import tkinter as tk

import matplotlib.pyplot as plt

from tkinter.filedialog import askopenfilename

from tensorflow.keras.models import Sequential

import cv2

from PIL import Image, ImageTk

from skimage.io import imshow,imread

from tkinter import \*

import pickle

from skimage import data

import tkinter

from tkinter import \*

from PIL import ImageTk, Image

from tkinter import filedialog

import os

from PIL import Image

from numpy import asarray

import numpy as np

from PIL import Image as im

import scipy.io

from skimage import color

from skimage import io

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import confusion\_matrix

from skimage import color

from skimage import io

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import numpy as np

import matplotlib.pyplot as plt

from tkinter.filedialog import askopenfilename

from tensorflow.keras.models import Sequential

import cv2

from skimage.io import imshow

from skimage.feature import hog

import os

import argparse

from tensorflow.keras.models import Sequential

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

from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense, Dropout

from tkinter.filedialog import askopenfilename

import numpy as np

import matplotlib.pyplot as plt

import cv2

from PIL import ImageTk, Image

from cv2 import \*

import random

from skimage import color

from skimage import io

import os

import numpy as np

import cv2

from matplotlib import pyplot as plt

import os

import numpy as np

import cv2

from matplotlib import pyplot as plt

# Position text in frame

# Create a photoimage object of the image in the path

# Resize image to fit on button

# Position image on button

root = tk.Tk()

root.geometry("1080x600")

root.resizable(width=True, height=True)

img = None

resized\_image = None

canvas = Canvas(root, width=1080, height=700)

import tkinter.messagebox

def openfn():

global filename

filename = askopenfilename(title='open')

return filename

def write\_slogan():

global img

x = openfn()

img =Image.open(x)

resized\_ima = img.resize((300,300))

img1 = ImageTk.PhotoImage(resized\_ima)

canvas.pack(pady=20)

# Add Images to Canvas widget

canvas.create\_image(210, 1, anchor=NW, image=img1)

panel = Label(root, image=img1)

panel.image = img1

panel.pack()

def Preproce():

global img

global resized\_image

global imgGray

import numpy as np

import cv2

from matplotlib import pyplot as plt

# PRE-PROCESSING

h1=300

w1=300

resized\_images = img.resize((h1,w1))

resized\_image = asarray(img.resize((h1, w1), Image.ANTIALIAS))

resized\_image1 = ImageTk.PhotoImage(resized\_images)

SP = np.shape(resized\_image)

try:

Red = resized\_image[:,:,0]

Green = resized\_image[:,:,1]

Blue = resized\_image[:,:,2]

plt.imshow(Red)

plt.title('RED IMAGE')

plt.show()

plt.imshow(Green)

plt.title('GREEN IMAGE')

plt.show()

plt.imshow(Blue)

plt.title('BLUE IMAGE')

plt.show()

except:

None

R, G, B = resized\_image[:,:,0], resized\_image[:,:,1], resized\_image[:,:,2]

imgGray = 0.2989 \* R + 0.5870 \* G + 0.1140 \* B

plt.imshow(imgGray, cmap='gray')

plt.title('GRAY IMAGE')

plt.show()

img = mpimg.imread(filename)

image\_input = img

image\_input = color.rgb2gray(image\_input)

(x1, y) = image\_input.shape

image\_input = image\_input.astype(float) \*255

plt.figure()

plt.imshow(image\_input)

plt.show()

# print(image\_input)

mu, sigma = 0, 0.1 # mean and standard deviation

key = np.random.normal(mu, sigma, (x1, y)) + np.finfo(float).eps

# print(key)

image\_encrypted = image\_input \* key

# imwrite('image\_encrypted.jpg', image\_encrypted \* 255)

plt.figure()

plt.imshow(image\_encrypted)

plt.title('Attacked')

plt.show()

image\_encrypted = image\_encrypted.astype('uint8')

images\_arr = im.fromarray(image\_encrypted)

image\_encryptedss = (images\_arr.resize((300, 300), Image.ANTIALIAS))

gray1 = ImageTk.PhotoImage(image\_encryptedss)

# canvas = Canvas(root, width=100, height=600)

canvas.pack(pady=1)

# Add Images to Canvas widget

img3 = canvas.create\_image(512, 1, anchor=NW, image=gray1)

panel1 = Label(root,image=gray1)

panel1.pack(side=tk.TOP)

panel1.image = gray1

panel1.pack()

plt.title('RESIZED IMAGE')

plt.imshow(resized\_image)

plt.show()

print('Preprocess completed...')

def gray():

global resized\_image

global imgGray

global filename

global image\_output

GRAY = resized\_image

import numpy as np

import cv2

from matplotlib import pyplot as plt

img = mpimg.imread(filename)

image\_input = img

image\_input = color.rgb2gray(image\_input)

(x1, y) = image\_input.shape

image\_input = image\_input.astype(float) \*255

plt.figure()

plt.imshow(image\_input)

plt.show()

# print(image\_input)

mu, sigma = 0, 0.1 # mean and standard deviation

key = np.random.normal(mu, sigma, (x1, y)) + np.finfo(float).eps

# print(key)

image\_encrypted = image\_input \* key

# imwrite('image\_encrypted.jpg', image\_encrypted \* 255)

plt.figure()

plt.imshow(image\_encrypted)

plt.title('Attacked')

plt.show()

image\_output = image\_encrypted / key

image\_output /= 255.0

# imwrite('image\_output.jpg', image\_output\*255)

plt.figure()

plt.imshow(image\_output)

plt.show()

# image\_output = image\_output.astype(np.uint8)

#

h = 300

w = 300

# image\_outputa = image\_output.resize((h,w))

# plt.figure()

# plt.imshow(image\_outputa)

# plt.title('sdsdf')

# plt.show()

#

####################################################################

#

# image\_encryptedss = image\_encrypted.resize((h,w))

# image\_encryptedaa = im.fromarray(image\_encrypted)

image\_outputaas = im.fromarray(resized\_image)

print(1)

image\_encryptedss = (image\_outputaas.resize((h, w), Image.ANTIALIAS))

gray1s = ImageTk.PhotoImage(image\_encryptedss)

MN\_val = np.mean(GRAY)

ST\_val = np.std(GRAY)

VR\_val = np.var(GRAY)

########################### CNN #######################################

test\_data = os.listdir('Samp/')

train\_data = os.listdir('Dataset/')

dot= []

labels = []

for img in train\_data:

try:

img\_1 = plt.imread('Dataset/' + "/" + img)

img\_resize = cv2.resize(img\_1,((50, 50)))

dot.append(np.array(img\_resize))

labels.append(1)

except:

None

for img in test\_data:

try:

img\_2 = plt.imread('Samp/'+ "/" + img)

img\_resize = cv2.resize(img\_2,(50, 50))

dot.append(np.array(img\_resize))

labels.append(0)

except:

None

x\_train, x\_test, y\_train, y\_test = train\_test\_split(dot,labels,test\_size = 0.2, random\_state = 101)

x\_train1=np.zeros((len(x\_train),50,50,3))

for i in range(0,len(x\_train)):

x\_train1[i,:,:]=x\_train[i]

x\_test1=np.zeros((len(x\_test),50,50,3))

for i in range(0,len(x\_test)):

x\_test1[i,:,:]=x\_test[i]

from keras.utils import to\_categorical

model=Sequential()

model.add(Conv2D(filters=16,kernel\_size=2,padding="same",activation="relu",input\_shape=(50,50,1)))

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

model.add(Conv2D(filters=32,kernel\_size=2,padding="same",activation="relu"))

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

model.add(Conv2D(filters=64,kernel\_size=2,padding="same",activation="relu"))

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

model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(500,activation="relu"))

model.add(Dropout(0.2))

model.add(Dense(2,activation="softmax"))#2 represent output layer neurons

model.summary()

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

y\_train1=np.array(y\_train)

train\_Y\_one\_hot = to\_categorical(y\_train1)

test\_Y\_one\_hot = to\_categorical(y\_test)

Features = [MN\_val,ST\_val,VR\_val]

import pickle

with open('Trainfea.pickle', 'rb') as fp:

Train\_features = pickle.load(fp)

y\_trains = np.arange(0,77)

from sklearn.neighbors import KNeighborsClassifier

neigh = KNeighborsClassifier(n\_neighbors=3)

neigh.fit(Train\_features, y\_trains)

Class\_KNN = neigh.predict([Features])

import numpy as np

import matplotlib.pyplot as plt

Labeled = y\_trains

Labeled[0:11] = 1

Labeled[11:21] = 2

Labeled[21:33] = 3

Labeled[33:44] = 4

Labeled[44:56] = 5

Labeled[56:67] = 6

Labeled[67:77] = 7

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

Name = Labeled[Class\_KNN]

if Name == 1:

print('Before Defence - Recognition Result')

print('Identified - Sea Image')

print(' ')

print('Recognition Result - After Defence')

print('Re-Labeled - Boat, Sea, Sky')

elif Name == 2:

print('Before Defence - Recognition Result')

print('Identified - Car Image')

print(' ')

print('Recognition Result - After Defence')

print('Re-Labeled - Traffic Image')

elif Name == 3:

print('Before Defence - Recognition Result')

print('Identified - Kitchen Image')

print(' ')

print('Recognition Result - After Defence')

print('Re-Labeled - Mug an Jars')

elif Name == 4:

print('Before Defence - Recognition Result')

print('Identified - Snow and Hill Image')

print(' ')

print('Recognition Result - After Defence')

print('Re-Labeled - Human (Couple Image)')

elif Name == 5:

print('Before Defence - Recognition Result')

print('Identified - House Image')

print(' ')

print('Recognition Result - After Defence')

print('Re-Labeled - Visitors place')

elif Name == 6:

print('Before Defence - Recognition Result')

print('Identified - Grass Land Image')

print(' ')

print('Recognition Result - After Defence')

print('Re-Labeled - Human Photography')

elif Name == 7:

print('Before Defence - Recognition Result')

print('Identified - Beach Image')

print(' ')

print('Recognition Result - After Defence')

print('Re-Labeled - Human, Beach and Objects')

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

if Class\_KNN == 1:

print('Identified - Non Attack')

else:

print('Identified - Attack')

print('Copy Move Attack is found...')

Name = Labeled[Class\_KNN]

resized\_image\_ref = mpimg.imread('Samp/'+str(int(Name))+'.jpg')

resized\_image\_ref = cv2.resize(resized\_image\_ref,(h,w))

my\_dpi = 60

IBG = resized\_image\_ref - resized\_image

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

from skimage import io, feature

from scipy import ndimage

import numpy as np

def correlation\_coefficient(patch1, patch2):

product = np.mean((patch1 - patch1.mean()) \* (patch2 - patch2.mean()))

stds = patch1.std() \* patch2.std()

if stds == 0:

return 0

else:

product /= stds

return product

#

sh\_row, sh\_col = resized\_image[:,:,0].shape

#

d = 1

correlation = np.zeros\_like(resized\_image)

correlation = correlation\_coefficient(resized\_image[:,:,0],

resized\_image\_ref[:,:,0])

plt.show()

print('Accuracy of Non Tamper = ',abs(correlation\*100), ' %')

correlation\_t = correlation\_coefficient(resized\_image[:,:,0],

IBG[:,:,0])

print('Accuracy Of Adversial / Tamper = ',abs(correlation\_t\*100), ' %')

image\_output = cv2.resize(image\_output,(h,w))

correlation\_tz = correlation\_coefficient(resized\_image\_ref[:,:,0],

image\_output)

print('Accuracy Of Defence Image = ',abs(correlation\_tz\*100), ' %')

ot.configure(text="Accuracy of Non Tamper: " + str(abs(correlation\*100)))

but.configure(text="Accuracy Of Adversial / Tamper: " + str(abs(correlation\_t\*100)))

but1.configure(text="Accuracy Of Defence Image: " + str(abs(correlation\_tz\*100)))

canvas.pack(pady=1)

# Add Images to Canvas widget

img3 = canvas.create\_image(210, 300, anchor=NW, image=gray1s)

panel1 = Label(root,image=gray1s)

panel1.pack(side=tk.TOP)

panel1.image = gray1s

panel1.pack()

print('Segmentation completed...')

#def feat():

#

#

# ot.configure(text="Features: " + str(Features))

#

#def classi():

# but1.configure(text="Accuracy is: " +str(ACC\*100)+ ' %')

def Close():

root.destroy()

# root.pack()

btn = tk.Button(root, text='Input image',width=25, command=write\_slogan)

# .pack()

btn.pack(side=tk.TOP)

btn.place(x=20, y=25)

btn = tk.Button(root, text='Attack Detection',width=25, command=Preproce)

btn.pack(side=tk.TOP)

btn.place(x=20, y=65)

btn = tk.Button(root, text='Reconstruction',width=25, command=gray)

btn.pack(side=tk.TOP)

btn.place(x=20, y=105)

#btn = tk.Button(root, text='Reconstruction',width=25, command=feat)

#btn.pack(side=tk.TOP)

#btn.place(x=20, y=145)

#btn = tk.Button(root, text='Performance',width=25, command=classi)

#btn.pack(side=tk.TOP)

#btn.place(x=20, y=185)

btn = tk.Button(root, text='QUIT',width=25, command=Close)

btn.pack(side=tk.TOP)

btn.place(x=20, y=225)

ot = Label(root, text="",font=("Arial Bold", 10))

# ot.grid(column=1, row=19)

ot.place(x=512, y=300)

but = Label(root, text="",font=("Arial Bold", 10))

# ot.grid(column=1, row=19)

but.place(x=512, y=350)

but1 = Label(root, text="",font=("Arial Bold", 10))

# ot.grid(column=1, row=19)

but1.place(x=512, y=400)

root.mainloop()