INFORMATICS INSTITUE OF TECHNOLOGY

DEPARTMENT OF COMPUTING

Module: 5COSC009C.2

Software Development Group Project

Module Leader: Mr. Guhanathan Poravi

Paddy Weed Detector

Team Hypesters		
Name	Iit ID	UoW ID
Akila Nanayakkara	2018400	w1742308
Kisal Wedage	2018368	w1742101
Moveen Wijerathne	2018541	w1742328
Amilakelum Kodikara	2018395	w1742107
Danushka Samarakoon	2018382	w1742106
Malshama Perera	2018446	w1742313

Contents

❖ Data Science Model Code

• Data Set Importing and Pre-Processing

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
from tqdm import tqdm
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

DATA_DIRECTORY = "D:\iit\year-2\SDGP\weed-
detector\paddyWeedDetector\Data_set\datatrain" #path where the data set is
located
CATEGORIES = ["paddy", "weed"]

for category in CATEGORIES:
    path = os.path.join(DATA_DIRECTORY, category)  # create path to paddy and
weed
    for img in os.listdir(path):  # iterate over each image per paddy and weed
        gray_img_array = cv2.imread(os.path.join(path, img),
cv2.IMREAD_GRAYSCALE)  # convert to array and gray scaling
        plt.imshow(gray_img_array, cmap='gray')  # graph it
        plt.show()  # display
        break  # to display one picture in gray scale
    break

print(gray_img_array)

print(gray_img_array.shape)

IMAGE_SIZE = 50 #resizing images to 50 by 50

resize_img_array = cv2.resize(gray_img_array, (IMAGE_SIZE, IMAGE_SIZE))
#adding the resized images to an array
plt.imshow(resize_img_array, cmap='gray')
plt.show()

training_data_array = [] #training data array

def create_training_data():
    for category in CATEGORIES:

        path = os.path.join(DATA_DIRECTORY, category)  # create path to paddy
and weed
        class_num = CATEGORIES.index(category)  # get the classification  (0
or a 1). 0=paddy 1=weed

        for image in tqdm(os.listdir(path)):  # iterate over each image per
paddy and weed
            try:
                image_array = cv2.imread(os.path.join(path, image),
cv2.IMREAD_GRAYSCALE)  # convert to array
                image_array_2 = cv2.resize(image_array, (IMAGE_SIZE,
IMAGE_SIZE))  # resize to normalize data size
                training_data_array.append([image_array_2, class_num])  # add
this to training_data
            except Exception as e:  #to keep the output clean...
                pass

create_training_data()

print(len(training_data_array))

import random

#to balance the training data input to the model
random.shuffle(training_data_array) #this will suffle the data and input to
the modle.

for sample in training_data_array[:10]: #checking if the shuffling is working
    print(sample[1])

#making the modle (pickles)
X = []
y = []

for features, label in training_data_array: #packeting the shuffled data to
arrays
    X.append(features)
    y.append(label)

print(X[0].reshape(-1, IMAGE_SIZE, IMAGE_SIZE, 1))

X = np.array(X).reshape(-1, IMAGE_SIZE, IMAGE_SIZE, 1)

y = np.array(y)

import pickle #saving the processed data inside a pickle

pickle_out = open("X.pickle","wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("y.pickle","wb")
pickle.dump(y, pickle_out)
pickle_out.close()
```

- Training the CNN Data Science Model

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard
import pickle
import time
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.333)
sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))

#importing the pickles we create
pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)

pickle_in = open("y.pickle", "rb")
y = pickle.load(pickle_in)

X = X/255.0 #normalizing the data by scailing, min is 0 and max is 255

dense_layers = [2]
layer_sizes = [64, 128]
conv_layers = [3]

for dense_layer in dense_layers:
    for layer_size in layer_sizes:
        for conv_layer in conv_layers:
            NAME = "paddyweedCNN-{}-conv-{}-nodes-{}-dense-
{}".format(conv_layer, layer_size, dense_layer, int(time.time()))
            tensorboard = TensorBoard(log_dir='logs\ {}'.format(NAME))
            print(NAME)

            # training the model
            model = Sequential()

            model.add(Conv2D(layer_size, (3, 3), input_shape=X.shape[1:]))  #
Convolutional layer
            model.add(Activation('relu'))  # activation layer
            model.add(MaxPooling2D(pool_size=(2, 2)))  # max pooling layer

            for l in range(conv_layer - 1):
                model.add(Conv2D(layer_size, (3, 3)))
                model.add(Activation('relu'))
                model.add(MaxPooling2D(pool_size=(2, 2)))

            model.add(Flatten())  # this converts our 3D feature maps to 1D
feature vectors

            for l in range(dense_layer):
                model.add(Dense(512))  # dense layer
                model.add(Activation('relu'))
                model.add(Dropout(0.2))

            model.add(Dense(1))
            model.add(Activation('sigmoid'))

            model.compile(loss='binary_crossentropy',
                          optimizer='adam',
                          metrics=['accuracy'])

#how many items per round
model.fit(X, y, batch_size=30, epochs=10, validation_split=0.3,
callbacks=[tensorboard])

#model.save('paddyWeedDetectionModelCNN.model') #saving the model

model.save('paddyWeedDetectorModelWithArch.h5') #saving the model with
architecture to be upload
```

- Testing the Data Science Model in the Console

```python
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import cv2
import tensorflow as tf

CATEGORIES = ["paddy", "weed"]  # will use this to convert prediction num to string value

def prepare(filepath):
    IMAGE_SIZE = 50  # 50 in txt-based
    image_grayscale_array = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)  # read in the image, convert to grayscale
    image_resize_array = cv2.resize(image_grayscale_array, (IMAGE_SIZE, IMAGE_SIZE))  # resize image to match model's expected sizing
    return image_resize_array.reshape(-1, IMAGE_SIZE, IMAGE_SIZE, 1)  # return the image with shaping that TF wants.

model = tf.keras.models.load_model("paddyWeedDetectorModelWithArch.h5")

prediction = model.predict([prepare('African-Lovegrass-BuckleyEcologyLab.jpeg')])  # PASSING A LIST OF THINGS YOU WISH TO PREDICT
print(CATEGORIES[int(prediction[0][0])])
```