C00229681

Younis Ghirfani

Research manual

Research manual

Utilizing Benford’s Law to detect deep fake images

Table of Contents

[Abstract 2](#_Toc70718297)

[Introduction 2](#_Toc70718298)

[Deep fake images 3](#_Toc70718299)

[What is Deep Fake images? 3](#_Toc70718300)

[Benford’s Law 5](#_Toc70718301)

[What is Benford’s Law? 5](#_Toc70718302)

[Benford’s Law Mathematics 5](#_Toc70718303)

[Uses of Benford’s Law 7](#_Toc70718304)

[Accounting fraud detection 7](#_Toc70718305)

[Election data 8](#_Toc70718306)

[Real vs Fake 8](#_Toc70718307)

[Jpeg Compression 9](#_Toc70718308)

[What is Jpeg Compression? 9](#_Toc70718309)

[How does JPEG compression work? 10](#_Toc70718310)

[Discrete Cosine Transform (DCT) 11](#_Toc70718311)

[What is DCT? 11](#_Toc70718312)

[How to use DCT? 12](#_Toc70718313)

[Decompression 15](#_Toc70718314)

[Python 15](#_Toc70718315)

[Tkinter 15](#_Toc70718316)

[Matplotlib 15](#_Toc70718317)

[NumPy 16](#_Toc70718318)

[Skimage.io & Skimage.color 16](#_Toc70718319)

[Scipy.fftpack 16](#_Toc70718320)

[Collections 16](#_Toc70718321)

[References: 16](#_Toc70718322)

[Paper Reference: 17](#_Toc70718323)

# Abstract

***Problem:*** Image can be altered to depict a target in a compromising scenario. These doctored images can be used to blackmail the target. These are called Deep fake images

***Objectives:*** Create a software program that verifies if an image has been altered an any way

***Methods:*** Using a mathematical equation called Benford’s Law, images can be determined to be real or fake

***Results:***

***Conclusions:***

# Introduction

This document outlines the research undertaken to create a deep fake image detection program. A deep fake is an image or video that has been doctored to depict a targets likeness that never occurred. This can be in a malicious way or something as simple as a social media filter, it depends on the intent of the creator. Research began by watching a tv show that was recommended by a project supervisor called “Connected” on Netflix. One of the episodes, describes the use of Benford’s Law in the detection of deep fake images.

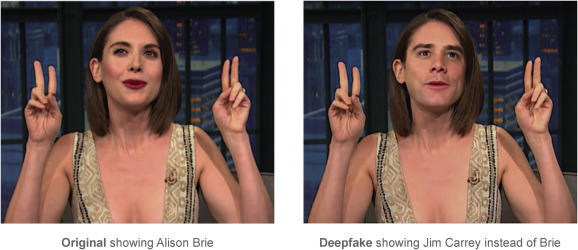
It features an image forensics expert called Hany Faird. He is recognised for assisting multiple international intelligence agencies. He is an image forensics professor at the University of California, USA .

Upon investigating his research, he used jpeg compression to show if an image has been edited from when it was captured to when it was saved on the same or different device. The idea is that when compressing images, detailed areas of the image will become doctored and changed which is an area a malicious user can add.

# Deep fake images

## What is Deep Fake images?

Deep fake images are the manipulation of images to show a different target on the image. While not always conducted in a malicious style, many times they are made with malicious intent.



Figure

As seen above Figure 1 has one original image and another image with the famous actor Jim Carrey edited above Alison’s face. A malicious user can use this technique to incriminate a person’s reputation and career.



Deep Fake real-world example

* In January 2019, Fox affiliate KCPQ released a deep fake of Donald Trump during one of his Oval Office addresses, mocking his appearance and his skin colour.



* In April 2020, the Belgian branch of Extinction Rebellion published a deep fake video of Belgian Prime Minister Sophie Wilmes on Facebook



Most cases are targeted towards high level individuals are in many ways

# Benford’s Law

## What is Benford’s Law?

Benford’s Law is a mathematical equation used across the world from tax fraud(not officially) to election data. It was discovered in 1881 when Simon Newcomb noticed that his logarithm tables were more were in the first few pages compared to the rest of the book. After this he though to check if it isn’t just for his logbooks but for starting numbers of any set of numbers. Only based off the first number from 1-9. He then made the equation: P = log(n + 1) – log(n)

After many years Frank Benford noticed the same phenomenon and decided to look into it, and he found a similar result. He tested the theory against the sizes of US populations, 1800 molecular weights, 5000 entries from a mathematical handbook, 342 persons listed on American men of science etc. It was later named after his name Benford. As years went on, it was verified and used in more practical ways across the world.

According to Benford’s Law the scale works like this:

## Benford’s Law Mathematics

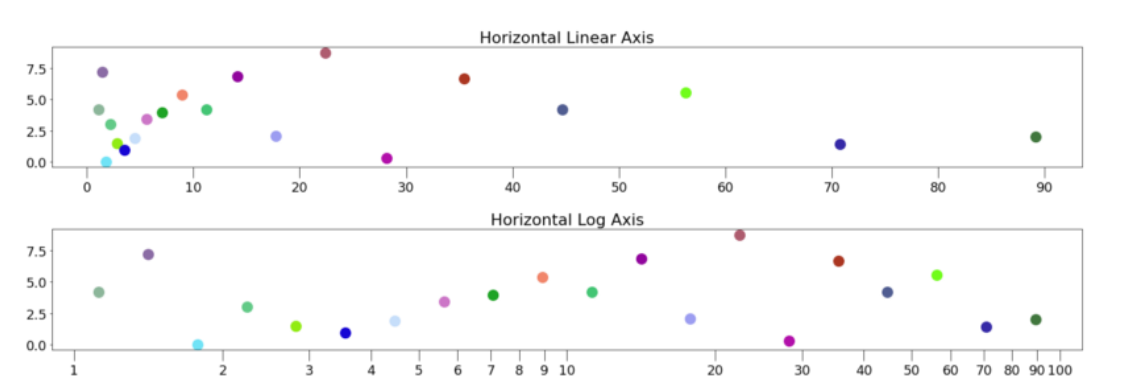
Frank Benford noticed that using log for a large amount of data causes it to be the same amount of data but in a smaller scale. To do this Bedford uses Logarithms to make his data in a scale he can work with. Logarithms is an inverse function of exponentiation if we take:

10 to the power of 4 it will be 1,000. So, the log(1,000) will give us 4.

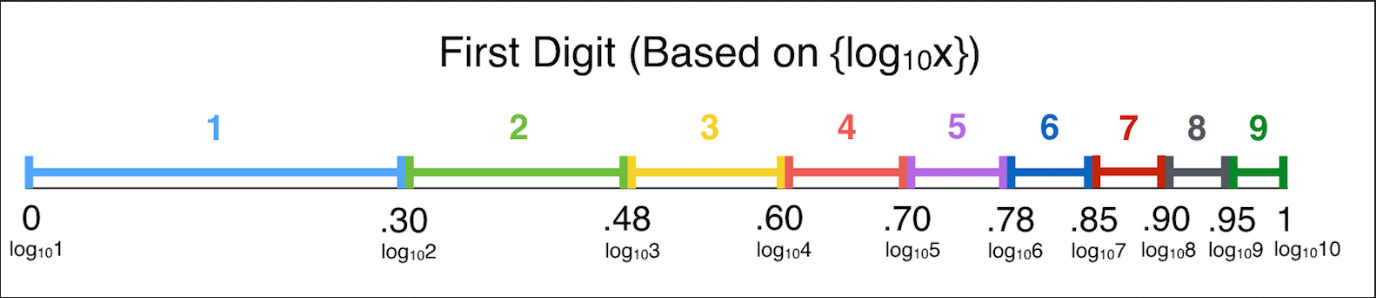
10 to the power of 8.7 will give us 10,000. So, the log(10,000) will give us 5.

Notice how using log a large number can become a log smaller, as log is the opposite of adding a

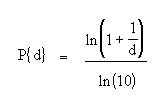
If you have a large amount of number let’s say you have a lot of data between 0 – 10 , but then smaller data across the board from 2 onwards to 100



These tables are an example, the different is that log10 has been applied which makes the distance between each input in the scale of each other. By doing this the Y axis is not scaled normally but is now scaled to the distance between the data. Notice 1 to 2 is now much wider as there is more data there compared to 5 – 6 or 7 – 8.



Here is Benford’s Law, and as you can see its quite similar to how the Log Axis scale looks like. Frank notices this and made an equation is describe it

[](https://www.statisticshowto.com/wp-content/uploads/2016/07/benford-formula.png)

Or in simpler terms:

P = log(*n* + 1) – log(*n*)

P = log(n + 1) – log(n)

P = log(8 + 1) – log(8)

P = 0.9542425094 - 0.903089987

P = 0.05115252245

Probability of the first digit being 8 is 5.11%

We can do an example:

Using some excel, I added the population of the cities of the USA and applied Benford’s first digit Law, in some places Benford’s Law works well but in other not so much. Benford Law has a weakness and that’s the sample size much is quite a size for it to be accurate. Or the Law simply doesn’t work with the data.

## Uses of Benford’s Law

Benford’s Law can be used for multiple applications from:

### Accounting fraud detection

One of the main uses for Benford’s law is applying it to accounting/tax fraud. When a person tries to alter a company’s books it will find a lot more ones and twos compared to other. The reason this works is because humans are of habit so when we made “random” numbers it is actually connected to who we are which creates patterns. When comparing it to Benford’s curve it will show either a huge difference or a small difference when it comes to the curve gradient

### Election data

Elections is a long and complicated process and has been rigger many times in the past, in 2008 Benford’s law was used to defraud the Iranian election. The only deviation was a 0.5% deviation but because of how many votes were made it creates a huge range for Benford’s law to handle. Even a deviation of 0.5% is concerning it shows how accurate Benford’s Law can be.

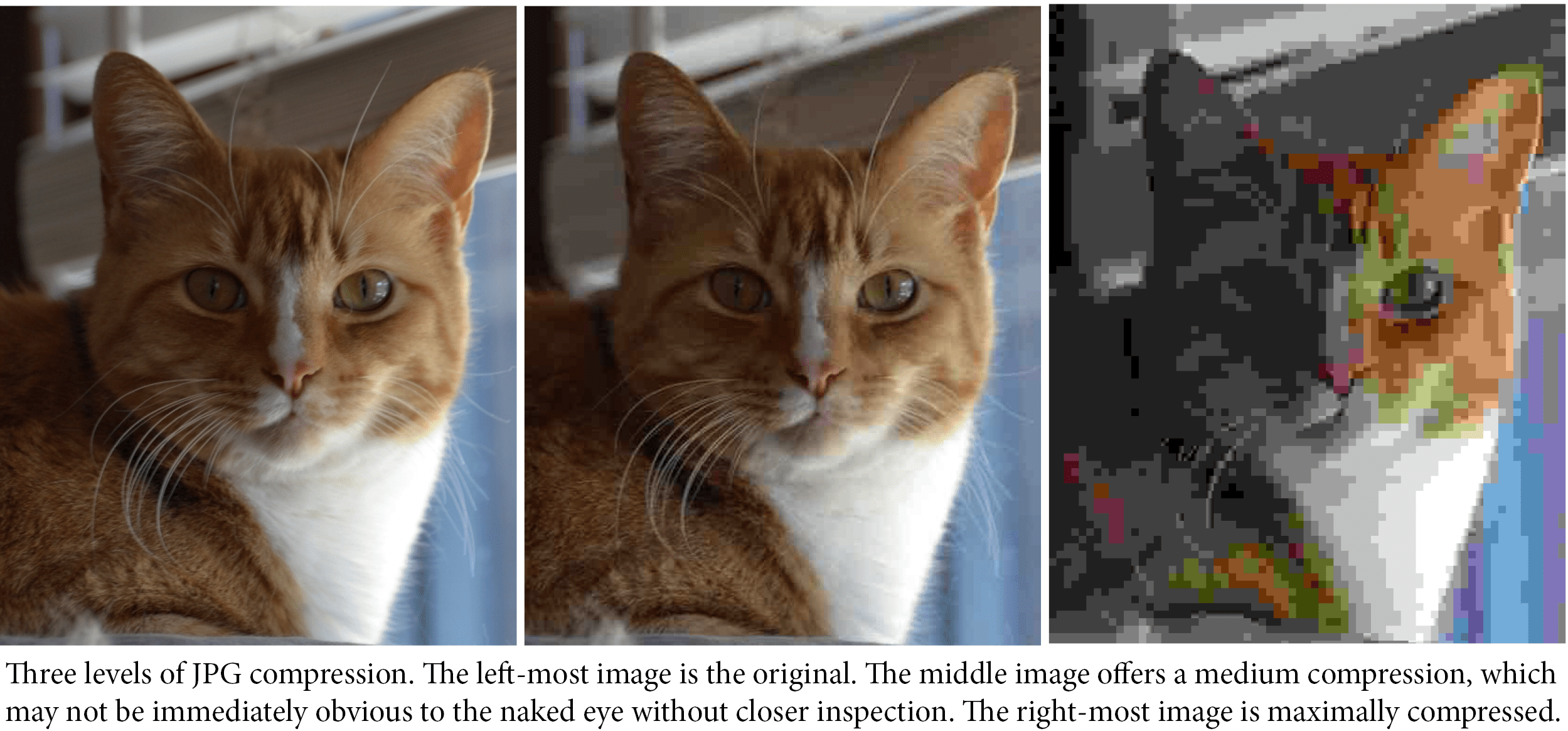
Recently there was a presidential election in America and following the predicted winner being Biden, trump and his administration has come out and said the election was rigged to Biden favour. As you have read before, Benford’s Law has been used to check if previous elections, but for the most recent election experts are saying it’s not rigged.

There is also many people who believe that Benford’s law doesn’t work for elections as its either x or y Biden vs trump which doesn’t work for a 1-9 based number formula, either way it is used and tested on a lot of major elections.

## Real vs Fake

To determine the different between a real image and a fake image is calculated by applying Benford’s Law with JPEG to determine if an image has been saved multiple times.

When an image is taken and saved, it is compressed using JPEG once. This means it hasn’t been edited / modified. If a user wants to apply deep fake techniques to an image, they will need to save that image again, which will cause it to compress the image again. This makes the image less quality, but more importantly Benford’s Law will detect that the compression has occurred twice



Here is an example, the difference is very little, but the first image is compressed with the other image being compressed again. Visually there isn’t much difference, but it the image is zoomed in small difference can be made.

# Jpeg Compression

## What is Jpeg Compression?

Jpeg compression is a commonly used method of lossy compression a digital image (Hanks, 2013)

First started in 1986, a team called the Joint Photographic Experts Group (JPEG) stated experiment with different types of Jpeg compression through the 1980’s. Why use jpeg compression? the main reason is storage, as higher resolution images are getting more reasonable it makes the images bigger in size which cause storage issues for SD cards to hard drives that stores thousands of images such as a social media site or your local hard drive with pictures from last summer’s holiday

|  |  |  |
| --- | --- | --- |
| File type | 3 MP image | 15 MP image |
| JPEG 100% - 24-bit RGB | 2.6 Mb | 10.2 Mb |
| JPEG 97% - 24-bit RGB | 1.2 Mb | 4.5 Mb |
| JPEG 75% - 24-bit RGB | 0.5 Mb | 1.8 Mb |

For context, the 100 , 97 and 75% is the quality mark after compression. obviously the more it compressed the more lose in quality it does to the image



As you can see even the 50 quality is still a good photo by no means, under 50 quality it does mess with the colours and textures. But after being Compressed too much the textures and contrast become tainted.

Having high-res images can be as big as 68mb which in terms of downloading is a lot, especially with 1080p 60Fps video. Having high resolution images are good but they carry the issue of hard drive space and upload time from downloading image catalogues to saving months of photograph which can be 1000’s of images that needs to be saved. Now a days with better grade cameras it’s not uncommon to see 2160 pixels of an image to 4k or even 8k images that need to be stored.

## How does JPEG compression work?

Colour Transform

DCT

Image

JPEG

Encoding

Quantization

**Image:** The raw freshly captured image

**Colour Transform:** Using YCbCr the image is converted to a grey scale type image, which saves a lot of mb storage space

**DCT:** depending on the image and type of DCT, an images values are changed using cosine waves. Allowing the images to be compressed using different values without the image changing too much to where human eyes will see the difference

**Quantization:** Using a quantization table, divide the dct coefficient value by the quantization value. This creates a long string of number that if similar can be compressed even more using encoding

**Encoding:** Huffman encoding is then used on the resulting numbers to further compress the data that is in a string

**E.g.** -23-2-2164002110-10000000000000000000000000000000000000000000000000000

Converts to: 22669B4D26498CB660C832CA10F295AFFFFFFFFFFFFF

**JPEG:** Image is then created from the final string of numbers

This process is repeated to each 8x8 block of the image

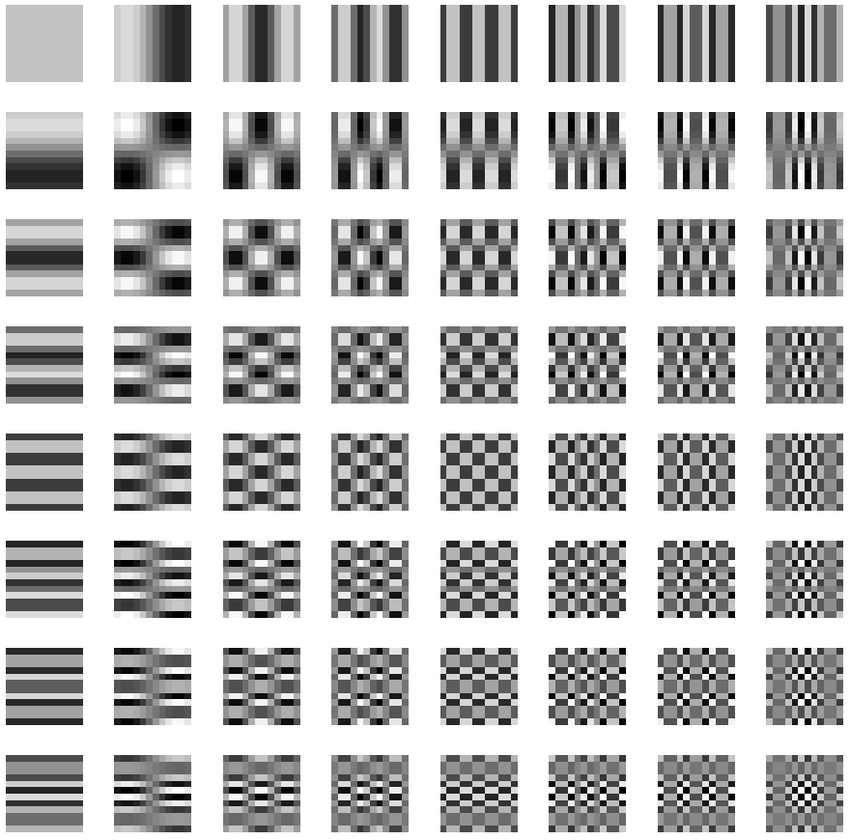
# Discrete Cosine Transform (DCT)

## What is DCT?

Discrete cosine transform is a sequence of data points converted from an initial value, it is commonly used in signal processing, signal compression, digital television, digital radio and partial differential equations. Jpeg uses DCT to compress images efficiently and quickly. For this project DCT is a crucial part of how the detection process is made

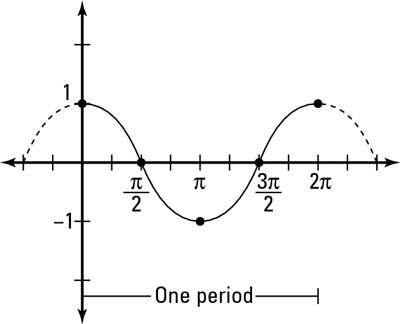
DCT ranges from DCT-I to DCT VIII. As JPEG uses DCT – II, that’s the area research will be focused on.

Here is a DCT table:

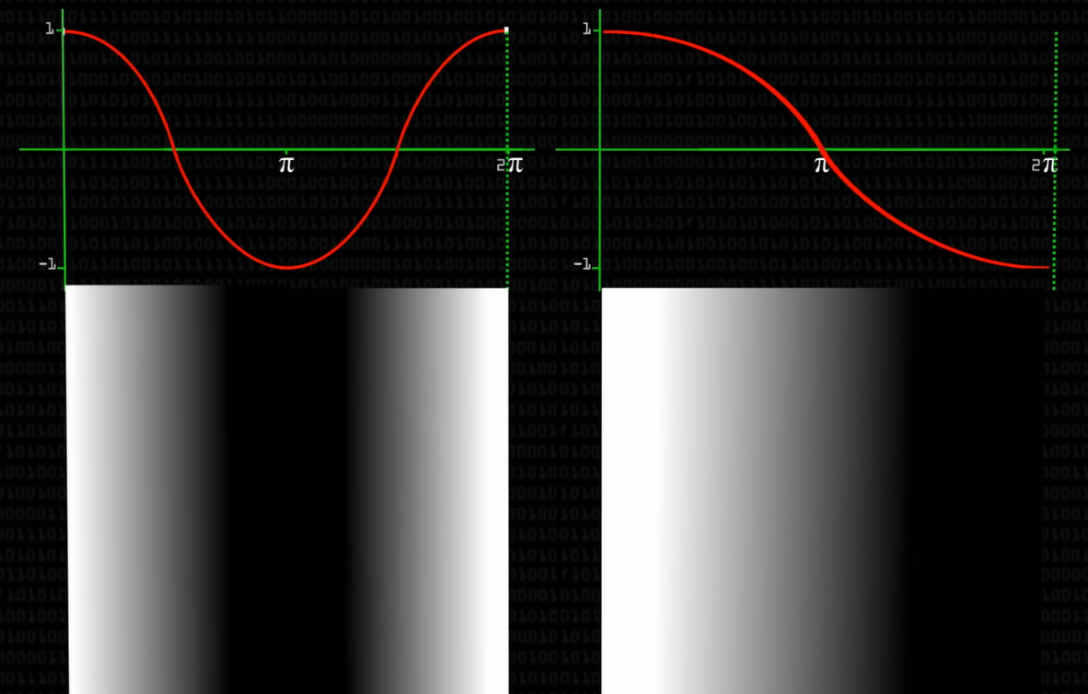


## How to use DCT?

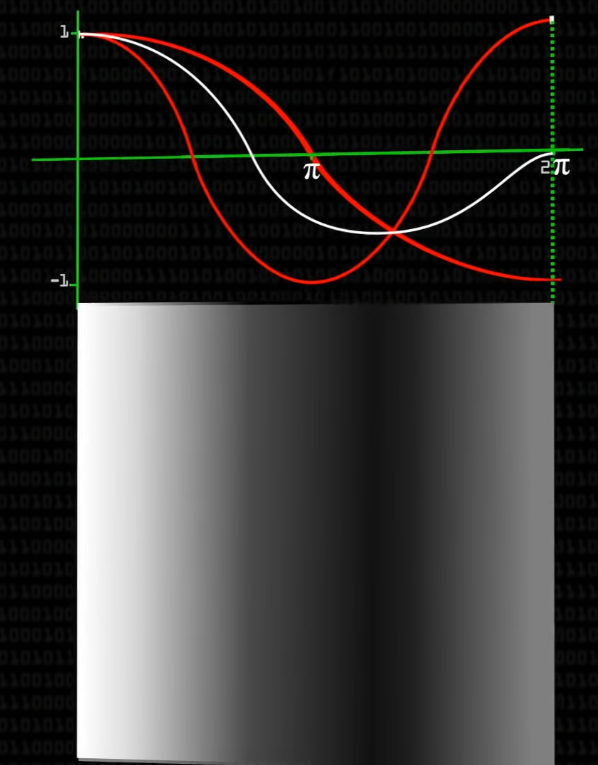
To make an image ready for compression, DCT will be applied to it so the result will still make an image, not quite the same image. But that image can be compressed due to its new values. To get these digits, coefficient is applied to the image. A coefficient means it will check the 8x8 block of pixels and check how much each part of the DCT table contributes to that block, it is then centre all the values to 0 because the graph is centred from -1 to 1. This is done by -128 from each pixel value. Here DCT -2 is now applied.



Here is coefficient graph showing an example of one block taken from the DCT table, each DCT square has a graph representing that part of the DCT table.

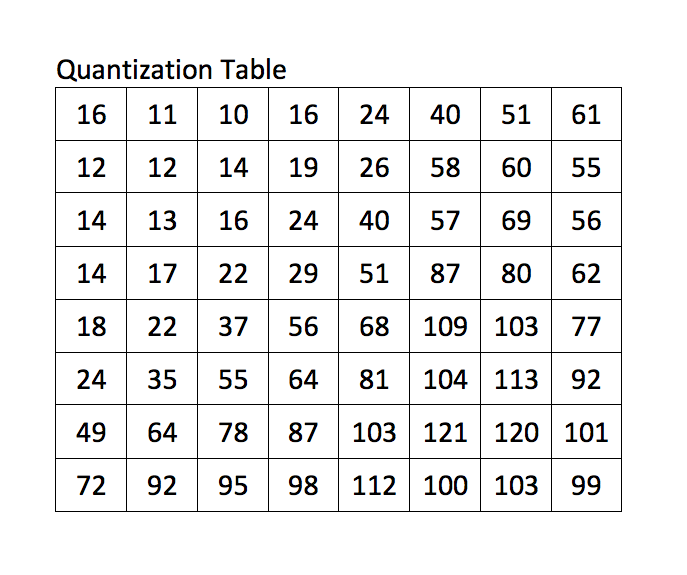


The lower the graph goes to -1 the darker the image gets on that side. Using coefficients each pixel will be a combination.

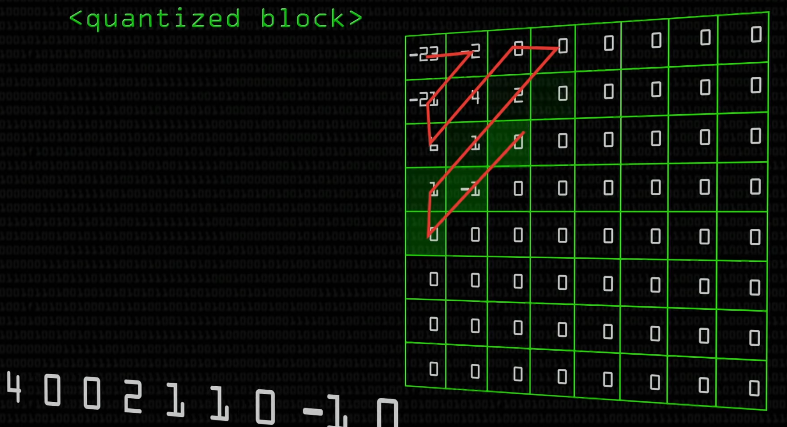
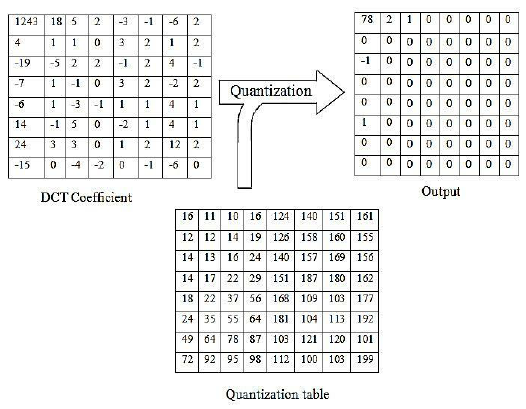


As the graphs combine the pixel will get either darker or lighter, coefficient will calculate the appropriate value. Each pixel will be a combination of all if not most DCT table types, from completely white or a type of checkered image.

A JPEG quantization table of E.G., 50 is applied to the digits, meaning each number is divided by the table and rounded to the nearest digit. As high frequency parts of the image is mostly used it will have higher digits, while the bottom right side of the image may have 20 0’s or 2 0’s depending on the image



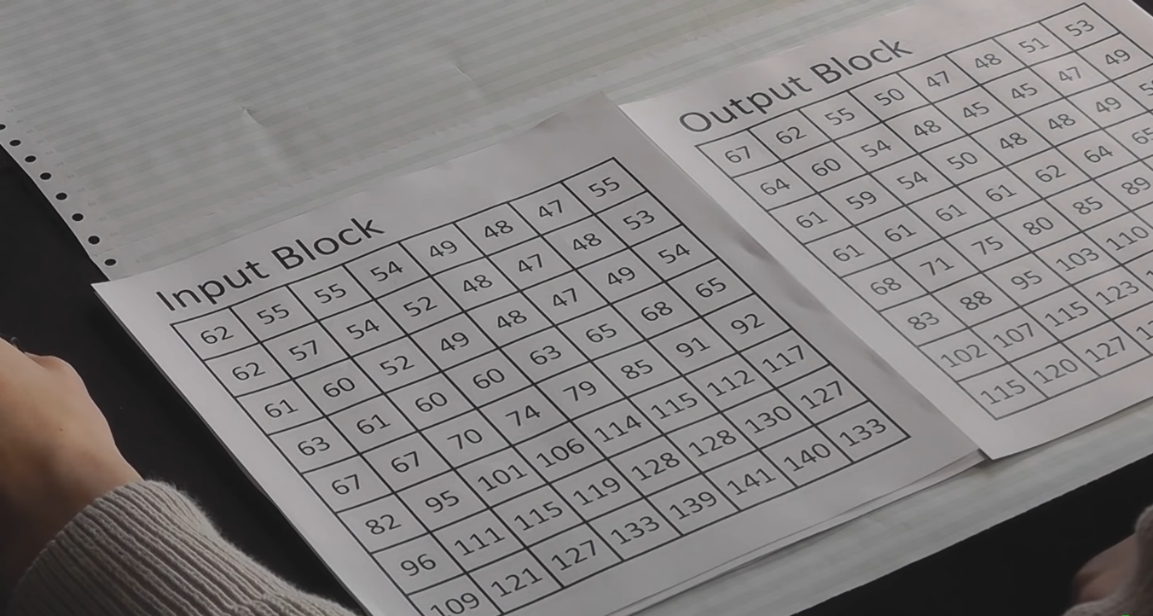
This is the default quantization table, there is different types based off ow much compression is wanted



This will create a string of numbers used that will be compressed using Huffman encoding. This will reduce the length of the string value. Once compressed that string of numbers is used to represent one pixel the in the JPEG image. This is repeated for each pixel from the 8x8 block, then each 8x8 block of the entire images

## Decompression

Each pixel is multiplied with the previous quantisation table used before, that is stored in the image metadata. Once completed, 128 will then be added to each number as 128 was removed previously to align with the DCT table.



The image values won’t be the same as the original digits used before decompressed is not the true values, but more a shorter value for each pixel

# Python

## Tkinter

Tkinter is a GUI based around python, I found it to be the most user friendly with some trouble with text placement and some threading problems, most importantly it gave an entry level user of python to understand how it works compared to other coding languages. Using PhotoImage and filedialog the user can add an image to the program and see the result. I found it difficult to place the matplotlib onto the tkinter GUI until I used Tkinter. Canvas as a buffer between the two packages and it worked correctly

## Matplotlib

Matplotlib is also a GUI based package allowing creative graphs and displaying numerical values in a user-friendly way. Using line1.set\_ydata() I can update consistently the graph and let the user know of changes in the calculation and a final result. This was only possible because of Tkinter canvas

## 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

# References:

**Benford’s Law:**

[**https://en.wikipedia.org/wiki/Benford%27s\_law**](https://en.wikipedia.org/wiki/Benford%27s_law)

[**https://www.statisticshowto.com/benfords-law/**](https://www.statisticshowto.com/benfords-law/)

[**https://www.dcode.fr/benford-law**](https://www.dcode.fr/benford-law)

(Hanks, 2013)

(Scipy DCT, n.d.)

(matlab - Issiue with implementation of 2D Discrete Cosine Transform in Python - Stack Overflow, n.d.)

(Image-Processing/DCT.py at master · anantham/Image-Processing, n.d.)

(Two-dimensional lists (arrays) - Learn Python 3 - Snakify, n.d.)

(python - img = Image.open(fp) AttributeError: class Image has no attribute 'open' - Stack Overflow, n.d.)

(How to Place Matplotlib Charts on a Tkinter GUI - Data to Fish, n.d.)

(Graph Plotting in Python | Set 1 - GeeksforGeeks, n.d.)

(matplotlib scatter plot annotate / set text at / label each point | by pythonmembers.club | Medium, n.d.)

(opencv - How do I apply a DCT to an image in Python? - Stack Overflow, n.d.)

(Jpeg: Colorspace Transform, Subsampling, DCT and Quantisation — Multimedia Codec Excercises 1.0 documentation, n.d.)

(JPEG DCT Demo, n.d.)

(YCbCr - Wikipedia, n.d.)

(Python - scipy.fft.dct() method - GeeksforGeeks, n.d.)(Benford's Law - How mathematics can detect fraud! - YouTube, n.d.)

(Revealing the Traces of JPEG Compression Anti-Forensics | Request PDF, n.d.)

(18 Impressive Applications of Generative Adversarial Networks (GANs), n.d.)

(2004.07682.pdf, n.d.)

(Image Generation in 10 Minutes with Generative Adversarial Networks | Towards Data Science, n.d.)

# Paper Reference:

<https://www.spiedigitallibrary.org/conference-proceedings-of-spie/6505/65051L/A-generalized-Benfords-law-for-JPEG-coefficients-and-its-applications/10.1117/12.704723.short>

(A generalized Benford's law for JPEG coefficients and its applications in image forensics, n.d.) (image compression and the discrete cosine transform, n.d.)

<https://abhayk1201.github.io/files/EE604_tp.pdf>

(Analisis of Benford's Law in Digital Image Forensis, n.d.)

(image compression and the discrete cosine transform, n.d.)

**Logarithm:**

[**https://en.wikipedia.org/wiki/Common\_logarithm**](https://en.wikipedia.org/wiki/Common_logarithm)

[**https://en.wikipedia.org/wiki/Logarithm**](https://en.wikipedia.org/wiki/Logarithm)

[**https://en.wikipedia.org/wiki/Logarithmic\_scale**](https://en.wikipedia.org/wiki/Logarithmic_scale)

**Charts:**

**https://towardsdatascience.com/benfords-law-a-simple-explanation-341e17abbe75**

[**https://www.infoplease.com/us/states/state-population-by-rank**](https://www.infoplease.com/us/states/state-population-by-rank)

**Jpeg Compression:**

**https://en.wikipedia.org/wiki/JPEG**

[**https://uk.mathworks.com/help/images/jpeg-image-deblocking-using-deep-learning.html**](https://uk.mathworks.com/help/images/jpeg-image-deblocking-using-deep-learning.html)

[**https://kinsta.com/blog/jpg-vs-jpeg/**](https://kinsta.com/blog/jpg-vs-jpeg/)