Younis Ghirfani

ghirfani.younis@gmail.com

Implementing Benford’s Law to Deep fake images

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.
When you’re ready to add your content, just click here and start typing.]

Technical Document

Younis Ghirfani

Contents

[Major Technologies 2](#_Toc70719386)

[Python 2](#_Toc70719387)

[Tkinter 2](#_Toc70719388)

[Matplotlib 2](#_Toc70719389)

[NumPy 2](#_Toc70719390)

[Skimage.io & Skimage.color 2](#_Toc70719391)

[Scipy.fftpack 2](#_Toc70719392)

[Collections 2](#_Toc70719393)

[Code Snippets 3](#_Toc70719394)

[Import statements 3](#_Toc70719395)

[GUI and variables declarations 4](#_Toc70719396)

[addFIles() 5](#_Toc70719397)

[Calculations() 5](#_Toc70719398)

[Save dct data 8](#_Toc70719399)

[DCT 8](#_Toc70719400)

[Threads 8](#_Toc70719401)

[Matplotlib 9](#_Toc70719402)

[Canvas and Tkinter 10](#_Toc70719403)

[Buttons and closing GUI 11](#_Toc70719404)

# Major Technologies

## Tkinter

Tkinter is a GUI based around python,

## Matplotlib

Matplotlib is also a GUI based package allowing creative graphs and displaying numerical values in a user-friendly way.

## NumPy

NumPy allowed me to use high value arrays effectively, it also worked alongside with matplotlib and Tkinter quite well. With NumPy I can save arrays to .txt files.

## Skimage.io & Skimage.color

Skimage.io gives many options when it comes to image processing, including imread, rgb2grey and dct. To find the dct coefficients, the image is turned to greyscale from 0,255. Once done dct is applied to the array of digits

## Scipy.fftpack

Scipy.fftpack allow me to use their dct package, which converts an array of numbers to the dct compressed standard set based on variables, this is the most important art of the project

## Collections

To count the number of numbers from 1-9 I used Counter

#

# Code Snippets

## Import statements

#Deep Fake Detection Tool

#Name: Younis Ghirfani

#Student number: C00229681

from array import array

import PIL.ImageTk

import PIL.Image

import cv2, cv2

from skimage.io import imread

from skimage.color import rgb2gray

from scipy.fftpack import dct, idct

import numpy as np

from numpy import fft, pi, sin, zeros, r\_, loadtxt, asarray, savetxt, empty,arange,exp,real,imag,pi

import matplotlib.pylab as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk

from matplotlib import pyplot as plt

from matplotlib.figure import \*

from numpy import empty,arange,exp,real,imag,pi

from numpy.fft import rfft,irfft

from collections import Counter, OrderedDict

from IPython.display import clear\_output

import tkinter as tk

from tkinter import \*

from tkinter import PhotoImage, filedialog

import os

from pylab import \*

from tkthread import \*

from tkinter import \*

from tkinter.ttk import \*

from scipy.fft import dctn, idctn

##Games

import pygame

##Threading

import logging

import time

from time import sleep

from threading import \*

#import matlab.engine

## GUI and variables declarations

# root is the windows window that the GUI will be based on

gui = tk.Tk()

gui.title("DeepFake Image Detection Tool")

gui.geometry("1200x700")

gui.resizable(width = False, height = False)

apps = []

global photo

global ArrayCounter

e = 0

new2 = 0

list1 = 0

number = 0

array\_length = 0

Total\_length = 0

Original\_Total\_Of\_Values = 0

positive\_number\_array = []

positive\_number\_float = []

positive\_number\_int = []

first\_number\_array = []

difference\_array = [0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1]

new =[]

ACounter = []

Number\_Of\_Numbers = []

Total\_Of\_Values = []

extra\_array = []

percentages = [0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1]

## addFIles()

# Function

def addFiles():

    canvas.delete("all")

    inputed\_Image = filedialog.askopenfilename(initialdir = "/", title = "Select Image", filetypes = (("Image Files","\*.jpeg;\*.jpg"), ("all files", "\*.\*")))

    img = PIL.Image.open(inputed\_Image)

    width, height = img.size

    width = str(width)

    height = str(height)

    print(width)

    print(height)

    img = img.resize((680,400))

    rgb\_im = img.convert('RGB')

    rgb\_im.save('inputedimage.jpg')

    photo = PIL.ImageTk.PhotoImage(rgb\_im)

    canvas.create\_image(15, 15, image = photo, anchor = tk.NW)

    canvas.create\_text(15, 425, anchor = tk.NW, fill = "blue", font = "Arial 15", text = "Width of inputed image: " + width)

    canvas.create\_text(15, 455, anchor = tk.NW, fill = "blue", font = "Arial 15", text = "Height of inputed image: " + height)

    canvas.get\_tk\_widget().pack(side = tk.BOTTOM, fill = tk.BOTH, expand = True)

    canvas.\_tkcanvas.pack(side=TOP, fill=BOTH, expand=True)

## Calculations()

def calculations():

    print()

    id = canvas.create\_text(15, 550, anchor = tk.NW, fill = "blue", font = "Arial 15", text = "loading.....")

    im = rgb2gray(imread('inputedimage.jpg'))

    imF = dct2(im)

    index = 0

    for i in range(len(imF)):

        for j in range(len(imF[i])):

            number = imF[i][j]

            #savetxt('text1.txt', imF, delimiter=',')

            extra\_array.append(number)

            rows = len(imF)

            columns = len(imF[0])

            Original\_Total\_Of\_Values = rows \* columns

            #Step 1

            positive\_number\_array = abs(number);

            if(positive\_number\_array >= 1):

                #Step 2

                positive\_number\_float = float(str(positive\_number\_array)[:1])

                #Step 3

                first\_number\_array.append(positive\_number\_float)

                ACounter = Counter(first\_number\_array)

                ACounter1 = ACounter[1]

                ACounter2 = ACounter[2]

                ACounter3 = ACounter[3]

                ACounter4 = ACounter[4]

                ACounter5 = ACounter[5]

                ACounter6 = ACounter[6]

                ACounter7 = ACounter[7]

                ACounter8 = ACounter[8]

                ACounter9 = ACounter[9]

                Number\_Of\_Numbers = [ACounter1,ACounter2,ACounter3,ACounter4,ACounter5,ACounter6,ACounter7,ACounter8,ACounter9]

                Total\_Of\_Values = sum(Number\_Of\_Numbers)

                print()

                print("Original Number:", number)

                print("Positive number array:", positive\_number\_array)

                print("Positive number float:", positive\_number\_float)

                print("Counter:", ACounter)

                print("Counter in order:", Number\_Of\_Numbers)

                print("Sum:", Total\_Of\_Values)

                #STEP 9

                #Find the percent of each value from 1-9 compared to the sum of the values

                index = 0

                for index in range(0,9):

                    per = (Number\_Of\_Numbers[index] / Total\_Of\_Values) \* 100

                    percentages[index] = per

                index1 = 0

                for index1 in range(0,9):

                    difference = (percentages[index1] / Benford\_y\_coords[index1]) \* 100

                    difference\_array[index1] = difference

                DCT\_Y\_coords = percentages

                line1.set\_ydata(DCT\_Y\_coords)

                complex\_graph.canvas.draw()

                complex\_graph.canvas.flush\_events()

                print("i: [",i,"] j: [",j,"]")

                plus = i + j / 245

                progress['value'] = plus

                gui.update\_idletasks()

                sleep(0.000001)

                print("plus:", plus)

                print("Original\_Total\_Of\_Values:", Original\_Total\_Of\_Values)

        print("difference\_array[0]", difference\_array[0])

        if(80 < difference\_array[0]):

            print("difference\_array[0]", difference\_array[0])

            canvas.itemconfig(id, text="This image is authentic ")

        if(0 < difference\_array[0] < 80):

            print("difference\_array[0]", difference\_array[0])

            canvas.itemconfig(id, text="This is a Deep Fake image")

## Save dct data

    a\_file = open("test2.txt", "w")

    for row in imF:

        np.savetxt(a\_file, row)

    a\_file.close()

## DCT

# implement 2D DCT

def dct2(a):

    #return dct(dct(a, norm='ortho'), norm='ortho')

    #return dct(dct(a,np.transpose),np.transpose)

    return dct(dct(a).T)

    #return dct(dct(a,np.transpose, norm='ortho'),np.transpose, norm='ortho')

    #return b

## Threads

################################################Thredding

def threadCalculations():

    t1 = Thread(target=calculations)

    t1.start()

    t1.join()

def test1():

    calculations()

class Hello(Thread):

    def run(self):

        for i in range(5):

            print("Hello")

            sleep(0.2)

class Hi(Thread):

    def run(self):

        calculations()

        t2.run()

t1 = Hello()

t2 = Hi()

t1.run()

sleep(0.1)

## Matplotlib

#Like body, its a square inside a square

complex\_graph = plt.figure("Benfords Law", figsize=(5,5))

Benford\_x\_coords = np.array([1,2,3,4,5,6,7,8,9])

Benford\_y\_coords = np.array([30.1,17.6,12.5,9.7,7.9,6.7,5.8,5.1,4.6])

plt.plot(Benford\_x\_coords, Benford\_y\_coords, color = "g", label = "Benfords Law")

DCT\_X\_coords = np.array([1,2,3,4,5,6,7,8,9])

DCT\_Y\_coords = np.array(percentages)

line1, = plt.plot(DCT\_X\_coords, DCT\_Y\_coords, color = "r", label = "DCT")

#plt.plot(DCT\_X\_coords, DCT\_Y\_coords, color = "r", label = "DCT")

plt.xlabel('First Digit Range')

plt.ylabel('Probability %')

plt.title('Graph of Benford Law and DCT Coefficents percentages ')

plt.legend(['Benfords Law', 'DCT'], loc="upper right")

## Canvas and Tkinter

##Orange canvas

canvas = tk.Canvas(gui, bg = "#8de879")

canvas.pack(expand = YES, fill = BOTH)

progress = Progressbar(gui, orient = HORIZONTAL,length = 1100, mode = 'determinate')

progress.pack()

##Place plot to canvas

Image\_canvas = FigureCanvasTkAgg(complex\_graph, canvas)

Image\_canvas.draw()

print (dct2([45,18,47,41,14,11,37,32]))

#img = PIL.Image.open('C:\Lenna.png')

#photo = img.resize((680,400))

#width, height = img.size

#width = str(width)

#height = str(height)

#canvas.create\_image(15, 15, image = photo, anchor = tk.NW)

canvas.create\_text(10, 300, anchor = tk.NW, fill = "blue", font = "Arial 20", text = "Please input an image ")

#canvas.get\_tk\_widget().pack(side = tk.BOTTOM, fill = tk.BOTH, expand = 1)

canvas.pack(side = tk.TOP, fill = tk.BOTH, expand = 1)

Image\_canvas.get\_tk\_widget().pack(side = tk.RIGHT)

Image\_canvas.draw()

## Buttons and closing GUI

# Creating Open image button and setting button to the canvas of the gui

openFile = tk.Button(gui, text="Open Image", padx=30, pady=5, fg="white", bg="#3209e6", command = addFiles).pack()

# Creating Run Application button and setting button to the canvas of the gui

runApp = tk.Button(gui, text="Run Application", padx=23, pady=5, fg="white", bg="#3209e6", command = test1).pack()

#root.resizable(False,False)

#root.update()

gui.mainloop()