PROJECT REPORT
ON
Plant Disease Classification using CNN (leaf images)

SUBMITTED BY :-

C S UNNIKRISHAN
PANKAJ
AKSHAY K BABY

UNDER GUIDANCE OF :-

Mrs.VIMALA MATHEW
(Scientist/Engineer 'D')

TABLE OF CONTENTS

1. ACKNOWLEDGEMENT

We wish to acknowledge our sincere gratitude to Mrs.Vimala Mathew (Scientist/Engineer ‘D’) for guiding us in various statges of the project.

2. ABSTRACT

Early detection of diseases in plants can help in taking the necessary remedial measure to save the plant. Inexperienced pesticide usage can cause the development of long-term resistance of the pathogens, severely reducing the ability to fight back. Timely and accurate diagnosis of plant diseases is one of the pillars of precision agriculture. It is crucial to prevent unnecessary waste of financial and other resources, thus achieving healthier production.

3. 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 the ability to fight back. Timely and accurate diagnosis of plant diseases is 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 effects 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.

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. An automated system designed to help identify plant diseases by the plant's appearance and visual symptoms could be of great help in disease diagnostics. 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. Convolution neural network (CNN) was used to classify input leaf image as either healthy or having disease. Exploiting common digital image processing techniques were used with the aim of detection and classification of plant diseases.

4. TOOLS USED

A computer with **Quadro P5000 GPU** was utilized for the entire process of training and testing the plant disease detection model. **Python** was the programming language used for the project.

Below were the python libraries used for the project :-

OpenCV : For Image processing

Keras : For building CNN layers

Numpy :To convert image to array

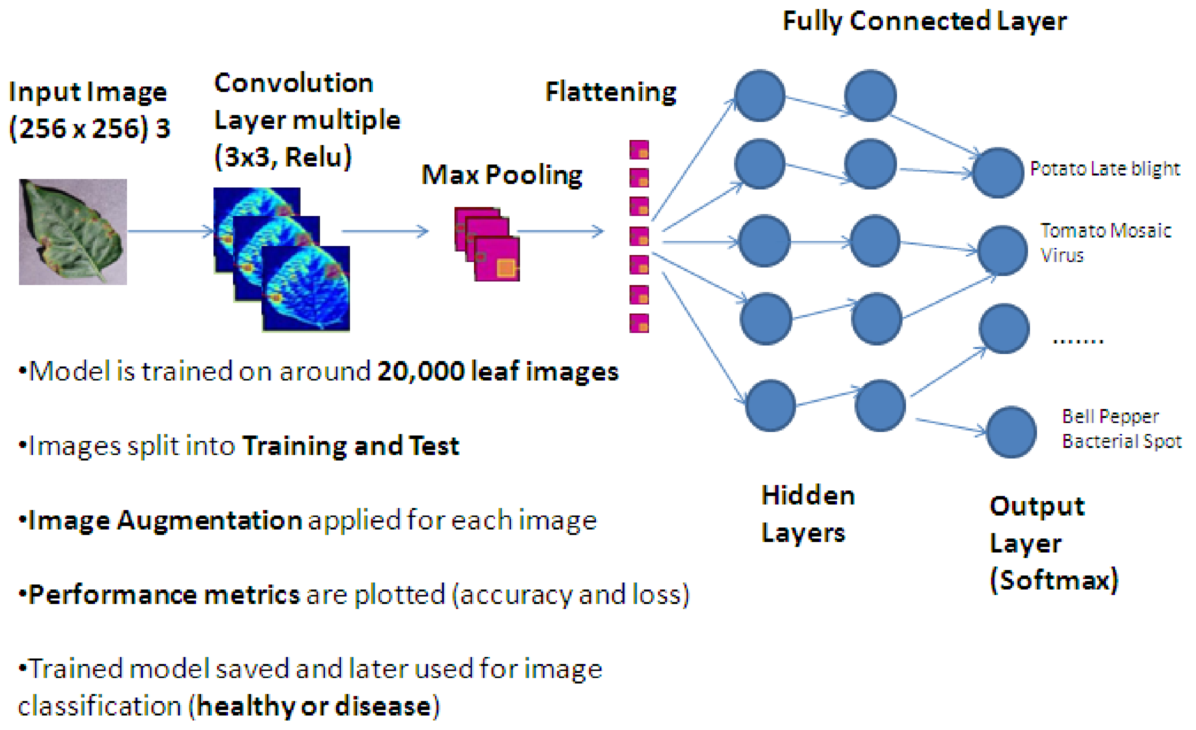
Sklearn :To convert categorical to numeric values

Matplotlib : To plot graphs

5.CONVOLUTIONAL NEURAL NETWORK (CNN)

CNNs consist of convolutional layers, which are sets of image filters convoluted to images or feature maps, along with other (e.g., pooling) layers. In image classification, feature maps are extracted through convolution and other processing layers repetitively and the network eventually outputs a label indicating an estimated class. Given a training dataset, CNN, unlike traditional machine learning techniques that use hand-crafted features , optimizes the weights and filter parameters in the hidden layers to generate features suitable to solve the classification problem. In principle, the parameters in the network are optimized by back-propagation and gradient descent approaches to minimize the classification error.

The convolutional layer is the essential building block of the convolutional neural network. The layer's parameters are comprised of a set of learnable kernels which possess a small receptive field but extend through the full depth of the input volume. Rectified Linear Units (ReLU) are used as substitute for saturating nonlinearities. This activation function adaptively learns the parameters of rectifiers and improves accuracy at negligible extra computational cost. It is defined aswhere represents the input of the nonlinear activation function on the th channel. Another important layer of CNNs is the pooling layer, which is a form of nonlinear downsampling. Pooling operation gives the form of translation invariance; it operates independently on every depth slice of the input and resizes it spatially.



•Model is trained on around **20,000 leaf images**

•Images split into **Training and Test**

•**Image Augmentation** applied for each image

•**Performance metrics** are plotted (accuracy and loss)

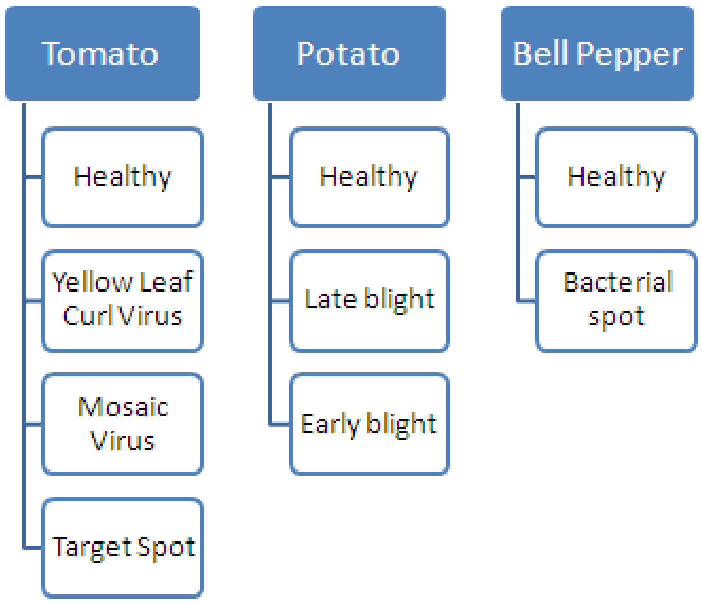
•Trained model saved and later used for image classification (**healthy or disease**)

6.DATASET

Appropriate datasets are required at all stages of object recognition research, starting from training phase to evaluating the performance of recognition algorithms. Plantvillage dataset uploaded in Kaggle was used for the current project.

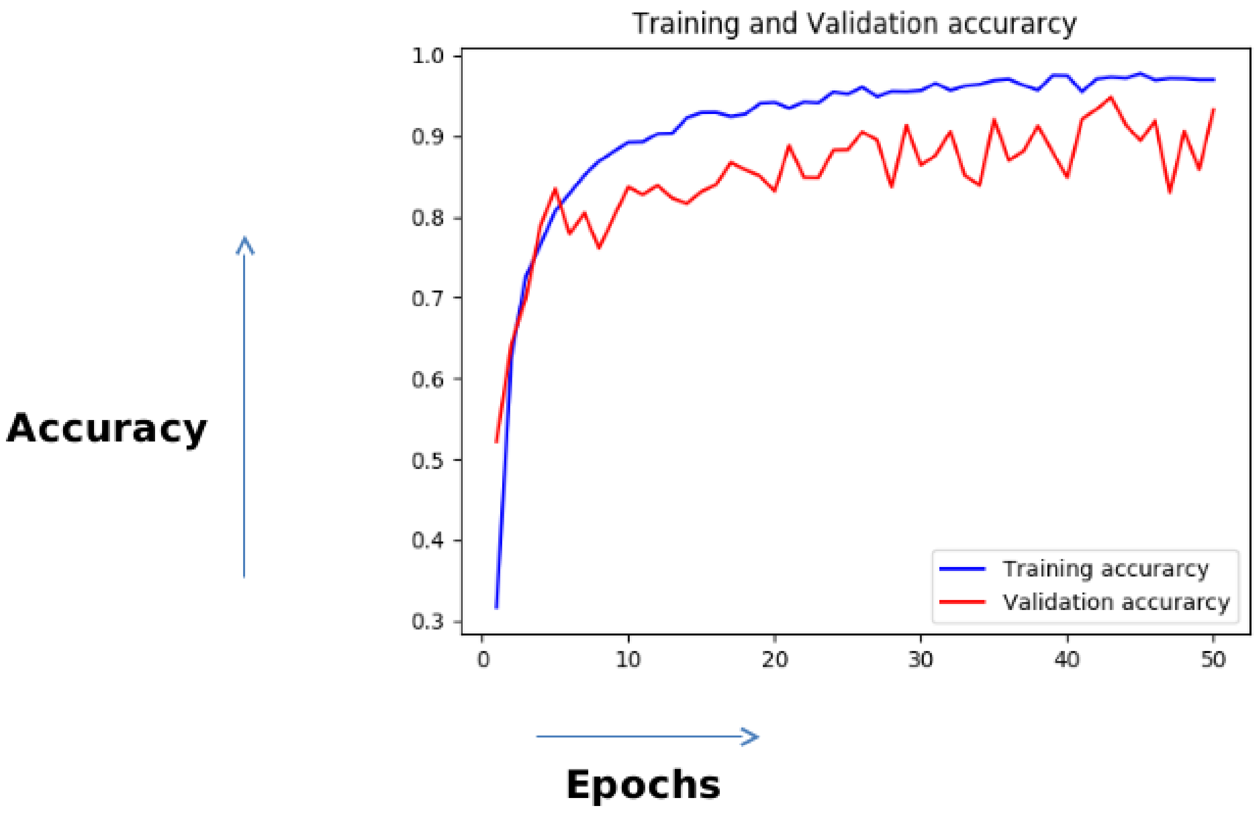
Next step was to enrich the dataset with augmented images. The main goal of the presented study is to train the network to learn the features that distinguish one class from the others. Therefore, when using more augmented images, the chance for the network to learn the appropriate features has been increased. The main purpose of applying augmentation is to increase the dataset and introduce slight distortion to the images which helps in reducing overfitting during the training stage. In machine learning, as well as in statistics, overfitting appears when a statistical model describes random noise or error rather than underlying relationship. The image augmentation contained one of several transformation techniques including horizontal flip, perspective transformation, and simple image rotations. For the augmentation process, simple image rotations were applied, as well as rotations on the different axis by various degrees.

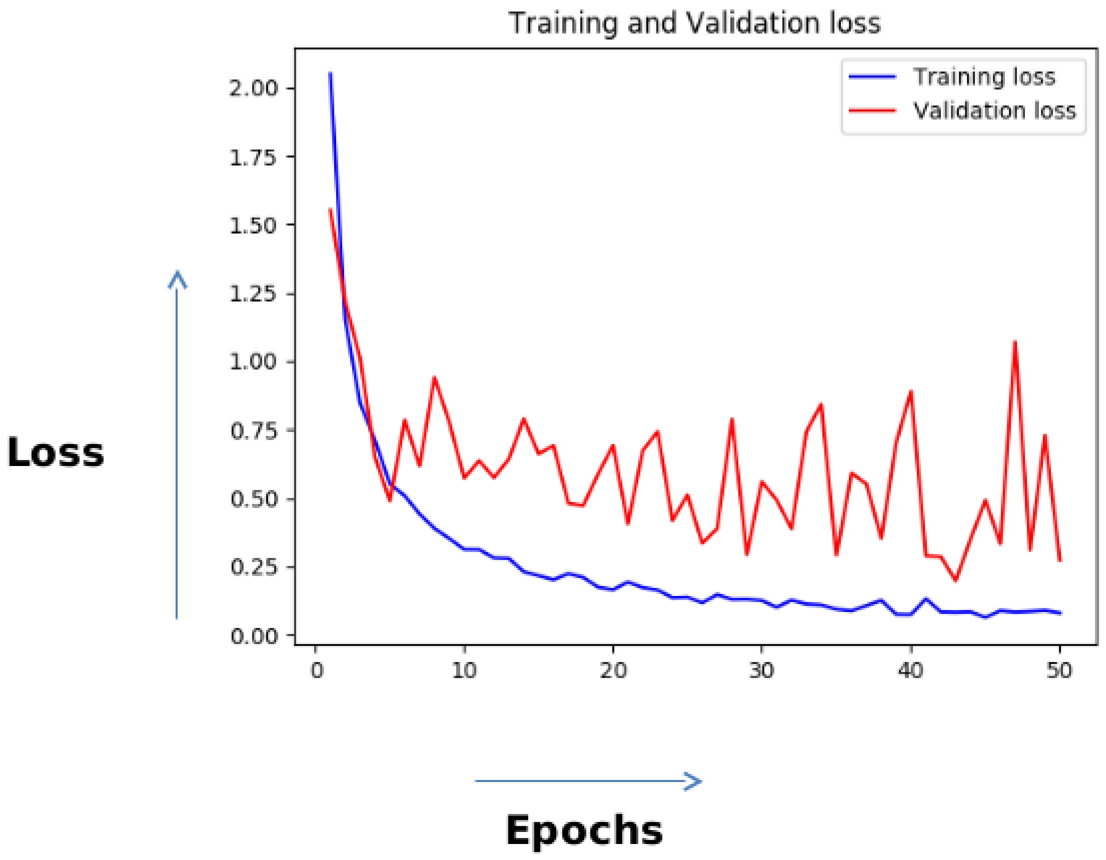
Below three plants and corresponding leaf images (healthy and disease) were in scope for the project.



7. TRAINING APPROACH

All the leaf images were placed in the corresponding folders and folder names were taken as the label names. Images from each folder were taken and converted to array of size 256x256 before giving it to the Convolution layer. Filters of size 3x3 were used for feature extraction. The image were split into training and test set. Image augmentation was applied to each image. Relu was the activation function used at Convolution layer and Softmax at the output layer. Model was fit and performance metrics were generated. Plots for Accuracy and Loss were plotted

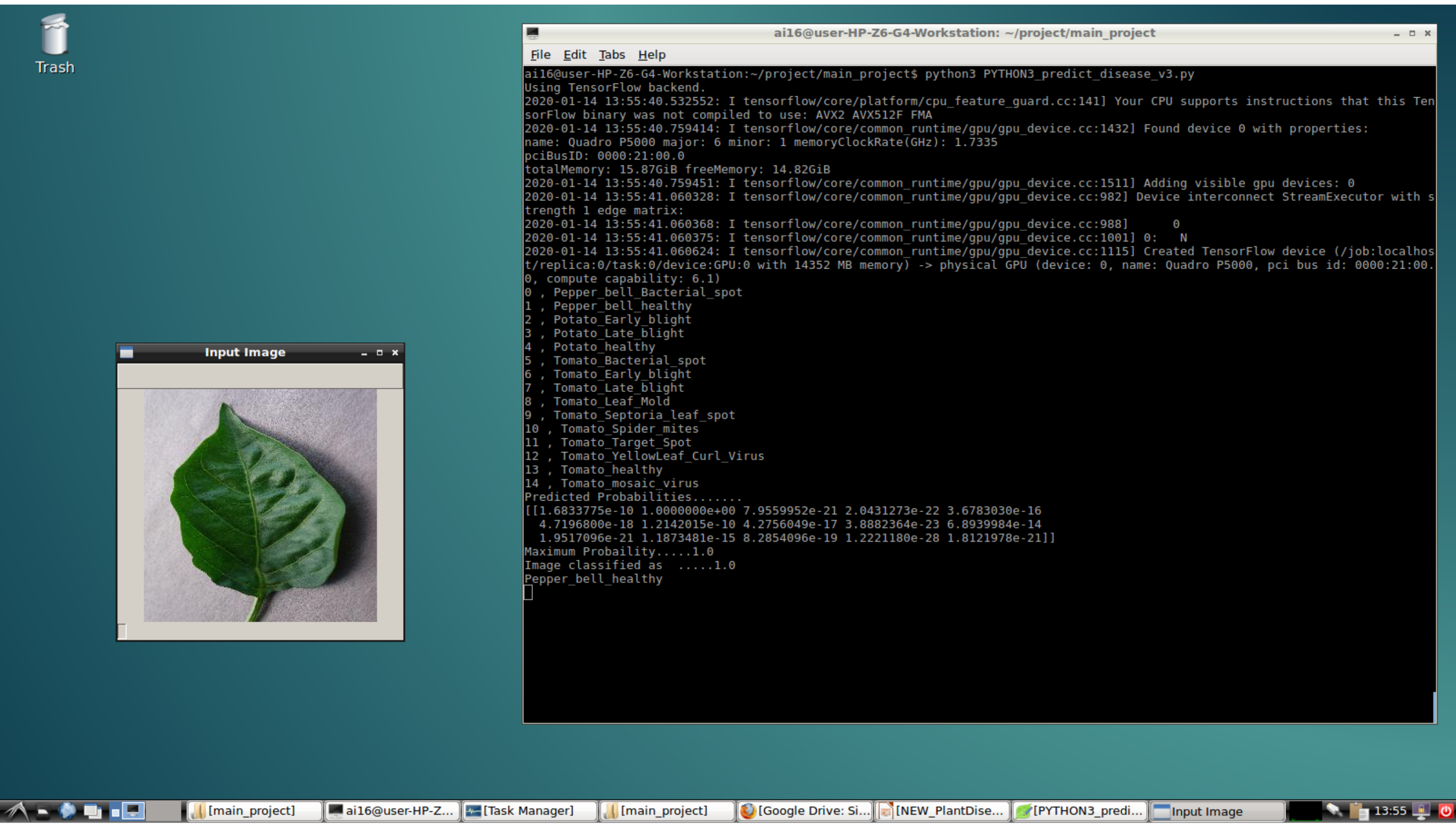



Training and Validation loss
Training loss
Validation loss
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50
Loss
Epochs

8. TEST RESULTS

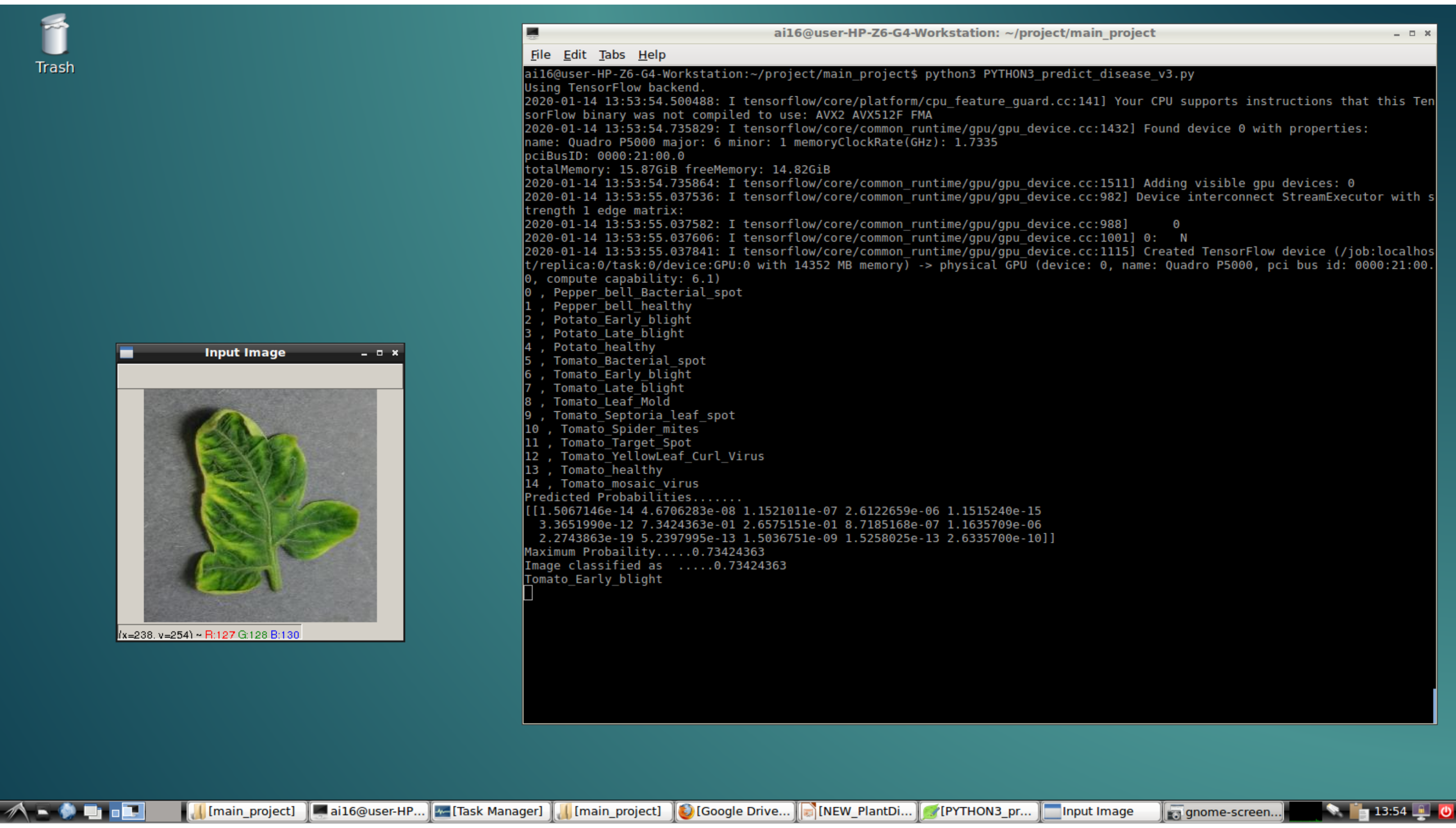
The trained model was used to predict a new leaf image. Below is the output from the prediction.

Correct Classification for a leaf. CNN model is able to correctly predict the input leaf image as Pepper Bell Healthy



```
ai16@user-HP-Z6-G4-Workstation:~/project/main_project$ python3 PYTHON3_predict_disease_v3.py
Using TensorFlow backend.
2020-01-14 13:55:40.532552: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this Ten
sorFlow binary was not compiled to use: AVX2 AVX512F FMA
2020-01-14 13:55:40.759414: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with properties:
name: Quadro P5000 major: 6 minor: 1 memoryClockRate(GHz): 1.7335
pciBusID: 0000:21:00.0
totalMemory: 15.87GiB freeMemory: 14.82GiB
2020-01-14 13:55:40.759451: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2020-01-14 13:55:41.060328: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with s
trength 1 edge matrix:
2020-01-14 13:55:41.060368: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988]      0
2020-01-14 13:55:41.060375: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0:   N
2020-01-14 13:55:41.060624: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhos
t/replica:0/task:0/device:GPU:0 with 14352 MB memory) -> physical GPU (device: 0, name: Quadro P5000, pci bus id: 0000:21:00.
0, compute capability: 6.1)
0 , Pepper_bell_Bacterial_spot
1 , Pepper_bell_healthy
2 , Potato_Early_blight
3 , Potato_Late_blight
4 , Potato_healthy
5 , Tomato_Bacterial_spot
6 , Tomato_Early_blight
7 , Tomato_Late_blight
8 , Tomato_Leaf_Mold
9 , Tomato_Septoria_leaf_spot
10 , Tomato_Spider_mites
11 , Tomato_Target_Spot
12 , Tomato_YellowLeaf_Curl_Virus
13 , Tomato_healthy
14 , Tomato_mosaic_virus
Predicted Probabilities.......
[[1.6833775e-10 1.0000000e+00 7.9559952e-21 2.0431273e-22 3.6783030e-16
  4.7196800e-18 1.2142015e-10 4.2756049e-17 3.8882364e-23 6.8939984e-14
  1.9517096e-21 1.1873481e-15 8.2854096e-19 1.2221180e-28 1.8121978e-21]]
Maximum Probaility.....1.0
Image classified as  .....1.0
Pepper_bell_healthy
```

In some situations , the CNN model predicted the leaf incorrectly. In the below case, the input image is Tomato Yellow leaf curl virus but the prediction is Tomato Early Blight



```
ai16@user-HP-Z6-G4-Workstation:~/project/main_project$ python3 PYTHON3_predict_disease_v3.py
Using TensorFlow backend.
2020-01-14 13:53:54.500488: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this Ten
sorFlow binary was not compiled to use: AVX2 AVX512F FMA
2020-01-14 13:53:54.735829: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with properties:
name: Quadro P5000 major: 6 minor: 1 memoryClockRate(GHz): 1.7335
pciBusID: 0000:21:00.0
totalMemory: 15.87GiB freeMemory: 14.82GiB
2020-01-14 13:53:54.735864: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2020-01-14 13:53:55.037536: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with s
trength 1 edge matrix:
2020-01-14 13:53:55.037582: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988]      0
2020-01-14 13:53:55.037606: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0:   N
2020-01-14 13:53:55.037841: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhos
t/replica:0/task:0/device:GPU:0 with 14352 MB memory) -> physical GPU (device: 0, name: Quadro P5000, pci bus id: 0000:21:00.
0, compute capability: 6.1)
0 , Pepper_bell_Bacterial_spot
1 , Pepper_bell_healthy
2 , Potato_Early_blight
3 , Potato_Late_blight
4 , Potato_healthy
5 , Tomato_Bacterial_spot
6 , Tomato_Early_blight
7 , Tomato_Late_blight
8 , Tomato_Leaf_Mold
9 , Tomato_Septoria_leaf_spot
10 , Tomato_Spider_mites
11 , Tomato_Target_Spot
12 , Tomato_YellowLeaf_Curl_Virus
13 , Tomato_healthy
14 , Tomato_mosaic_virus
Predicted Probabilities.......
[[1.5067146e-14 4.6706283e-08 1.1521011e-07 2.6122659e-06 1.1515240e-15
  3.3651990e-12 7.3424363e-01 2.6575151e-01 8.7185168e-07 1.1635709e-06
  2.2743863e-19 5.2397995e-13 1.5036751e-09 1.5258025e-13 2.6335700e-10]]
Maximum Probaility.....0.73424363
Image classified as  .....0.73424363
Tomato_Early_blight
```

9.PYTHON CODE

Python code is split into two . First python code to train the model and another to predict a new leaf image. Code for both of them are as below

(a) Python code to train the model

```
#Import neccessary packages

from os import listdir
import sys

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

from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers.core import Activation, Flatten, Dense
from keras import backend as K
from keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator,img_to_array
from keras.utils import to_categorical

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split


#Initialise few vars
EPOCHS = 50
INIT_LR = 1e-3
BS = 96 #Changed from 32 to 96
default_image_size = tuple((256, 256))
image_size = 0
directory_root = '/home/ai16/project/main_project/Data/Plantvillage'
width=256
height=256
depth=3

#Function to convert images to array
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}")
        print("convert_image_to_array() : Error when converting Image to Array:"+e)
        return None

#Fetch images from directory and convert each to an array...assign labels to it (folder name)
image_list, label_list = [], []
try:
    print("[INFO] Loading images from folders...")
    root_dir = listdir(directory_root)
    for plant_folder in root_dir :
        print("Processing images from folder... ->: "+ plant_folder)
        Images_In_Folder = listdir(directory_root+"/"+plant_folder)

        #for image in plant_disease_folder_list[:200]:
        for image in Images_In_Folder:
                image_filename = directory_root+"/"+plant_folder+"/"+image
                if image_filename.endswith(".jpg") == True or image_filename.endswith(".JPG") ==
True:
                    image_list.append(convert_image_to_array(image_filename))
                    label_list.append(plant_folder)



    print("[INFO] Image loading completed from all directories....")
except Exception as e:
    print("Try...Catch: Error when loding/processing images...: "+str(e))

#Convert labels to numeric values (Ex. 0,1,2,3...based on categories)
le=LabelEncoder()
label_list_num=le.fit_transform(label_list)
n_classes=len(np.unique(label_list_num))

#convert to binary values (one hot encoding)
label_list_num_bin=to_categorical(label_list_num)

np_image_list = np.array(image_list, dtype=np.float16) / 255.0

#Create Train and Test set
print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, label_list_num_bin, test_size=0.2,
random_state = 42)

#Image Augmentation
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")


#Build CNN layers
model = Sequential()

inputShape = (height, width, depth)

model.add(Conv2D(32, (3, 3), input_shape=inputShape))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3, 3)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation("relu"))

model.add(Conv2D(64, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(Activation("relu"))

model.add(Conv2D(128, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

#Fully Connected Layer
model.add(Dense(128))
model.add(Activation("relu"))

#Output layer
model.add(Dense(n_classes))
model.add(Activation("softmax"))

model.summary()

opt = Adam(lr=INIT_LR)

# distribution...Complile the model
model.compile(loss="categorical_crossentropy", optimizer=opt,metrics=["accuracy"])

# train the network
```

```
print("[INFO] training network...")
history = model.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['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

#Plot the train and val curve

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()


#Model Accuracy
print("[INFO] Calculating model accuracy")
scores = model.evaluate(x_test, y_test)
print("Test Loss: "+str(scores[0]))
print("Test Accuracy: "+str(scores[1]*100))
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

#Save the model to disk
print("[INFO] Saving model...")
model.save("cnn_model.h5")
```

(b)Python code to predict an input leaf image

```
from keras.preprocessing import image
from keras.models import load_model
import matplotlib.pyplot as plt
import numpy as np
import os
import cv2 as cv

def load_image(img_path, show=False):

    img = image.load_img(img_path, target_size=(256, 256))
    img_array = image.img_to_array(img)                    # (height, width, channels)
    img_array = np.expand_dims(img_array, axis=0)         # (1, height, width, channels), add a
dimension because the model expects this shape: (batch_size, height, width, channels)
    img_array /= 255.                                      # imshow expects values in the range [0, 1]

    if show:
        plt.imshow(img_array[0])
        plt.axis('off')
        plt.show()

    return img_array

if __name__ == "__main__":

    # load model
    model = load_model("/home/ai16/project/main_project/cnn_model_50epochs_12ksamples.h5")

    # image path
    img_path = '/home/ai16/project/main_project/Data/TEST/Pepper_bell_healthy/0a3f2927-4410-
46a3-bfda-5f4769a5aaf8___JR_HL 8275.JPG'

    # load a single image
    new_image = load_image(img_path)

    new_list = ['Pepper_bell_Bacterial_spot', 'Pepper_bell_healthy', 'Potato_Early_blight',\
 'Potato_Late_blight', 'Potato_healthy', 'Tomato_Bacterial_spot',\
 'Tomato_Early_blight', 'Tomato_Late_blight', 'Tomato_Leaf_Mold',\
 'Tomato_Septoria_leaf_spot', 'Tomato_Spider_mites', 'Tomato_Target_Spot',\
 'Tomato_YellowLeaf_Curl_Virus', 'Tomato_healthy', 'Tomato_mosaic_virus']
    i=0
    #Index and Categories
    while(i<15):
      print(str(i)+" , "+new_list[i])
      i=i+1

    #Check prediction
    #Generates output predictions for the input samples.
```

```
#Computation is done in batches
print("Predicted Probabilities.......")
pred = model.predict(new_image)
print(pred)

print("Maximum Probaility....."+ str(pred.max()))

if pred.max()>0.5:
 print("Image classified as  ....."+str(pred.max()))
 #Index with largest value across axes of tensor..
 classes_names=pred.argmax(axis=-1)
 #print(classes_names)
 print(new_list[classes_names[0]])
else:
 print("Not a leaf (Tomato, Potato, Bell Pepper)...")

img_vw=cv.imread(img_path)
cv.imshow('Input Image',img_vw)
cv.waitKey(10000)
```