A picture containing clipart

Description automatically generated

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 3](#_Toc39428510)

[ Data Set Importing and Pre-Processing 3](#_Toc39428511)

[ Training the CNN Data Science Model 5](#_Toc39428512)

[ Testing the Data Science Model in the Console 7](#_Toc39428513)

# 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

**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***])])***