**LEAF DISEASE PREDICTION USING DEEP LEARNING**

PROGRAM

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from matplotlib.image import imread

import cv2

import random

import os

from os import listdir

from PIL import Image

from sklearn.preprocessing import label\_binarize, LabelBinarizer

from keras.preprocessing import image

from keras.preprocessing.image import img\_to\_array, array\_to\_img

from tensorflow.keras.optimizers import Adam

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D

from keras.layers import Activation, Flatten, Dropout, Dense

from sklearn.model\_selection import train\_test\_split

from keras.models import model\_from\_json

from tensorflow.keras.utils import to\_categorical

plt.figure(figsize=(12,12))

path = "../input/leaf-image-dataset/Plant\_images/Potato\_\_\_Early\_blight"

for i **in** range(1,17):

plt.subplot(4,4,i)

plt.tight\_layout()

rand\_img = imread(path +'/'+ random.choice(sorted(os.listdir(path))))

plt.imshow(rand\_img)

plt.xlabel(rand\_img.shape[1], fontsize = 10)

plt.ylabel(rand\_img.shape[0], fontsize = 10)

def convert\_image\_to\_array(image\_dir):

try:

image = cv2.imread(image\_dir)

if image **is** **not** None :

image = cv2.resize(image, (256,256))

*#image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)*

return img\_to\_array(image)

else :

return np.array([])

except **Exception** as e:

print(f"Error : **{**e**}**")

return None

dir = "../input/leaf-image-dataset/Plant\_images"

root\_dir = listdir(dir)

image\_list, label\_list = [], []

all\_labels = ['Corn-Common\_rust', 'Potato-Early\_blight', 'Tomato-Bacterial\_spot']

binary\_labels = [0,1,2]

temp = -1

*# Reading and converting image to numpy array*

*#Now we will convert all the images into numpy array.*

for directory **in** root\_dir:

plant\_image\_list = listdir(f"**{**dir**}**/**{**directory**}**")

temp += 1

for files **in** plant\_image\_list:

image\_path = f"**{**dir**}**/**{**directory**}**/**{**files**}**"

image\_list.append(convert\_image\_to\_array(image\_path))

label\_list.append(binary\_labels[temp])

label\_counts = pd.DataFrame(label\_list).value\_counts()

label\_counts.head()

image\_list[0].shape

label\_list = np.array(label\_list)

label\_list.shape

x\_train, x\_test, y\_train, y\_test = train\_test\_split(image\_list, label\_list, test\_size=0.2, random\_state = 10)

x\_train = np.array(x\_train, dtype=np.float16) / 225.0

x\_test = np.array(x\_test, dtype=np.float16) / 225.0

x\_train = x\_train.reshape( -1, 256,256,3)

x\_test = x\_test.reshape( -1, 256,256,3)

y\_train = to\_categorical(y\_train)

y\_test = to\_categorical(y\_test)

model = Sequential()

model.add(Conv2D(32, (3, 3), padding="same",input\_shape=(256,256,3), activation="relu"))

model.add(MaxPooling2D(pool\_size=(3, 3)))

model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(8, activation="relu"))

model.add(Dense(3, activation="softmax"))

model.summary()

model.compile(loss = 'categorical\_crossentropy', optimizer = Adam(0.0001),metrics=['accuracy'])

x\_train, x\_val, y\_train, y\_val = train\_test\_split(x\_train, y\_train, test\_size = 0.2)

epochs = 50

batch\_size = 128

history = model.fit(x\_train, y\_train, batch\_size = batch\_size, epochs = epochs,

validation\_data = (x\_val, y\_val))

plt.figure(figsize=(12, 5))

plt.plot(history.history['accuracy'], color='r')

plt.plot(history.history['val\_accuracy'], color='b')

plt.title('Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epochs')

plt.legend(['train', 'val'])

plt.show()

print("[INFO] Calculating model accuracy")

scores = model.evaluate(x\_test, y\_test)

print(f"Test Accuracy: **{**scores[1]\*100**}**")

y\_pred = model.predict(x\_test)

img = array\_to\_img(x\_test[10])

img

print("Originally : ",all\_labels[np.argmax(y\_test[10])])

print("Predicted : ",all\_labels[np.argmax(y\_pred[10])])

INPUT



OUTPUT

A white background with black text

Description automatically generated