ymax\_y)

print ("brown to black Bmax\_x =",Bmax\_x)

print ("brown to black Bmax\_y =",Bmax\_y)

print ("only yellow TYmax\_x =",TYmax\_x)

print ("only yellow TYmax\_y =",TYmax\_y)

print ("only brown TBmax\_x =",TBmax\_x)

print ("only brown TBmax\_y =",TBmax\_y)

# print Ymax\_x

print ("Yellow = ", TYmax\_y)

#print "Yellowx = ", TYmax\_x

print ("Brown = ", TBmax\_y)

#print "Brownx = ", TYmax\_x

def image\_disk():

Tk().withdraw() # we don't want a full GUI, so keep the root window from appearing

filename = filedialog.askopenfilename() # show an "Open" dialog box and return the path to the selected file

img = cv2.imread(filename)

main(img)

def image\_camera():

cam = cv2.VideoCapture(0)

cv2.namedWindow("test")

img\_counter = 0

while True:

ret, frame = cam.read()

cv2.imshow("test", frame)

if not ret:

break

k = cv2.waitKey(1)

if k % 256 == 27: # PRESS ESCAPE TO CLOSE THE WEBCAM WINDOW

# ESC pressed

print("Escape hit, closing...")

break

elif k % 256 == 32: # TAKE IMAGE PY PRESSING SPACE (TAKE ONLY 1 AT A TIME)

# SPACE pressed

img\_name = str(img\_counter) + ".png"

cv2.imwrite(img\_name, frame)

print("{} written!".format(img\_name))

img\_counter += 1

cam.release()

cv2.destroyAllWindows()

my\_img = cv2.imread("0.png") # CV2 will only read image named "0.png"

main(my\_img)

button = Button(root,

text="Select Image From Disk",

fg="red",

command=image\_disk)

button.place(relx=0.5, rely=0.5, anchor=CENTER)

button.pack(side=TOP)

slogan = Button(root,

text="Take Image from WebCam",

fg="red",

command=image\_camera)

slogan.place(relx=1, rely=1, anchor=CENTER)

slogan.pack(side=TOP)

root.mainloop()

**6.2 MODULES :**

**List of modules:**

• Image acquisition.

• Image pre-processing.

• Image enhancement.

• Image segmentation.

• Image analysis.

• Feature extraction.

• Disease classification.

**6.3 MODULES DESCRIPTION:**

**Image Acquisition:**

The first step is to gather data from a publicly accessible repository. The picture is used as the input for further processing. We've chosen the most common image domains so that we can accept any format as input to our method, including.bmp,.jpg, and.gif. The camera feeds the real-time images directly. Since most leaves colour varies from red to green for exact segmentation, a white background is provided for further study, proper visibility, and easy image analysis. Cotton images are captured using an image capturing system in this process. The picture is taken in such a way that any distortion is avoided. The photo was not taken in direct sunlight because it would distort the picture.

**Image Pre-processing:**

The use of computer algorithms to perform image processing on digital images is known as image pre-processing. We can detect the plant by analyzing the image with a specific algorithm. We use a similar approach for image processing and detection with a specific algorithm. The image quality is critical in this process; we can't use the algorithm if the image isn't clear.

**Image Enhancement:**

The process of modifying digital images so that the effects are more appropriate for display or further image processing is known as image enhancement. Any of the following can be used to improve an image.

• Histogram Equalization.

• Noise removal using filters.

• Unsharp mask filtering.

• Decorrelation stretch etc.

**Image Segmentations:**

The method of segmenting a digital image into multiple segments is known as image segmentation (sets of pixels, also known as image objects). Image segmentation is used to make image identification and analysis simpler by dividing the image into several segments and analysing each segment individually. Color, texture, and intensity are all common characteristics among the various segments.

**Image analysis:**

In this step, image segmentation is used to locate the region of interest. The technique used in segmentation is region-based segmentation, which uses the color of the leaf to distinguish between healthy and diseased regions of the plant leaf.

**Feature Extraction:**

Feature extraction is a part of the dimensionally reduction method in machine learning, which divides and reduces a large collection of raw data into smaller classes. When we have a large amount of data and need to minimize the number of resources while avoiding errors, this step is critical. As a result, function extraction aids in the extraction of the best feature from large data sets by selecting and combining variables into functions.

**Disease Classifications:**

It is the method of using our qualified deep learning model to recognize plant disease. A digital camera or equivalent system should be used to take an image of the contaminated plant's leaf. OpenCV was used to scan the image. Then it determines what kind of plant it is. It determines what kind of disease the plant has after finding it.

**CHAPTER 7**

**SYSTEM TESTING**

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.

## TYPE 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 knowledgeof its construction and is invasive. Unit tests perform basic tests at component level andtest 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 specificallyaimed at exposing the problems that arise from the combination of component.

## 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 rejected Functions : identified functions must be exercised.

Output : identified classes of application outputs 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 systemtesting is the configuration oriented system integration test. System testing is based onprocess 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 ofthe inner workings, structure and language of the software, or at least its purpose. It ispurpose. 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 testingin which the software under test is treated, as a black box .you cannot “see” into it. Thetest 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 thesoftware 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.

## 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.

**CHAPTER 8**

**RESULTS**

**8.1 OUTPUT :**

** **

**Output:-**

heightLB = 400

light brown LBmax\_x = 10

light brown LBmax\_y = 21

yellow to light brown Ymax\_x = 0

yellow to light brown Ymax\_y = 0

brown to black Bmax\_x = 0

brown to black Bmax\_y = 0

only yellow TYmax\_x = 0

only yellow TYmax\_y = 0

only brown TBmax\_x = 0

only brown TBmax\_y = 0

Yellow = 0

Brown = 0

**Output:-**

heightLB = 400

light brown LBmax\_x = 316

light brown LBmax\_y = 235

yellow to light brown Ymax\_x = 16

yellow to light brown Ymax\_y = 21

brown to black Bmax\_x = 8

brown to black Bmax\_y = 12

only yellow TYmax\_x = 16

only yellow TYmax\_y = 21

only brown TBmax\_x = 8

only brown TBmax\_y = 10

Yellow = 21

Brown = 10

**CHAPTER 9**

**CONCLUSION**

Plant diseases have long been a major issue in agriculture. Making the best judgments possible based on the outcomes of DL techniques has made precision agriculture possible, enabling early disease identification and the minimizing of losses. Recent developments in DL offer solutions with extremely precise findings, and the technology that is now accessible permits quick processing. The decision-making procedure, nevertheless, may be enhanced. The models that are now available perform poorly when evaluated under actual circumstances.

**CHAPTER 10**

**REFERENCES**

1. Jiang, Peng, et al. "Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks." IEEE Access 7 (2019): 59069-59080.
2. Wang, Qimei, et al. "Identification of Tomato Disease Types and Detection of Infected Areas Based on Deep Convolutional Neural Networks and Object Detection Techniques." Computational Intelligence and Neuroscience 2019 (2019).
3. Kumar, Akshay, and M. Vani. "Image Based Tomato Leaf Disease Detection." 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2019.
4. Ozguven, Mehmet Metin, and Kemal Adem. "Automatic detection and classification of leaf spot disease in sugar beet using deep learning algorithms." Physica A: Statistical Mechanics and its Applications 535 (2019): 122537.
5. Karthik, R., et al. "Attention embedded residual CNN for disease detection in tomato leaves." Applied Soft Computing 86 (2020): 105933.
6. Ashok, Surampalli, et al. "Tomato Leaf Disease Detection Using Deep Learning Techniques." 2020 5th International Conference on Communication and Electronics Systems (ICCES). IEEE, 2020.
7. Kaushik, M., et al. "Tomato Leaf Disease Detection using Convolutional Neural Network with Data Augmentation." 2020 5th International Conference on Communication and Electronics Systems (ICCES). IEEE, 2020
8. Salih, Thair A. "Deep Learning Convolution Neural Network to Detect and Classify Tomato Plant Leaf Diseases." Open Access Library Journal 7.05 (2020): 1.
9. Zhang, Yang, Chenglong Song, and Dongwen Zhang. "Deep learning-based object detection improvement for tomato disease." IEEE Access 8 (2020): 56607-56614.
10. Khan, Asifullah, et al. "A survey of the recent architectures of deep convolutional neural networks." Artificial Intelligence Review 53.8 (2020): 5455-5516.
11. Zhang, Keke, et al. "Can deep learning identify tomato leaf disease?." Advances in Multimedia 2018 (2018).
12. Tm, Prajwala, et al. "Tomato leaf disease detection using convolutional neural networks." 2018 Eleventh International Conference on Contemporary Computing (IC3). IEEE, 2018.
13. de Luna, R.G., Dadios, E.P. and Bandala, A.A., 2018, October. Automated image capturing system for deep learning-based tomato plant leaf disease detection and recognition. In TENCON 2018-2018 IEEE Region 10 Conference (pp. 1414-1419). IEEE.
14. Sardogan, Melike, Adem Tuncer, and Yunus Ozen. "Plant leaf disease detection and classification based on CNN with LVQ algorithm." 2018 3rd International Conference on Computer Science and Engineering (UBMK). IEEE, 2018.
15. Rangarajan, Aravind Krishnaswamy, Raja Purushothaman, and Aniirudh Ramesh. "Tomato crop disease classification using pre-trained deep learning algorithm." Procedia computer science 133 (2018): 1040-1047.
16. Ashqar, Belal AM, and Samy S. Abu-Naser. "Image-Based Tomato Leaves Diseases Detection Using Deep Learning." (2018).