Younis Ghirfani

Functional Specifications

[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.]

Table of Contents

[Abstract: 2](#_Toc70619495)

[Users 2](#_Toc70619496)

[Purpose of this project 2](#_Toc70619497)

[How it works? 2](#_Toc70619498)

[Input 2](#_Toc70619499)

[Processing 2](#_Toc70619500)

[Prepare for DCT Coefficient 2](#_Toc70619501)

[For Loops 3](#_Toc70619502)

[Positive Number 3](#_Toc70619503)

[First Digit 3](#_Toc70619504)

[Counter and Number of Numbers 3](#_Toc70619505)

[Percentage values 4](#_Toc70619506)

[Update Graph 4](#_Toc70619507)

[Metrics 4](#_Toc70619508)

[Possible functionalities: 5](#_Toc70619509)

[Cross platform across multiple Operating Systems. 5](#_Toc70619510)

[Video analysis 5](#_Toc70619511)

[Artificial intelligence 5](#_Toc70619512)

[The market for this product? 6](#_Toc70619513)

[Is there other application similar to this one? 6](#_Toc70619514)

[Use Case: 7](#_Toc70619515)

[Deep fake image detection tool: 7](#_Toc70619516)

[DCT Coefficients 8](#_Toc70619517)

# Abstract:

This Function Document with explain the inner working of my deep fake detection tool. The functionalities of the program and what can/can’t be done using it. While this project is mostly software based, I plan on making a website for exposure/ advertising if needed. This includes my perspective, the user’s perspective and the publics perspectives. It works by uploading an image and comparing it to Benford’s Law to check if its real or fake.

# Users

The consumer/customer

# Purpose of this project

My project is about a software application that anyone can download and use to verify if an image is real or fake, using Benford’s Law as a measurement tool. Some images will be edited more than others so there will be a fine line between real or fake. A user will submit an image they want to verify as real or fake, after which the program will deliver an answer

# How it works?

## Input

A window will appear asking the user to search in the pc’s local files and upload an image in the jpeg and jpg format. These formats allow compression on an image. The image is then checked for size and set as a variable name to allow access to it.

## Processing

## Prepare for DCT Coefficient

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

imF = dct2(im)

# implement 2D DCT

def dct2(a):

    #return dct(dct(a,type = 2, norm='ortho').T, norm='ortho')

    return dct(dct(a,type = 1,).T)

First the image is changed to grayscale to allow adjustments with DCT Coefficients. DCT is then applied to the image. Once completed the imF array is now an array holding all the dct digits

## For Loops

    index = 0

    for i in range(len(imF)):

        for j in range(len(imF[i])):

create a double for loop to process each dct number

## Positive Number

number = imF[i][j]

positive\_number\_array = abs(number);

Each number is made positive just to make selecting the next step easier

## First Digit

if(positive\_number\_array >= 1):

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

Benford’s Law works from 1-9, when applying dct it can create many 0’s number. This is skipped with a simple if statement. Most number will look like this 252.43245436543, I get just the first digit by adding the number 1 it means the first digit, 252.43245436543 turns to 2.0

## Counter and Number of Numbers

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)

Once the first digit is forum of a number it is then added to an array and then a counter is checking the number of numbers from 1-9. This is will be useful later when plot data is needed. Also, the sum is found of all the number processed excluding the 0’s. As the counter shows the values as highest number to least, I set each value to a variable to manage the order

## Percentage values

index = 0

for index in range(0,9):

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

      percentages[index] = per

Benford’s Law works by having a scale from 1-9 in a percentage amount. This for loop changes it so it will show the number of 1’s compared to the rest of the found digits.

## Update Graph

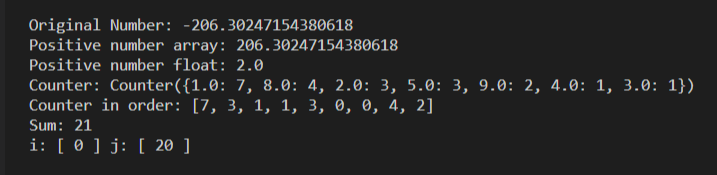
DCT\_Y\_coords = percentages

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

complex\_graph.canvas.draw()

complex\_graph.canvas.flush\_events()

Here, im setting the array of percentages as the new plot data, every number is processed and displayed on the graph live for the user. Here is an example:

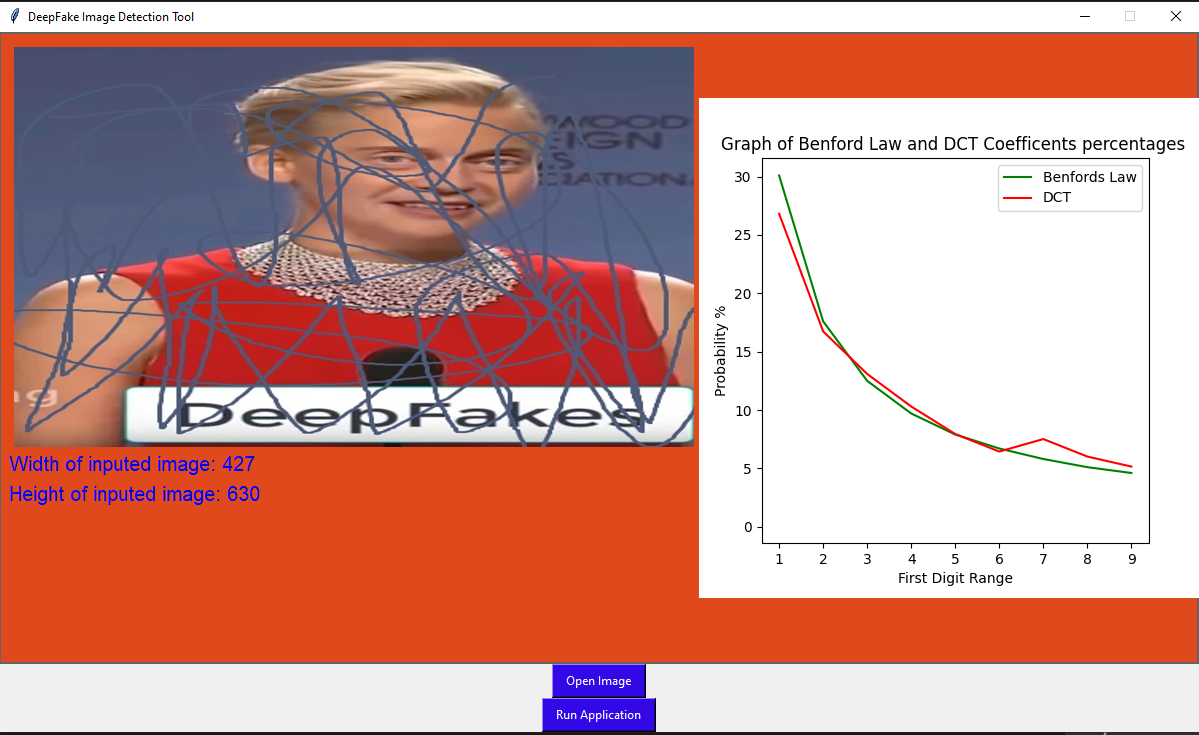


The for loop is then looped and the next number is added, there is approximately 270,000 numbers put image

## Metrics

To determine the image is real or fake, the dct Coefficients numbers compared to Benford’s Law graph line will be lower in the number 1.2 and 3 and even more. But the key is that there should be less 1’s compared to Benford’s law curve

## Print DCT Coefficient



Here, the image has been compressed enough to allow the number of 1’s to reduce, making it a deep fake image

# Possible functionalities:

## Cross platform across multiple Operating Systems.

Having the program run on Windows and Linux will allow most PC users to download it and use it. I predict most users will be Windows based but some may be Linux based too. Having a universal software between OS types will allow more exposure of the software

## Video analysis

I’m currently working on an image only style of detection, but deep fake videos also exist, and they may be used even more criminally then the images.

Display Images data like pixel size, quality, image format and copyrights ownership

Uploading an image will allow, the GUI to show the size of the image E.g. 1920 x 1080. Also, the format of the image E.g. JPEG, PNG etc. For copyright ownership, it will do search on the internet for a reference to the image and see if there is a clear owner, not maybe a URL link to the images

## Artificial intelligence

A.I being a commonly mentioned part of technology, it will allow more accurate reading to real of fake, by analysing the difference between the two measurement readings

# The market for this product?

Yes, as its s growing market and the technology being made for the deep fake images becoming more and more advanced. There will be a need for software with the same methods as this one or a different style of verifying its authenticity

# Is there other application similar to this one?

My first reference was a Netflix show called “Connected”, there was an episode called “Digits”. In this episode they discuss the use of Benford’s Law including using it as a deep fake image detection tool. These studies showed me that DCT Coefficients is the key to determine if the image is real or fake.

I used many Articles and thesis’s :

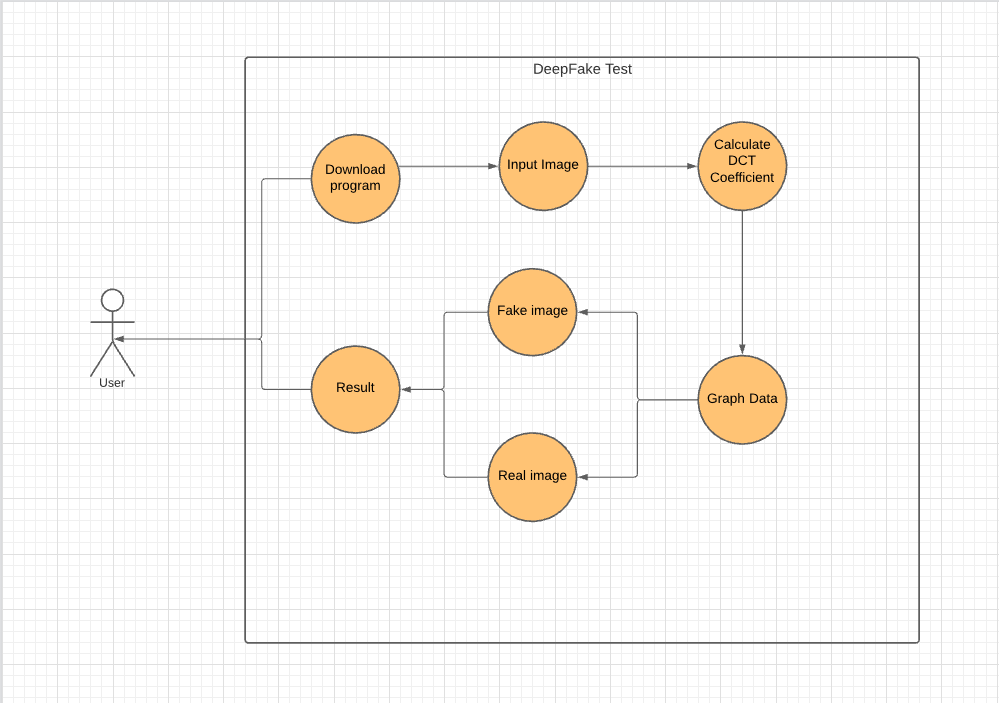
*A generalized Benford's law for JPEG coefficients and its applications in image*

*forensics - art. no. 65051L. By Y.Q Shi, Dongdong Fu and Wei Fang Su*

*Analysis f Benford’s Law in Digital Image Forensics. By, Abhay Kumar, Abhay Agarwal, Abhiskek Kumar Singh*

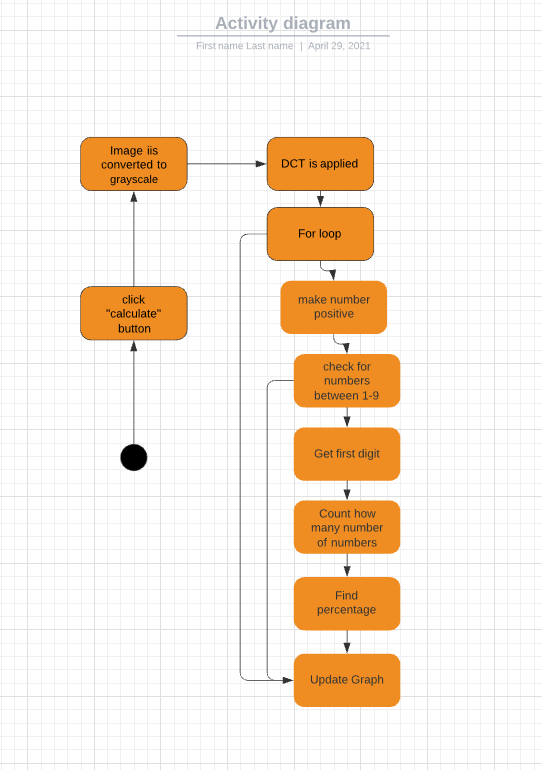
# Use Case:

## Deep fake image detection tool:



Once a few procedures, step by step the images go through processes to reach a Boolean true or false

## DCT Coefficients



A visually added diagram of how DCT Coefficients work with Benford’s Law.