**Cropdisease.py**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

import matplotlib.pyplot as plt

import numpy as np

from tkinter import ttk

from tkinter import filedialog

from keras.utils.np\_utils import to\_categorical

from keras.models import Sequential

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

from sklearn.metrics import accuracy\_score

import os

import cv2

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

import pickle

from keras.models import model\_from\_json

main = Tk()

main.title("Research on Recognition Model of Crop Diseases and Insect Pests Based on Deep Learning in Harsh Environments")

main.geometry("1300x1200")

global filename

global X, Y

global model

global accuracy

plants = ['Pepper\_\_bell\_\_\_Bacterial\_spot', 'Pepper\_\_bell\_\_\_healthy', 'Potato\_\_\_Early\_blight', 'Potato\_\_\_healthy', 'Potato\_\_\_Late\_blight', 'Tomato\_\_Target\_Spot',

'Tomato\_\_Tomato\_mosaic\_virus', 'Tomato\_\_Tomato\_YellowLeaf\_\_Curl\_Virus', 'Tomato\_Bacterial\_spot', 'Tomato\_Early\_blight', 'Tomato\_healthy',

'Tomato\_Late\_blight', 'Tomato\_Leaf\_Mold', 'Tomato\_Septoria\_leaf\_spot', 'Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite']

def uploadDataset():

global X, Y

global filename

text.delete('1.0', END)

filename = filedialog.askdirectory(initialdir=".")

text.insert(END,'dataset loaded\n')

def imageProcessing():

text.delete('1.0', END)

global X, Y

X = np.load("model/myimg\_data.txt.npy")

Y = np.load("model/myimg\_label.txt.npy")

Y = to\_categorical(Y)

X = np.asarray(X)

Y = np.asarray(Y)

X = X.astype('float32')

X = X/255

indices = np.arange(X.shape[0])

np.random.shuffle(indices)

X = X[indices]

Y = Y[indices]

text.insert(END,'image processing completed\n')

img = X[20].reshape(64,64,3)

cv2.imshow('ff',cv2.resize(img,(250,250)))

cv2.waitKey(0)

def cnnModel():

global model

global accuracy

text.delete('1.0', END)

if os.path.exists('model/model.json'):

with open('model/model.json', "r") as json\_file:

loaded\_model\_json = json\_file.read()

model = model\_from\_json(loaded\_model\_json)

json\_file.close()

model.load\_weights("model/model\_weights.h5")

model.\_make\_predict\_function()

print(model.summary())

f = open('model/history.pckl', 'rb')

accuracy = pickle.load(f)

f.close()

acc = accuracy['accuracy']

acc = acc[9] \* 100

text.insert(END,"CNN Crop Disease Recognition Model Prediction Accuracy = "+str(acc))

else:

model = Sequential() #resnet transfer learning code here

model.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu'))

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

model.add(Convolution2D(32, 3, 3, activation = 'relu'))

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

model.add(Flatten())

model.add(Dense(output\_dim = 256, activation = 'relu'))

model.add(Dense(output\_dim = 15, activation = 'softmax'))

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

print(model.summary())

hist = model.fit(X, Y, batch\_size=16, epochs=10, validation\_split=0.2, shuffle=True, verbose=2)

model.save\_weights('model/model\_weights.h5')

model\_json = model.to\_json()

with open("model/model.json", "w") as json\_file:

json\_file.write(model\_json)

json\_file.close()

f = open('model/history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

f = open('model/history.pckl', 'rb')

accuracy = pickle.load(f)

f.close()

acc = accuracy['accuracy']

acc = acc[9] \* 100

text.insert(END,"CNN Crop Disease Recognition Model Prediction Accuracy = "+str(acc))

def predict():

global model

filename = filedialog.askopenfilename(initialdir="testImages")

img = cv2.imread(filename)

img = cv2.resize(img, (64,64))

im2arr = np.array(img)

im2arr = im2arr.reshape(1,64,64,3)

test = np.asarray(im2arr)

test = test.astype('float32')

test = test/255

preds = model.predict(test)

predict = np.argmax(preds)

img = cv2.imread(filename)

img = cv2.resize(img, (800,400))

cv2.putText(img, 'Crop Disease Recognize as : '+plants[predict], (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0, 255, 0), 2)

cv2.imshow('Crop Disease Recognize as : '+plants[predict], img)

cv2.waitKey(0)

def graph():

acc = accuracy['accuracy']

loss = accuracy['loss']

plt.figure(figsize=(10,6))

plt.grid(True)

plt.xlabel('Iterations')

plt.ylabel('Accuracy/Loss')

plt.plot(acc, 'ro-', color = 'green')

plt.plot(loss, 'ro-', color = 'blue')

plt.legend(['Accuracy', 'Loss'], loc='upper left')

#plt.xticks(wordloss.index)

plt.title('Iteration Wise Accuracy & Loss Graph')

plt.show()

def close():

main.destroy()

text.delete('1.0', END)

font = ('times', 15, 'bold')

title = Label(main, text='Research on Recognition Model of Crop Diseases and Insect Pests Based on Deep Learning in Harsh Environments')

#title.config(bg='powder blue', fg='olive drab')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

ff = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload Crop Disease Dataset", command=uploadDataset)

uploadButton.place(x=20,y=100)

uploadButton.config(font=ff)

processButton = Button(main, text="Image Processing & Normalization", command=imageProcessing)

processButton.place(x=20,y=150)

processButton.config(font=ff)

modelButton = Button(main, text="Build Crop Disease Recognition Model", command=cnnModel)

modelButton.place(x=20,y=200)

modelButton.config(font=ff)

predictButton = Button(main, text="Upload Test Image & Predict Disease", command=predict)

predictButton.place(x=20,y=250)

predictButton.config(font=ff)

graphButton = Button(main, text="Accuracy & Loss Graph", command=graph)

graphButton.place(x=20,y=300)

graphButton.config(font=ff)

exitButton = Button(main, text="Exit", command=close)

exitButton.place(x=20,y=350)

exitButton.config(font=ff)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=85)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=450,y=100)

text.config(font=font1)

main.config()

main.mainloop()