

Plant Disease Identification using Convolutional Neural Network

**Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in**

Computer Science and Engineering

By

ABHINAV SINHA

(Reg. no. 37110005)

RATNADIP NATH

(Reg. no. 37110625)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119 TAMILNADU.

MARCH 2021



SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(Established under Section 3 of UGC Act, 1956)

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119

www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **RATNADIP NATH (Reg. No. 37110625)** and **ABHINAV SINHA (Reg. No. 37110005)** who carried out the project entitled **“Plant Disease Identification using Convolutional Neural Network”** under our supervision from **December 2020 to March 2021**.

Internal Guide

Mrs. M.D. Anto Praveena. M.C.A.,M.E.,(Ph.D).

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, RATNADIP NATH (Reg. No. 37110625) hereby declare that the Project Report on **“Plant Disease Identification using Convolutional Neural Network”** done by me under the guidance of **Mrs. M.D. Anto Praveena, M.C.A,M.E,(Ph.D)** at Sathyabama Institute of Science and Technology (Deemed to be University) is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D** , Dean of school of computing for providing me necessary support and details at the right time during the progressive reviews.

I convey my thanks to **Dr. S. Vigneshwari M.E., Ph.D** , Head of the Department, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mr. R. VIGNESH, M.E(Ph.D)** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of COMPUTER SCIENCE AND ENGINEERING who were helpful in many ways for the completion of the project.

ABSTRACT

When plants and crops are affected by pests it affects the agricultural production of the society and the country as a whole. Usually any disease is identified and detected by farmers and other personnel using their mere naked eyes. This forementioned event is time consuming, big fat-budgeted and inaccurate at the same time. On the other hand, automatic detection using high-tech machines and image processing techniques provide us with fast and more accurate results. The project “Plant Disease Identification using CNN” is concerned with a new approach to the development of plant disease recognition model, based on leaf image classification, by the use of deep convolutional networks. Advances in computer vision present an opportunity to enhance and widen the practice of precise plant protection and extend the market of computer vision and application in the field of agriculture.

This method paper is a new approach in detecting plant diseases using the deep convolutional neural network trained and fine-tuned to fit accurately to the database of a plant's leaves that was gathered independently for diverse plant diseases. The advance and novelty of the developed model lie in its simplicity; healthy leaves and background images are in line with other classes, enabling the model to distinguish between diseased leaves and healthy ones or from the environment by using CNN.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF ABBREVIATIONS	ix
	LIST OF FIGURES	x
1	INTRODUCTION	1
1.1	ABOUT NEURAL NETWORK	1
1.2	DESIGN	2
1.3	THE PURPOSE OF CNN	4
2.	AIM AND SCOPE	6
2.1	AIM	6
2.2	PROBLEM STATEMENT	6
2.3	SCOPE	7
2.4	ADVANTAGE	7
3.	ALGORITHMS AND METHODS	8
3.1	GENERAL	8
3.2	ALGORITHM	8
3.3	PROJECT ARCHITECTURE	9
4.	SYSTEM IMPLEMENTATION	12
4.1	REQUIREMENTS	12

4.2	MODULES	12
4.3	RESULTS AND DISCUSSIONS	13
5.	CONCLUSION AND FUTURE WORK	14
5.1	CONCLUSION	14
5.2	FUTURE WORK	14
	REFERENCES	15
	APPENDIX	17
A	SOURCE CODE	17
B	SCREENSHOTS	22
C	OUTPUT	26

LIST OF FIGURES

FIGURE NO.	NAME	PAGE NO.
1.1	SIMPLE NEURAL NETWORK	1
1.2	ARCHITECTURE OF NEURAL NETWORK	2
1.3	ARCHITECTURE OF LeNet-5	3
1.4	ARCHITECTURE OF ALEXNET	3
1.5	ARCHITECTURE OF GOOGLNET	4
1.6	FILTER APPLIED TO A TWO-DIMENSIONAL INPUT	4
2.1	NEURAL NETWORK OPERATION	7
3.1	IMAGE PREPROCESSING	9
3.2	CNN LAYERS	10
B.1	SCREENSHOT OF ACTUAL CODE	25
B.2	USER INPUT SCREEN	26

CHAPTER 1

INTRODUCTION

1.1 ABOUT NEURAL NETWORKS:

A neural network works similarly to the human brain's neural network. A “neuron” in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

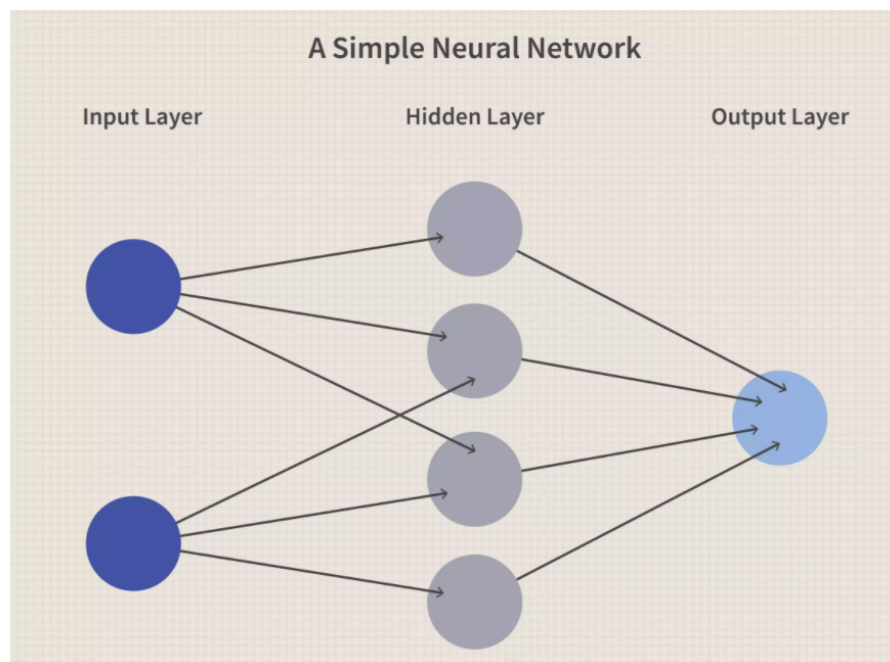


FIG 1.1 SIMPLE NEURAL NETWORK

The Fig 1.1 is an example of a neural network and how it is interconnected with the other node.

1.2 DESIGN AND TYPES OF NEURAL NETWORK:

ARCHITECTURE:

A Convolutional Neural Network (CNN) is a deep learning algorithm that can recognize and classify features in images for computer vision. It is a multi-layer neural network designed to analyze visual inputs and perform tasks such as image classification, segmentation and object detection, which can be useful for autonomous vehicles. CNNs can also be used for deep learning applications in healthcare, such as medical imaging.

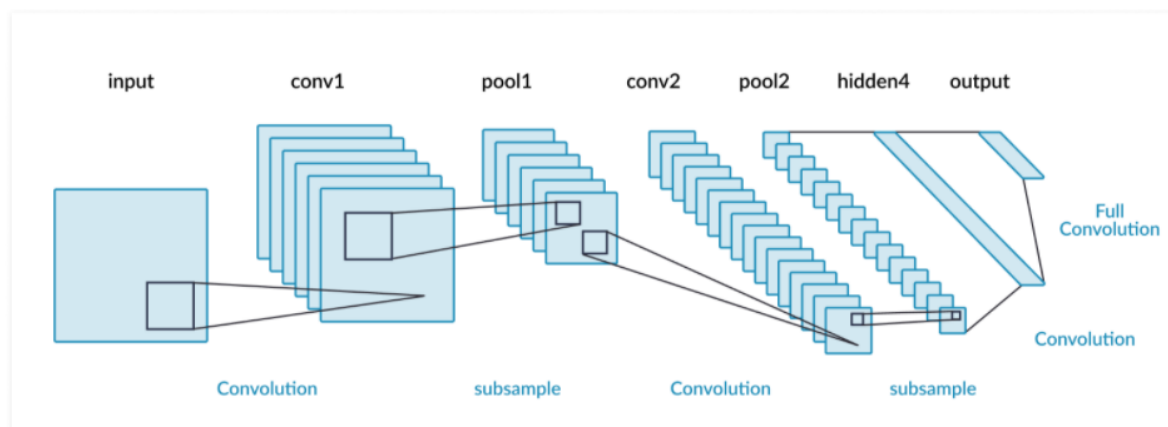


FIG 1.2 ARCHITECTURE OF A NEURAL NETWORK

There are two main parts to a CNN:

- A convolution tool that splits the various features of the image for analysis
- A fully connected layer that uses the output of the convolution layer to predict the best description for the image.

CNN architecture is inspired by the organization and functionality of the visual cortex and designed to mimic the connectivity pattern of neurons within the human brain. The neurons within a CNN are split into a three-dimensional structure, with each set of neurons analyzing a small region or feature of the image. In other words, each group of neurons specializes in identifying one part of the image. CNNs use the

predictions from the layers to produce a final output that presents a vector of probability scores to represent the likelihood that a specific feature belongs to a certain class.

DIFFERENT ARCHITECTURES:

There are many popular CNN architectures, many of them gained recognition by achieving good results at the ILSVRC.

LeNet-5 (1998):

This 7-layer CNN classified digits, digitized 32×32 pixel greyscale input images. it was used by several banks to recognize the hand-written numbers on checks.

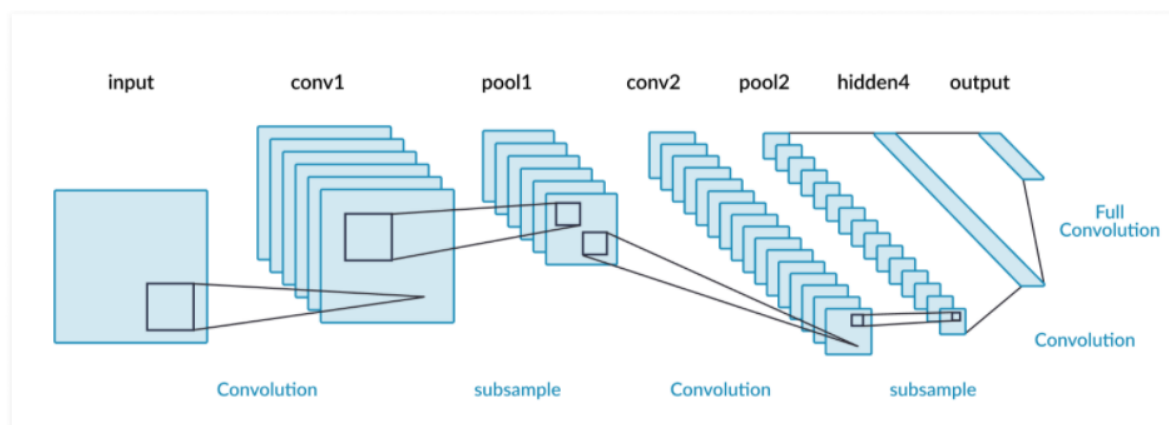


FIG 1.3 ARCHITECTURE OF LeNet-5

AlexNet (2012):

AlexNet is designed by SuperVision group, with a similar architecture to LeNet, but deeper – it has more filters per layer as well as stacked convolutional layers. It is composed of five convolutional layers followed by three fully connected layers.

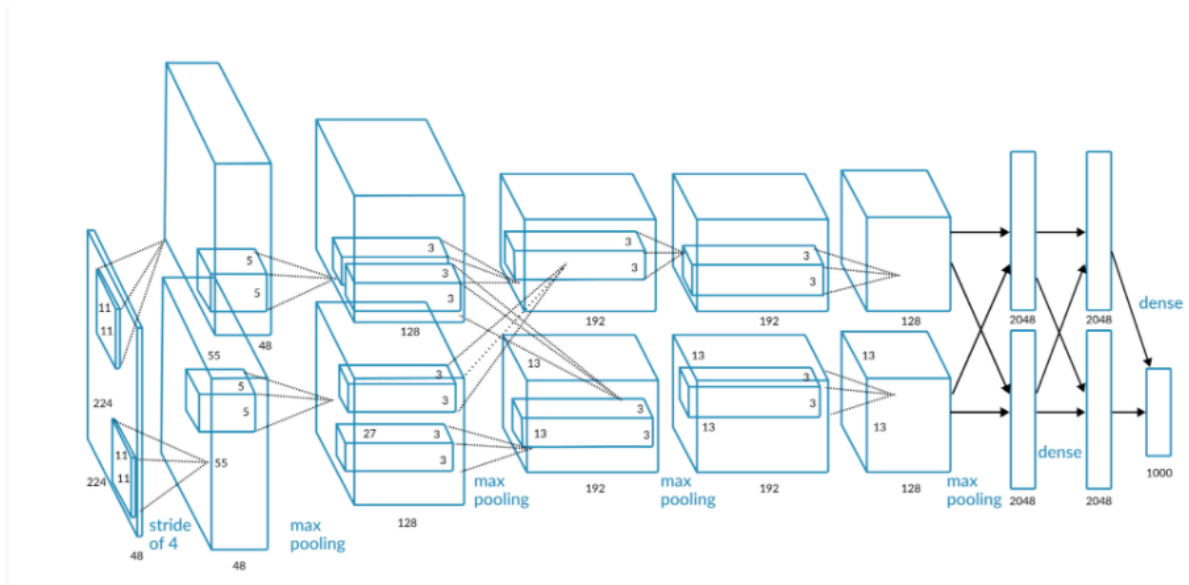


FIG 1.4 ARCHITECTURE OF ALEXNET

GoogleNet (2014):

GoogleNet architecture consisted of a 22 layer deep CNN used a module based on small convolutions, called “inception module”, which used batch normalization, RMSprop and image to reduce the number of parameters from 60 million like in AlexNet to only 4 million.

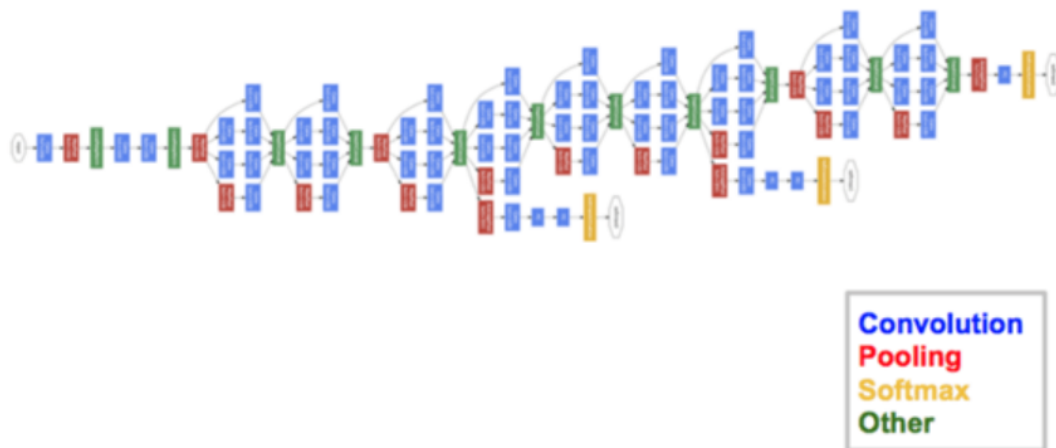


FIG 1.5 ARCHITECTURE OF GOOGLNET

1.3 THE PURPOSE OF CNN:

Convolution in Convolutional Neural Networks:

Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a “*convolution*”.

In the context of a convolutional neural network, a convolution is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network.

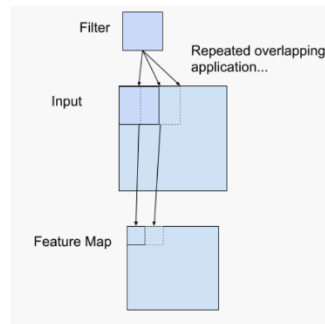


FIG 1.6 FILTER APPLIED TO A TWO-DIMENSIONAL INPUT.

Convolution in Computer Vision:

The idea of applying the convolutional operation to image data is not new or unique to convolutional neural networks; it is a common technique used in computer vision. Historically, filters were designed by hand by computer vision experts, which were then applied to an image to result in a feature map or output from applying the filter then makes the analysis of the image easier in some way.

Convolutional neural networks do not learn a single filter; they, in fact, learn multiple features in parallel for a given input.

For example, it is common for a convolutional layer to learn from 32 to 512 filters in parallel for a given input. This gives the model 32, or even 512, different ways of extracting features from an input, or many different ways of both “*learning to see*” and after training, many different ways of “*seeing*” the input data.

CHAPTER 2

AIM AND SCOPE

2.1 AIM:

The main objective is to find whether the plant or the leaf is diseased or not on the basis of its images and help them get cured before it is too late.

It is crucial to prevent unnecessary waste of financial and other resources, thus achieving healthier production in this changing environment.

The scope varies from trained professionals to the novice people who are related to agriculture.

2.2 PROBLEM STATEMENT:

Identifying and detecting a diseased plant or leaf requires much time if we go for it manually. The chances of the result being accurate also diminishes if not done by any expert. Thus, advanced technologies and algorithms come into action in such times. Detecting and identifying the diseases using advanced imaging techniques increase the accuracy and hence helps to boost up the agricultural produce.

The problem statement can be understood from a small example:

Suppose a farmer is observing for health condition of plants in his farmland. But with mere naked eyes, he is facing difficulties in order to get a clear idea about the particular disease. As such the result he is getting is biased and inaccurate. How can this problem be solved?

The answer is by using Neural Network.

If advanced imaging techniques are used in addition to the algorithm formed by connecting neurons, a much better result is obtained. Continuous scanning and filtering of the leaf images results in enhanced accuracy in the deep learning model.

Thus, this makes the whole process much more easy and hassle free.

The Figure 2 shows the process of fetching of data from the feeded image.

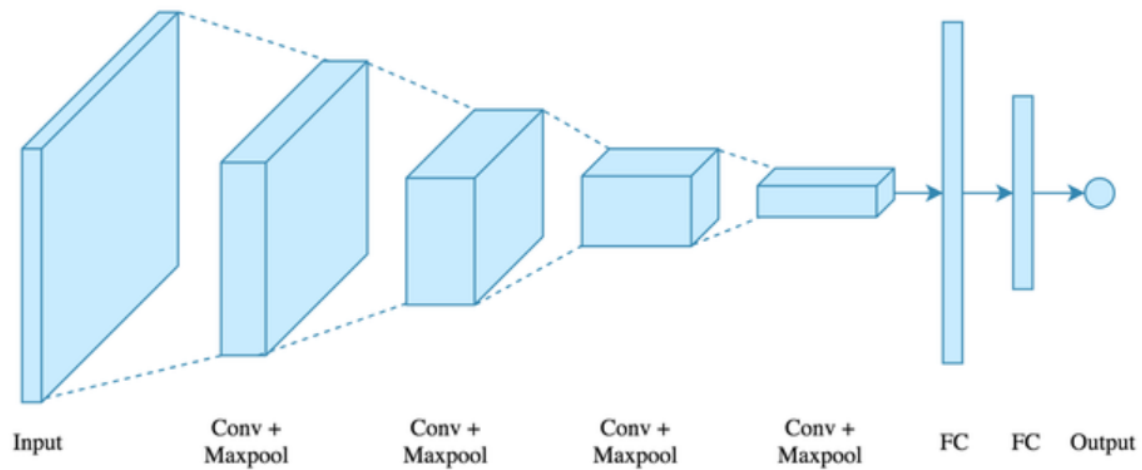


FIG 2.1 NEURAL NETWORK OPERATION.

2.3 SCOPE:

At an earlier stage of the project, the image classification and pre-processing is carried out, based on which the further algorithm is developed. The implication of deep neural networks makes the solution feasible and accessible to one and all. The accuracy is more than any previous solution and classification algorithm encountered.

2.4 ADVANTAGES:

The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs it learns distinctive features for each class by itself. CNN is also computationally efficient.

CHAPTER 3

ALGORITHMS AND METHODS

3.1 GENERAL:

In this project, the plant diseases are identified and detected based on image classification of plant leaves, which helps to reduce the disease and hence increase the agricultural output. The methodology followed is quite simple. Basic elements of machine learning and deep learning are used throughout the whole project.

Numpy-NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

Pickle-The pickle module can transform a complex object into a byte stream and it can transform the byte stream into an object with the same internal structure.

Keras-Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3 Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit.

Label Binarizer-It assigns a unique value or number to each label in a categorical feature.

Matplotlib-Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

3.2 ALGORITHM:

1. Open any open-source notebook like Jupyter or Google Colab.
2. Download the dataset from Kaggle consisting on leaf samples.
3. Import libraries like numpy, pickle, cv2, os, matplotlib, Keras.

4. Import scikit learn packages like preprocessing and LabelBinarizer.
5. Import and define various convolutional layer elements like Conv2D, MaxPooling2D.

6. Initialize a few parameters required for the image dataset preprocessing.
7. Load the dataset.
8. Split the data into training and test sets for validation purpose.
9. Define the hyperparameters of the plant disease classification model like epochs, steps, batch_size.
10. Create a sequential model and adding Convolutional, Normalization, Pooling, Dropout and Activation layers at the appropriate positions.
11. Initialize Adam optimizer with learning rate and decay parameters.
12. Compare the accuracy and loss by plotting the graph for training and validation.
13. Evaluate model accuracy by using the **evaluate** method.
14. Save the model for future purposes.
14. For testing purposes, randomly choose images from the dataset and try predicting class or disease of the plant image.
15. Detection of disease is completed.
16. Exit the notebook.

3.3 PROJECT ARCHITECTURE:



FIG 3.1 IMAGE PREPROCESSING

First the image is feeded into the system through the dataset. The images are converted into array in order to access them seamlessly. Next all the duplicate

photos are removed, and filters are applied so as to create a better and efficient dataset. Finally, the first phase, that is dataset is loaded so as to carry out further operations.

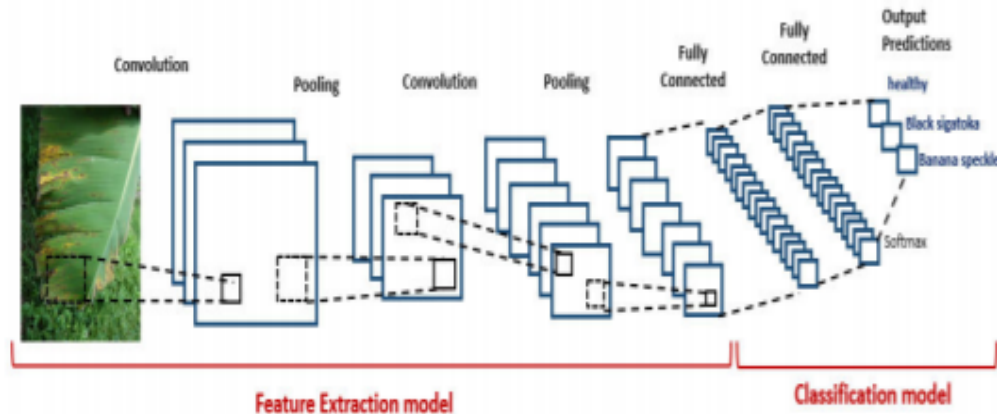


FIG 3.2 CNN LAYERS

Convolution Operation:

First Layer:

Input to Convolutional layer:

The image is resized to an optimal size and is fed as input to the convolutional layer. Let us consider the input as $32 \times 32 \times 3$ array of pixel values.

There exists a filter or neuron or kernel which lays over some of the pixels of the input image depending on the dimensions of the Kernel size. Let the dimensions of the kernel of the filter be $5 \times 5 \times 3$.

The Kernel actually slides over the input image, thus it is multiplying the values in the filter with the original pixel values of the image (aka computing element-wise multiplications).

The multiplications are summed up generating a single number for that particular receptive field and hence for sliding the kernel a total of 784 numbers are mapped to 28×28 array known as the feature map.

However, when we're talking about the 2nd conv. layer, the input is the activation map(s) that result from the first layer. So each layer of the input is basically describing the locations in the original image for where certain low-level features appear.

Now when you apply a set of filters on top of that (pass it through the 2nd conv. layer), the output will be activations that represent higher-level features. Types of these features could be semicircles (a combination of a curve and straight edge) or squares (a combination of several straight edges). As you go through the network and go through more conv. layers, you get activation maps that represent more and more complex features.

By the end of the network, you may have some filters that activate when there is anomalies in the image, filters that activate when they see disease features.

Pooling Operation.

A pooling layer is another building block of a CNN. Pooling. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently.

Fully Connected layer:

The way this fully connected layer works is that it looks at the output of the previous layer (which as we remember should represent the activation maps of high-level features) and the number of classes N (10 for digit classification).

For example, if the program is predicting that some image is a dog, it will have high values in the activation maps that represent high-level features like a paw or 4 legs, etc. Basically, an FC layer looks at what high level features most strongly correlate to a particular class and has particular weights so that when you compute the products between the weights and the previous layer, you get the correct probabilities for the different classes.

CHAPTER 4

SYSTEM IMPLEMENTATION:

4.1 REQUIREMENTS:

I. HARDWARE REQUIREMENTS:

A system with minimum 4 GB RAM and 50 GB free disk space.

Keyboard.

Monitor.

II. SOFTWARE REQUIREMENTS:

Python 3.6 installed in the system.

If using native notebook, then Modules should be pre-installed.

Stable internet connection of 2 MBPS if using online open-source notebook like Google Colab.

4.2 MODULES:

PYTHON:

Python is a popular, general-purpose programming language with an emphasis on being readable and allowing programmers to use fewer lines of code to accomplish tasks than in older languages. Libraries such as NumPy, SciPy, Keras and Matplotlib make it useful for scientific computing.

OS:

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. The `*os*` and `*os.path*` modules include many functions to interact with the file system.

NumPy-NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

Pickle-The pickle module can transform a complex object into a byte stream and it can transform the byte stream into an object with the same internal structure.

Keras-Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3 Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit.

Label Binarizer-It assigns a unique value or number to each label in a categorical feature.

Matplotlib-Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

4.3 RESULTS AND DISCUSSION:

The result of the project is the classified image and the accuracy graph. The image is classified based on the nature and type of the disease occurred. Healthy leaf gets displayed as a healthy one and the diseased leaf gets displayed as a diseased one with the particular disease mentioned beforehand. A graph is plotted based on the accuracy of the training model, which shows training loss and validation loss as per the epochs. The results are far far better than what can be gained by simple observation with the naked eye.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION:

This application is very simple. It is easily understandable to the users who have knowledge in Python and basics of deep learning neural networks. This project will be useful for the farmers and other organizations who deal in the agricultural aspect of the society and form a major portion of the economy. The model developed can be used flawlessly for any research purpose related to leaf diseases.

5.2 FUTURE WORK:

This project can be further improved by making the project a native windows and mac application. The project can also be improved by adding API's which provides the user with ease of access anytime anywhere just a fingertip away. The project can be made real-time by adding live photo feeder, such that whenever an image is encountered, the model can tell whether the leaf image loaded is a healthy leaf or a diseased one.

REFERENCES

- [1] Aakanksha Rastogi, Ritika Arora, Shanu Sharma, "Leaf Disease Detection and Grading using Computer Vision Technology &Fuzzy Logic," presented at the 2nd International Conference on Signal Processing and Integrated Networks (SPIN), IEEE, 2015, pp. 500–505
- [2] Garima Tripathi, Jagruti Save, "AN IMAGE PROCESSING AND NEURAL NETWORK BASED APPROACH FOR DETECTION AND CLASSIFICATION OF PLANT LEAF DISEASES," Int. J. Comput. Eng. Technol. IJCET, vol. 6, no. 4, pp. 14–20, Apr. 2015
- [3] S. Arivazhagan, R. Newlin Shebiah, S. Ananthi, S. Vishnu Varthini, "Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features," Agric Eng Int CIGR J., vol. 15, no. 1, pp. 211–217, Mar. 2013.
- [4] Prof. Sanjay B. Dhaygude, Mr.NitinP.Kumbhar, "Agricultural plant Leaf Disease Detection Using Image Processing"IJAREEIE, vol. 2(1),pp. 599-602, January 2013
- [5] K. Muthukannan, P. Latha, R. PonSelvi and P. Nisha, "CLASSIFICATION OF DISEASED PLANT LEAVES USING NEURAL NETWORK ALGORITHMS," ARPN J. Eng. Appl. Sci., vol. 10, no. 4, pp. 1913–1918, Mar. 2015.
- [6] Md. Nazrul Islam, M.A. Kashem, MahmudaAkter and Md. JamilurRahman, "An Approach to Evaluate Classifiers for Automatic Disease Detection and Classification of Plant Leaf," presented at the International Conference on Electrical, Computer and Telecommunication Engineering, RUET, Rajshahi-6204, Bangladesh, 2012, pp. 626–629.

[7] Hrishikesh Kanjalkar P. and Prof. Lokhande S. 2013. Detection and Classification of Plant Leaf Diseases using ANN. International Journal of Scientific & Engineering Research. ISSN: 2229 5518.

[8] Suhaili Beeran kutty. and Noor ezan Abdullah. 2013. 'Classification of Watermelon Leaf Diseases Using Neural Network Analysis', IEEE Business Engineering and Industrial Applications Colloquium (BEIAC). International Journal of Engineering Science and Computing, March 2017 5328 <http://ijesc.org/>

APPENDIX

A. SOURCE CODE:

```
# Download a file based on its file ID.
file_id = '18DbC6Xj4NP-hLzI14WuMaAEyq482vNfn'

# Download dataset
!gdown https://drive.google.com/uc?id={file_id}

# Unzip the downloaded file
!unzip -q PlantVillage.zip

import numpy as np
import pickle
import cv2
import os
import matplotlib.pyplot as plt
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

# Dimension of resized image
DEFAULT_IMAGE_SIZE = tuple((256, 256))

# Number of images used to train the model
N_IMAGES = 100

# Path to the dataset folder
root_dir = './PlantVillage'

train_dir = os.path.join(root_dir, 'train')
val_dir = os.path.join(root_dir, 'val')
```

```

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 ...")
    plant_disease_folder_list = listdir(train_dir)

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

        for image in plant_disease_image_list[:N_IMAGES]:
            image_directory = f"{train_dir}/{plant_disease_folder}/{image}"
            if image_directory.endswith(".jpg")==True or image_directory.endswith(".JPG"
)==True:
                image_list.append(convert_image_to_array(image_directory))
                label_list.append(plant_disease_folder)

    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

# Transform the loaded training image data into numpy array
np_image_list = np.array(image_list, dtype=np.float16) / 225.0
print()

# Check the number of images loaded for training
image_len = len(image_list)
print(f"Total number of images: {image_len}")

label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)

```

```

pickle.dump(label_binarizer,open('plant_disease_label_transform.pkl', 'wb'))
n_classes = len(label_binarizer.classes_)

print("Total number of classes: ", n_classes)

augment = 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")

print("[INFO] Splitting data to train and test...")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_size=0.2, random_state = 42)

EPOCHS = 25
STEPS = 100
LR = 1e-3
BATCH_SIZE = 32
WIDTH = 256
HEIGHT = 256
DEPTH = 3

model = Sequential()
inputShape = (HEIGHT, WIDTH, DEPTH)
chanDim = -1

if K.image_data_format() == "channels_first":
    inputShape = (DEPTH, HEIGHT, WIDTH)
    chanDim = 1

model.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))

```

```

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

model.summary()

# Initialize optimizer
opt = Adam(lr=LR, decay=LR / EPOCHS)

# Compile model
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# Train model
print("[INFO] Training network...")
history = model.fit_generator(augment.flow(x_train, y_train, batch_size=BATCH_SIZE),
                             validation_data=(x_test, y_test),
                             steps_per_epoch=len(x_train) // BATCH_SIZE,
                             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 accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and Validation accuracy')
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()

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

# Dump pickle file of the model
print("[INFO] Saving model...")
pickle.dump(model, open('plant_disease_classification_model.pkl', 'wb'))

# Dump pickle file of the labels
print("[INFO] Saving label transform...")
filename = 'plant_disease_label_transform.pkl'
image_labels = pickle.load(open(filename, 'rb'))

def predict_disease(image_path):
    image_array = convert_image_to_array(image_path)
    np_image = np.array(image_array, dtype=np.float16) / 225.0
    np_image = np.expand_dims(np_image, 0)
    plt.imshow(plt.imread(image_path))
    result = model.predict_classes(np_image)
    print((image_labels.classes_[result][0]))

predict_disease('/content/PlantVillage/val/Blueberry___healthy/008c85d0-a954-4127-
bd26-861dc8a1e6ff___RS_HL_2431.JPG')

```

***THE SOURCE CODE OF THE PROJECT HAS BEEN MENTIONED ABOVE**

B. SCREENSHOTS:

```
# Download a file based on its file ID.
file_id = '18DbC6Xj4NP-hLzI14WuMaAEyq482vNfn'

# Download dataset
!gdown https://drive.google.com/uc?id={file_id}

# Unzip the downloaded file
!unzip -q PlantVillage.zip
```

Downloading...

From: <https://drive.google.com/uc?id=18DbC6Xj4NP-hLzI14WuMaAEyq482vNfn>
To: /content/PlantVillage.zip
866MB [00:05, 148MB/s]

Importing necessary libraries and modules required to build the classification model.

```
[ ] import numpy as np
import pickle
import cv2
import os
import matplotlib.pyplot as plt
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
```

```

# Dimension of resized image
DEFAULT_IMAGE_SIZE = tuple((256, 256))

# Number of images used to train the model
N_IMAGES = 100

# Path to the dataset folder
root_dir = './PlantVillage'

train_dir = os.path.join(root_dir, 'train')
val_dir = os.path.join(root_dir, 'val')

```

We use the function `convert_image_to_array` to resize an image to the size `DEFAULT_IMAGE_SIZE` we defined above.

```

[ ] 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)

```

```

▶ image_list, label_list = [], []

try:
    print("[INFO] Loading images ...")
    plant_disease_folder_list = listdir(train_dir)

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

        for image in plant_disease_image_list[:N_IMAGES]:
            image_directory = f"{train_dir}/{plant_disease_folder}/{image}"
            if image_directory.endswith(".jpg")==True or image_directory.endswith(".JPG")==True:
                image_list.append(convert_image_to_array(image_directory))
                label_list.append(plant_disease_folder)

    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

# Transform the loaded training image data into numpy array
np_image_list = np.array(image_list, dtype=np.float16) / 225.0
print()

```

▼ Augment and Split Dataset

Using `ImageDataGenerator` to augment data by performing various operations on the training images.

```

[ ] augment = 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")

```

Splitting the data into training and test sets for validation purpose.

```

[ ] print("[INFO] Splitting data to train and test...")
    x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_size=0.2, random_state = 42)

[INFO] Splitting data to train and test...

```


Build Model

Defining the hyperparameters of the plant disease classification model.

```
[ ] EPOCHS = 25
    STEPS = 100
    LR = 1e-3
    BATCH_SIZE = 32
    WIDTH = 256
    HEIGHT = 256
    DEPTH = 3
```

```
model = Sequential()
inputShape = (HEIGHT, WIDTH, DEPTH)
chanDim = -1

if K.image_data_format() == "channels_first":
    inputShape = (DEPTH, HEIGHT, WIDTH)
    chanDim = 1

model.add(Conv2D(32, (3, 3), padding="same", input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))

model.summary()
```

```
# Initialize optimizer
opt = Adam(lr=LR, decay=LR / EPOCHS)

# Compile model
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# Train model
print("[INFO] Training network...")
history = model.fit_generator(augment.flow(x_train, y_train, batch_size=BATCH_SIZE),
                             validation_data=(x_test, y_test),
                             steps_per_epoch=len(x_train) // BATCH_SIZE,
                             epochs=EPOCHS,
                             verbose=1)
```

```
[INFO] Training network...
Epoch 1/25
97/97 [=====] - 792s 8s/step - loss: 0.0964 - accuracy: 0.9739 - val loss: 0.3557 - val accuracy
```



```

▶ 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 accuracy')
  plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
  plt.title('Training and Validation accuracy')
  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()

```

Evaluating model accuracy by using the `evaluate` method

```

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

```

[INFO] Calculating model accuracy
780/780 [=====] - 52s 66ms/step
Test Accuracy: 98.74754548072815

▼ Save Model

```

[ ] # Dump pickle file of the model
    print("[INFO] Saving model...")
    pickle.dump(model, open('plant_disease_classification_model.pkl', 'wb'))

[INFO] Saving model...

[ ] # Dump pickle file of the labels
    print("[INFO] Saving label transform...")
    filename = 'plant_disease_label_transform.pkl'
    image_labels = pickle.load(open(filename, 'rb'))

[INFO] Saving label transform...

```

We write the following `predict_disease` function to predict the class or disease of a plant image.

We just need to provide the complete path to the image and it displays the image along with its prediction class or plant disease.

```

[ ] def predict_disease(image_path):
    image_array = convert_image_to_array(image_path)
    np_image = np.array(image_array, dtype=np.float16) / 225.0
    np_image = np.expand_dims(np_image, 0)
    plt.imshow(plt.imread(image_path))
    result = model.predict_classes(np_image)
    print((image_labels.classes_[result][0]))

```

FIG. B.1 SCREENSHOT OF ACTUAL CODE

C.OUTPUT:

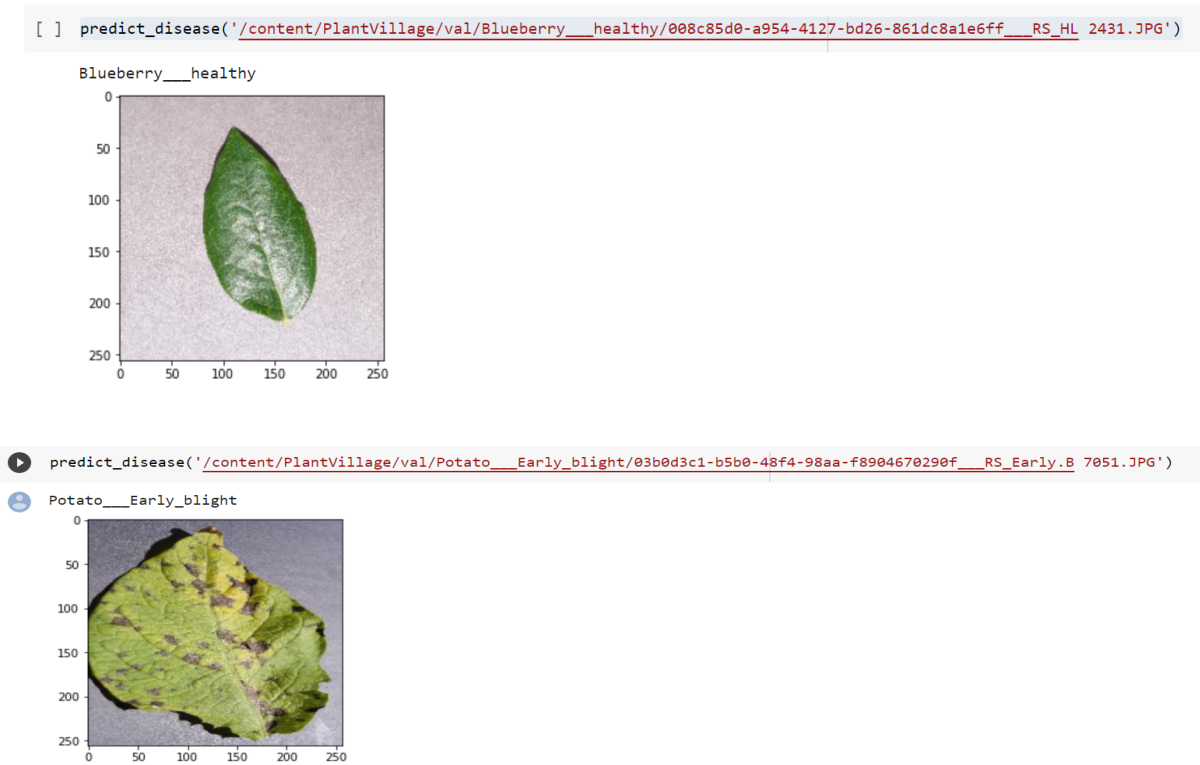


FIG B.2 USER INPUT SCREEN