VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A PROJECT REPORT **(15CSP85)** ON

"PLANT DISEASE DETECTION"

Submitted in Partial fulfillment of the Requirements for the Degree of

Bachelor of Engineering in Computer Science & Engineering

By

BURHANUDDIN KHOMOSI(1CR16CS034)

C M ARJUN(1CR16CS035)

HARSH KUMAR(1CR16CS056)

Under the Guidance of,

Dr. P N SINGH

Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled **"PLANT DISEASE DETECTION"** carried out by **Mr. BURHANUDDIN KHOMOSI**, USN **1CR16CS034**, **Mr. C M ARJUN**, USN **1CR16CS035**, **Mr. HARSH KUMAR**, USN **1CR16CS056,** bonafide students of CMR Institute of Technology, in partial fulfillment for the award of **Bachelor of Engineering** in Computer Science and Engineering of the Visveswaraiah Technological University, Belgaum during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Dr. P N Singh	**Dr. Prem Kumar Ramesh**	**Dr. Sanjay Jain**
Professor	**Professor & Head**	**Principal**
Dept. of CSE, CMRIT	**Dept. of CSE, CMRIT**	**CMRIT**

External Viva

Name of the examiners	Signature with date
1.	_____
2.	_____

DECLARATION

We, the students of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the work entitled "**PLANT DISEASE DETECTION**" has been successfully completed under the guidance of Dr. P N Singh, Computer Science and Engineering Department, CMR Institute of technology, Bangalore. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2019 - 2020. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place: Bangalore

Date: 10/06/2020

Team members:

BURHANUDDIN KHOMOSI (1CR16CS034) _____

C M ARJUN (1CR16CS035) _____

HARSH KUMAR (1CR16CS056) _____

ABSTRACT

The proposed system helps in identification of plant disease and provides remedies that can be used as a defense 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.

We capture images of the plants which will act as input for the software, based of which the software will tell us whether the plant is healthy or not. With our code and training model we have achieved an accuracy level of more than 95%. Our software gives us the name of the plant Disease with its confidence level and also the remedy that can be taken as a cure.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing me a platform to pursue my studies and carry out my final year project

I have a great pleasure in expressing my deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore, for his constant encouragement.

I would like to thank **Dr. Prem Kumar Ramesh,** Professor and Head, Department of Computer Science and Engineering, CMRIT, Bangalore, who has been a constant support and encouragement throughout the course of this project.

I consider it a privilege and honor to express my sincere gratitude to my guide **Dr. P N Singh, Professor,** Department of Computer Science and Engineering, for the valuable guidance throughout the tenure of this review.

I also extend my thanks to all the faculty of Computer Science and Engineering who directly or indirectly encouraged me.

Finally, I would like to thank my parents and friends for all their moral support they have given me during the completion of this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN	**Convolutional Neural Network**
RGB	**Red Green Blue**
HSV	**Hue Saturation Value**
HSI	**Hue Saturation Intensity**
GLCM	**Gray Level Co-occurrence Matrix**
SVM	**Support Vector Machine**
ANN	**Artificial Neural Network**
SURF	**Speeded Up Robust Features**
CCM	**Classification Manager**
ReLU	**Rectified Linear Unit**

CHAPTER 1

INTRODUCTION

1.1 Relevance of the Project

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).

1.2 Problem Statement

If a person or a farmer wants to know if their plant is healthy or not, they can use this program to get the basic idea about what disease a plant is having by uploading its leaf's image and getting the remedies so that the disease is identified and controlled.

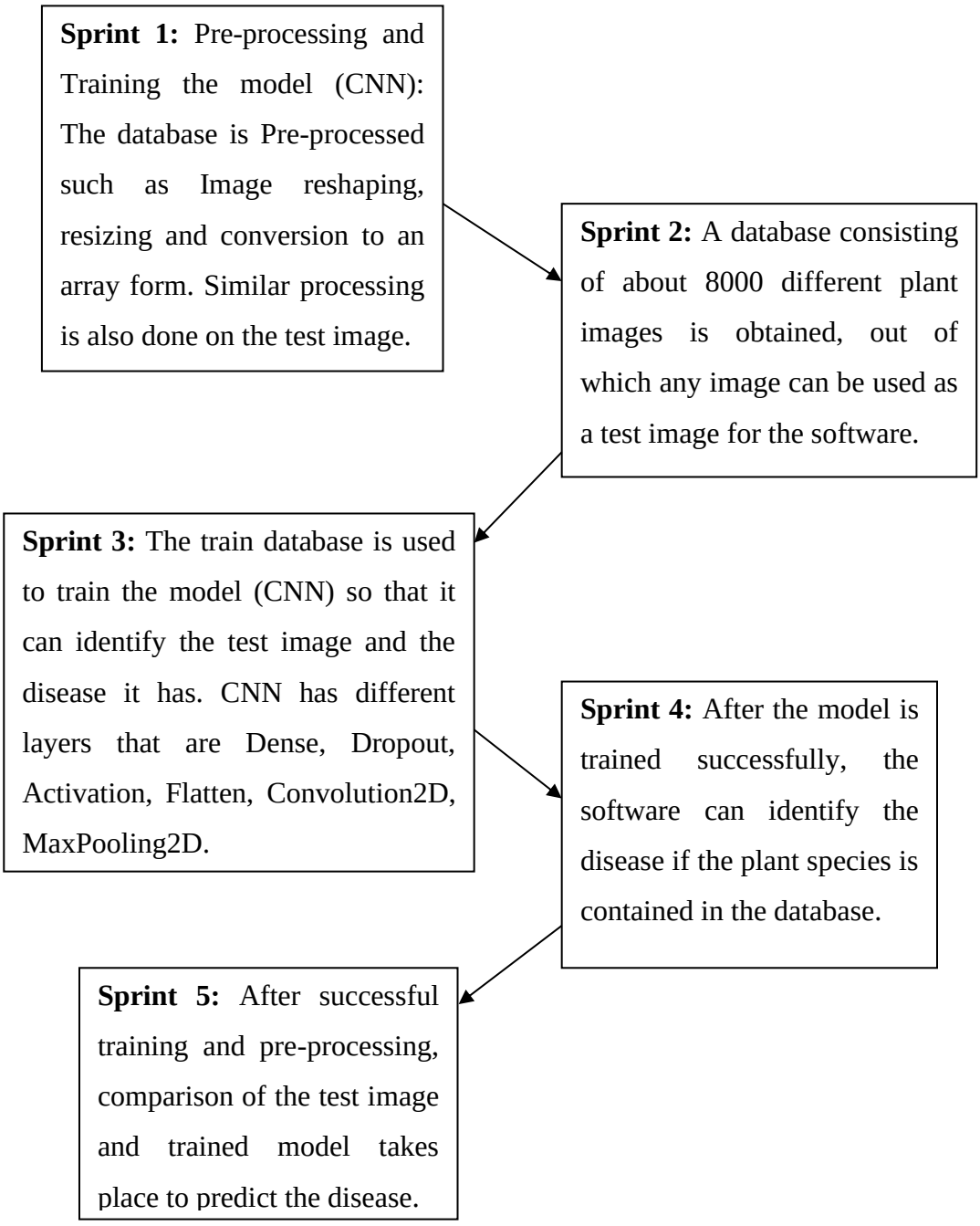
1.3 Objective

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, suggests the name of pesticide to be used. It also identifies the insects and pests responsible for epidemic. 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.

1.4 Methodology



1.5 Dataset

Dataset collection: Initial step for any image processing-based project is acquiring proper dataset which is valid. Most of the time the standard dataset is preferred but in certain circumstances we do not get proper dataset. So, in such conditions we can collect the images and can form our own dataset.

There is a huge dataset so basically the images with better resolution and angle are selected. After selection of images we should have deep knowledge about the different leaves and the disease they have. Huge research is done from plant village organization repository. Different types of plant images are studied and corresponding. After detailed study, labelling is done by segregating the images and with different diseases in data set write the name of the plant and disease we used.

CHAPTER 2

LITERATURE SURVEY

Paper [1] presents classification and detection techniques that can be used for plant leaf disease classification. Here, pre-processing is done before feature extraction. RGB images are converted into white and then converted into grey level image to extract the image of vein from each leaf. Then basic Morphological functions are applied on the image. Then the image is converted into binary image. After that if binary pixel value is 0 its converted to corresponding RGB image value. Finally, by using Pearson correlation and Dominating feature set and Naïve Bayesian classifier disease is detected.

In paper [2] there are four steps. Out of them the first one is gathering image from several part of the country for training and testing. Second part is applying Gaussian filter is used to remove all the noise and thresholding is done to get the all green color component. K-means clustering is used for segmentation. All RGB images are converted into HSV for extracting feature.

The paper [3] presents the technique of detecting jute plant disease using image processing. Image is captured and then it is realized to match the size of the image to be stored in the database. Then the image is enhanced in quality and noises are removed. Hue based segmentation is applied on the image with customized thresholding formula. Then the image is converted into HSV from RGB as it helps extracting region of interest. This approach proposed can significantly support detecting stem-oriented diseases for jute plant.

According to paper [4] they have proposed for a technique that can be used for detecting paddy plant disease by comparing it with 100 healthy images and 100 sample of disease1 and another 100 sample of disease2. It’s not sufficient enough to detect disease or classify it training data is not linearly separable.

In paper [5] detection of unhealthy plant leaves includes some steps are RGB image acquisition. Converting the input image from RGB to HSI format. Masking and removing the green pixels. Segment the components using Ostu’s

method. Computing the texture features using color-co-occurrence methodology and finally classifying the disease using Genetic Algorithm.

Paper [6] includes tomato disease detection using computer vision. A gray scale image is turned into binary image depending on threshold value. The threshold algorithm is used for image segmentation. The threshold values are given color indices like red, green, blue. But the thresholding is not a reliable method as this technique only distinguishes red tomatoes from other colors. It becomes difficult to distinguish ripe and unripe tomatoes. For this K-means clustering algorithm is used to overcome the drawbacks. K-means create a particular number of non-hierarchical clusters. This method is numerical, unsupervised, non-deterministic and iterative. Then separating the infected parts from the leaf, the RGB image was converted into YcbCr to enhance the feature of the image. The final step is the calculation of the percentage of infection and distinguishing the ripe and unripe tomatoes.

The methodology for cucumber disease detection is presented in paper [7]. The methodology includes image acquisition, image preprocessing, feature extraction with Gray level co-occurrence matrix (GLCM) and finally classified with two types: Unsupervised classification and supervised classification.

Paddy plant is an important plant in continental region. In paper [8] RGB images are converted into gray scale image using color conversion. Various enhancement techniques like histogram equalization and contrast adjustment are used for image quality enhancement. Different types of classification features like SVM, ANN, FUZZY classification are used here. Feature extract6ion uses different types of feature values like texture feature, structure feature and geometric feature. By using ANN and FUZZY classification, it can identify the disease of the paddy plant.

In paper [9] popular methods have utilized machine learning, image processing and classification-based approaches to identify and detect the disease of agricultural product.

In paper [10] image processing technique are used to detect the citrus leaf disease. This system includes: Image preprocessing, segmentation of the leaf using K-means clustering to determine the diseased areas, feature extraction and classification of disease. Uses Gray-Level Co-Occurrence matrix (GLCM) for feature extraction and classification is done using support vector machine (SVM).

Paper Name	Methods	Future Work
[1] Detection and Classification technique Of Yellow Vein Mosaic Virus Disease in Okra Leaf Images Using Naïve Bayesian Classifier.	k-means clustering, Basic Morphological functions, Naïve Bayesian classifier, Color Co-Occurrence method.	Nil
[2]Svm Classifier Based Grape Leaf Disease Detection	K-means clustering algorithm with SVM, Color co-occurrence method.	Developing combinations of more algorithms by using fusion classification technique so as to improve the detection the detection rate of the classification process.
[3] Detecting Jute Plant Disease Using Image Processing and Machine Learning	Color co-occurrence methods, Multi SVM classifier.	Nil

[4] Plant disease Analysis Using Histogram Matching based on Bhattacharaya's Distance Calculation	Bhattacharaya's similarity calculation	It identifies the burning and blast disease of paddy. In further enhancement this method can be used to find all kind of diseases in various leaves. This may help the farmers in identification of the disease in the leaf in feasible and accurate way, in short time span.
[5] Detection Of Unhealthy Plant Leaves Using Image Processing and Genetic Algorithm with Arduino.	Genetic algorithm, Arduino, Masking the green pixel and color co-occurrence method.	Nil
[6] Maturity and disease detection in tomato using computer vision.	Threshholding algorithm, K-means clustering	Nil
[7] Cucumber disease detection using artificial neural network.	ANN, GLCM(Gray level co-occurrence method)	Classification accuracy can be increased by using additional texture features.
[8] Detection and measurement of paddy leaf disease symptoms using image processing.	ANN, FUZZY classification, SVM, K-means algorithm, color co-occurrence method.	It evaluates the techniques in image processing, detecting diagnosing of crop leaf disease.
[9] Recent Machine Learning Based Approaches for Disease Detection and Classification of Agricultural products.	K-means, GLCM, ANN, SURF, CCM, SVM.	According to this paper there is a need of system in agriculture science can combinely detects the disease on all kinds of plants, Fruits and Vegetables.
[10] Detection of leaf disease and classification using digital image processing.	GLCM, SVM, K-means	Classifying different plant disease and improve the classification accuracy.

Table 2.1 List of reviewed papers with Name, Used methods and Future work

Paper Number	Methods	Accuracy Value
Paper 1	K-means clustering, basic morphological functions, Naïve Bayesian classifier, color co-occurrence method.	87%
Paper 2	K-means clustering algorithm with SVM, color co-occurrence method.	88.89%
Paper 3	Color co-occurrence method, Multi SVM classifier	86%
Paper 7	ANN, GLCM	80.45%
Paper 8	ANN, FUZZY classification, SVM, K-means algorithm, Color co-occurrence method.	94.70%
Paper 9	K-means, GLCM, ANN, SURF, CCM, SVM	95%
Paper 10	GLCM, SVM, K-means	90%

Table 2.2 List of reviewed papers with accuracy values and used methods

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Software Requirements

1. Python 3
2. Anaconda Navigator
3. Jupyter Notebook
4. Pycharm

3.2 Hardware Requirements

1. Windows 7/8/8.1/10 /Unix
2. RAM: 2GB
3. Hard Disk Space: 5 GB
4. Intel Core 2.0 Ghz or above
5. System Type: 32 bit or 64 bit processor
6. GPU

3.3 Packages Required

1. numpy (pip install numpy)
2. tqdm (pip install tqdm)
3. TensorFlow
4. OpenCV
5. Matplotlib
6. tflearn
7. tkinter
8. PIL

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

4.1 Flow Chart

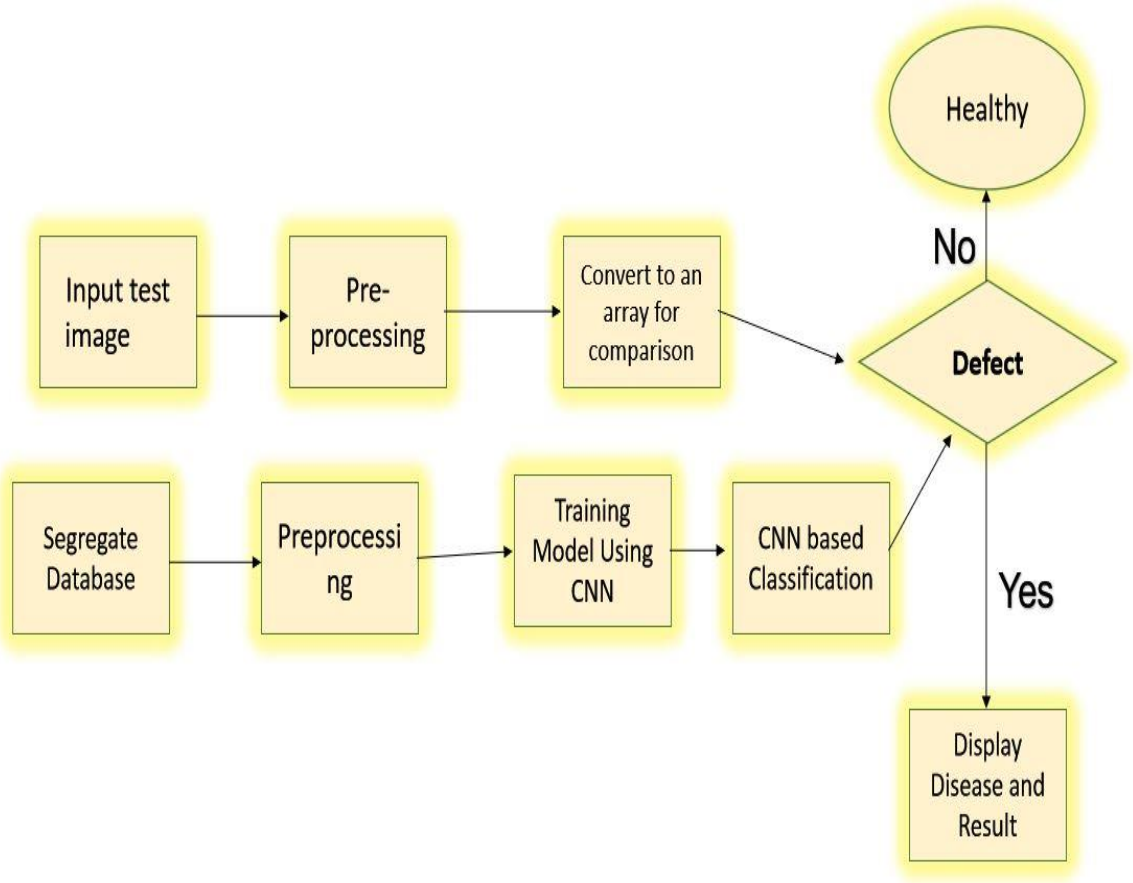


Fig 4.1 Flow Chart of Disease Detection

4.2 General Explanation

1. The input test image is acquired and pre-processed in the next stage and then it is converted into array form for comparison.

2. The selected database is properly segregated and pre-processed and then renamed into proper folders.
3. The model is properly trained using CNN and then classification takes place.
4. The comparison of the test image and the trained model take place followed by the display of the result.
5. If there is a defect or disease in the plant the software displays the disease along with the remedy.

CHAPTER 5

IMPLEMENTATION

5.1 Algorithm

- ⇨ Provide input image into convolution layer
- ⇨ Choose parameters, apply filters with strides, padding if required. Perform convolution on the image and apply ReLU (Rectified Linear Unit) activation to the matrix.
- ⇨ Perform pooling to reduce dimensionality size
- ⇨ Add as many convolutional layers until satisfied
- ⇨ Flatten the output and feed into a fully connected layer (FC Layer)
- ⇨ Output the class using an activation function (Logistic Regression with cost functions) and classifies images.

5.2 Program Flow

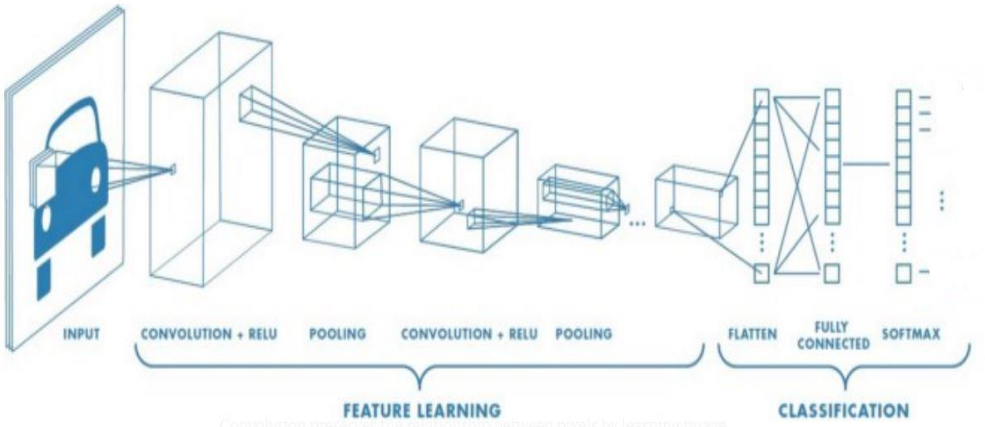


Fig 5.1 A Neural Network consisting of multiple convolutional layers

5.3 Explanation

CNN (Convolutional Neural Network) image classification takes an input image, processes it and classifies it under certain categories (eg., Healthy leaf, Bacterial spot, Late blight, Curl virus, etc). Computer sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d (h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

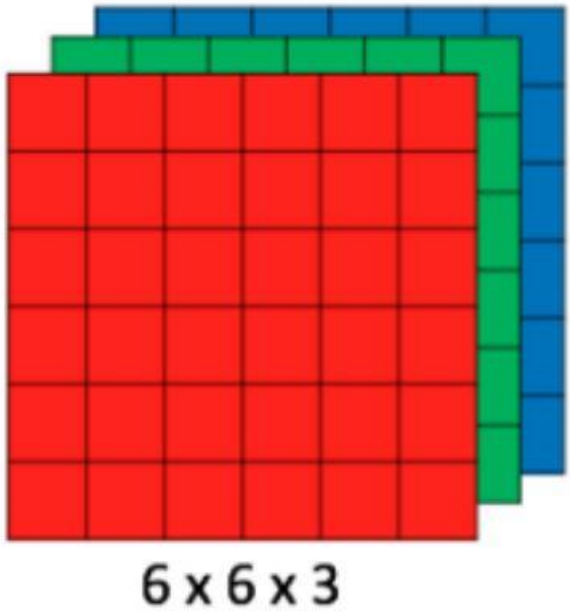


Fig 5.2 An Array of RGB matrix

A deep learning CNN model to train and test, each input image will pass it through a series of convolution layers with filters (kernels), Pooling, Fully Connected layers (FC Layers) and apply Softmax function in order to classify an object with probabilistic values between 0 and 1.

5.3.1 Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

- An image matrix of dimension **h x w x d**
- A filter matrix of dimension **f_h x f_w x d**
- Outputs a matrix of dimension **(h – f_h + 1) x (w – f_w + 1) x 1**

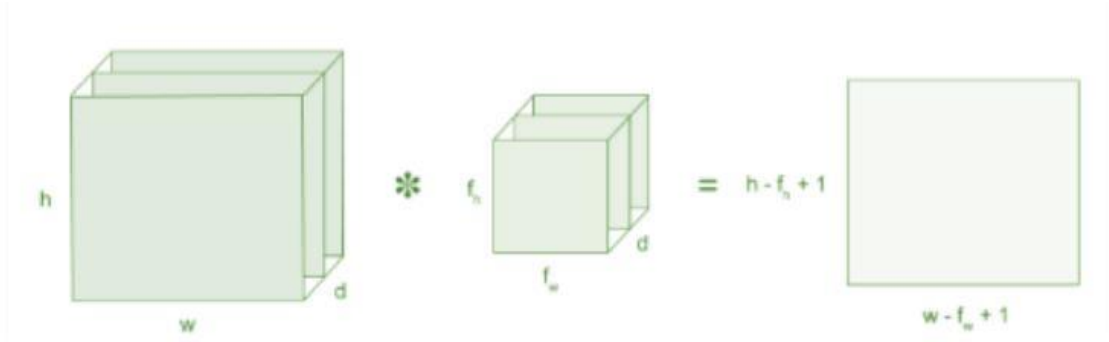


Fig 5.3 Multiplication of Image Matrix with Filter Matrix

Consider a 5 x 5 Image Matrix whose image pixel values are [0, 1] and a 3 x 3 Filter Matrix as shown below:

Fig 5.4 Image Matrix multiplies Filter Matrix

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called a **"Feature Map"** in a series of steps, some of which are shown below:

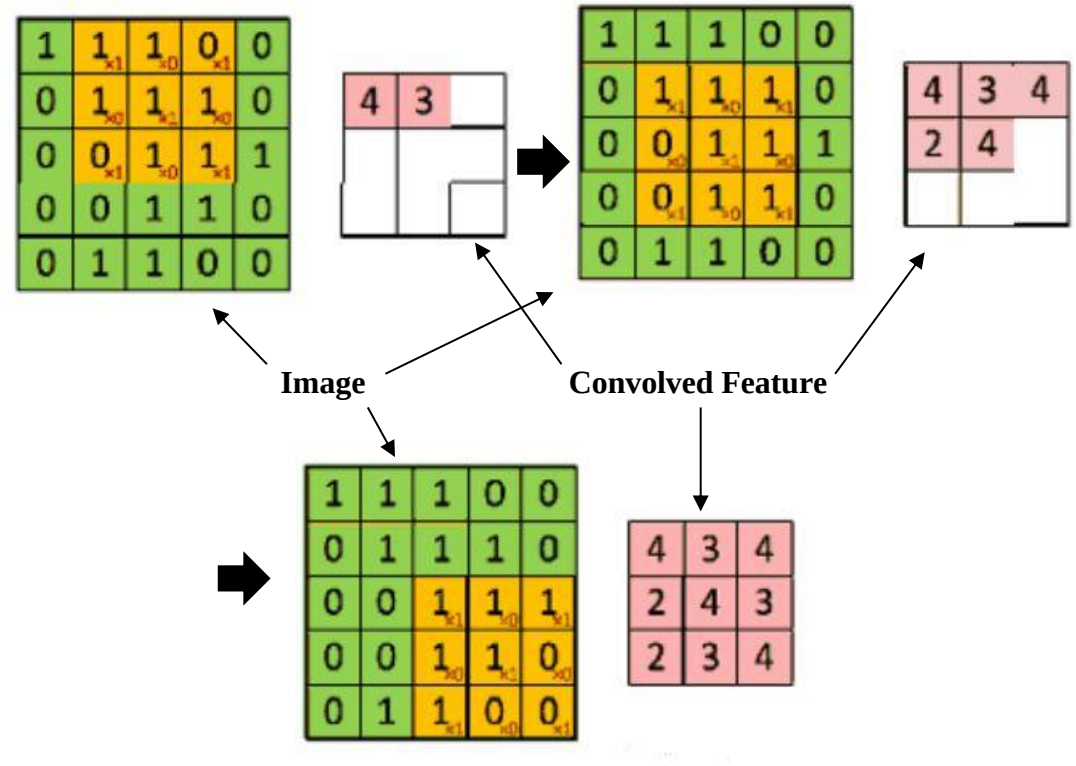


Fig 5.5 3 x 3 Output Matrix

5.3.2 Strides

Stride is the number of pixel shifts over the input matrix. When the stride is 1 then we move the filters 1 pixel at a time. When the stride is 2 then we move the filters 2 pixels at a time and so on. The below figure shows convolution working with a stride of 2.

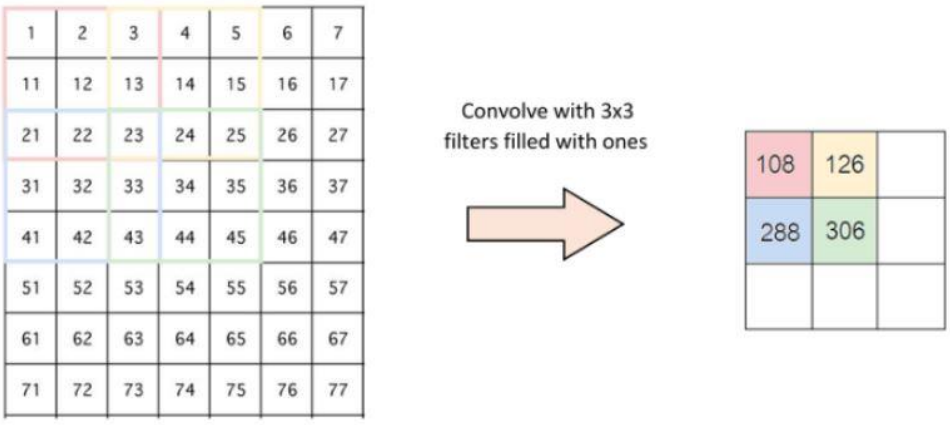


Fig 5.6 Stride of 2 pixels

5.3.3 Padding

Sometimes filter does not perfectly fit the input image. Hence, we have two options:

- Pad the picture with zeros (zero-padding) so that it fits.
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

5.3.4 Non-Linearity (ReLU)

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is ***f(x) = max (0, x).***

5.3.5 Max Pooling

Pooling layers section would reduce the number of parameters when the images are too large. Max pooling takes the largest element from the rectified feature map.

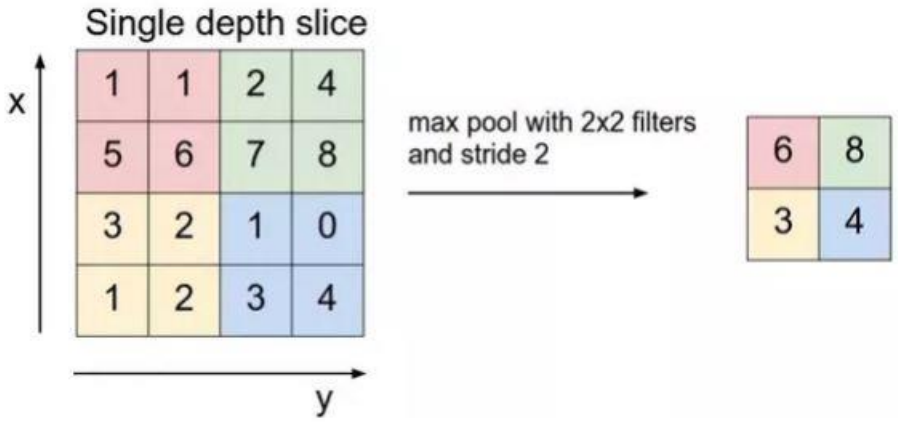


Fig 5.7 Max Pooling

5.3.6 Fully Connected Layer

To the layer we call as FC layer, we flatten our matrix into a vector and feed it into a Fully Connected layer like a neural network.

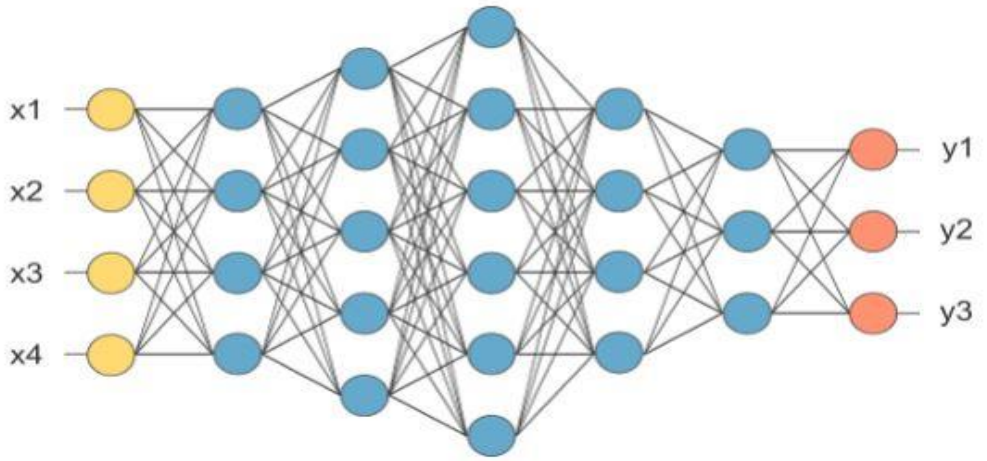


Fig 5.8 After Pooling, flattened as FC Layer

The feature map matrix will be converted as vector (x1, x2, x3, …). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax to classify the output as healthy leaf, having bacterial spots, undergoing late blight or having curl virus.

5.3.7 Code Snippets

cnn.py

We've got the data, but we can't exactly just stuff raw images right through our convolutional neural network. First, we need all of the images to be the same size, and then we also will probably want to just RGB them.

TensorFlow Fold makes it easy to implement deep-learning models that operate over data of varying size and structure.

First, we'll get our imports and constants for pre-processing:

```
import cv2             # working with, mainly resizing, images
import numpy as np     # dealing with arrays
import os              # dealing with directories
from random import shuffle # mixing up or currently ordered data that might lead our network astray in training.
from tqdm import tqdm      # a nice pretty percentage bar for tasks.
TRAIN_DIR = 'train/train'
TEST_DIR = 'test/test'
IMG_SIZE = 50
LR = 1e-3
MODEL_NAME = 'healthyvsunhealthy-{}-{}.model'.format(LR, '2conv-basic')
```

Now, our first order of business is to convert the images and labels to array information that we can pass through our network. To do this, we'll need a helper function to convert the image name to an array.

We also build another function to fully process the training images and their labels into arrays:

```
def label_img(img):
    word_label = img[0]

    if word_label == 'h': return [1,0,0,0]

    elif word_label == 'b': return [0,1,0,0]
    elif word_label == 'v': return [0,0,1,0]
    elif word_label == 'l': return [0,0,0,1]

def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR,img)
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data
```

The tqdm module is a really nice, pretty, way to measure where you are in a process, rather than printing things out at intervals...etc, it gives a progress bar.

Anyway, the above function converts the data for us into array data of the image and its label.

When we've gone through all of the images, we shuffle them, then save. Shuffle modifies a variable in place, so there's no need to re-define it here.

With this function, we will both save, and return the array data. This way, if we just change the neural network's structure, and not something with the images, like image size, etc. then we can just load the array file and save some processing time. We make a function to process the testing data. The data has no label.

```
def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        path = os.path.join(TEST_DIR,img)
        img_num = img.split('.')[0]
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        testing_data.append([np.array(img), img_num])

    shuffle(testing_data)
    np.save('test_data.npy', testing_data)
    return testing_data
```

Now, we can run the training:

```
train_data = create_train_data()
# If you have already created the dataset:
train_data = np.load('train_data.npy')
```

Next, we're ready to define our neural network and we need to reset the graph instance, since we're doing this in a continuous environment:

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import tensorflow as tf
tf.reset_default_graph()

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

convnet = conv_2d(convnet, 32, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 64, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 128, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 32, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 64, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 4, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')
#tensorboard --logdir=log
```

What we have here is a nice, 2 layered convolutional neural network, with a fully connected layer, and then the output layer.

First, we just want to see how 3 epochs trains, but then, after 3, maybe we're done, or maybe we want to see about 5 epochs. We want to be saving our model after every session, and reloading it if we have a saved version, so we added this:

```
if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('model loaded!')
```

Now, let's split out training and testing data:

```
train = train_data[:-500]
test = train_data[-500:]
```

Now, the training data and testing data are both labeled datasets. The training data is what we'll fit the neural network with, and the test data is what we're going to use to validate the results. The test data will be "out of sample," meaning the testing data will only be used to test the accuracy of the network, not to train it.

We also have "test" images that we downloaded. Those images are not labeled at all.

Next, we're going to create our data arrays.

```
X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
test_y = [i[1] for i in test]
```

Now we fit for 8 epochs:

```
model.fit({'input': X}, {'targets': Y}, n_epoch=8, validation_set=({'input': test_x}, {'targets': test_y}),
    snapshot_step=40, show_metric=True, run_id=MODEL_NAME)
```

Now, since we are happy with the accuracy retrieved from our model, we'll go ahead and save the model:

```
model.save(MODEL_NAME)
```

Moving on to the user interface section of our project, we have ui.py:

ui.py

We import the following packages to create a window-based UI

```
import tkinter as tk
from tkinter.filedialog import askopenfilename
import shutil
import os
import sys
from PIL import Image, ImageTk
```

We deploy a window titled PLANT DISEASE DETECTION as follows:

```
window = tk.Tk()

window.title("PLANT DISEASE DETECTION")

window.geometry("560x550")
window.configure(background ="gray")

title = tk.Label(text=" Click below to choose picture for testing disease", background = "gray", fg="black", font=("", 15))
title.grid()
```

We now begin to create separate windows using tkinter to display the remedies for the specific diseases that are identified.

We start off with the bacterial spot remedies:

```
def bact():
    window.destroy()
    window1 = tk.Tk()

    window1.title("PLANT DISEASE DETECTION")

    window1.geometry("600x370")
    window1.configure(background="gray")

    def exit():
        window1.destroy()
    rem = "The remedies for Bacterial Spot are:\n\n "
    remedies = tk.Label(text=rem, background="black",
                        fg="White", font=("", 15))
    remedies.grid(column=0, row=5, padx=10, pady=10)
    rem1 = " Discard or destroy any affected plants. \n  Do not compost them. \n  " \
           "Rotate your plants yearly to prevent re-infection next year. \n Use copper fungicites"
    remedies1 = tk.Label(text=rem1, background="gray",
                         fg="Black", font=("", 12))
    remedies1.grid(column=0, row=8, padx=10, pady=10)

    button = tk.Button(text="Exit", command=exit)
    button.grid(column=0, row=9, padx=20, pady=20)

    window1.mainloop()
```

Next, we have the remedies for the Yellow leaf curl virus that are displayed very subtly.

```
def vir():
    window.destroy()
    window1 = tk.Tk()

    window1.title("PLANT DISEASE DETECTION")

    window1.geometry("779x370")
    window1.configure(background="gray")

    def exit():
        window1.destroy()
    rem = "The remedies for Yellow leaf curl virus are: "
    remedies = tk.Label(text=rem, background="black",
                        fg="white", font=("", 15))
    remedies.grid(column=0, row=7, padx=10, pady=10)
    rem1 = " Monitor the field, handpick diseased plants and bury them. \n  Use sticky yellow plastic traps. \n  " \
           "Spray insecticides such as organophosphates, carbametes during the seedling stage. \n Use copper fungicites"
    remedies1 = tk.Label(text=rem1, background="gray",
                         fg="Black", font=("", 12))
    remedies1.grid(column=0, row=8, padx=10, pady=10)

    button = tk.Button(text="Exit", command=exit)
    button.grid(column=0, row=9, padx=20, pady=20)

    window1.mainloop()
```

Below is the code for the window designed for Late Blight disease on plants:

```
def latebl():
    window.destroy()
    window1 = tk.Tk()

    window1.title("PLANT DISEASE DETECTION")

    window1.geometry("560x350")
    window1.configure(background="gray")

    def exit():
        window1.destroy()
    rem = "The remedies for Late Blight are: "
    remedies = tk.Label(text=rem, background="black",
                        fg="white", font=("", 15))
    remedies.grid(column=0, row=7, padx=10, pady=10)

    rem1 = " Monitor the field, remove and destroy infected leaves. \n  Treat organically with copper spray. \n  " \
           "Use chemical fungicides,the best of which is chlorothalonil."
    remedies1 = tk.Label(text=rem1, background="gray",
                         fg="black", font=("", 12))
    remedies1.grid(column=0, row=8, padx=10, pady=10)

    button = tk.Button(text="Exit", command=exit)
    button.grid(column=0, row=9, padx=20, pady=20)

    window1.mainloop()
```

Below is the code snippet for testing whether the test image provided is either of the following:

- healthy
- unhealthy --Bacterial Spot

--Leaf Curl Virus

--Late Blight

```
if np.argmax(model_out) == 0:
    str_label = 'healthy'
elif np.argmax(model_out) == 1:
    str_label = 'bacterial'
elif np.argmax(model_out) == 2:
    str_label = 'viral'
elif np.argmax(model_out) == 3:
    str_label = 'lateblight'

if str_label =='healthy':
    status ="HEALTHY"
    message = tk.Label(text='Status: ' + status, background="gray",
                       fg="darkgreen", font=("", 15))
else:
    status = "UNHEALTHY"
    message = tk.Label(text='Status: ' + status, background="gray",
                       fg="red", font=("", 15))
```

```
message.grid(column=0, row=3, padx=10, pady=10)
if str_label == 'bacterial':
    diseasename = "Bacterial Spot "
    disease = tk.Label(text='Disease Name: ' + diseasename, background="gray",
                       fg="red", font=("", 15))
    disease.grid(column=0, row=4, padx=10, pady=10)
    r = tk.Label(text='Click below for remedies...', background="gray", fg="black", font=("", 15))
    r.grid(column=0, row=5, padx=10, pady=10)
    button3 = tk.Button(text="Remedies", command=bact)
    button3.grid(column=0, row=6, padx=10, pady=10)
elif str_label == 'viral':
    diseasename = "Yellow leaf curl virus "
    disease = tk.Label(text='Disease Name: ' + diseasename, background="gray",
                       fg="red", font=("", 15))
    disease.grid(column=0, row=4, padx=10, pady=10)
    r = tk.Label(text='Click below for remedies...', background="gray", fg="black", font=("", 15))
    r.grid(column=0, row=5, padx=10, pady=10)
    button3 = tk.Button(text="Remedies", command=vir)
    button3.grid(column=0, row=6, padx=10, pady=10)
```

```
elif str_label == 'lateblight':
    diseasename = "Late Blight "
    disease = tk.Label(text='Disease Name: ' + diseasename, background="gray",
                       fg="red", font=("", 15))
    disease.grid(column=0, row=4, padx=10, pady=10)
    r = tk.Label(text='Click below for remedies...', background="gray", fg="black", font=("", 15))
    r.grid(column=0, row=5, padx=10, pady=10)
    button3 = tk.Button(text="Remedies", command=latebl)
    button3.grid(column=0, row=6, padx=10, pady=10)
else:
    r = tk.Label(text='Plant is healthy', background="lightgreen", fg="darkgreen",
                 font=("", 15))
    r.grid(column=0, row=4, padx=10, pady=10)
    button = tk.Button(text="Exit", command=exit)
    button.grid(column=0, row=9, padx=20, pady=20)
```

Underlying code snippet demonstrates how to create a window that opens an image to be used as a test image:

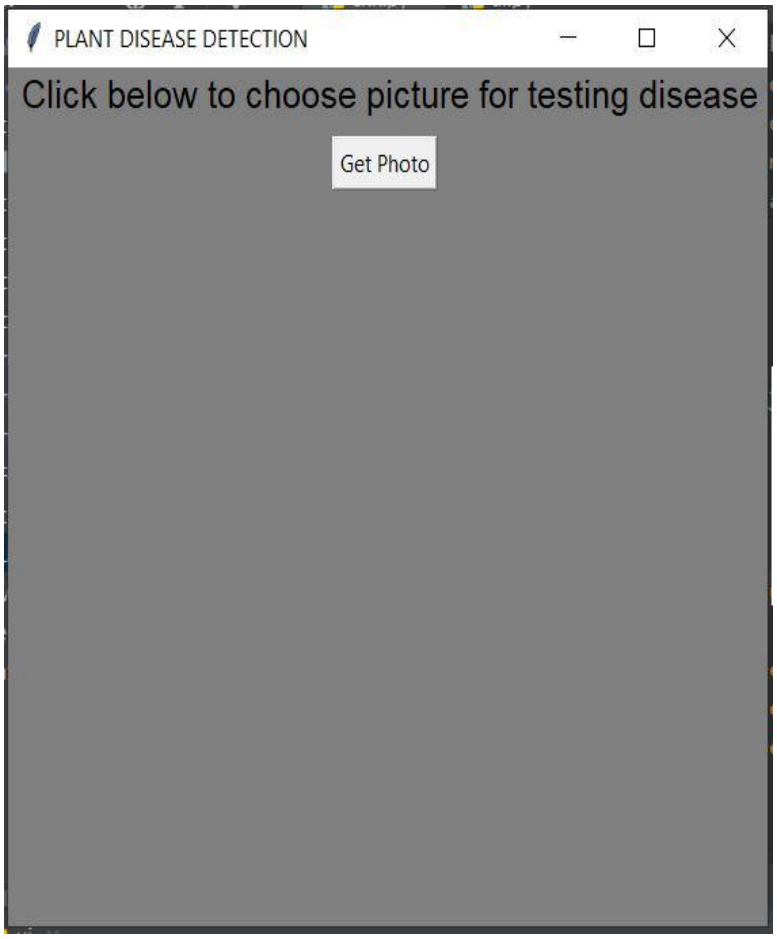
```
def openphoto():
    dirPath = "testpicture"
    fileList = os.listdir(dirPath)
    for fileName in fileList:
        os.remove(dirPath + "/" + fileName)

    fileName = askopenfilename(initialdir='C:/Users/C M ARJUN/Documents/final project testing/test', title='Select image for analysis ',
                       filetypes=[('image files', '.jpg')])
    dst = "C:/Users/C M ARJUN/Documents/final project testing/testpicture"
    shutil.copy(fileName, dst)
    load = Image.open(fileName)
    render = ImageTk.PhotoImage(load)
    img = tk.Label(image=render, height="250", width="537")
    img.image = render
    img.place(x=0, y=0)
    img.grid(column=0, row=1, padx=10, pady = 10)
    title.destroy()
    button1.destroy()
    button2 = tk.Button(text="Analyse Image", command=analysis)
    button2.grid(column=0, row=2, padx=10, pady = 10)
button1 = tk.Button(text="Get Photo", command = openphoto)
button1.grid(column=0, row=1, padx=10, pady = 10)

window.mainloop()
```

CHAPTER 6

RESULTS AND DISCUSSION

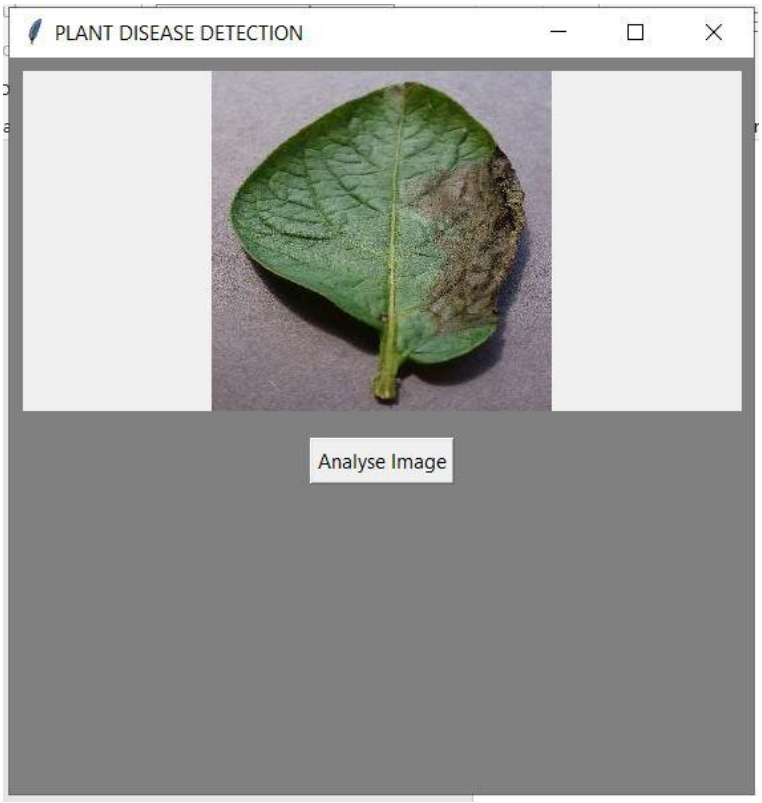
As a result, we have successfully been able to classify whether a plant is healthy or is affected by any disease by analyzing the plant leaf.

And if it is affected by any disease, then determining what the disease is and also providing the user with possible remedies that he/she can carry out to prevent the development of the disease.



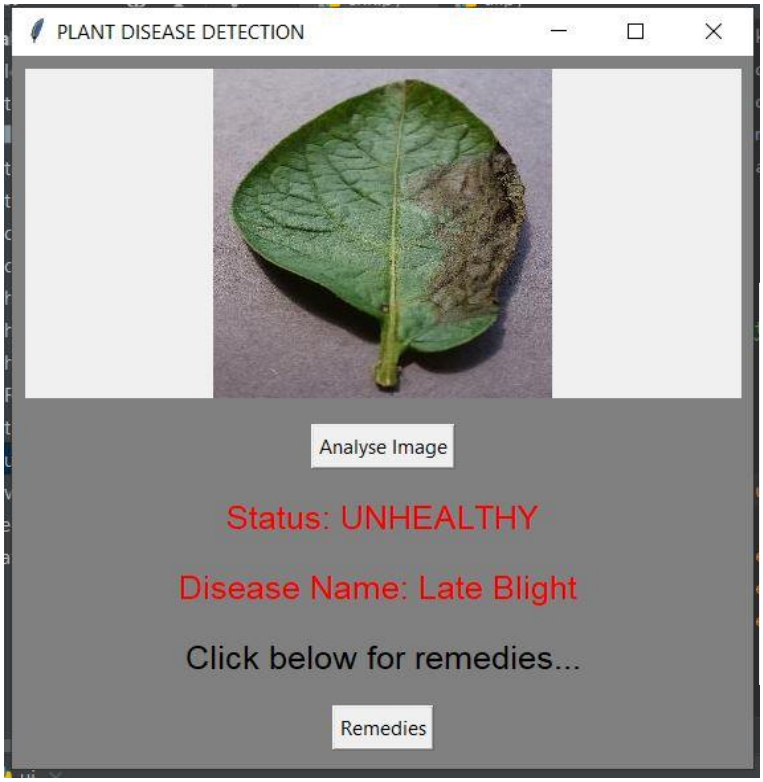
Here, we choose a picture of a plant leaf of whose disease, if any, is to be tested.

Fig 6.1 Initial Window to select an image to test



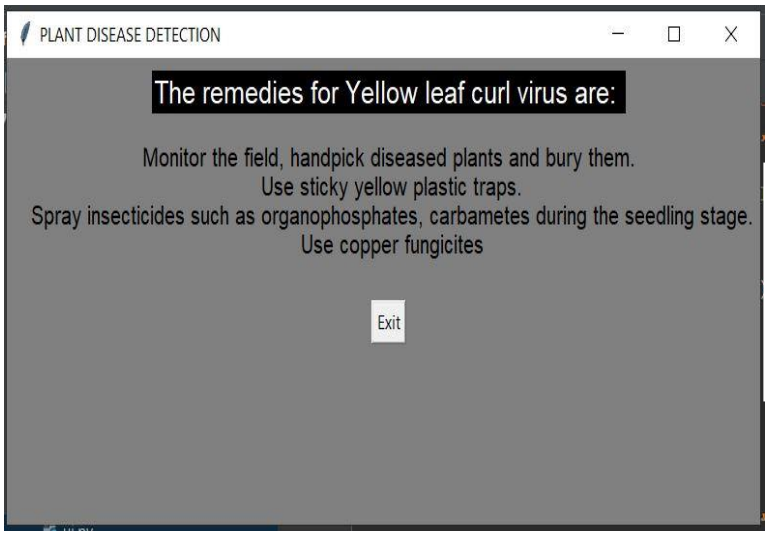
The plant leaf image is loaded and displayed and a button to analyse the image appears.

Fig 6.2 Window that displays the loaded image



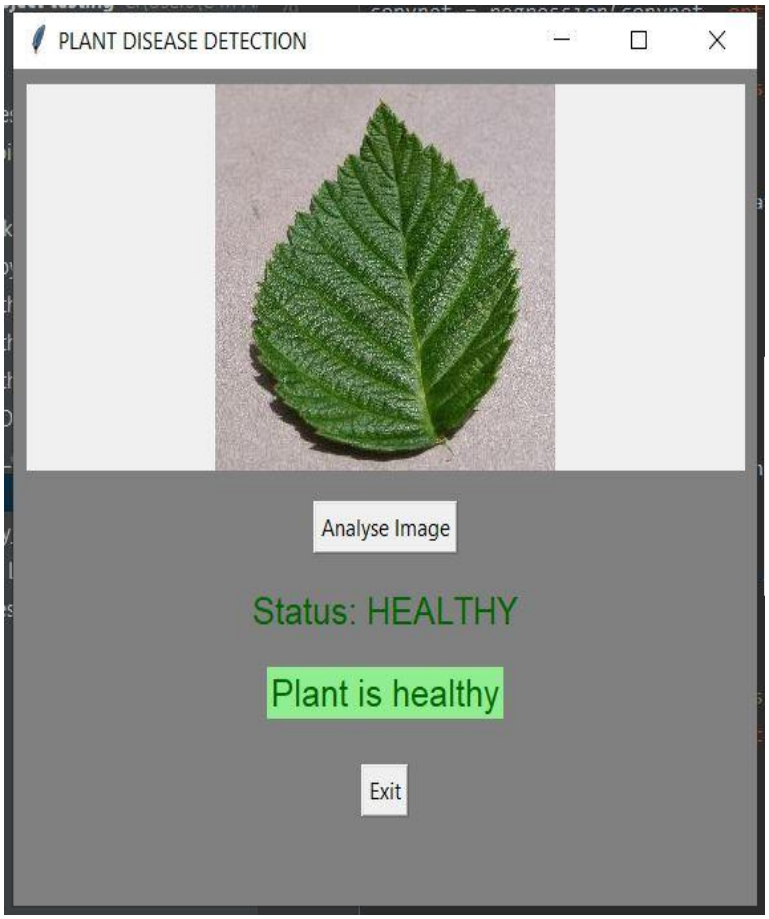
The plant leaf image is analysed and the status along with the disease (if any) is displayed along with a button that reveals the remedies.

Fig 6.3 Window that displays the disease and a button to reveal remedies



The remedies for one of the diseases we have taken into consideration called a yellow leaf curl virus is displayed.

Fig 6.4 Window that displays the remedies for the respective disease



This is one of the alternatives where the plant turns out to be healthy

Fig 6.5 Window that displays that the plant is healthy

CHAPTER 7

TESTING

We have used an open source tool called **TensorBoard** that provides the visualization and tooling needed for machine learning experimentation.

TensorBoard is a suite of web applications for inspecting and understanding your TensorFlow runs and graphs. It is designed to run entirely offline, without requiring any access to the Internet. For instance, this may be on your local machine, behind a corporate firewall, or in a datacenter.

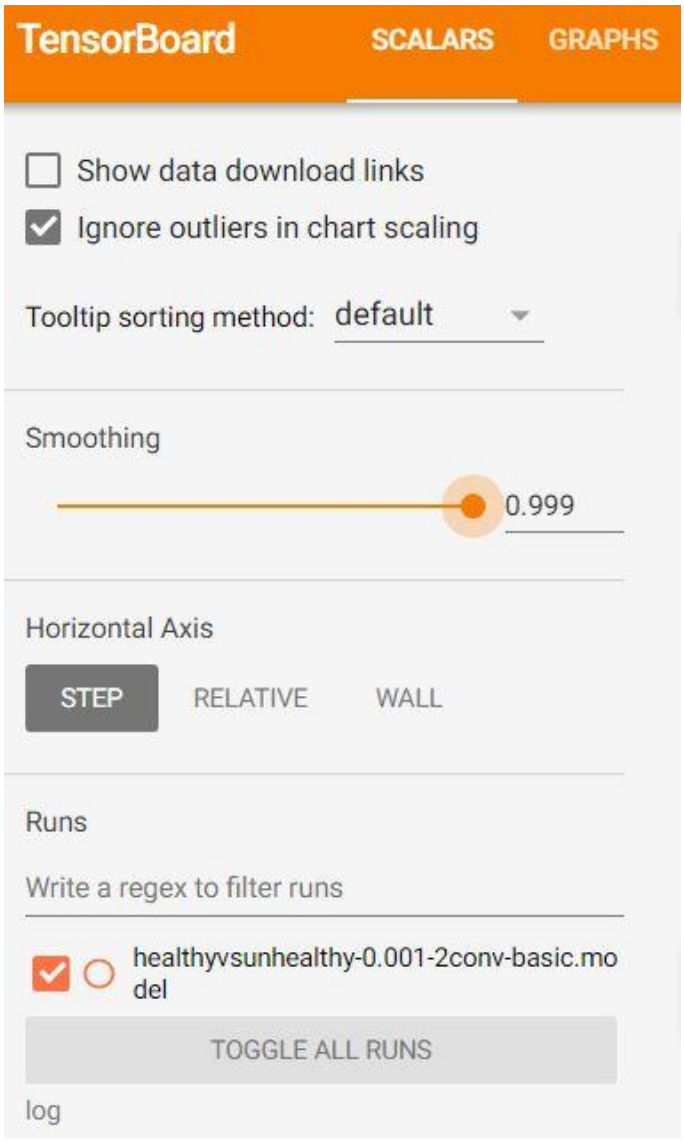


Fig 7.1 TensorBoard Interface

TensorBoard is used for the following:

- Tracking and visualizing metrics such as loss and accuracy
- Visualizing the model graph (ops and layers)
- Viewing histograms of weights, biases, or other tensors as they change over time
- Projecting embeddings to a lower dimensional space
- Displaying images, text, and audio data

A command called “tensorboard --logdir=<name of the log directory>” is sent on the command line interface.

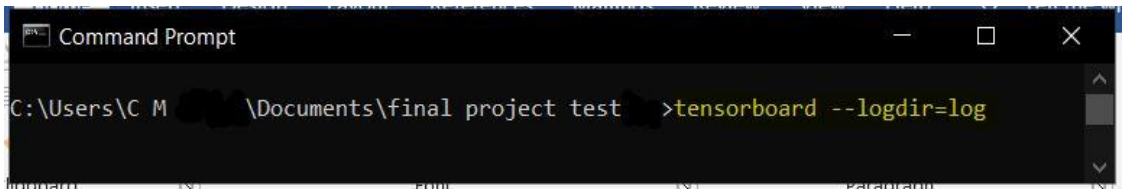


Fig 7.2 TensorBoard Command Line Interface

The accuracy and loss are visualized as follows:

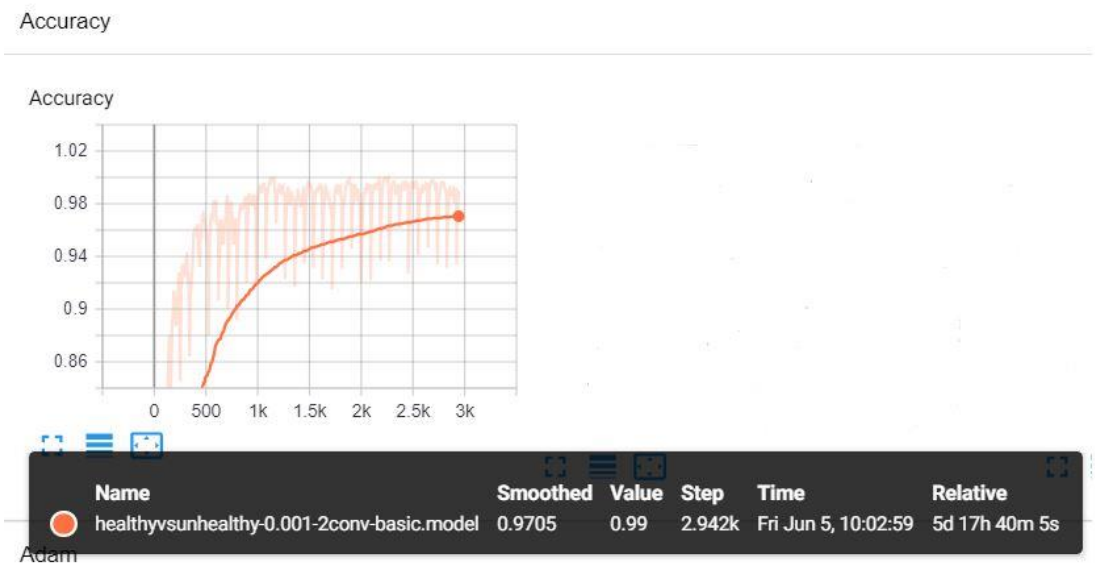


Fig 7.3 The Accuracy averaging to 99%

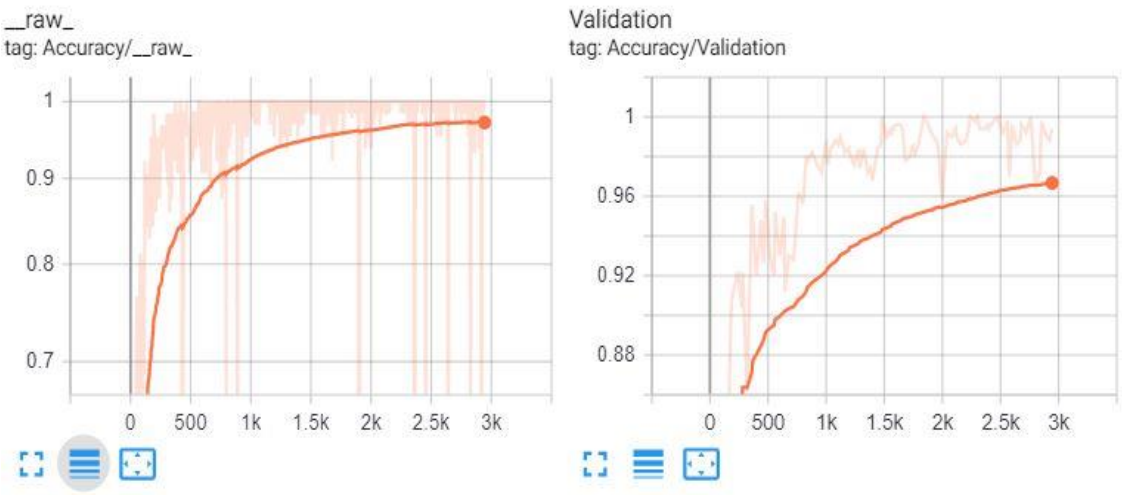


Fig 7.4 The raw and validation accuracy graphs

Fig 7.5 The Loss averaging to 1.88%

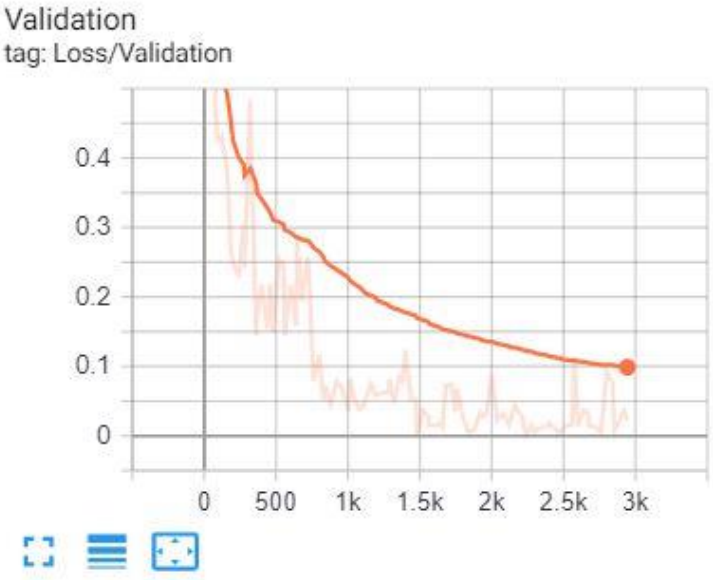


Fig 7.6 The validation loss graph

CHAPTER 8

CONCLUSION

This paper gives the executed results on different diseases classification techniques that can be used for plant leaf disease detection and an algorithm for image segmentation technique used for Automatic detection as well as classification of plant leaf diseases which has been described later. Tomato, Pepper Bell, Potato and Raspberry are the four species on which proposed algorithm was tested. Therefore, related diseases for these plants were taken for identification. With very less computational efforts the optimum results were obtained, which also shows the efficiency of the proposed algorithm in recognition and classification of the leaf diseases. Another advantage of using this method is that the plant diseases can be identified at an early stage or the initial stage.

The proposed system was developed taking in mind the benefits of the farmers and agricultural sector. The developed system can detect disease in plant and also provide the remedy that can be taken against the disease. By proper knowledge of the disease and the remedy can be taken for improving the health of the plant. The proposed system is based on python and gives an accuracy of more than 90%.

As a future work, the system can be installed on Drones so that aerial surveillance of crop fields can be done.

REFERENCES

[1] Dhiman Mondal, Dipak Kumar Kole, Aruna Chakraborty, D. Dutta Majumder" Detection and Classification Technique of Yellow Vein Mosaic Virus Disease in Okra Leaf Imagesusing Leaf Vein Extraction and Naive Bayesian Classifier., 2015, International Conference on Soft Computing Techniques and Implementations- (ICSCTI) Department of ECE, FET, MRIU, Faridabad, India, Oct 8-10, 2015.

[2] Pranjali B. Padol, Prof. AnjilA.Yadav, "SVM Classifier Based Grape Leaf Disease Detection" 2016 Conference on Advances in Signal Processing(CAPS) Cummins college of Engineering for Women, Pune. June 9-11, 2016.

[3] Detecting jute plant disease using image processing and machine learning 2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)

[4] Tejoindhi M.R, Nanjesh B.R, JagadeeshGujanuru Math, AshwinGeetD'sa "Plant Disease Analysis Using Histogram Matching Based On Bhattacharya's Distance Calculation" International Conference on Electrical, Electroniocs and Optimization Techniques(ICEEOT)-2016

[5] Detection of unhealthy plant leaves using image processing and genetic algorithm with Arduino2018 International Conference on Power, Signals, Control and Computation (EPSCICON)

[6] Tanvimehera, vinaykumar,pragyagupta "Maturity and disease detection in tomato using computer vision" 2016 Fourth international conference on parallel, distributed and grid computing(PDGC)

[7] Ms.Poojapawer ,Dr.varshaTukar, prof.parvinpatil "Cucumber Disease detection using artificial neural network"

[8] Mukesh Kumar Tripathi, Dr.Dhananjay, D.Maktedar" Recent Machine Learning Based Approaches for Disease Detection and Classification of Agricultural Products" International Conference on Electrical, Electronics and Optimization Techniques (ICEEOT)-2016.

[9] Detection of leaf diseases and classification using digital image processing2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)