Plant Disease Classification

(using leaf images)

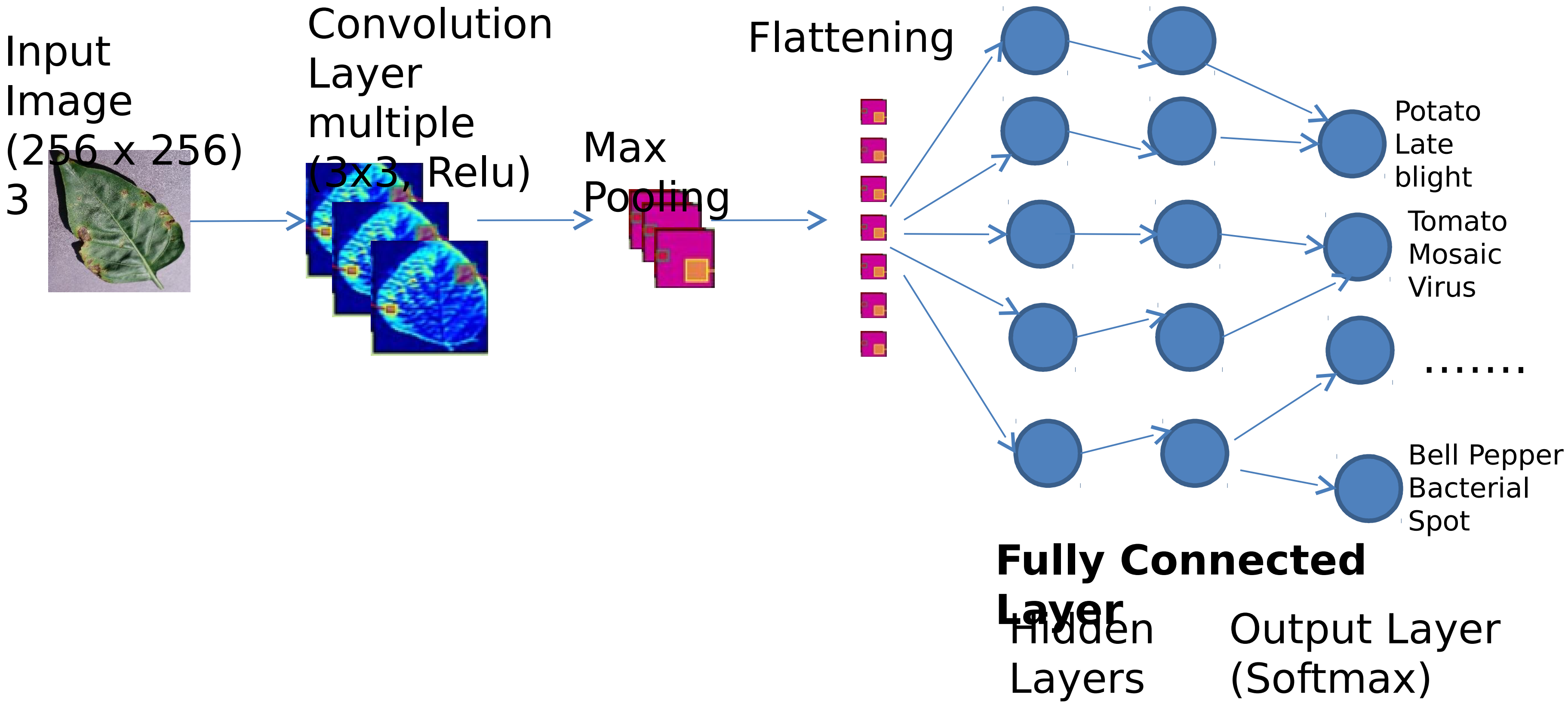
Submitted By:-
C S UNNIKRISHAN
PANKAJ
AKSHAY K BABY

Under the guidance of :-
Mrs. Vimala Mathew

14th January 2020

Solution Design

Objective : To classify leaf images (healthy and disease) using CNN



Input Image (256 x 256) 3

Convolution Layer multiple (3x3, Relu)

Max Pooling

Flattening

Potato Late blight

Tomato Mosaic Virus

....

Bell Pepper Bacterial Spot

Fully Connected Layer

Hidden Layers

Output Layer (Softmax)

Training Approach

•Model is trained on around **20,000 leaf images** (10 Epochs and 32 as Batch size)

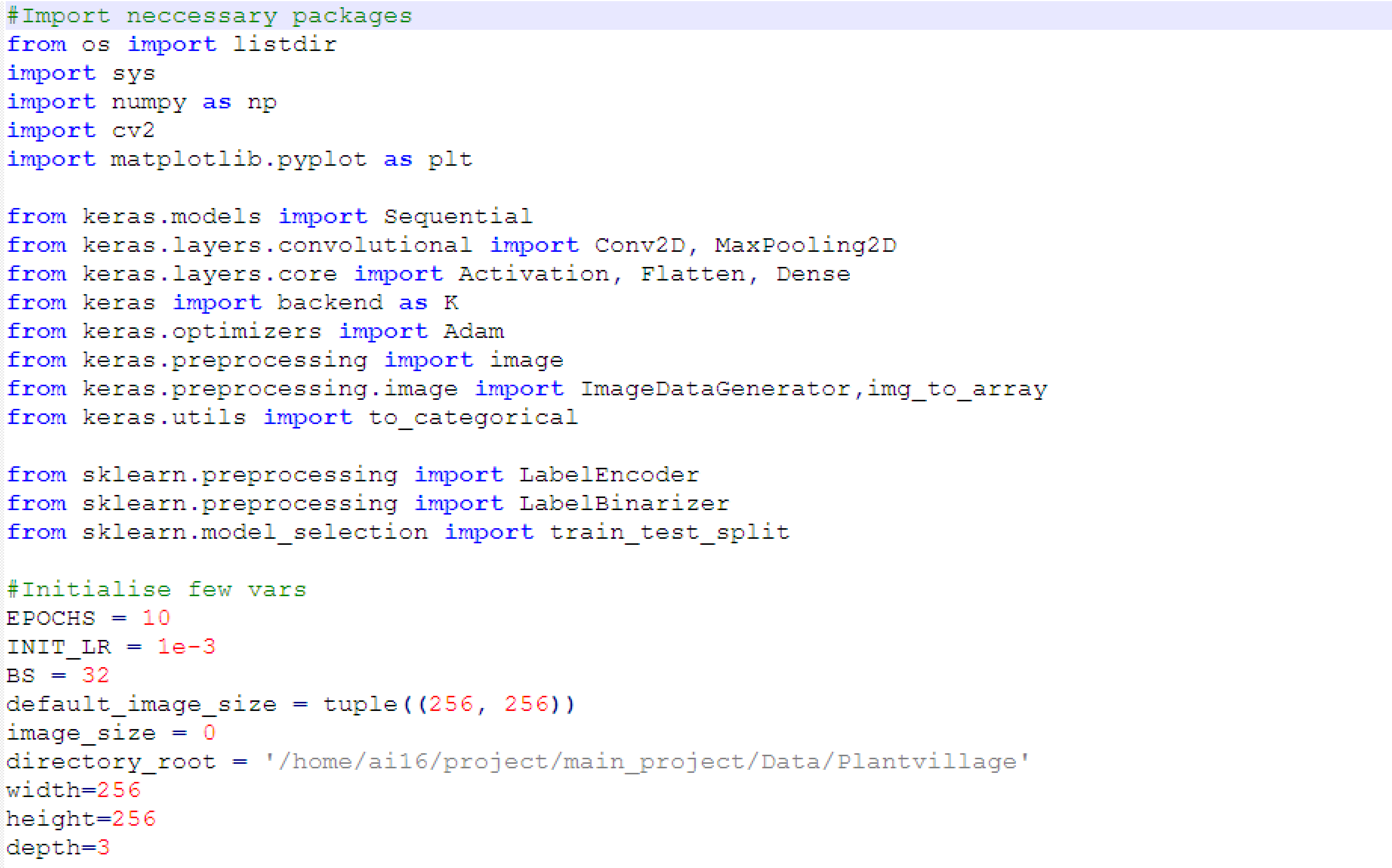
•Images split into **Training and Test**

•**Image Augmentation** applied for each image

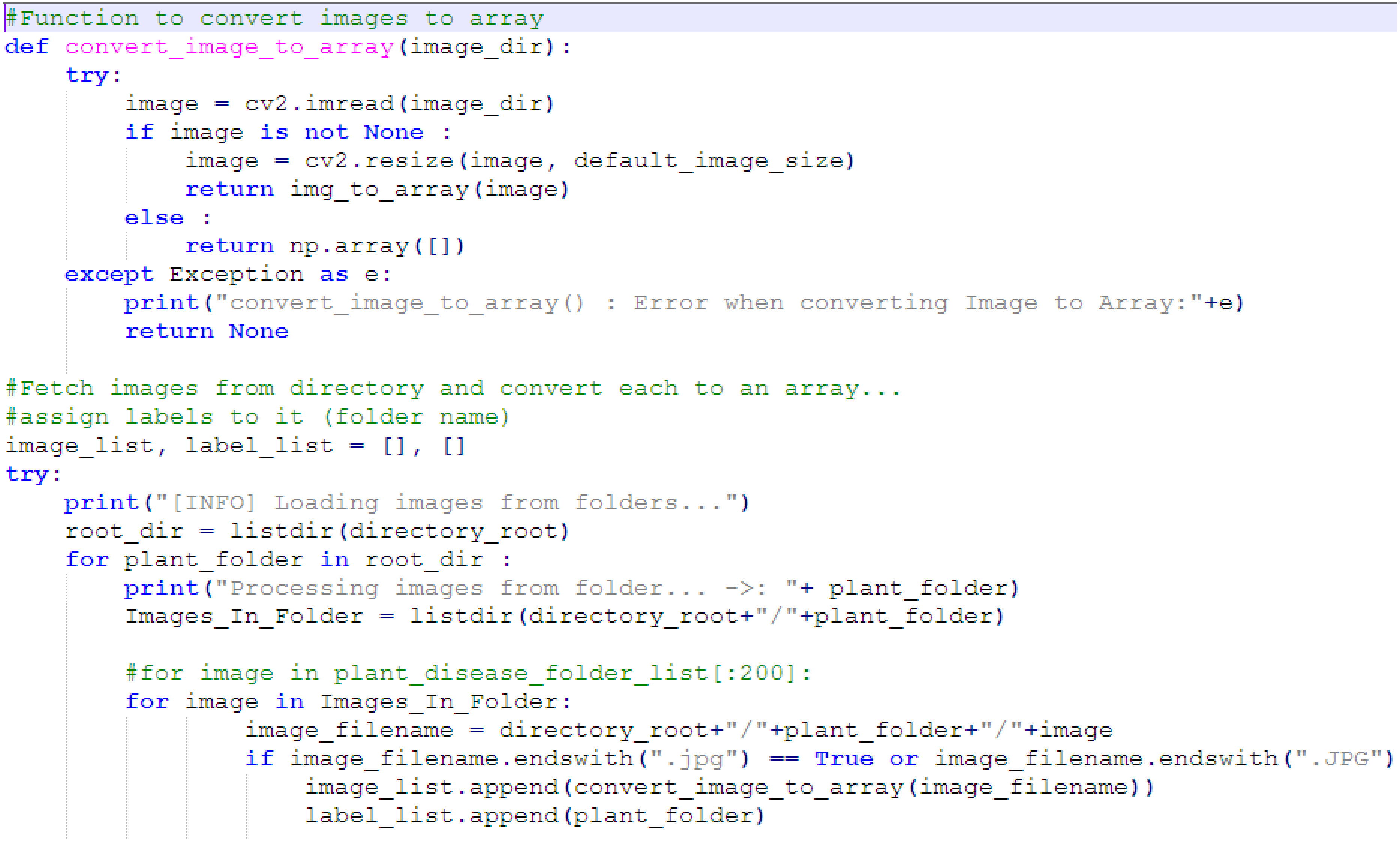
•**Performance metrics** are plotted (accuracy and loss)

•Trained model saved and used to classify a leaf image (**healthy or disease**)

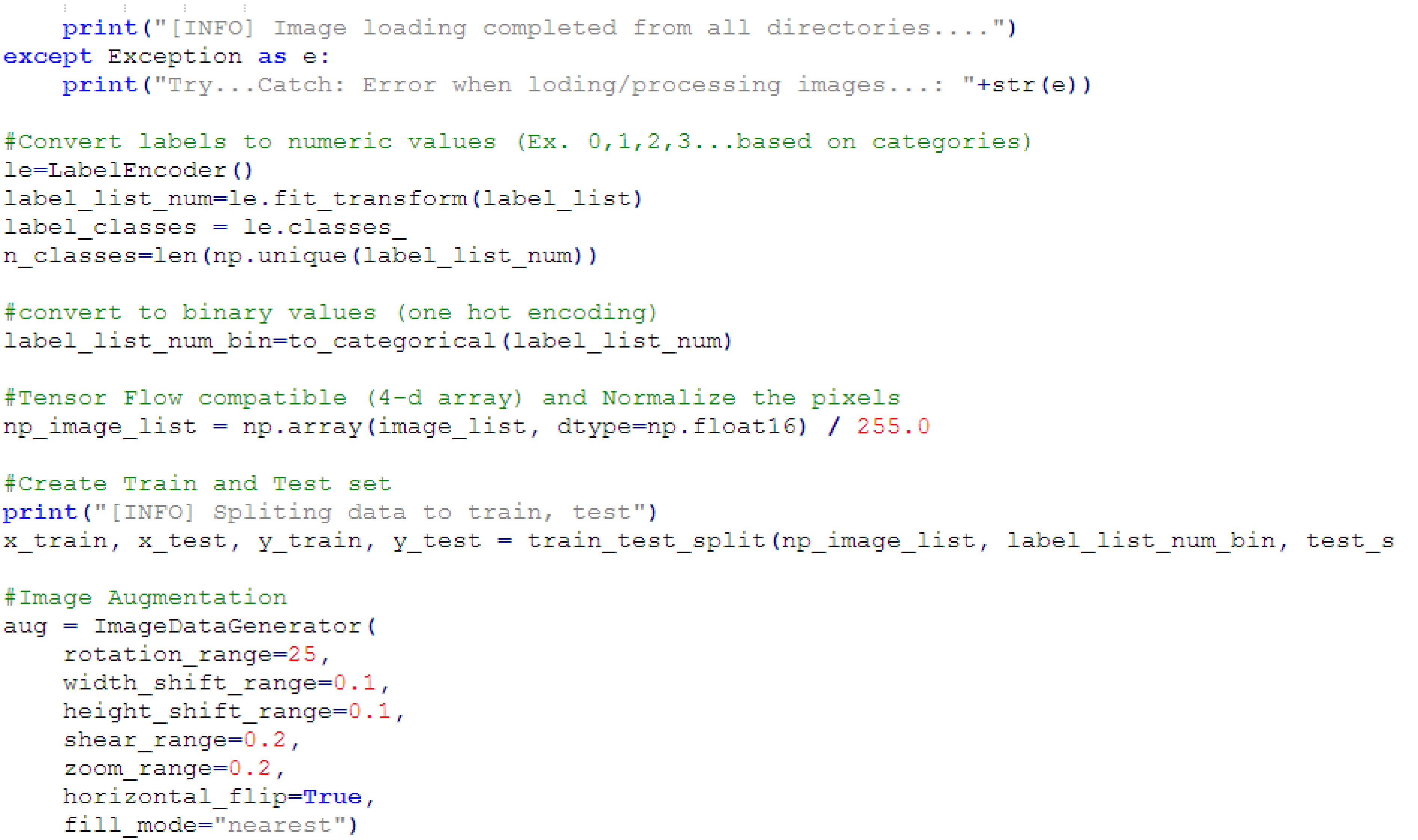
Program: Train model for Classification



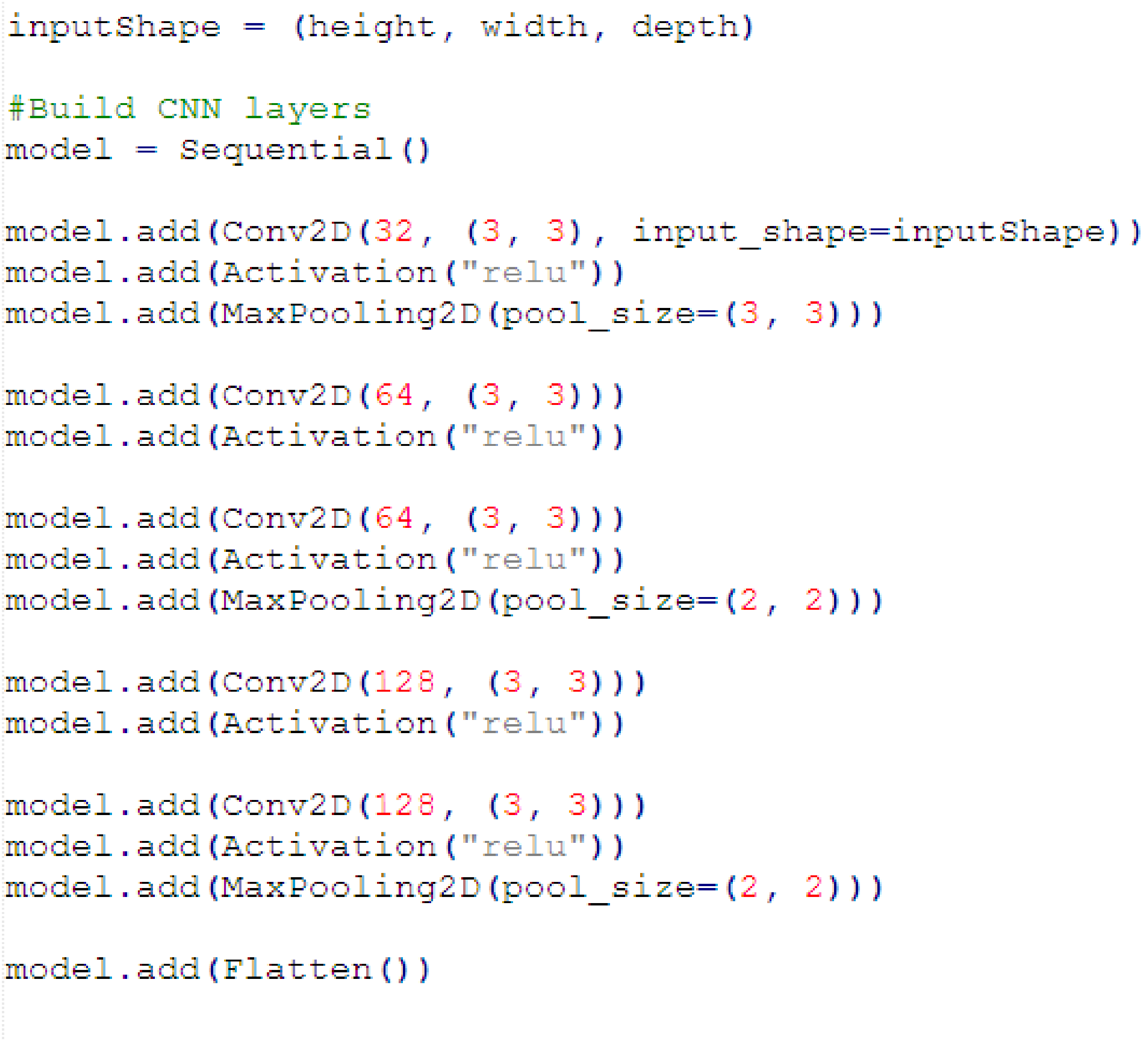
```
#Import neccessary packages
from os import listdir
import sys
import numpy as np
import cv2
import matplotlib.pyplot as plt

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 = 10
INIT_LR = 1e-3
BS = 32
default_image_size = tuple((256, 256))
image_size = 0
directory_root = '/home/ai16/project/main_project/Data/Plantvillage'
width=256
height=256
depth=3
```



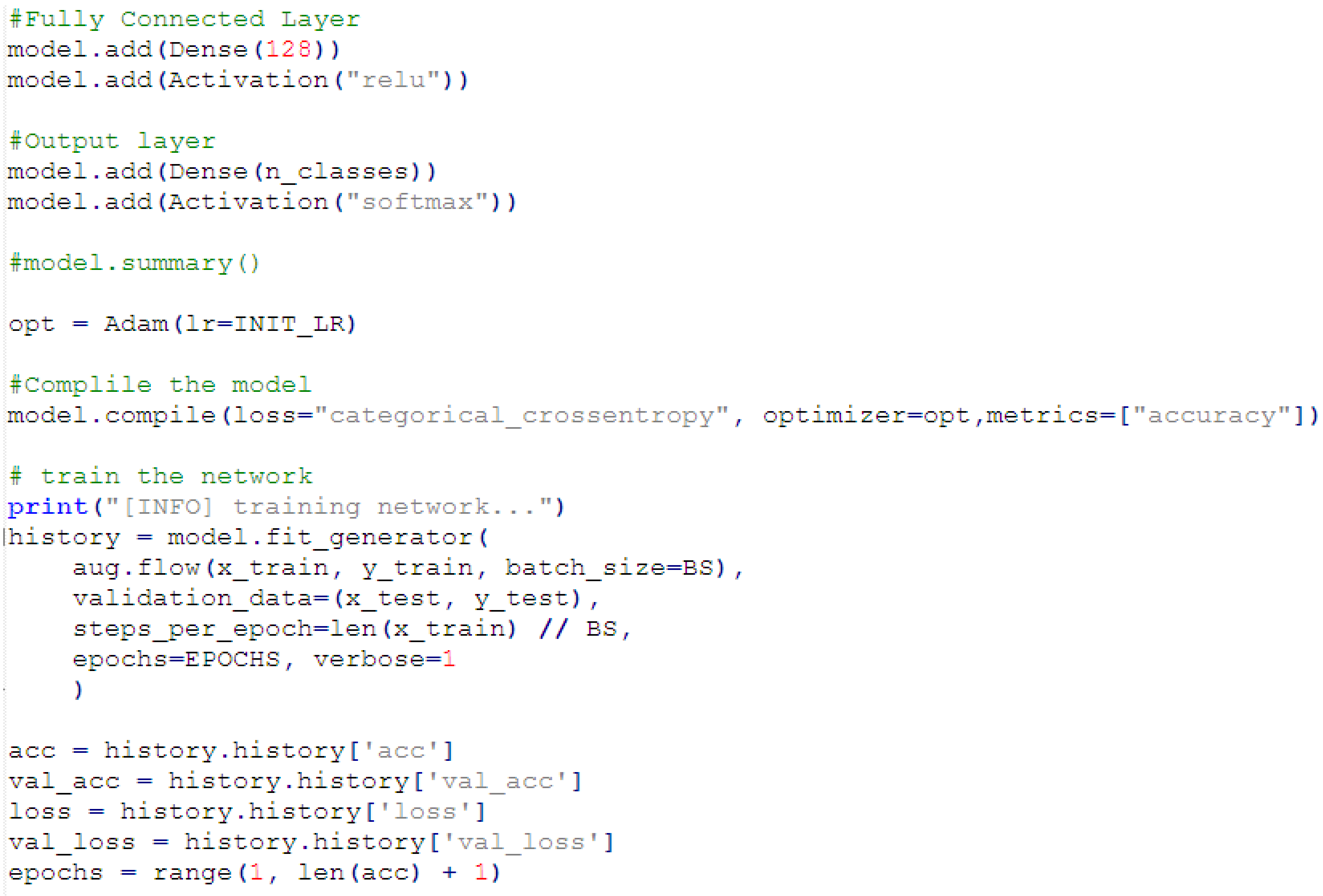
```
#Function to convert images to array
def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        if image is not None :
            image = cv2.resize(image, default_image_size)
            return img_to_array(image)
        else :
            return np.array([])
    except Exception as e:
        print("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")
                    image_list.append(convert_image_to_array(image_filename))
                    label_list.append(plant_folder)
```



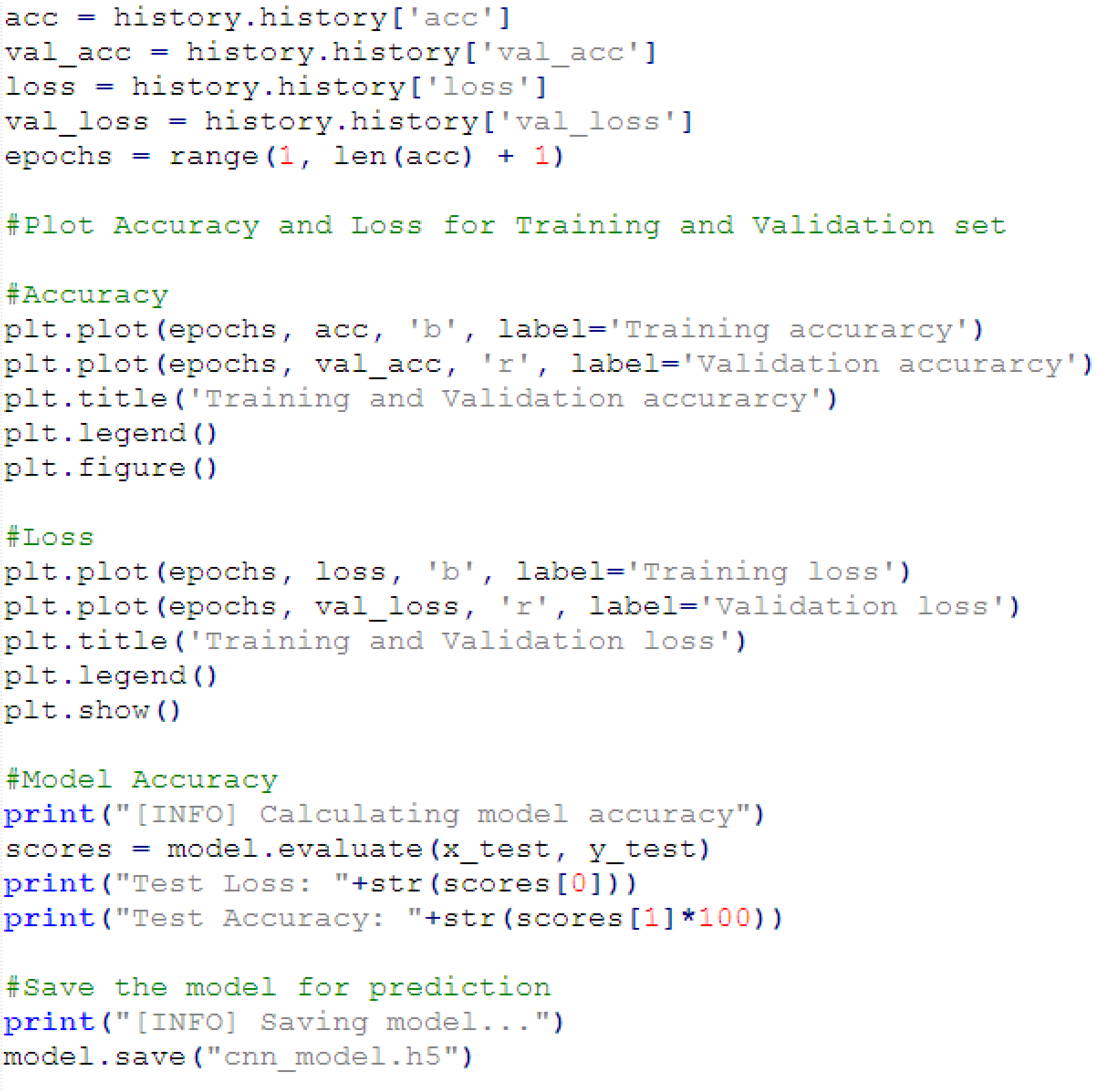
```
    print("[INFO] Image loading completed from all directories....")
except Exception as e:
    print("Try...Catch: Error when loding/processing images...: "+str(e))

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

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

#Tensor Flow compatible (4-d array) and Normalize the pixels
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_s

#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")
```



```
inputShape = (height, width, depth)

#Build CNN layers
model = Sequential()

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())
```

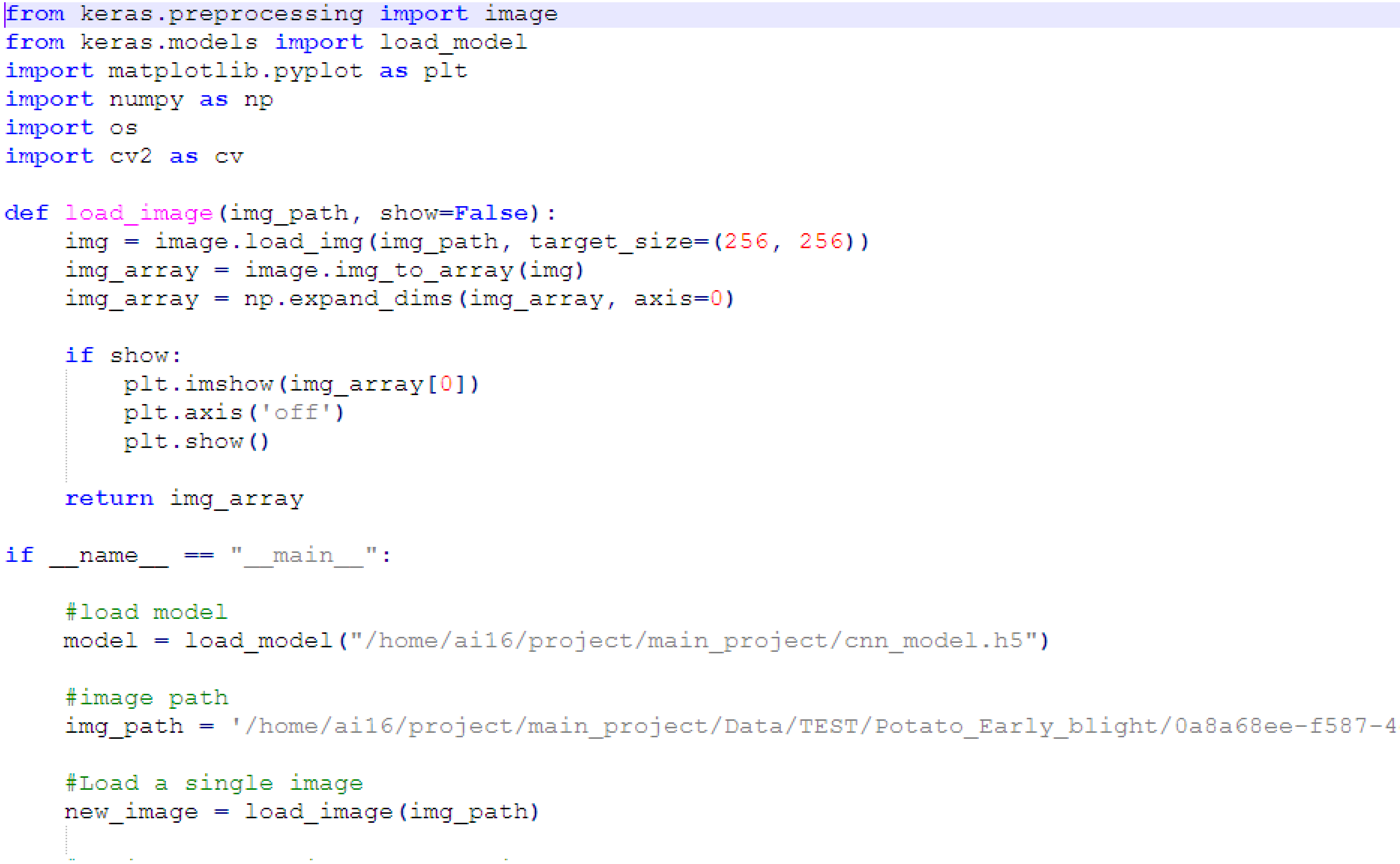


```
#Fully Connected Layer
model.add(Dense(128))
model.add(Activation("relu"))

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

#model.summary()

opt = Adam(lr=INIT_LR)

#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)
```

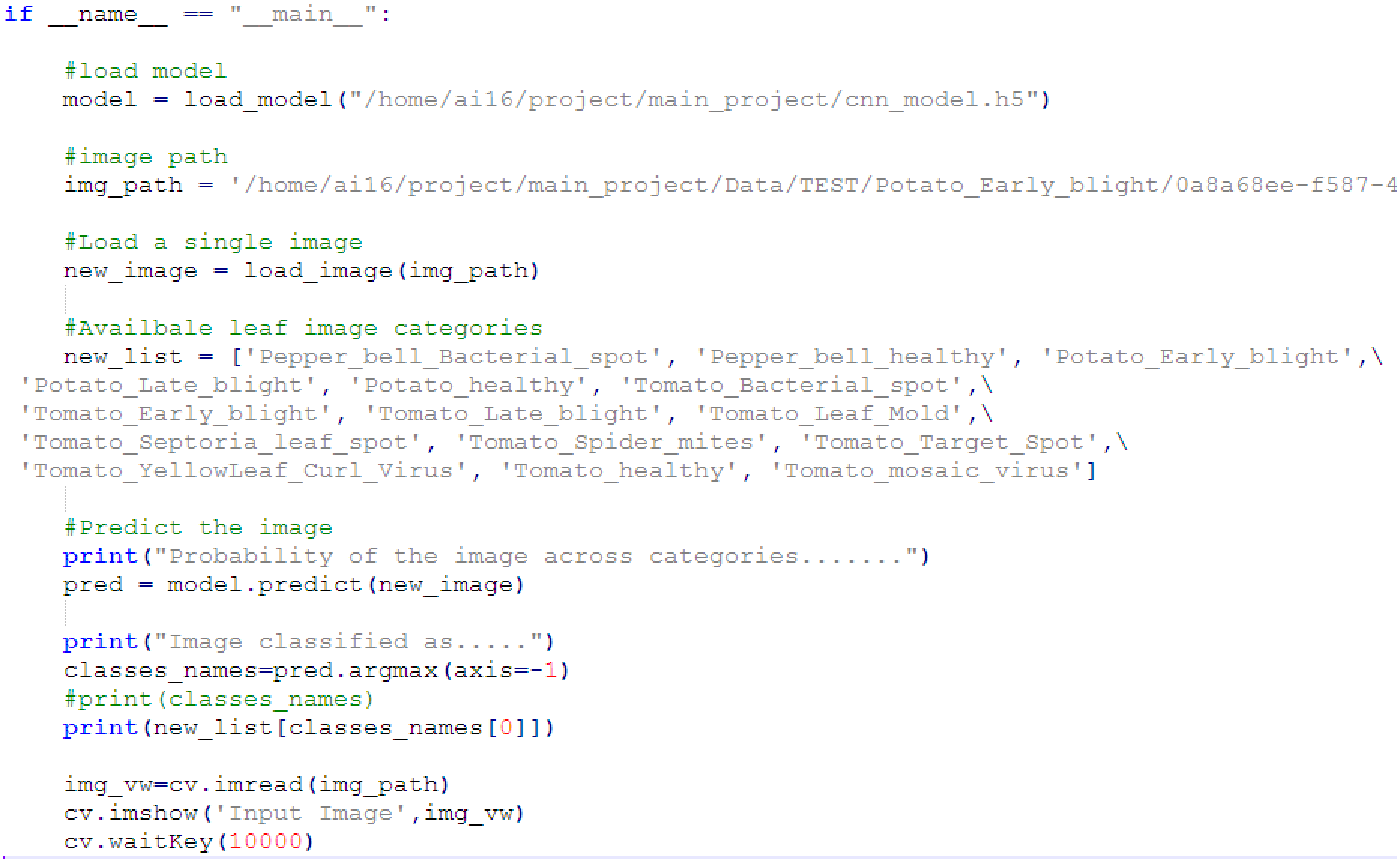


```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

#Plot Accuracy and Loss for Training and Validation set

#Accuracy
plt.plot(epochs, acc, 'b', label='Training accurarcy')
plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()
plt.figure()

#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))

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

Program : Image classification based on Trained model

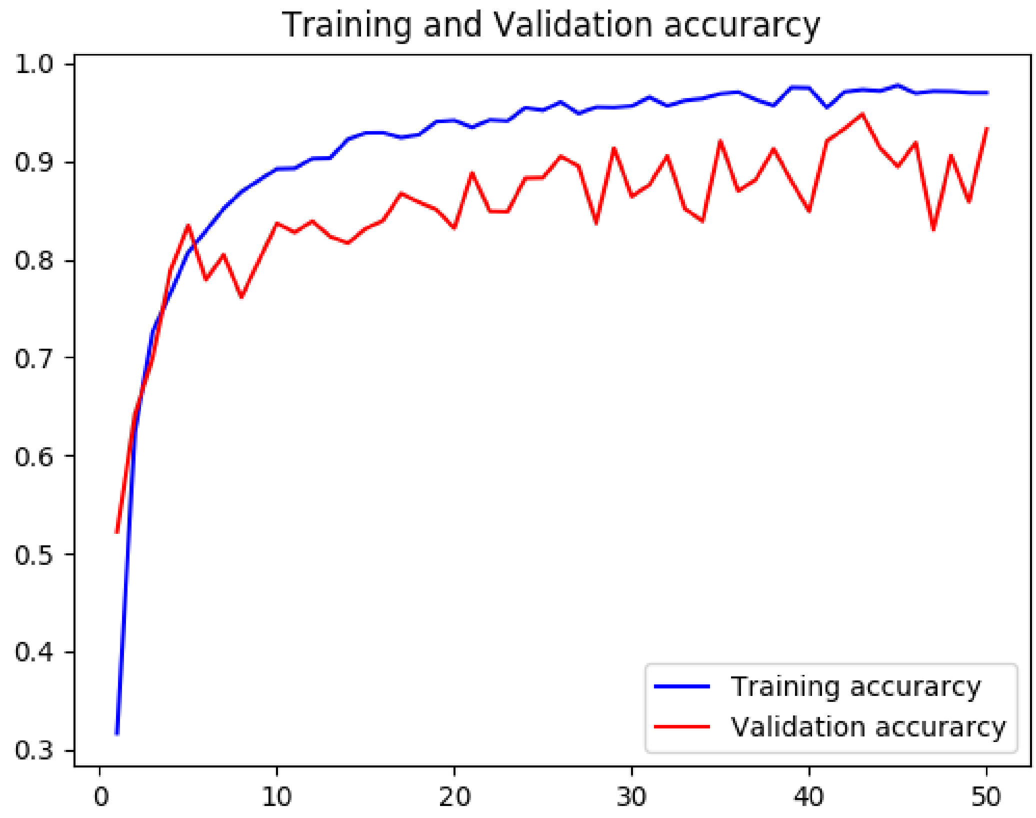


```
from keras.preprocessing import image
from keras.models import load_model
import matplotlib.pyplot as plt
import numpy as np
import os
import cv2 as cv

def load_image(img_path, show=False):
    img = image.load_img(img_path, target_size=(256, 256))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)

    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.h5")

    #image path
    img_path = '/home/ai16/project/main_project/Data/TEST/Potato_Early_blight/0a8a68ee-f587-4

    #Load a single image
    new_image = load_image(img_path)
```



```
if __name__ == "__main__":

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

    #image path
    img_path = '/home/ai16/project/main_project/Data/TEST/Potato_Early_blight/0a8a68ee-f587-4

    #Load a single image
    new_image = load_image(img_path)

    #Availbale leaf image categories
    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']

    #Predict the image
    print("Probability of the image across categories.......")
    pred = model.predict(new_image)

    print("Image classified as.....")
    classes_names=pred.argmax(axis=-1)
    #print(classes_names)
    print(new_list[classes_names[0]])

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

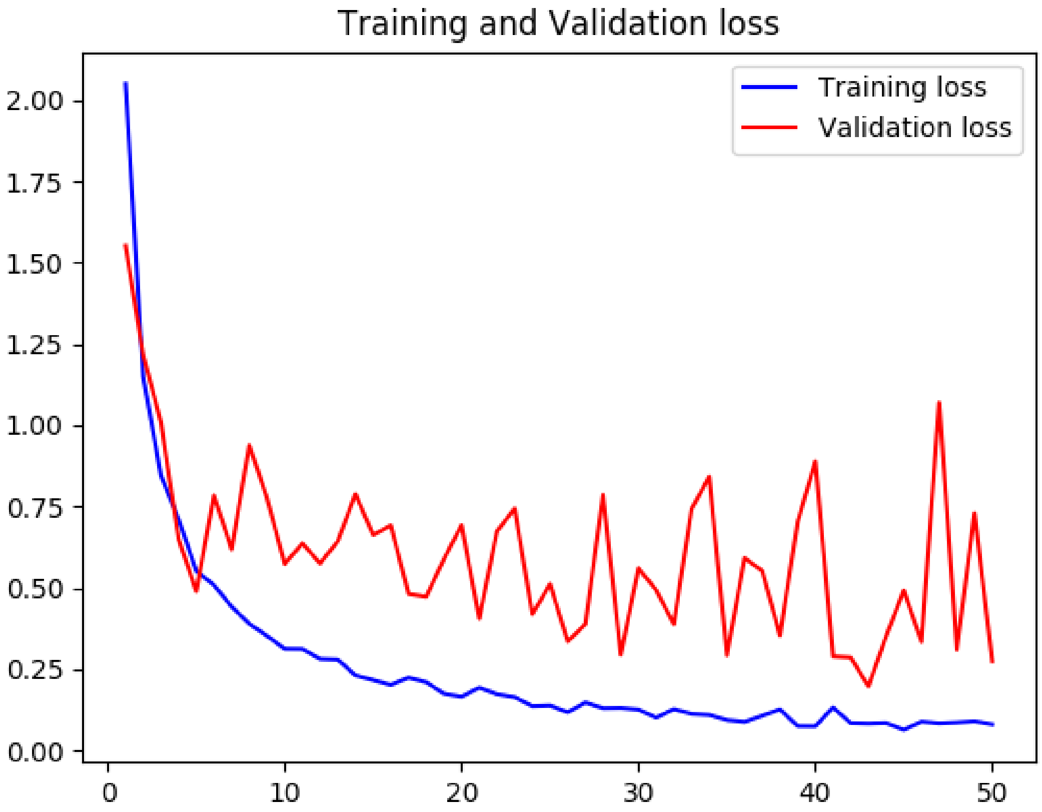
Training & Validation Accuracy (12,000 leaf images)



Accuracy

Epochs

Training & Validation Loss (12,000 leaf images)

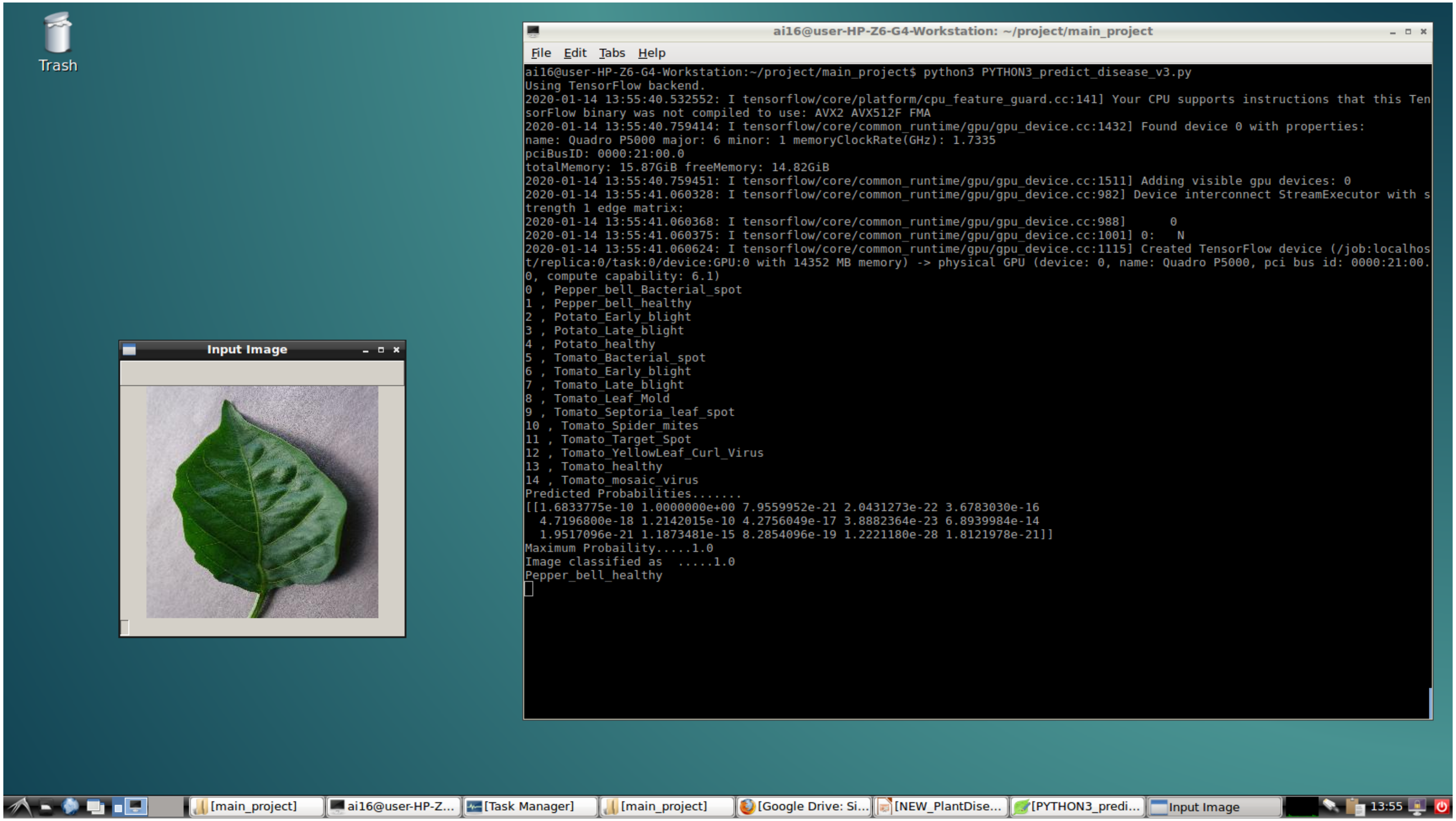


Loss

Epochs

Output

Correct classification



```
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
```

Future Scope

- **Highlight** the infected leaf part

- **Increase coverage of plants** for disease identification

- **Remedy** for a plant disease can be provided

- Model can be integrated with a **mobile app** and farmers can use (with local language support)

Conclusion

- CNN model was able to classify the leaf images as healthy or disease with probability

- Training accuracy improved and loss decreased with increase in number of samples (from 1500 to 20,000)

Thank You