# Plant Disease Detection using CNN

Submitted as Major Project in partial fulfilment of the requirements for the award of the Degree of

## Bachelor of Technology

In

## Computer Science and Engineering

By

**Swetha Annavarapu-16011A0537**

**Tailor Afreen Sultana-16011A0538**

**Thara Chand Chowdary-16011A0539**

**Sachi Pandita-16011A0559**

Under the guidance of

**Dr.D.Vasumathi**

**Professor**


**Department of Computer Science and Engineering,**

**JNTUH College of Engineering Hyderabad**

**Kukatpally, Hyderabad-500085.**

**Department of Computer Science and Engineering**

**JNTUH College of Engineering, Hyderabad**

**Kukatpally, Hyderabad - 500 085**

## DECLARATION BY THE CANDIDATES

We, **Swetha Annavarapu (16011A0537), Tailor Afreen Sultana (16011A0538), Thara Chand Chowdary (16011A0539), Sachi Pandita (16011A0559)**, hereby declare that the major project report entitled "**Plant Disease Detection using CNN**" carried out under the guidance of **Dr.D.Vasumathi,** is submitted in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**. This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced/ copied from any source and have not been submitted to any other University or Institute for the award of any other degree or diploma.

**Swetha Annavarapu - (16011A0537)**

**Tailor Afreen Sultana - (16011A0538)**

**Thara Chand Chowdary - (16011A0539)**

**Sachi Pandita - (16011A0559)**

**Date: 11-06-2020**

**Department of Computer Science and Engineering**

**JNTUH College of Engineering, Hyderabad**

**Kukatpally, Hyderabad - 500 085**

## CERTIFICATE BY THE SUPERVISOR

This is to certify that the project report entitled "**Plant Disease Detection using CNN**", being submitted by **Swetha Annavarapu (16011A0537), Tailor Afreen Sultana (16011A0538), Thara Chand Chowdary (16011A0539), Sachi Pandita (16011A0559)** in the **Department of Computer Science and Engineering of JNTUH COLLEGE OF ENGINEERING HYDERABAD** is a record of bonafide work carried out by them under my guidance and supervision. The results are verified and found to be satisfactory.

**Dr.D.Vasumathi,**

**Professor,**

**Dept of CSE, JNTUH College of Engineering,**

**Hyderabad.**

**Date: 11-06-2020**

**Department of Computer Science and Engineering**

**JNTUH College of Engineering, Hyderabad**

**Kukatpally, Hyderabad - 500 085**

## CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that the Major Project report entitled "**Plant Disease Detection using CNN**", being submitted by **Swetha Annavarapu (16011A0537), Tailor Afreen Sultana (16011A0538), Thara Chand Chowdary (16011A0539), Sachi Pandita (16011A0559)**, in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**.

**Dr.M.Chandra Mohan,**

**Professor & Head of the Department,**

**Department of Computer Science & Engineering,**

**JNTUH College of Engineering,**

**Hyderabad.**

**Date: 11-06-2020**

# Acknowledgement

We would like to express sincere thanks to our Supervisor **Dr.D.Vasumathi**, Professor, for her admirable guidance and inspiration both theoretically and practically and most importantly for the drive to complete project successfully. Working under such an eminent guide was our privilege.

We owe a debt of gratitude to **Dr.M.Chandra Mohan**, Professor & Head of the department of Computer Science & Engineering and **Dr.B N Bhandari** for their unending cooperation.

We are grateful to the **Project Review Committee members** and Department of Computer Science & Engineering who have helped in successfully completing this project by giving their valuable suggestions and support.

We express thanks to **our parents** for their love, care and moral support without which we would have not been able to complete this project. It has been a constant source of inspiration for all our academic endeavours. Last but not the least, we thank the **Almighty** for making us a part of the world.

**Swetha Annavarapu - (16011A0537)**

**Tailor Afreen Sultana - (16011A0538)**

**Thara Chand Chowdary - (16011A0539)**

**Sachi Pandita - (16011A0559)**

# DECLARATION

We declare that this written submission represents our ideas in our own words and other ideas or words have been included we have adequately cited and referenced the original sources. We also declare that we have adhered to all the principles of the academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/data/fact/sources in our submission. We understand that any violation of the above will be cause for disciplinary action by the institute and also can evoke penal action from the sources which we have thus not been properly cited or from whom proper permission has not been taken when needed.

**Swetha Annavarapu - (16011A0537)**

**Tailor Afreen Sultana - (16011A0538)**

**Thara Chand Chowdary - (16011A0539)**

**Sachi Pandita - (16011A0559)**

# ABSTRACT

The primary occupation in India is agriculture. India ranks second in the agricultural output worldwide. Here in India, farmers cultivate a great diversity of crops. Various factors such as climatic conditions, soil conditions, various disease, etc affect the production of the crops. The existing method for plants disease detection is simply naked eye observation which requires more man labour, properly equipped laboratories, expensive devices, etc. And improper disease detection may lead to inexperienced pesticide usage that can cause development of long-term resistance of the pathogens, reducing the ability of the crop to fight back. The plant disease detection can be done by observing the spot on the leaves of the affected plant. The method we are adopting to detect plant diseases is image processing using Convolution neural network (CNN).

Thus, accurate disease prediction is one of the greatest challenges, despite many advances in agriculture department in recent decades. Plants means “Food” and Food means “Life”. Plant Disease Detection is closely related to agriculture sector, which contributes significantly to the economy of the nation. The proposed system helps in identification of plant disease and provides remedies.

It can be used as a defence mechanism against the disease. The database obtained from the Internet is properly segregated and the different plant species are identified and are renamed to form a proper database then obtain test-database which consists of various plant diseases that are used for checking the accuracy and confidence level of the project .Then using training data we will train our classifier and then output will be predicted with optimum accuracy . We use Convolution Neural Network (CNN) which comprises of different layers which are used for prediction. With our code and training model we have achieved an accuracy level of 97%. Our software gives us the name of the plant species with its confidence level.

The data for experimental purpose is collected from Kaggle.

# Table of Contents

# CHAPTER 1
# INTRODUCTION

The problem of efficient plant disease protection is closely related to the problems of sustainable agriculture and climate change. Research results indicate that climate change can alter stages and rates of pathogen development; it can also modify host resistance, which leads to physiological changes of host - pathogen interactions. The situation is further complicated by the fact that, today, diseases are transferred globally more easily than ever before. New diseases can occur in places where they were previously unidentified and, inherently, where there is no local expertise to combat them.

Inexperienced pesticide usage can cause the development of long-term resistance of the pathogens, severely reducing ability to fight back. Timely and accurate diagnosis of plant diseases are one of the pillars of precision agriculture. It is crucial to prevent unnecessary waste of financial and other resources, thus achieving healthier production, by addressing the long-term pathogen resistance development problem and mitigating the negative effect of climate change.

In this changing environment, appropriate and timely disease identification including early Prevention has never been more important. There are several ways to detect plant pathologies. Some diseases do not have any visible symptoms, or the effect becomes noticeable too late to act, and in those situations, a sophisticated analysis is obligatory.

However, most diseases generate some kind of manifestation in the visible spectrum, so the naked eye examination of a trained professional is the prime technique adopted in practice for plant disease detection. In order to achieve accurate plant disease diagnostics a plant pathologist should possess good observation skills so that one can identify characteristic symptoms. Variations in symptoms indicated by diseased plants may lead to an improper diagnosis since amateur gardeners and hobbyists could have more difficulties determining it than a professional plant pathologist. An automated system designed to help identify plant diseases by the plant's appearance and visual symptoms could be of great help to amateurs in the gardening process and also trained professionals as a verification system in disease diagnostics.

Advances in computer vision present an opportunity to expand and enhance the practice of precise plant protection and extend the market of computer vision applications in the field of precision agriculture.

Exploiting common digital image processing techniques such as colour analysis and thresholding were used with the aim of detection and classification of plant diseases.

Various different approaches are currently used for detecting plant diseases and most common are artificial neural networks (ANNs) and Support Vector Machines (SVMs). They are combined with different methods of image pre-processing in favour of better feature extraction.

In machine learning and cognitive science, ANN is an information-processing paradigm that was inspired by the way biological nervous systems, such as the brain, process information. The brain is composed of a large number of highly interconnected neurons working together to solve specific problems.

An artificial neuron is a processing element with many inputs and one output. Although artificial neurons can have many outputs, only those with exactly one output will be considered. Their inputs can also take on any value between 0 and 1. Also, the neuron has weights for each input and an overall bias.

The weights are real numbers expressing importance of the respective inputs to the output. The bias is used for controlling how easy the neuron is getting to output 1. For a neuron with really big bias it is easy to output 1, but when the bias is very negative then it is difficult to output 1.

## 1. Objective:

We can reduce the attack of pests by using proper pesticides and remedies. We can reduce the size of the images by proper size reduction techniques and see to it that the quality is not compromised to a great extent. We can expand the projects of the earlier mentioned authors such that the remedy to the disease is also shown by the system.

The main objective is to identify the plant diseases using image processing. It also, after identification of the disease, suggest the name of pesticide to be used. It also identifies the insects and pests responsible for epidemic. Apart from these parallel objectives, this drone is very time saving. The budget of the model is quite high for low scale farming purposes but will be value for money in large scale farming. It completes each of the process sequentially and hence achieving each of the output. Thus, the main objectives are:

1) To design such system that can detect crop disease and pest accurately.

2) Create database of insecticides for respective pest and disease.

3) To provide remedy for the disease that is detected.

## 2. Scope and Limitations:

Through this project work, we try to detect the plant disease using the pre-determined dataset containing the images of a variety of plant leaves. The model constructed in this project can show an accuracy up to 93% in detecting the disease.

## 3. Organization:

In chapter 1, we introduce to the reader the basic introduction about the Plant Disease Detection System to get them started, the object and the scope and limitations of the project are also covered here.

In chapter 2, we discuss the different research conducted by different people earlier and the points they mentioned.

In chapter 3, the problem is stated i.e. the purpose of using Convolutional neural networks in detecting plant diseases.

In Chapter 4, the implementation of the model is shown under this section, the overview of the model and the layers used in the neural network are mentioned here.

In Chapter 5, the results of the experiments i.e. the detection of the plant disease with the test dataset, along with the glimpse of the dataset used is illustrated.

In Chapter 6, the conclusion and the directions for conducting future work on this project are listed.

In Chapter 7, the various sources like the IEEE journals, the articles on the web and other references have been listed.

# CHAPTER 2

# LITERATURE SURVEY

**Authors: - Sachin D. Khirade.**
**Paper: - A survey on Plant Disease Detection using Convolutional neural network. International Journal of Computer Applications, 72(16).**

- Identification of the plant diseases is the key to preventing the losses in the yield and quantity of the agricultural product.

- It requires tremendous amount of work, expertise in the plant diseases, and also require the excessive processing time.

- Hence, image processing is used for the detection of plant diseases. Disease detection involves the steps like image acquisition, image pre-processing, image segmentation, feature extraction and classification.

- This paper discussed the methods used for the detection of plant diseases using their leaves images. This paper discussed various techniques to segment the disease part of the plant.

- This paper also discussed some Feature extraction and classification techniques to extract the features of infected leaf and the classification of plant diseases.

- The accurately detection and classification of the plant disease is very important for the successful cultivation of crop and this can be done using image processing.

- This paper discussed various techniques to segment the disease part of the plant. This paper also discussed some Feature extraction and classification techniques to extract the features of infected leaf and the classification of plant diseases.

- The use of ANN methods for classification of disease in plants such as self- organizing feature map, back propagation algorithm, SVMs etc. can be efficiently used. From these methods, we can accurately identify and classify various plant diseases using image processing technique.

**Authors:- Amandeep Singh.**

- The most significant challenge faced during the work was capturing the quality images with maximum detail of the leaf colour.

- It is very typical task to get the image with all the details within a processable memory.

- Such images are formed a through high resolution and thus are of 6-10MB of size. This was handled by using a Nikon made D5200 camera which served the task very well.

- Second challenge faced was to get rid of illumination conditions as from the start to the end of paddy crop season, illumination varies a lot even when the image acquiring time is fixed.

- However, the solution to this is variable user defined thresholding and making necessary adjustments to the shades of LCC.

**Authors: - Satish Madhgoria.**

- Proposed automatic pixel-based classification method for detecting unhealthy regions in leaf images is presented.

- The algorithms have been tested extensively. Linear SVM has been used to classify each pixel. We have also shown hoe the results from SVM could be improved remarkably using the neighbourhood check technique.

- The presented algorithm could well extend for other detection tasks which also mainly rely on colour information, but extension to other features is easily possible.

- The task is performed in three steps. First, we perform segmentation to divide the image into foreground and background. In the second step, support vector machines are applied to predict the class of each pixel belonging to the foreground.

- And finally, we do further refinement by neighbourhood-check to omit all falsely-classified pixels from second step. Satish Madhgoria, MarekSchikora & et at. Proposed automatic pixel-based classification method for detecting unhealthy regions in leaf images is presented

.

- The algorithms have been tested extensively. Linear SVM has been used to classify each pixel. We have also shown hoe the results from SVM could be improved remarkably using the neighbourhood check technique.

- The presented algorithm could well extend for other detection tasks which also mainly rely on colour information, but extension to other features is easily possible. The task is performed in three steps.

- First, we perform segmentation to divide the image into foreground and background. In the second step, support vector machines are applied to predict the class of each pixel belonging to the foreground. And finally, we do further refinement by neighbourhood-check to omit all falsely-classified pixels from second step.

# CHAPTER 3

# PROBLEM STATEMENT

Modern technologies have given human society the ability to produce enough food to meet the demand of more than 7 billion people. However, food security remains threatened by a number of factors including climate change (Tai et al., 2014), the decline in pollinators (Report of the Plenary of the Intergovernmental Science-Policy Platform on Biodiversity Ecosystem and Services on the work of its fourth session, 2016), plant diseases (Strange and Scott, 2005), and others. Plant diseases are not only a threat to food security at the global scale, but can also have disastrous consequences for smallholder farmers whose livelihoods depend on healthy crops. In the developing world, more than 80 percent of the agricultural production is generated by smallholder farmers (UNEP, 2013), and reports of yield loss of more than 50% due to pests and diseases are common (Harvey et al., 2014). Furthermore, the largest fraction of hungry people (50%) live in smallholder farming households (Sanchez and Swaminathan, 2005), making smallholder farmers a group that's particularly vulnerable to pathogen-derived disruptions in food supply.

Various efforts have been developed to prevent crop loss due to diseases. Historical approaches of widespread application of pesticides have in the past decade increasingly been supplemented by integrated pest management (IPM) approaches (Ehler, 2006). Independent of the approach, identifying a disease correctly when it first appears is a crucial step for efficient disease management. Historically, disease identification has been supported by agricultural extension organizations or other institutions, such as local plant clinics. In more recent times, such efforts have additionally been supported by providing information for disease diagnosis online, leveraging the increasing Internet penetration worldwide. Even more recently, tools based on mobile phones have proliferated, taking advantage of the historically unparalleled rapid uptake of mobile phone technology in all parts of the world (ITU, 2015).

Smartphones in particular offer very novel approaches to help identify diseases because of

their computing power, high-resolution displays, and extensive built-in sets of accessories, such as advanced HD cameras. It is widely estimated that there will be between 5 and 6 billion smartphones

on the globe by 2020. At the end of 2015, already 69% of the world's population had access to mobile broadband coverage, and mobile broadband penetration reached 47% in 2015, a 12-fold increase since 2007 (ITU, 2015).The combined factors of widespread smartphone penetration, HD cameras, and high performance processors in mobile devices lead to a situation where disease diagnosis based on automated image recognition, if technically feasible, can be made available at an unprecedented scale.Here, we demonstrate the technical feasibility using a deep learning approach utilizing 54,306 images of 30 crop species with 50 diseases (or healthy) made openly available

# CHAPTER 4

# PROPOSED WORK

Implementing the appropriate management strategies like fungicide applications, disease- specific chemical applications, and vector control through pesticide applications could lead to early information on crop health and disease detection. This could facilitate the control of diseases and improve productivity. Authors present, review, and recognize the demand for developing a rapid, cost-effective, and reliable health-monitoring sensor that facilitates advancements in agriculture. They described the currently used technologies that include spectroscopic and imaging-based and volatile profiling-based plant disease detection methods for the purpose of developing ground-based sensor system to assist in monitoring health and diseases in plants under field conditions.

After analysis of the work, it was decided to use image processing disease recognition approach among other approaches commonly used for plant disease diagnostics, for instance, double-stranded ribonucleic acid (RNA) analysis, nucleic acid probes, and microscopy.

Numerous procedures are currently in use for plant disease detection applying computer vision. One of them is disease detection by extracting colour feature. In this report YcbCr, HSI, and CIELB colour models were used in the study; as a result, disease spots were successfully detected and remained unaffected by the noise from different sources, such as camera flash.

In addition, plant disease detection could be achieved by extracting shape features method. Combination of all these features provides a robust feature set for image improvement and better classification. There are some approaches which apply the feed-forward back propagation of neural networks consisting of one input, one output, and one hidden layer for the needs of identifying the species of leaf, pest, or disease

Another technique proposed is incorporates the features extracted by Particle Swarm Optimization (PSO) and forward neural network in direction of determining the injured leaf spot of cotton and improving the accuracy of the system with the final overall accuracy of 95%.

Also, detection and differentiation of plant diseases can be achieved using Support Vector Machine algorithms. This technique was implemented for sugar beet diseases and presented, where, depending on the type and stage of disease, the classification accuracy was between 65% and 90%.

Likewise, there are methods that combine the feature extraction and Neural Network Ensemble (NNE) for plant disease recognition. Through training a definite number of neural networks and combining their results after that, NNE offers a better generalization of learning ability. Such method was implemented only for recognizing tea leaf diseases with final testing accuracy of 91%.

Another approach based on leaf images and using ANNs as a technique for an automatic detection and classification of plant diseases was used in conjunction with -means as a clustering procedure proposed by the authors. ANN consisted of 10 hidden layers. The number of outputs was 6 which was the number of classes representing five diseases along with the case of a healthy leaf. On average, the accuracy of classification using this approach was 94.67%.

They presented the deep learning methods for solving most complex tasks in different areas of research in biology, bioinformatics, biomedicine, robotics, and 3D technologies. In our study, we exploit the deep learning method for plant disease recognition, driven by evolvement of deep learning techniques and their application in practice. Extensive search of the state-of-the-art literature yielded no evidence that researchers explored deep learning approach for plant diseases recognition from the leaf images. Our method of recognition by applying deep CNN is presented in the sections below.

## PROCESS

The following displays the process:

**1.Exploratory data analysis**: Exploratory Data Analysis (EDA) is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to maximize insight into a data set; uncover underlying structure; detect outliers and anomalies; and also checked for the skewness of data.

**2.Building the model**: The output from the above feature selection, we first split the data into train and test data by using default function train-test split and checked for accuracy with respective algorithms.

**3.Feature Selection**: Feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction.

**4.Fine tuning**: Grid SearchCV is an approach to parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid.

**5.Convoluation neural network** –Here we constructed the neural networks with two different activation functions and tested for accuracy

**6.Integrating this with the UI to detect the disease** – The above functionality is being integrated with UI interface to detect the disease.

# CHAPTER 5

## DEFINITIONS

### NEURAL NETWORKS :

Neural networks are multi-layer networks of neurons (the blue and magenta nodes in the chart below) that we use to classify things, make predictions, etc. Below is the diagram of a simple neural network with five inputs, 5 outputs, and two hidden layers of neurons.

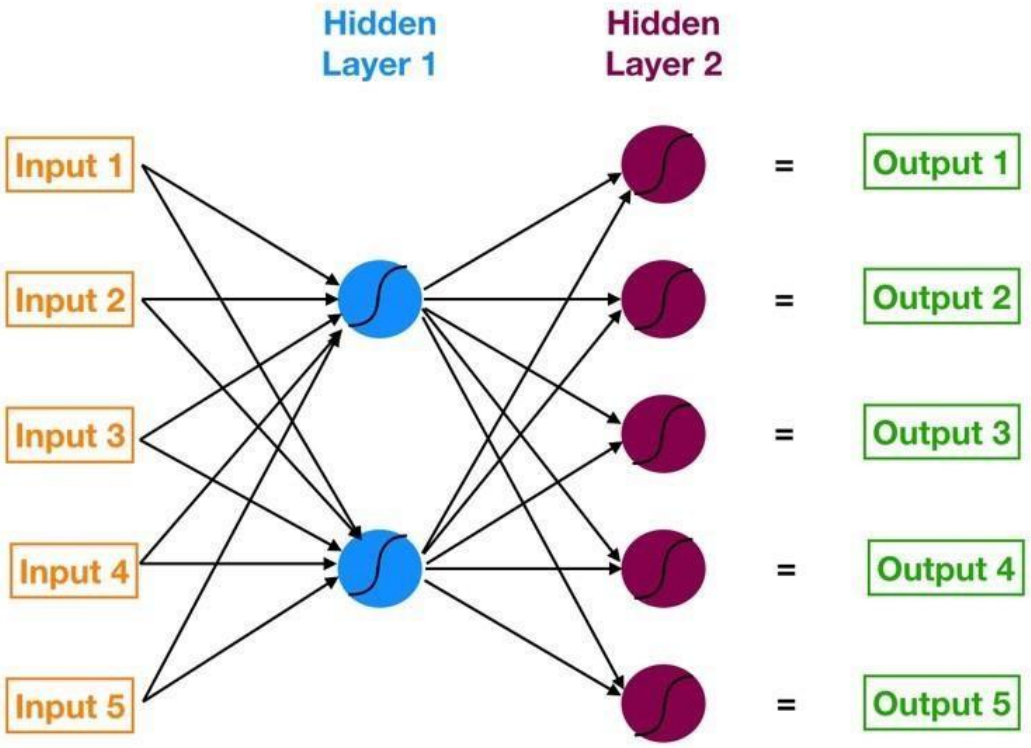



**Figure 2: The Layers in the Neural Network**

The arrows that connect the dots in the above figure shows how all the neurons are interconnected and how data travels from the input layer all the way through to the output layer.

**NEURONS:** An artificial neuron (also referred to as a perceptron) is a mathematical function. It takes one or more inputs that are multiplied by values called "weights" and added together. This value is then passed to a non-linear function, known as an activation function, to become the neuron's output.

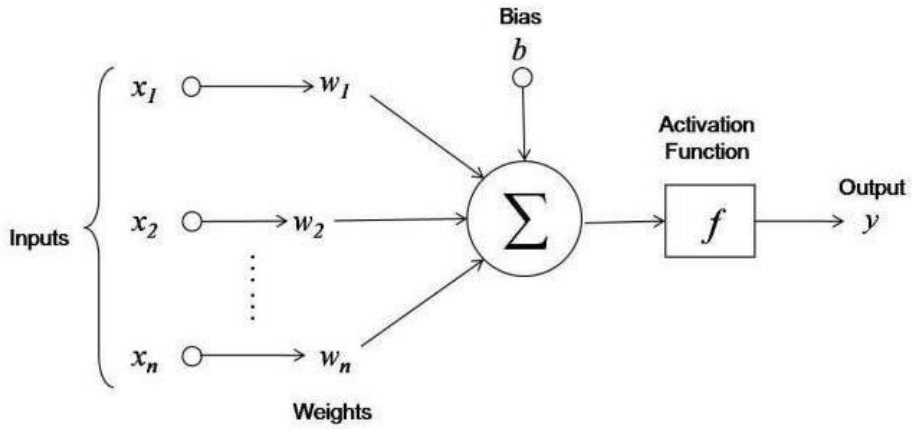


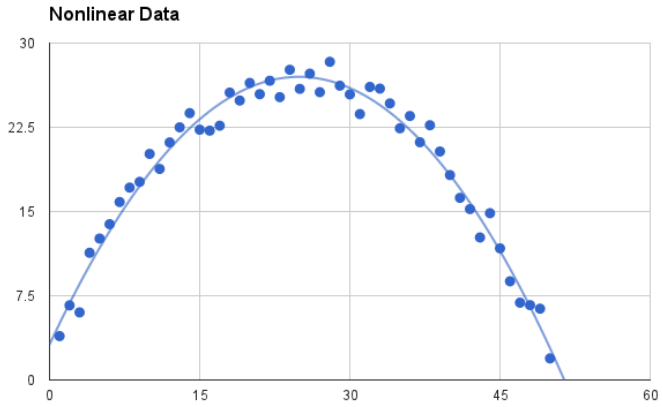

**Figure 3: The internal structure of neural network with bias and activation function**

- First, it adds up the value of every neurons from the previous column it is connected to. In the above figure, there are 3 inputs (x1, x2, x3) coming to the neuron, so 3 neurons of the previous column are connected to our neuron.
- This value is multiplied, before being added, by another variable called "weight" (w1, w2, w3) which determines the connection between the two neurons. Each connection of neurons has its own weight, and those are the only values that will be modified during the learning process.
- Moreover, a bias value may be added to the total value calculated. It is not a value coming from a specific neuron and is chosen before the learning phase, but can be useful for the network.
- After all those summations, the neuron finally applies a function called "activation function" to the obtained value.
- The so-called activation function usually serves to turn the total value calculated before to a number between 0 and 1 (done for example by a sigmoid function shown by Figure 3). Other function exist and may change the limits of our function, but keeps the same aim of limiting the value.

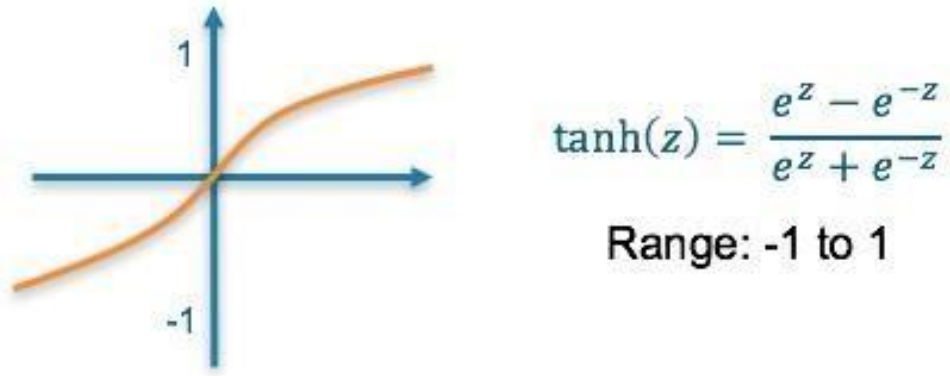
- That's all a neuron does! Take all values from connected neurons multiplied by their respective weight, add them, and apply an activation function. Then, the neuron is ready to send its new value to other neurons.

- After every neurons of a column did it, the neural network passes to the next column. In the end, the last values obtained should be one usable to determine the desired output.

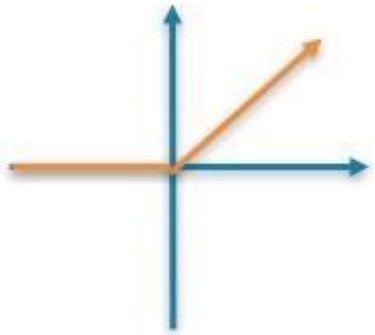
**ACTIVATION FUNCTIONS:** An activation function is a non-linear function applied by a neuron to introduce non-linear properties in the network. Here we use two nonlinear activation functions they are Hyperbolic Tangent Activation Function (Tanh) and Rectified Linear Unit (ReLU) Activation Function.




- **Hyperbolic Tangent Activation Function (Tanh):** The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped). The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph. The function is differentiable and monotonic while its derivative is not monotonic.



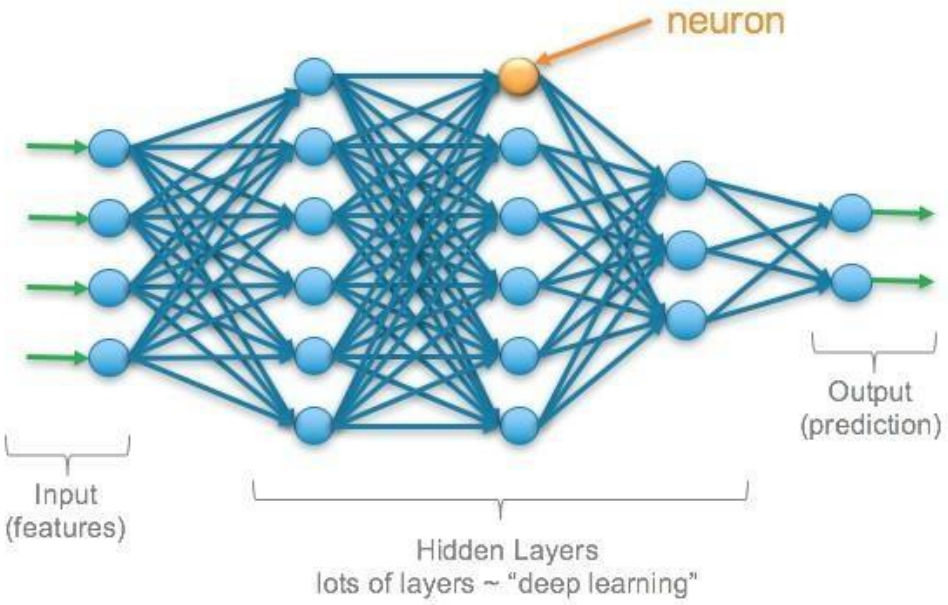

- **Rectified Linear Unit (ReLU) Activation Function :** The ReLU is the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning. The function and its derivative both are monotonic.



$$relu(z) = \max(0, z)$$

Range: 0 to infinity

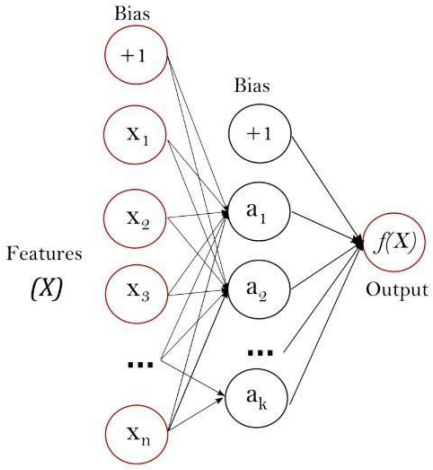
**MULTI-LAYER PERCEPTRON (MLP) :** A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to refer to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptron (with threshold activation).




An MLP must have at least three layers: the input layer, a hidden layer and the output layer. They are fully connected; each node in one layer connects with a weight to every node in the next layer. Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function $f(\cdot):R^m \rightarrow R^o$ by training on a dataset, where m is the number of dimensions for

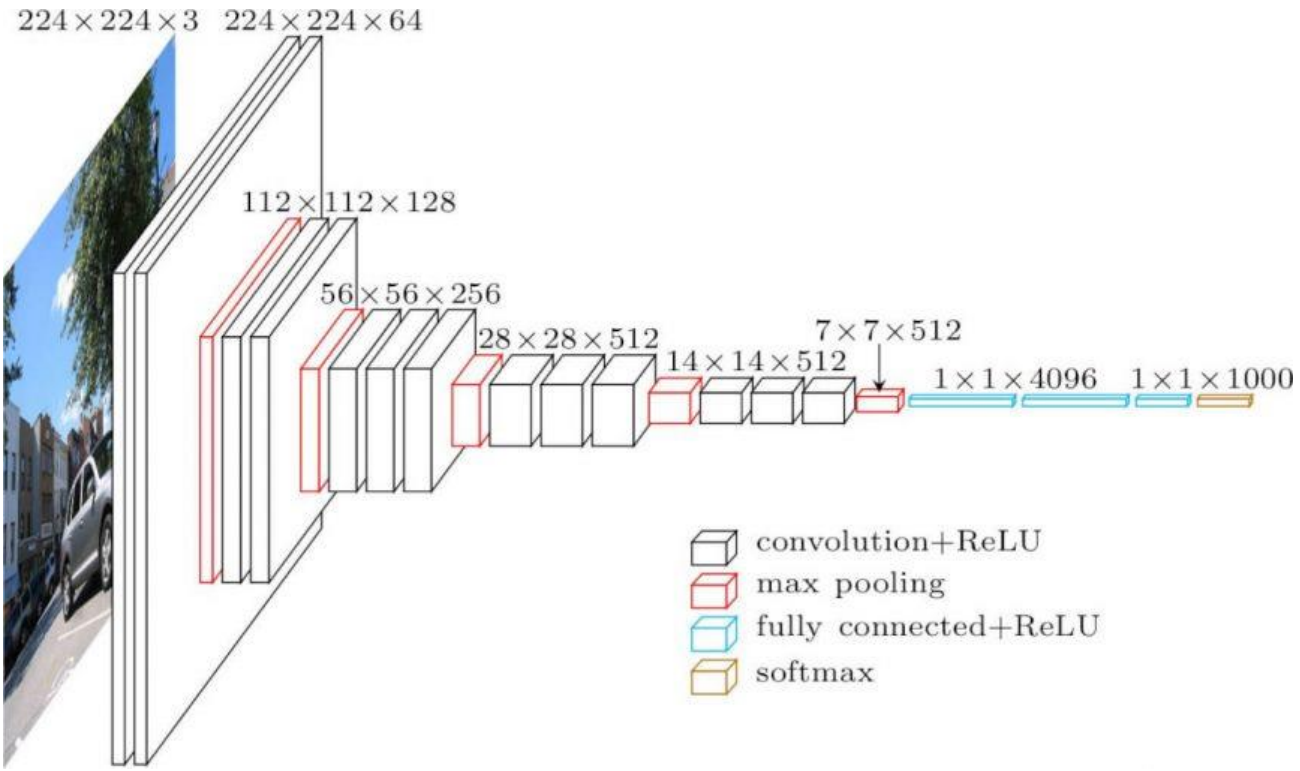
input and o is the number of dimensions for output. Given a set of features $X=x1,x2,...,xm$ and a target y, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or

more non-linear layers, called hidden layers. The figure below shows a one hidden layer MLP with scalar output.




The leftmost layer, known as the input layer, consists of a set of neurons {xi|x1,x2…xm} representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation w1x1+w2x2+…wmxm , followed by a non-linear activation function g(.): R− −> R - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values. The module contains the public attributes coefs and intercepts. Coefs is a list of weight matrices, where weight matrix at index represents the weights between layer i and layer i+1. intercepts_ is a list of bias vectors, where the vector at index i represents the bias values added to layer i+1.

**5.vgg16:** VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU’s.


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

## 1. PACKAGES :

- **NumPy:**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. In this project NumPy has been used to hold the image as a multidimensional – array to be passed as an input to the model designed.

- **Keras:**

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

- **Matplotlib:**

Matplotlib is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes.

- **Sklearn:**

Scikit-learn is an open source Python library that implements a range of machine learning, pre-processing, cross-validation and visualisation algorithms using a unified interface. The important features of scikit-learn are it is simple and efficient tool for data mining and analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, kmeans, etc.

- **Tkinter:**

It is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

- **Tensorflow:**

A free and opensource softwarelibrary for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as *tensors*. During the Google I/O conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

## 2. PLATFORMS:

- **Google colab:**

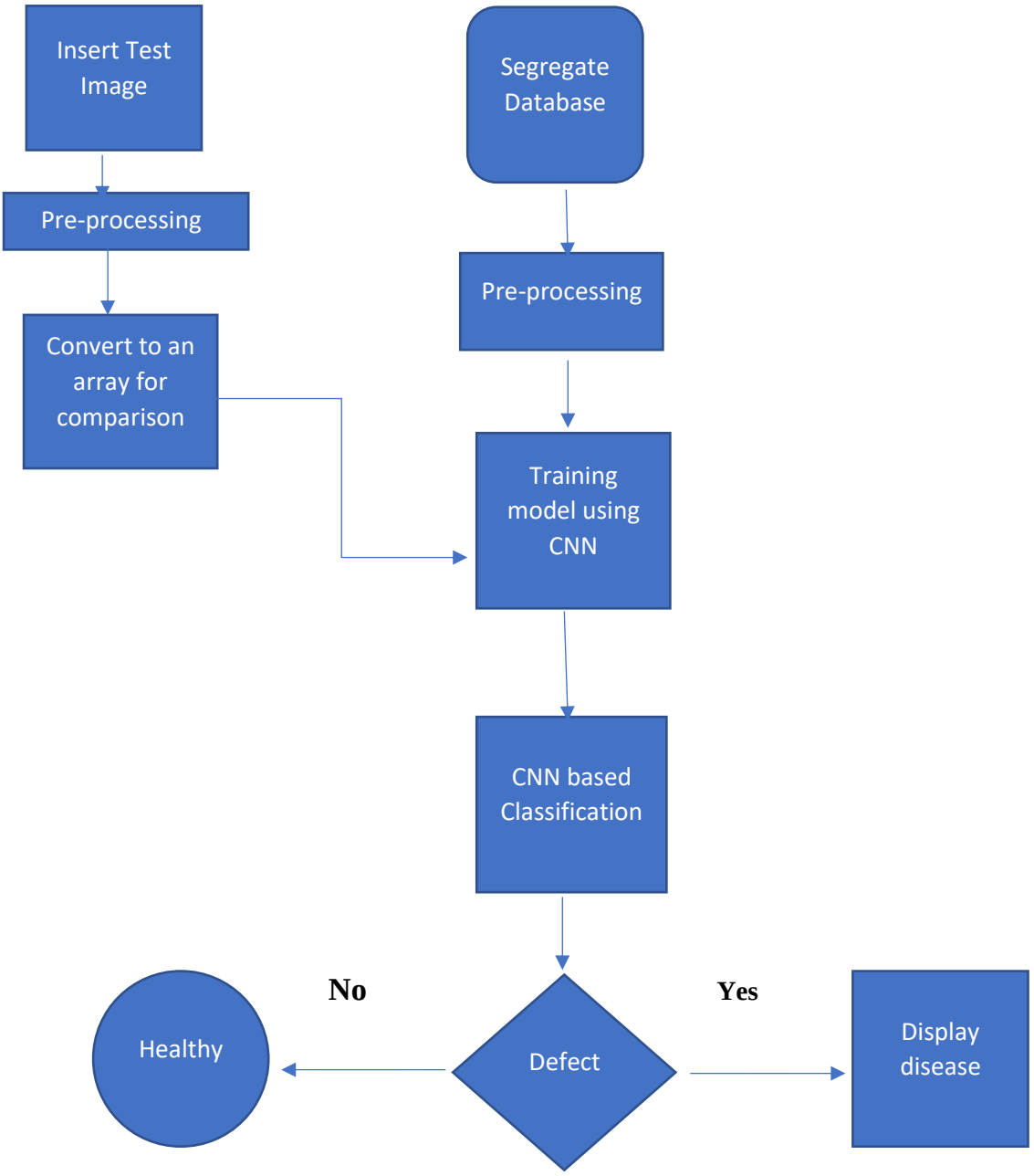
It is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs. Colab is able to provide free resources in part by having dynamic usage limits that sometimes fluctuate, and by not providing guaranteed or unlimited resources. This means that overall usage limits as well as idle timeout periods, maximum VM lifetime, GPU types available, and other factors vary over time. Colab does not publish these limits, in part because they can (and sometimes do) vary quickly.

GPUs and TPUs are sometimes prioritized for users who use Colab interactively rather than for long-running computations, or for users who have recently used less resources in Colab. As a result, users who use Colab for long-running computations, or users who have recently used more resources in Colab, are more likely to run into usage limits and have their access to GPUs and TPUs temporarily restricted. Users with high computational needs may be interested in using Colab's UI with a local runtime running on their own hardware. Users interested in having higher and more stable usage limits may be interested in Colab Pro.

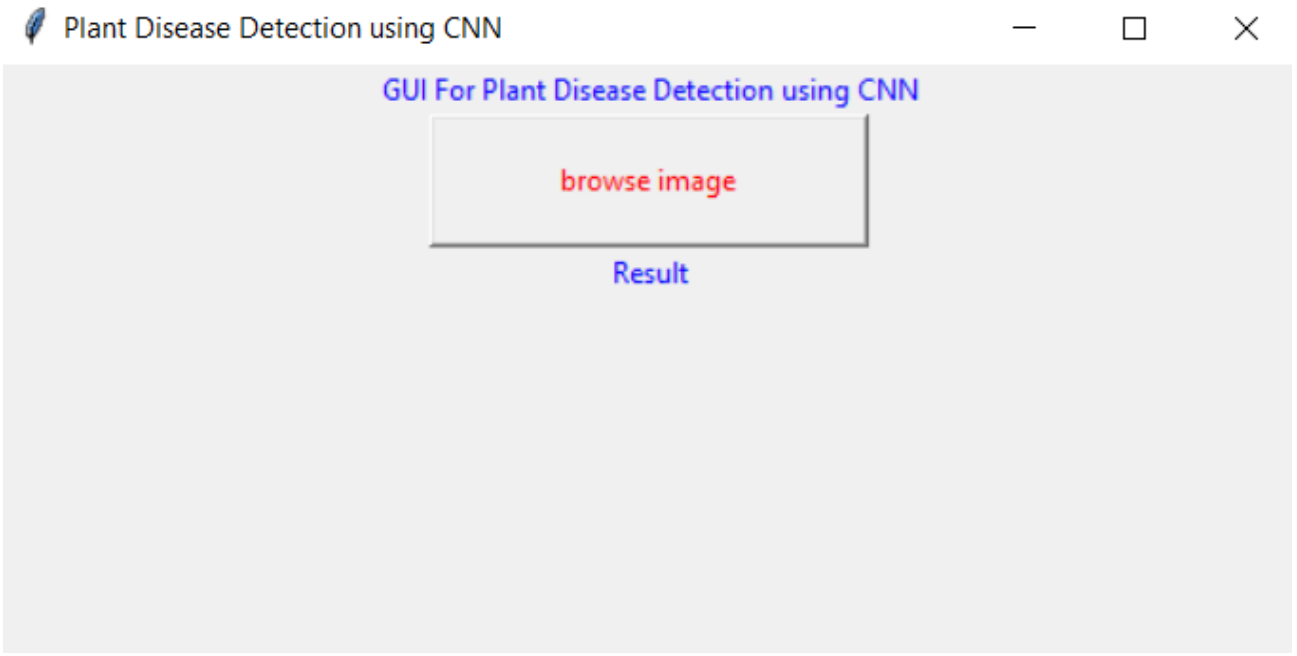
- **Jupyter notebook :**

The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook combines two components: web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output, notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

## 3. FLOW CHART:


Insert Test Image
Segregate Database
Pre-processing
Pre-processing
Convert to an array for comparison
Training model using CNN
CNN based Classification
No
Yes
Healthy
Defect
Display disease

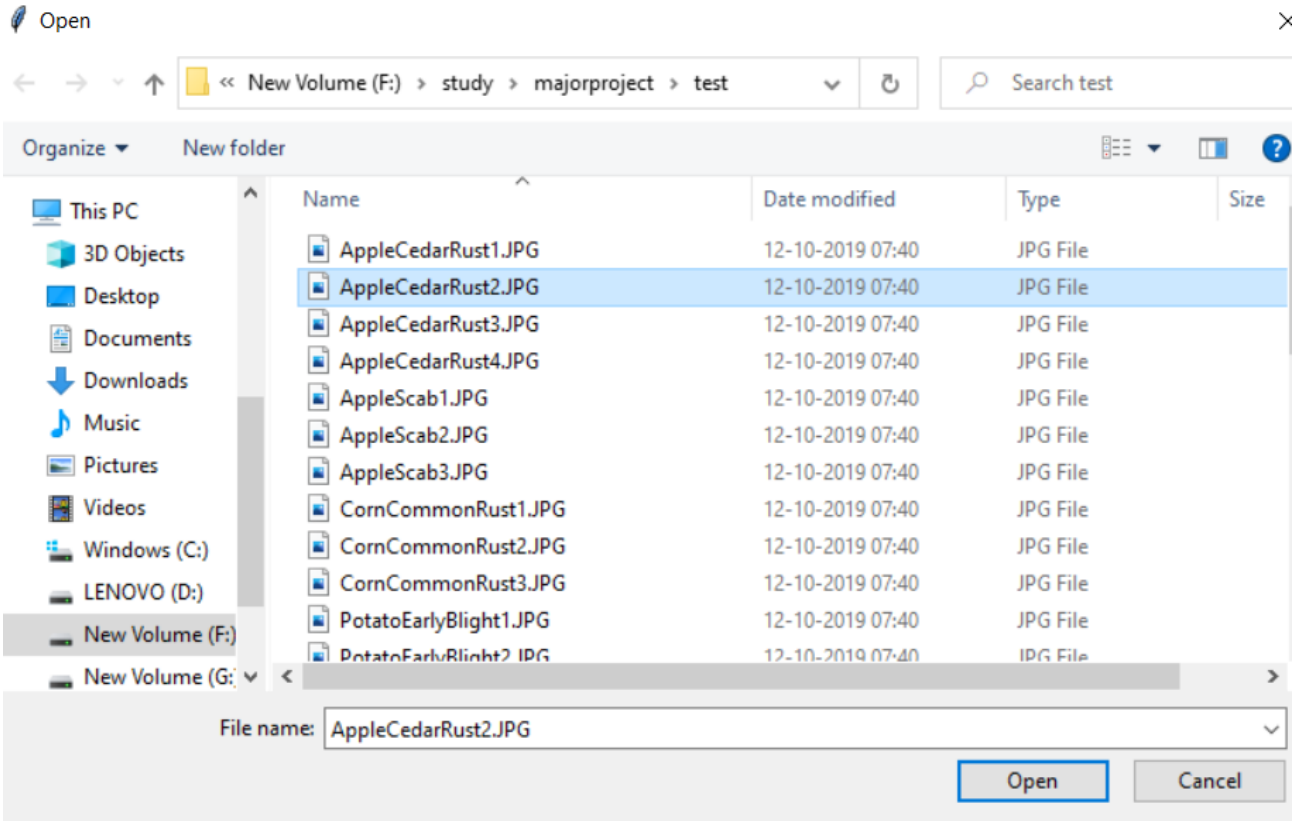
**4. RESULT :**

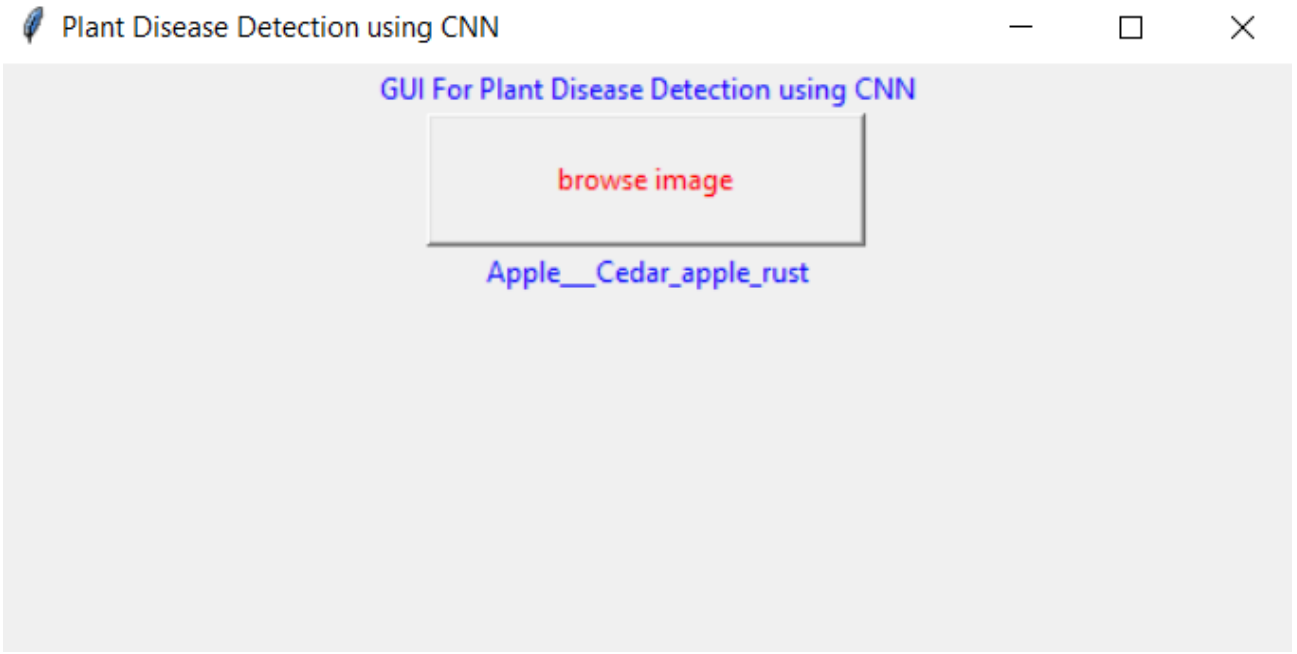



# UI

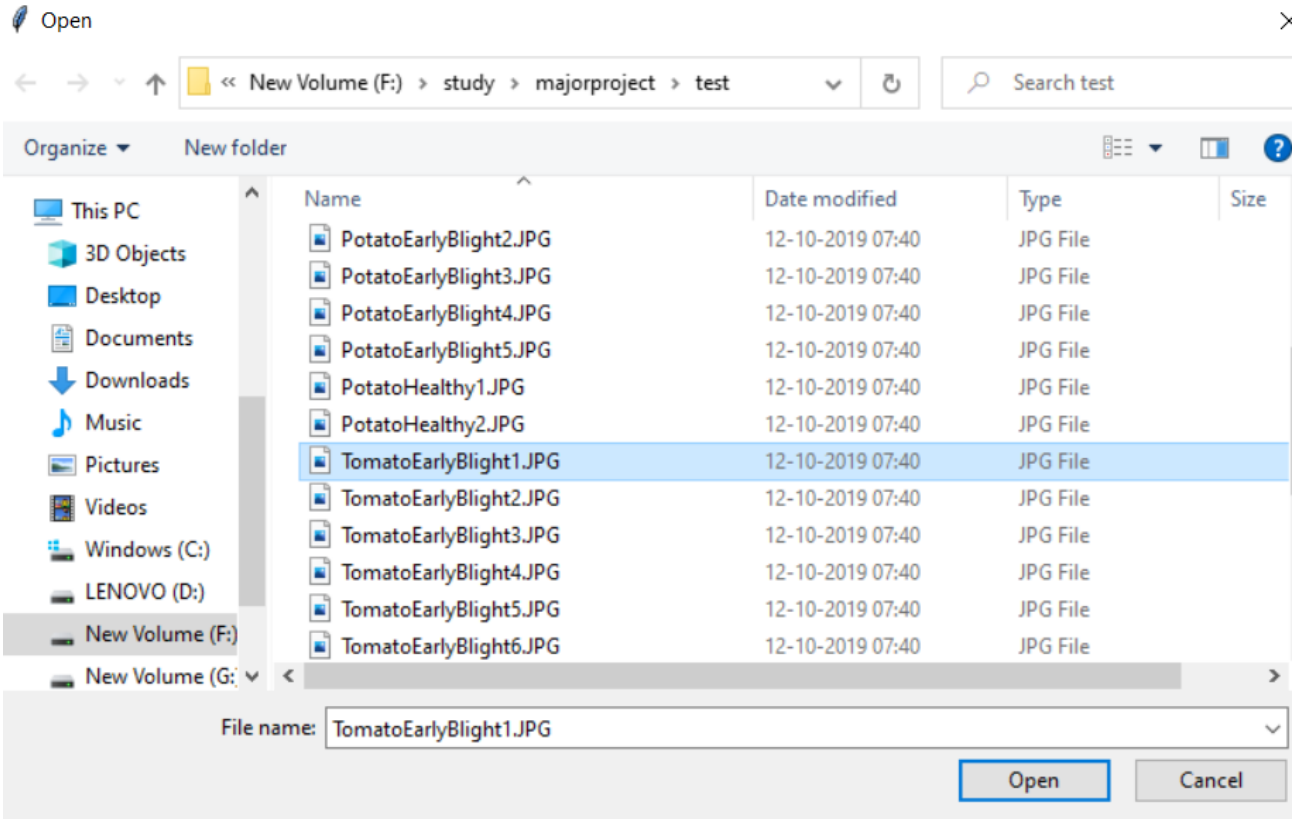
The pictures represent some examples we used to predict the diseases of the plant leaf images we uploaded.

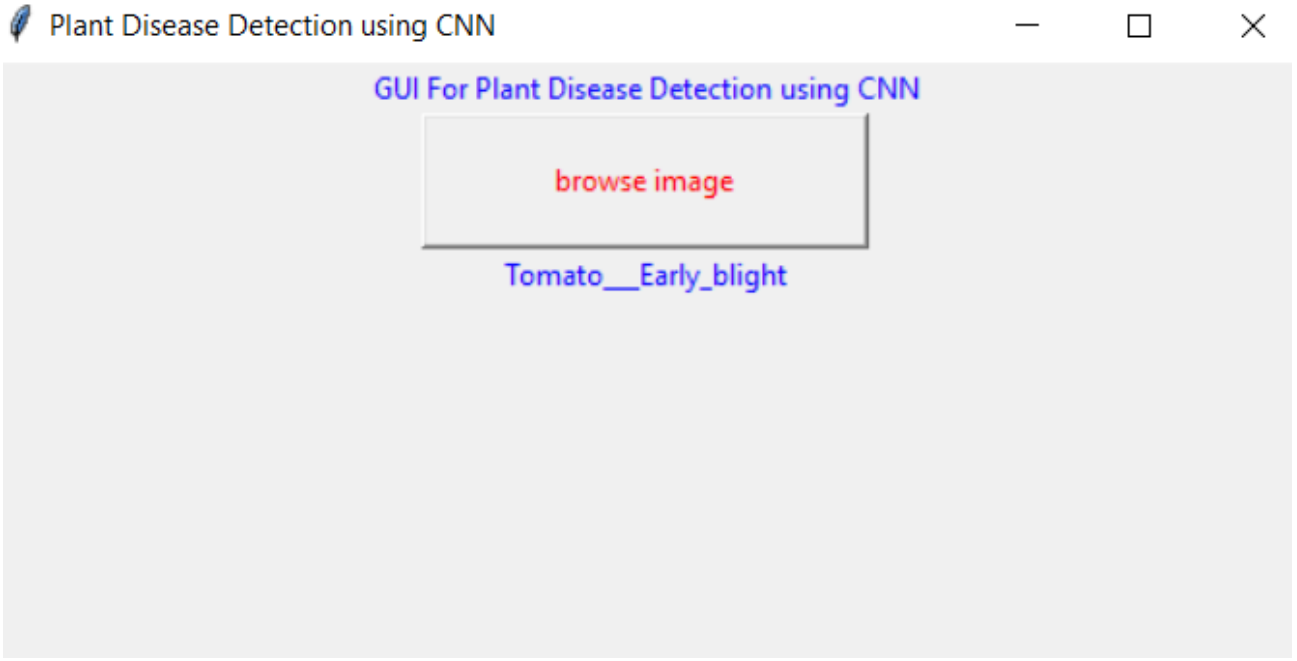
**Test Image 1)**


Open
New Volume (F:) › study › majorproject › test
Search test
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
LENOVO (D:)
New Volume (F:)
New Volume (G:
Name
Date modified
Type
Size
AppleCedarRust1.JPG 12-10-2019 07:40 JPG File
AppleCedarRust2.JPG 12-10-2019 07:40 JPG File
AppleCedarRust3.JPG 12-10-2019 07:40 JPG File
AppleCedarRust4.JPG 12-10-2019 07:40 JPG File
AppleScab1.JPG 12-10-2019 07:40 JPG File
AppleScab2.JPG 12-10-2019 07:40 JPG File
AppleScab3.JPG 12-10-2019 07:40 JPG File
CornCommonRust1.JPG 12-10-2019 07:40 JPG File
CornCommonRust2.JPG 12-10-2019 07:40 JPG File
CornCommonRust3.JPG 12-10-2019 07:40 JPG File
PotatoEarlyBlight1.JPG 12-10-2019 07:40 JPG File
File name: AppleCedarRust2.JPG
Open
Cancel

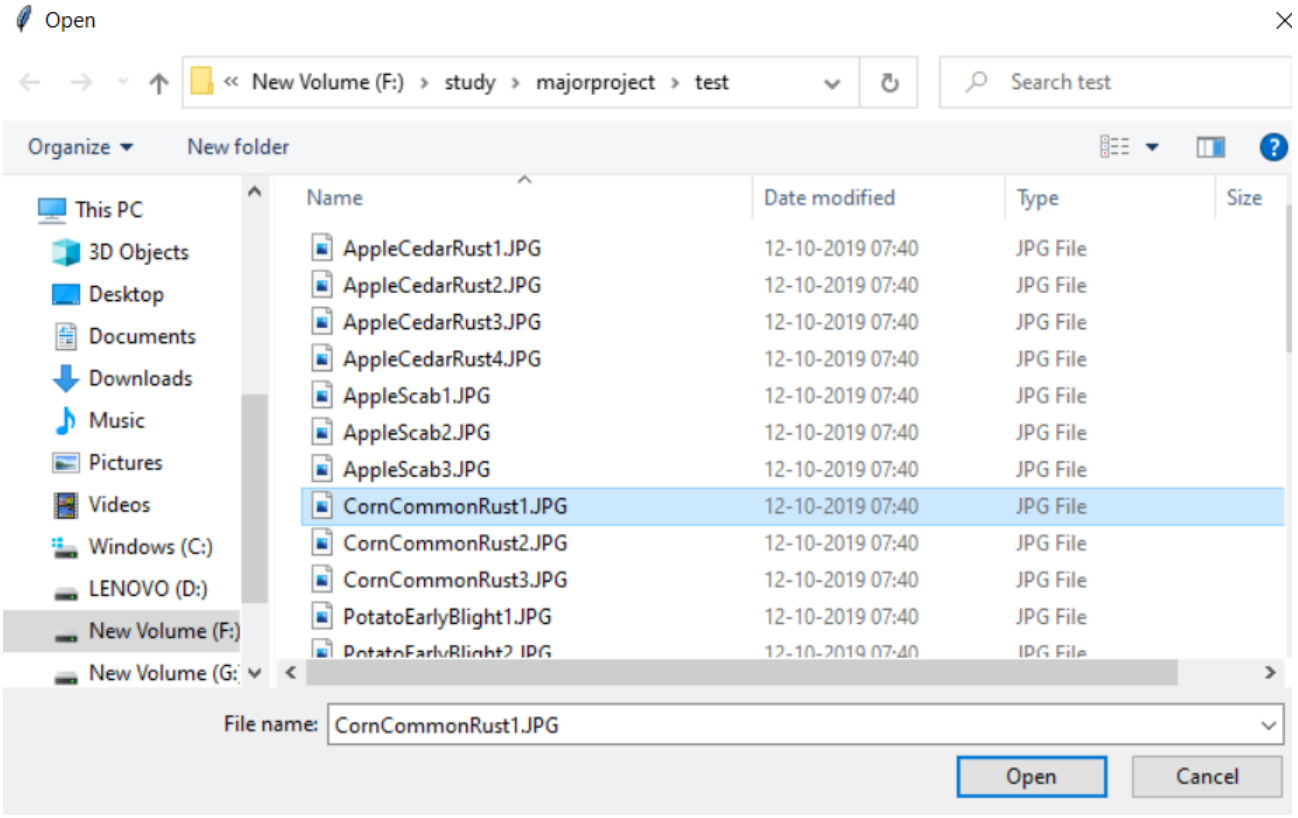

Plant Disease Detection using CNN
GUI For Plant Disease Detection using CNN
browse image
Apple___Cedar_apple_rust

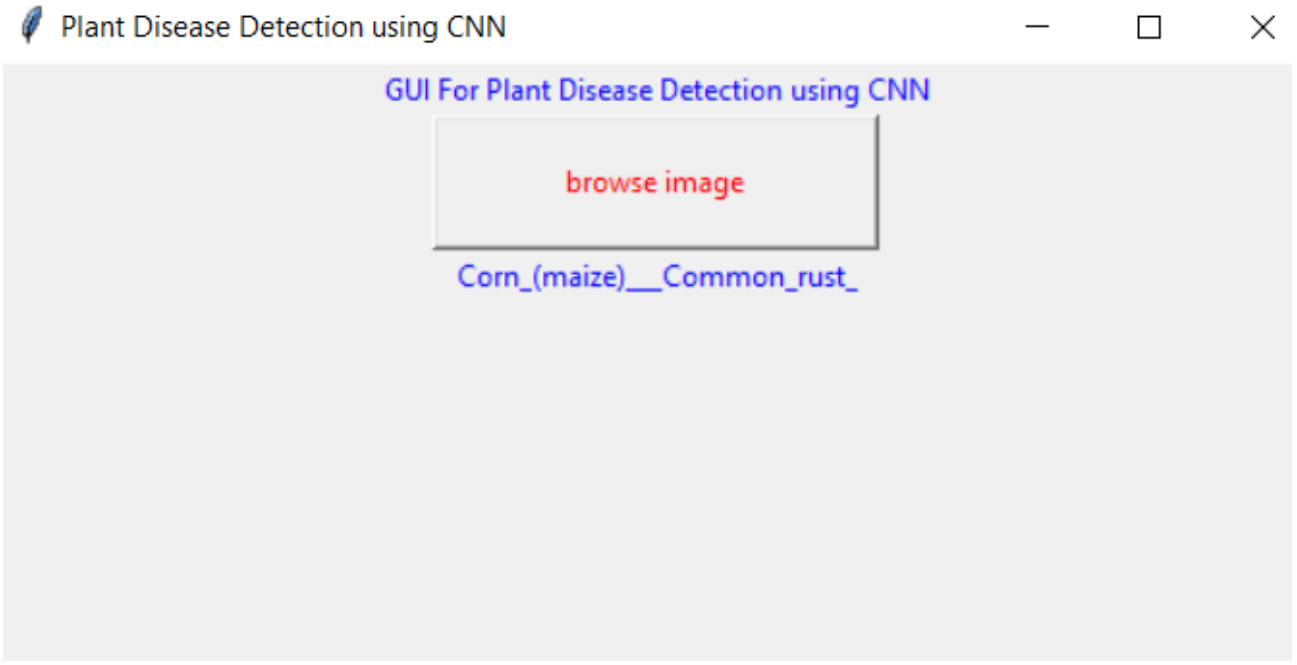
**Test Image 2)**


Open
« New Volume (F:) › study › majorproject › test
Search test
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
LENOVO (D:)
New Volume (F:)
New Volume (G:)
Name
Date modified
Type
Size
PotatoEarlyBlight2.JPG 12-10-2019 07:40 JPG File
PotatoEarlyBlight3.JPG 12-10-2019 07:40 JPG File
PotatoEarlyBlight4.JPG 12-10-2019 07:40 JPG File
PotatoEarlyBlight5.JPG 12-10-2019 07:40 JPG File
PotatoHealthy1.JPG 12-10-2019 07:40 JPG File
PotatoHealthy2.JPG 12-10-2019 07:40 JPG File
TomatoEarlyBlight1.JPG 12-10-2019 07:40 JPG File
TomatoEarlyBlight2.JPG 12-10-2019 07:40 JPG File
TomatoEarlyBlight3.JPG 12-10-2019 07:40 JPG File
TomatoEarlyBlight4.JPG 12-10-2019 07:40 JPG File
TomatoEarlyBlight5.JPG 12-10-2019 07:40 JPG File
TomatoEarlyBlight6.JPG 12-10-2019 07:40 JPG File
File name: TomatoEarlyBlight1.JPG
Open
Cancel


Plant Disease Detection using CNN
GUI For Plant Disease Detection using CNN
browse image
Tomato___Early_blight

**Test Image 3)**


Open
« New Volume (F:) › study › majorproject › test
Search test
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
LENOVO (D:)
New Volume (F:)
New Volume (G:
Name
Date modified
Type
Size
AppleCedarRust1.JPG 12-10-2019 07:40 JPG File
AppleCedarRust2.JPG 12-10-2019 07:40 JPG File
AppleCedarRust3.JPG 12-10-2019 07:40 JPG File
AppleCedarRust4.JPG 12-10-2019 07:40 JPG File
AppleScab1.JPG 12-10-2019 07:40 JPG File
AppleScab2.JPG 12-10-2019 07:40 JPG File
AppleScab3.JPG 12-10-2019 07:40 JPG File
CornCommonRust1.JPG 12-10-2019 07:40 JPG File
CornCommonRust2.JPG 12-10-2019 07:40 JPG File
CornCommonRust3.JPG 12-10-2019 07:40 JPG File
PotatoEarlyBlight1.JPG 12-10-2019 07:40 JPG File
File name: CornCommonRust1.JPG
Open
Cancel


Plant Disease Detection using CNN
GUI For Plant Disease Detection using CNN
browse image
Corn_(maize)___Common_rust_

# CHAPTER 6

# CODE

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

```
Using TensorFlow backend.
```

```
EPOCHS = 30
INIT_LR = 1e-3
BS = 32
default_image_size = tuple((256, 256))
image_size = 0
directory_root = '../input/'
width=256
height=256
depth=3
```

```
train_datagen = ImageDataGenerator(rescale = None,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
```

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

```
image_list, label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
```

```
image_list, label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)

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

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

            for image in plant_disease_image_list[:150]:
                image_directory = f"{directory_root}/{plant_folder}/{plant_
disease_folder}/{image}"
                if image_directory.endswith(".jpg") == True or image_direct
ory.endswith(".JPG") == True:
                    image_list.append(convert_image_to_array(image_director
y))
                    label_list.append(plant_disease_folder)
    print("[INFO] Image loading completed")
```

```
            print(f"[INFO] Processing {plant_disease_folder} ...")
            plant_disease_image_list = listdir(f"{directory_root}/{plant_f
lder}/{plant_disease_folder}/")

            for image in plant_disease_image_list[:150]:
                image_directory = f"{directory_root}/{plant_folder}/{plant
disease_folder}/{image}"
                if image_directory.endswith(".jpg") == True or image_direc
ory.endswith(".JPG") == True:
                    image_list.append(convert_image_to_array(image_directo
y))
                    label_list.append(plant_disease_folder)
    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

sorted(label_list)
```

```
[INFO] Loading images ...
[INFO] Processing Tomato___Target_Spot ...
[INFO] Processing Tomato___Early_blight ...
[INFO] Processing Apple___healthy ...
[INFO] Processing Tomato___healthy
```

```
[INFO] Loading images ...
[INFO] Processing Tomato___Target_Spot ...
[INFO] Processing Tomato___Early_blight ...
[INFO] Processing Apple___healthy ...
[INFO] Processing Tomato___healthy ...
[INFO] Processing Blueberry___healthy ...
[INFO] Processing Grape___healthy ...
[INFO] Processing Peach___healthy ...
[INFO] Processing Corn_(maize)___Common_rust_ ...
[INFO] Processing Cherry_(including_sour)___Powdery_mildew ...
[INFO] Processing Tomato___Leaf_Mold ...
[INFO] Processing Apple___Black_rot ...
[INFO] Processing Squash___Powdery_mildew ...
[INFO] Processing Corn_(maize)___Northern_Leaf_Blight ...
[INFO] Processing Strawberry___Leaf_scorch ...
[INFO] Processing Pepper,_bell___healthy ...
[INFO] Processing Orange___Haunglongbing_(Citrus_greening) ...
[INFO] Processing Potato___Late_blight ...
[INFO] Processing Tomato___Late_blight ...
[INFO] Processing Strawberry___healthy ...
[INFO] Processing Tomato___Tomato_Yellow_Leaf_Curl_Virus ...
[INFO] Processing Raspberry___healthy ...
[INFO] Processing Tomato___Tomato_mosaic_virus
```

```
['Apple___Apple_scab' 'Apple___Black_rot' 'Apple___Cedar_apple_rust'
 'Apple___healthy' 'Blueberry___healthy'
 'Cherry_(including_sour)___Powdery_mildew'
 'Cherry_(including_sour)___healthy'
 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot'
 'Corn_(maize)___Common_rust_' 'Corn_(maize)___Northern_Leaf_Blight'
 'Corn_(maize)___healthy' 'Grape___Black_rot'
 'Grape___Esca_(Black_Measles)'
 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)' 'Grape___healthy'
 'Orange___Haunglongbing_(Citrus_greening)' 'Peach___Bacterial_spot'
 'Peach___healthy' 'Pepper,_bell___Bacterial_spot'
 'Pepper,_bell___healthy' 'Potato___Early_blight' 'Potato___Late_blight'
 'Potato___healthy' 'Raspberry___healthy' 'Soybean___healthy'
 'Squash___Powdery_mildew' 'Strawberry___Leaf_scorch'
 'Strawberry___healthy' 'Tomato___Bacterial_spot' 'Tomato___Early_bligh
t'
 'Tomato___Late_blight' 'Tomato___Leaf_Mold' 'Tomato___Septoria_leaf_spo
t'
 'Tomato___Spider_mites Two-spotted_spider_mite' 'Tomato___Target_Spot'
 'Tomato___Tomato_Yellow_Leaf_Curl_Virus' 'Tomato___Tomato_mosaic_virus'
 'Tomato___healthy']
```

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

```
print("[INFO] Spliting data to train, test")

x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_l
bels, test_size=0.2, random_state = 42)
```

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

```
del image_list
del np_image_list
```

```
import gc
gc.collect()
```

```
142
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 256, 256, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 256, 256, 32)      0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256, 256, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 85, 85, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 85, 85, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 85, 85, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 85, 85, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 85, 85, 64)        256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 85, 85, 64)        36928
_________________________________________________________________
activation_3 (Activation)    (None, 85, 85, 64)        0
```

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

```
fmodel = Sequential()
inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (depth, height, width)
    chanDim = 1
fmodel.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
fmodel.add(Activation("relu"))
fmodel.add(BatchNormalization(axis=chanDim))
fmodel.add(MaxPooling2D(pool_size=(3, 3)))
fmodel.add(Dropout(0.25))
fmodel.add(Conv2D(64, (3, 3), padding="same"))
fmodel.add(Activation("relu"))
fmodel.add(BatchNormalization(axis=chanDim))
fmodel.add(Conv2D(64, (3, 3), padding="same"))
```

```
fmodel.add(Activation("relu"))
fmodel.add(BatchNormalization(axis=chanDim))
fmodel.add(MaxPooling2D(pool_size=(2, 2)))
fmodel.add(Dropout(0.25))
fmodel.add(Conv2D(128, (3, 3), padding="same"))
fmodel.add(Activation("relu"))
fmodel.add(BatchNormalization(axis=chanDim))
fmodel.add(Conv2D(128, (3, 3), padding="same"))
fmodel.add(Activation("relu"))
fmodel.add(BatchNormalization(axis=chanDim))
fmodel.add(MaxPooling2D(pool_size=(2, 2)))
fmodel.add(Dropout(0.25))
fmodel.add(Flatten())
fmodel.add(Dense(1024))
fmodel.add(Activation("relu"))
fmodel.add(BatchNormalization())
fmodel.add(Dropout(0.5))
fmodel.add(Dense(n_classes))
fmodel.add(Activation("softmax"))

fmodel.summary()
```

```
Model: "sequential_1"
```

```
batch_normalization_3 (Batch (None, 85, 85, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 42, 42, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 42, 42, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 42, 42, 128)       73856
_________________________________________________________________
activation_4 (Activation)    (None, 42, 42, 128)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 42, 42, 128)       512
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 42, 42, 128)       147584
_________________________________________________________________
activation_5 (Activation)    (None, 42, 42, 128)       0
_________________________________________________________________
batch_normalization_5 (Batch (None, 42, 42, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 21, 21, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 21, 21, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 56448)             0
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 256, 256, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 256, 256, 32)      0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256, 256, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 85, 85, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 85, 85, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 85, 85, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 85, 85, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 85, 85, 64)        256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 85, 85, 64)        36928
_________________________________________________________________
activation_3 (Activation)    (None, 85, 85, 64)        0
```

```
dense_1 (Dense)              (None, 1024)              57803776
_________________________________________________________________
activation_6 (Activation)    (None, 1024)              0
_________________________________________________________________
batch_normalization_6 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 38)                38950
_________________________________________________________________
activation_7 (Activation)    (None, 38)                0
=================================================================
Total params: 58,126,246
Trainable params: 58,123,366
Non-trainable params: 2,880
_________________________________________________________________
```

```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
# distribution
fmodel.compile(loss="categorical_crossentropy", optimizer=opt,metrics=["acc
uracy"])
# train the network
```

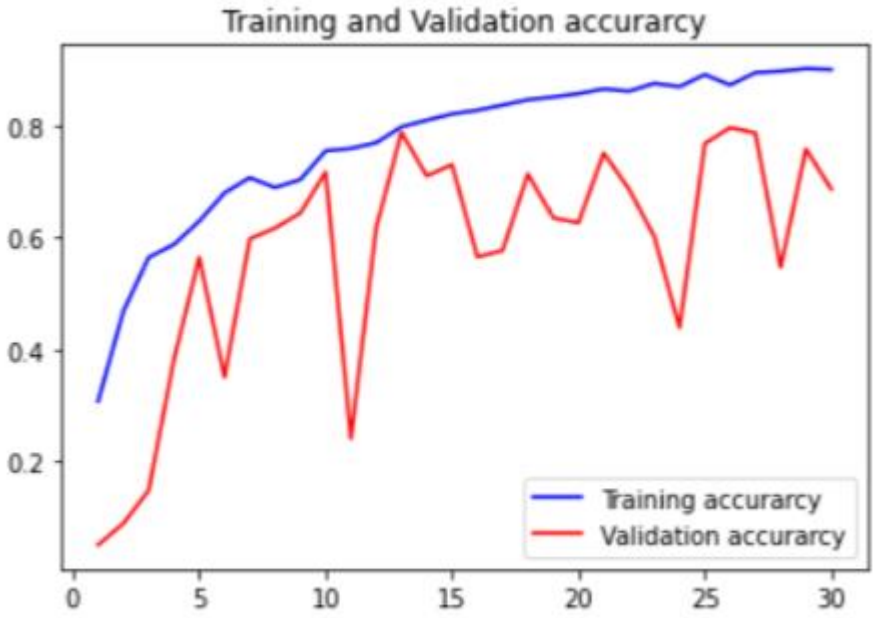
```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
# distribution
fmodel.compile(loss="categorical_crossentropy", optimizer=opt,metrics=["acc
uracy"])
# train the network
print("[INFO] training network...")
```

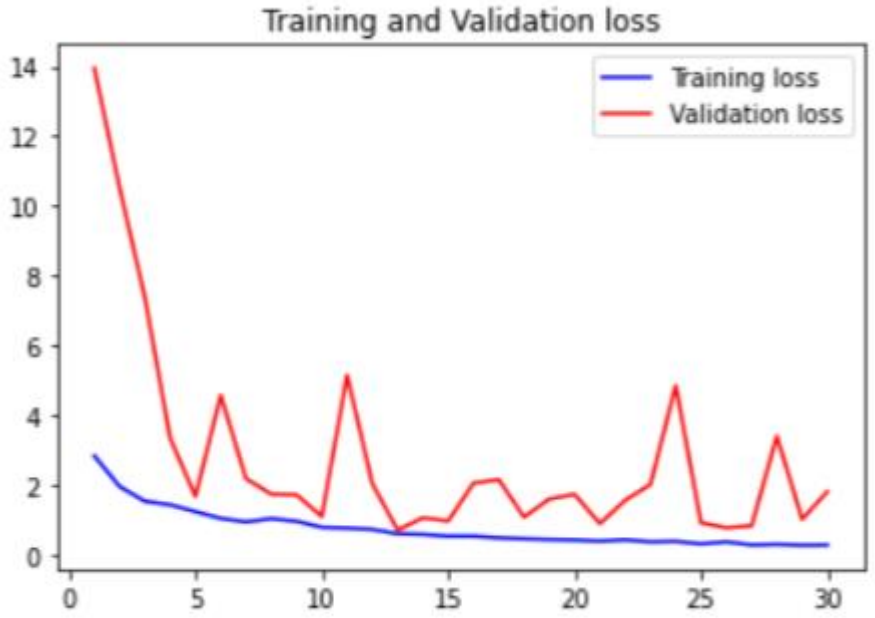
```
[INFO] training network...
```

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

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accurarcy')
plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
```


Training and Validation accurarcy
0.8
0.6
0.4
0.2
0
5
10
15
20
25
30
Training accurarcy
Validation accurarcy


Training and Validation loss
14
12
10
8
6
4
2
0
0
5
10
15
20
25
30
Training loss
Validation loss

```
print("[INFO] Calculating model accuracy")
scores = fmodel.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
```

```
[INFO] Calculating model accuracy
1140/1140 [==============================] - 2s 2ms/step
Test Accuracy: 68.77192854881287
```

```
model_json=fmodel.to_json()
with open("plantfmodel6.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
    fmodel.save_weights("plantfmodel6.h5")
    print("Saved model to disk")
```

```
Saved model to disk
```

# Vgg16 code

In [4]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
```

```
Using TensorFlow backend.
```

In [5]:
```
trdata = ImageDataGenerator(rescale = None,
                                    shear_range = 0.2,
                                    zoom_range = 0.2,
                                    horizontal_flip = True)
traindata = trdata.flow_from_directory(directory="/content/drive/My Drive/dataset/train",target_size=(224,224),batch_size = 32
                                           class_mode = 'categorical')
tsdata = ImageDataGenerator(rescale = 1./255)
testdata = tsdata.flow_from_directory(directory="/content/drive/My Drive/dataset/test", target_size=(224,224),batch_size = 32,
                                      class_mode = 'categorical')
```

```
Found 3990 images belonging to 38 classes.
Found 1330 images belonging to 38 classes.
```

In [6]:
```
from keras.layers import BatchNormalization,Dropout
model = Sequential()
model.add(Conv2D(input_shape=(224,224,3),filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(units=4096,activation="relu"))
model.add(Dropout(0.25))
model.add(Dense(units=4096,activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(units=38, activation="softmax"))
```

```
from keras.optimizers import Adam
#opt = Adam(lr=0.001)
model.compile(optimizer="adam", loss=keras.losses.categorical_crossentropy, metrics=['accuracy'])
```

```
hist = model.fit_generator(steps_per_epoch=375,generator=traindata, validation_data= testdata, validation_steps=125,epochs=25)
```

```
model_json=model.to_json()
with open("plantmodel6.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
    model.save_weights("plantmodel6.h5")
    print("Saved model to disk")
```

```
acc = hist.history['accuracy']
val_acc = hist.history['val_accuracy']
loss = hist.history['loss']
val_loss = hist.history['val_loss']
epochs = range(1, len(acc) + 1)
#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accurarcy')
plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
```

# Code for UI

```
import tkinter as tk
from tkinter import filedialog
import numpy as np
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import Dense
from keras.models import model_from_json
import os

import cv2
from tkinter import *
from PIL import ImageTk, Image
import _tkinter # with underscore, and lowercase 't'

win=tk.Tk()

def b1_click():
    global path2
    try:
        json_file = open('model2.json', 'r')
        loaded_model_json = json_file.read()
        json_file.close()
        loaded_model = model_from_json(loaded_model_json)
        # load weights into new model
        loaded_model.load_weights("model2.h5")
        print("Loaded model from disk")
        label=['Apple___Apple_scab', 'Apple___Black_rot', 'Apple___Cedar_apple_rust',
               'Apple___healthy', 'Blueberry___healthy',
 'Cherry_(including_sour)___Powdery_mildew',
 'Cherry_(including_sour)___healthy',
 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
 'Corn_(maize)___Common_rust_', 'Corn_(maize)___Northern_Leaf_Blight',
 'Corn_(maize)___healthy', 'Grape___Black_rot',
 'Grape___Esca_(Black_Measles)',
 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)', 'Grape___healthy',
 'Orange___Haunglongbing_(Citrus_greening)', 'Peach___Bacterial_spot',
 'Peach___healthy', 'Pepper,_bell___Bacterial_spot',
 'Pepper,_bell___healthy', 'Potato___Early_blight' ,'Potato___Late_blight',
 'Potato___healthy', 'Raspberry___healthy', 'Soybean___healthy',
 'Squash___Powdery_mildew', 'Strawberry___Leaf_scorch',
 'Strawberry___healthy', 'Tomato___Bacterial_spot', 'Tomato___Early_blight',
 'Tomato___Late_blight' ,'Tomato___Leaf_Mold' ,'Tomato___Septoria_leaf_spot',
 'Tomato___Spider_mites Two-spotted_spider_mite', 'Tomato___Target_Spot',
 'Tomato___Tomato_Yellow_Leaf_Curl_Virus' ,'Tomato___Tomato_mosaic_virus',
 'Tomato___healthy']


        #lbl2=tk.Label(win,image=img)

        #lbl2.pack(side = "bottom", fill = "both", expand = "yes")
        #img1=('F:/py/leaf_disease_final( COMPLETE )/1.jpg')


        #lbl2=tk.Label(win,image=img1)
        #lbl2.pack(side = "bottom", fill = "both", expand = "yes")
        #loading image
        path2=filedialog.askopenfilename()
        print(path2)


        #img = ImageTk.PhotoImage(Image.open(path2))

        #lbl2=tk.Label(win,image=img)
        #lbl2.pack(side = "bottom", fill = "both", expand = "yes")

        #The Label widget is a standard Tkinter widget used to display a text or image on the screen.
        #panel = tk.Label(win, image = img)
        #panel.pack( fill = "both", expand = "yes")
        #imr=cv2.imread(path2)
        #a=cv2.imshow(imr)
        #print(imr)
```

```
#img1='1.JPG'
#lb2 = Label(win,image=image)
#lb2.pack()
#if ("Result" ==('Grape___Black_rot')):
#lse:
   # print ('normal leaf')


#lbl.grid(column=0, row=0)
win.geometry("550x250")
win.title("Plant Disease Detection using CNN")
win.bind("<Return>", b1_click)
win.mainloop()
```

```
        test_image = image.load_img(path2, target_size = (128, 128))
        test_image = image.img_to_array(test_image)
        test_image = np.expand_dims(test_image, axis = 0)
        result = loaded_model.predict(test_image)
        #print(result)
        print(result)
        fresult=np.max(result)
        label2=label[result.argmax()]
        #print(label2)
        #lb2.configure(image=img)
        #lbl2.image=img
        lbl.configure(text=label2)


        #lbl2(ent.config(state='disabled'))
        win.mainloop()


    except IOError:
        pass


#button

#labelframe = LabelFrame(win, text="Plant Disease Detection using CNN")
#labelframe.pack(fill="both", expand="yes")
label1 = Label(win, text="GUI For Plant Disease Detection using CNN", fg ='blue')
label1.pack()

b1=tk.Button(win, text= 'browse image',width=25, height=3,fg ='red', command=b1_click)
b1.pack()
lbl = Label(win, text="Result", fg ='blue')
lbl.pack()

#image =ImageTk.PhotoImage(file='a.JPG')
```

```
#img1='1.JPG'
#lb2 = Label(win,image=image)
#lb2.pack()
#if ("Result" ==('Grape___Black_rot')):
#lse:
   # print ('normal leaf')


#lbl.grid(column=0, row=0)
win.geometry("550x250")
win.title("Plant Disease Detection using CNN")
win.bind("<Return>", b1_click)
win.mainloop()
```

# CHAPTER 7

# CONCLUSION

## CONCLUSION AND FUTURE WORK

The accurately detection and classification of the plant disease is very important for the successful cultivation of crop and this can be done using image processing. This paper discussed various techniques to segment the disease part of the plant. This report also discussed some Feature extraction and classification techniques to extract the features of infected leaf and the classification of plant diseases. can be efficiently used. From these methods, we can accurately identify and classify various plant diseases using image processing techniques. The classifying accuracy of our algorithm is 97%.

# CHAPTER 8

# REFERENCES


- Mrunalini R. et al., An application of K-means clustering and artificial intelligence in pattern recognition for crop diseases ,2011.

- S.Raj Kumar , S.Sowrirajan," Automatic Leaf Disease Detection and Classification using Hybrid Features and Supervised Classifier", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 5, Issue 6,2016.

- Tatem, D. J. Rogers, and S. I. Hay, "Global transport networks and infectious disease spread," Advances in Parasitology, vol. 62, pp. 293–343, 2006. View at Publisher · View at Google Scholar · View at Scopus.

- J. R. Rohr, T. R. Raffel, J. M. Romansic, H. McCallum, and P. J. Hudson, "Evaluating the links between climate, disease spread, and amphibian declines," Proceedings of the National Academy of Sciences of the United States of America, vol. 105, no. 45, pp. 17436–17441, 2008. View at Publisher · View at Google Scholar View at Scopus.

- T. Van der Zwet, "Present worldwide distribution of fire blight," in Proceedings of the 9th International Workshop on Fire Blight, vol. 590, Napier, New Zealand, October 2001.

- H. Cartwright, Ed., Artificial Neural Networks, Humana Press, 2015.

- Steinwart and A. Christmann, Support Vector Machines, Springer Science & Business Media, New York, NY, USA, 2008. View at MathSciNet.

- Steinwart and A. Christmann, Support Vector Machines, Springer Science & Business Media, New York, NY, USA, 2008. View at MathSciNet.

- S. Sankaran, A. Mishra, R. Ehsani, and C. Davis, "A review of advanced techniques for detecting plant diseases," Computers and Electronics in Agriculture, vol. 72, no. 1, pp. 1–13, 2010. View at Publisher ·View at Google Scholar · View at Scopus.

- P. R. Reddy, S. N. Divya, and R. Vijayalakshmi, "Plant disease detection technique tool—a theoretical approach," International Journal of Innovative Technology and Research, pp. 91–93, 2015. View at Google Scholar.

- A.-K. Mahlein, T. Rumpf, P. Welke et al., “Development of spectral indices for detecting and identifying plant diseases,” Remote Sensing of Environment, vol. 128, pp. 21–30, 2013. View at Publisher · View at Google Scholar · View at Scopus