Evaluating the Relevancy of JFIF Image Compression on the Internet

Uzen

Jul 22

`this is a DRAFT!!!!!!`

Contents

1 Introduction

Compression is the cornerstone of the internet. Behind the millions of bytes flowing between computers everyday around the world, compression algorithms are behind every single data packet. Compression algorithms are tools that can seemingly magically shrink a digital document, sometimes until they are a tenth of their size. Due the many uses of compression algorithms, they have been used in many different fields. From shrinking large backups for storage to speeding up the transfer of data across the web, compression algorithms are crucial to modern computing.

One specific field that I was interested in was website hosting. A personal website can be a good representation of themselves, similar to a more polished social media page. For a period of time, I was quite interested in starting my own website, renting virtual servers from Aliyun to start my project. However, I quickly ran into an issue. In the modern times, technology has progressed at an exponential rate, storage and bandwidth speeds are increasing at an extraordinary rate. It is common to see computers with hundreds of gigabytes of storage and gigabit connection to the internet. So it was quite a surprise for me when I found out that adding bandwidth and storage to my server costed money. A small, affordable server would only come with 10 to 50 gigabytes of storage a measly 1 megabit bandwidth connection (or purchase bandwidth by the gigabit used).

Due to this limitation, the website I made was slow. I couldn't upload many images or videos because the server would run out of storage, or make the website load at a snail's pace. This made me very careful about the files I upload, especially the sizes of image and videos. My curiosity eventually lead me to image compression. When an image is stored, there's different file types: `.png, .jpg, .gif`. These are actually different methods of storing images, using different methods image compression techniques. When creating my website, I had to choose what method of image compression to use. Each different method has its advantages and disadvantages.

One format of compression I am very interested in is JFIF compression (commonly found in JPEG files). JFIF uses a mixture of clever math and the characteristics of the human eye to shrink photos. It is commonly used in digital cameras. However, JFIF was originally invented over 30 years ago (August 1990 (Joint Photographic Experts Group)), and have not account for the many modern technological advances. For example, JFIF is bad at dealing with digital art and digital text due to limitation of the compression techniques. In the recent years, newer and more efficient image compression algorithms have been created, such as Webp. In fact, JPEG has released a newer version of JFIF, called JPEG-2000. Some has even used Artificial Intelligence to enhance compression rates (Johnston and Minnen).

Nevertheless, JFIF compression still remains as a dominant image compression technique. This paper aims to compare JFIF with modern image compression formats to evaluate whether it is still a competitive algorithm. The results of the paper would help website creators decide the best format to store images on their websites.

2 Background Information

2.1 Lossless and Lossy Compression

There are two major types of compression algorithms: lossy and lossless.

2.1.1 Lossless Compression

Lossless compression finds repetitive, similar patterns inside a file, condenses the data to the smallest possible size. One example of lossless compression is LZ77 (Brailsford). Commonly used on plain text documents[1], LZ77 finds repeated words, and stores it for reference. The next time the repeated word is used, LZ77 stores a pointer that points towards the last instance of word. In this paragraph, the word `compression` is used 5 times. LZ77 will only store the first `compression`, and save space from the rest.

Lossless compression is good for data with a lot of repetition. It also preserves the data it compresses, which lossy compression cannot. On the other hand, lossless compression is less efficient than lossy compression.

2.1.2 Lossy Compression

Lossy compression finds and remove redundant data. Lossy compression is used often in audio and video data. Due to its nature, lossy compression techniques varies greatly depending on the input data. While some lossless compression algorithms are generic, lossy compression algorithms cannot be interchange across different data types.

Lossy compression is more efficient than lossless compression. But it can degrade the quality of the data, as some data is thrown away. It is not suitable for all types of data. For example, we cannot use lossy compression on text, as every character matters in a sentence.

2.2 Storing Images

A pixel is a square block with a specific color. When a grid of pixels are placed together, it forms an image. An image is simply a 2D array of pixels. And each pixel can be stored as color. This method of storing images is called `bitmap`. `Add cool diagram showing pixels and image.`

Color is generally represented in a color space[2]. One commonly used color space is `RGB`[3], which stores the mixture of red, green, and blueness of a color. Imagine that a color is composed by shining red, green, blue light on a black wall. Where the number associated with each value

[1] `.txt` files
[2] method of storing color as data
[3] find rgb iso documents

File Type	Extension	C. Method	C. Strength	Uses Case	Availability	Cons
JFIF	.jpg	lossy	medium	still images	high	can cause compression artifacts
PNG	.png	lossless	medium	high quality images	high	worse compression than lossless compression
GIF	.gif	lossless	low	animated graphics	high	low performance
AVIF	.avif	both	high	all	medium	basic support on some web browsers
WebP	.webp	both	high	all	high	slightly worse compression than AVIF
TIFF	.tiff	lossless	low	high quality images	medium	support on some browsers only

`split two tables, add full name`

is the brightness of the light. The resulting image would be stored as a 2D array, each element containing a RGB value. `Add cool image of RGB. Add table to demonstrate.`

An alternative method of storing images is as mathematical equations. Images created this way are called `vector images`. Vector images have widely different use cases then bitmap images. However, this image type is severely limited at depicting complicated images, and is only suitable for digital art. Thus, this paper will work with bitmap images only.

2.2.1 Need for Compression

Storing images as arrays is fast and convenient, however it also results in large files. For example, a 1920 by 1080 image would take up 5.8MiB of data. `add in appendix`. Although this seems small, considering that most webpages have only kilobytes of data, images occupy a large amount of space. `Find website (reddit, amazon, apple, CSDN) to do example, use appendix to show calculations`. Videos are stored as a series of pictures. Using the same metrics as above, a 10 minute video would take up 104GiB of space. Uploading this to a websites would be extremely impractical.

2.3 Image Formats

Modern computer systems compresses images by default. Different images files have been developed with different algorithms and uses in mind. Below is a table of common image file types.

2.4 Websites

Since this paper focuses on the use of compression on the internet, this subsection will briefly introduce the concepts of websites. Websites are hosted on rented cloud servers, with limitations on bandwidth, storage, processing power, and memory. Images are rendered client side by web browser, thus processing power and memory is irrelevant. Images require storage on the cloud server, large images may reduce the size of the website. When serving a website, bandwidth may limit the speed of transfer. Large images can increase loading times. Web developers have to balance image quality with image size when creating a website.

Due to the many available web browsers, the standards for web development varies. Web browsers support different image formats. The most common web browser is Google Chrome[4], whilst many web browsers are based on Chrome. When displaying an image, web browsers have to decompress, taking up processing power. Although processing power is abundant on modern desktops, mobile devices and older computers may suffer. Web developers should be aware of the requirements of different image formats.

[4] requires citation

3 JPEG Methodology

JFIF compression take advantages of two assumptions: (Townsend)

1. Humans are more sensitive to brightness than color.
2. Humans are not sensitive to high frequency contents.

The first step in the algorithm is to transform the color space from `RGB` to `YCbCr`. `Chroma Subsampling` (3.1) is then performed on the image. The image then undergoes `2D Discrete Cosine Transform` (3.2), and is `quantized` (3.3). Lastly, the resulting data matrix are compressed with `Huffman Encoding` (3.5).

This section will only detail the steps to compression. As the steps to decompressing is simply the reverse of the compression process.

3.1 Chroma Subsampling

Since the human eye is less sensitive to color than brightness, JFIF can downsample[5] the color element of an image. The commonly used `RGB` colorspace is convenient for projecting images, but does not separate the color from the luminance[6] of an image, `YCbCr` is used instead. YCbCr separates color into `Y` (luminance, brightness of a pixel), `Cb` (chroma blue, blueness), and `Cr` (chroma red, redness). Chroma green is not stored, as it can be calculated from Cb and Cr (LEARN MEDIA TECH). The resulted image will be a matrix of pixels with color data stored in YCbCR.

3.2 Discrete Cosine Transform

Discrete Cosine Transform (DCT) separates an image into frequency components. DCT is a subset of Fourier Transformation (Pound, "JPEG DCT, Discrete Cosine Transform (JPEG Pt2)- Computerphile"), which takes a time based function and transforms it into the frequency and amplitude components. This is performed by first splitting an image into small 8 by 8 pixel chunks, then performing the algorithm. Although the DCT used on imaged are 2 dimensions, we can explain the concept from a 1 dimension perspective first.

Lets take a single row of 8 pixels, and plot them on a graph with pixels on the x axis and intensity on the y axis. The resulted data points forms a signal. If we were to recreate the curve mathematically, we can plot a curve through the data points. DCT reconstructs the signal by stacking differently weighted cosine curves of different frequency and amplitude.

For example, figure 1 shows a cosine curve with a period of 8 pixels. Lets say this curve has a frequency of 1 unit. The purple coordinates represents the resultant signal.

[5] reduce the size of an image
[6] brightness

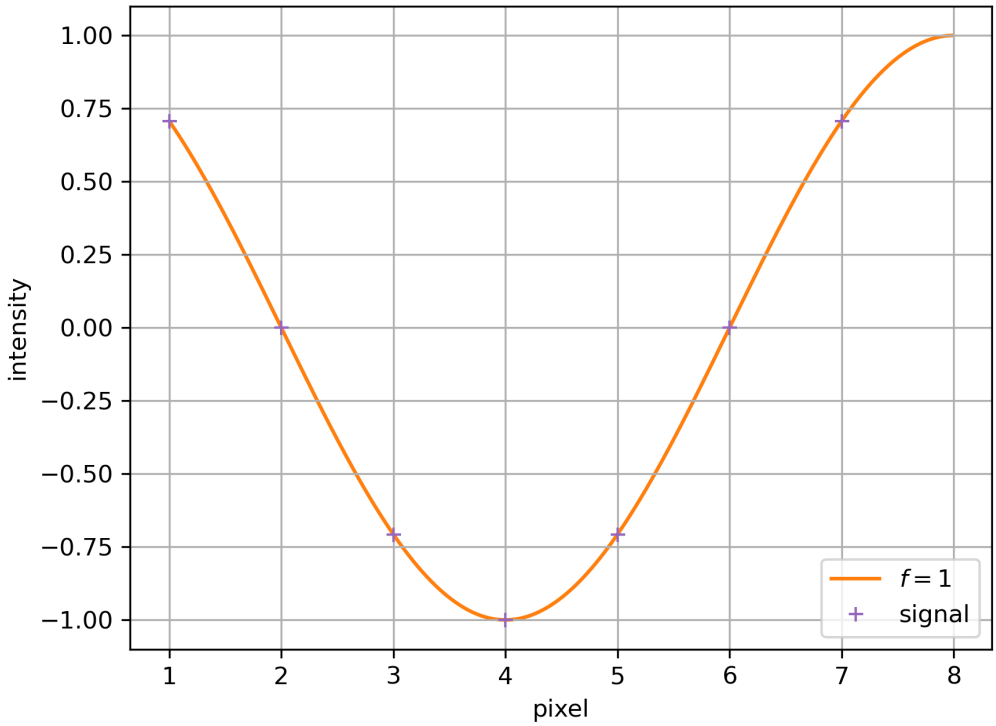

Figure 1: Cosine curve with frequency of 1 unit

If we add an additional cosine curve with a frequency of 2 units, the signal will be changed. In this case, the signal is the average of both curves.

If we change to weighting of the average, we can change the signal. In this case, the cosine curve with a frequency of 1 unit is weighed 3 times as much as the other cosine curve.

DCT[7] propose that we can represent any signal by adding 8 weighted cosine curves of increasing frequency. The resulted signal is modeled by this mathematical equation. The exact mathematics of DCT is too advance for the scope of this paper, implementation of DCT will be done through code.

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)\right] \qquad k = 0, \ldots, N-1 \tag{1}$$

`Continue with an example here (use random number and find the frequency)`

The example showed 1 dimension DCT, 2 dimension DCT is used on photos. Similar to decomposing a signal into many stacking cosine waves, 2D DCT decompose an 8 by 8 image

[7]More specifically DCT-II, used by JIFF compression

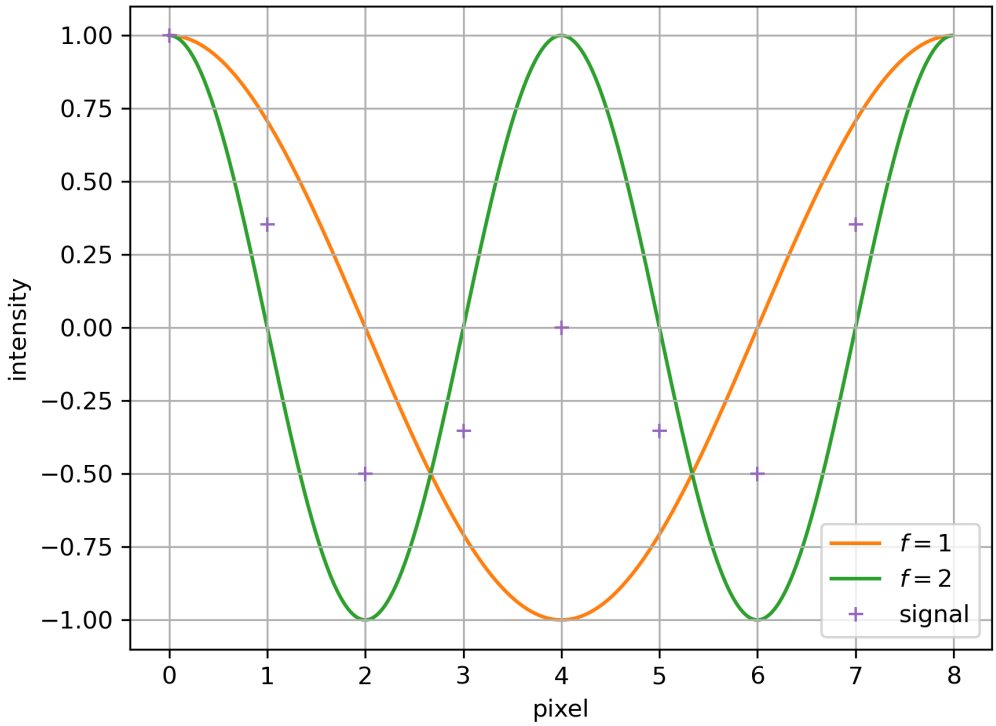

Figure 2: Cosine curves with frequency 1 and 2 unit

into a stack of 8 by 8 grids. These grids are formed by 2D cosine waves, each with increasing frequency. 2D DCT proposes that an 8 by 8 image can be taken apart into 64 different cosine grids. The results generated by this operation is called the DCT coefficients.

3.3 Quantization

The beauty of DCT lies in that most of the image can be recreated by the lower frequency cosine blocks ("JPEG 'files' & Colour (JPEG Pt1)- Computerphile"). This is exploiting the second flaw in the human eye, reduced ability to see high frequency content. Through this logical, we can theoretically create an importance value for each cosine grid (from DCT). Using this importance value, we can throw away the less important data, while keeping the more important data. This process is called quantization.

This is done mathematically by dividing the DCT coefficients by a quantization table. Each element in the DCT coefficients is divided by their responding quantization table, the resulted value rounded and stored as the quantized data matrix. The process of rounding quantized data is removing information depending on the importance of each frequency. Notice how in figure 3.3, the element in the quantization table increases as we move to the bottom right corner.

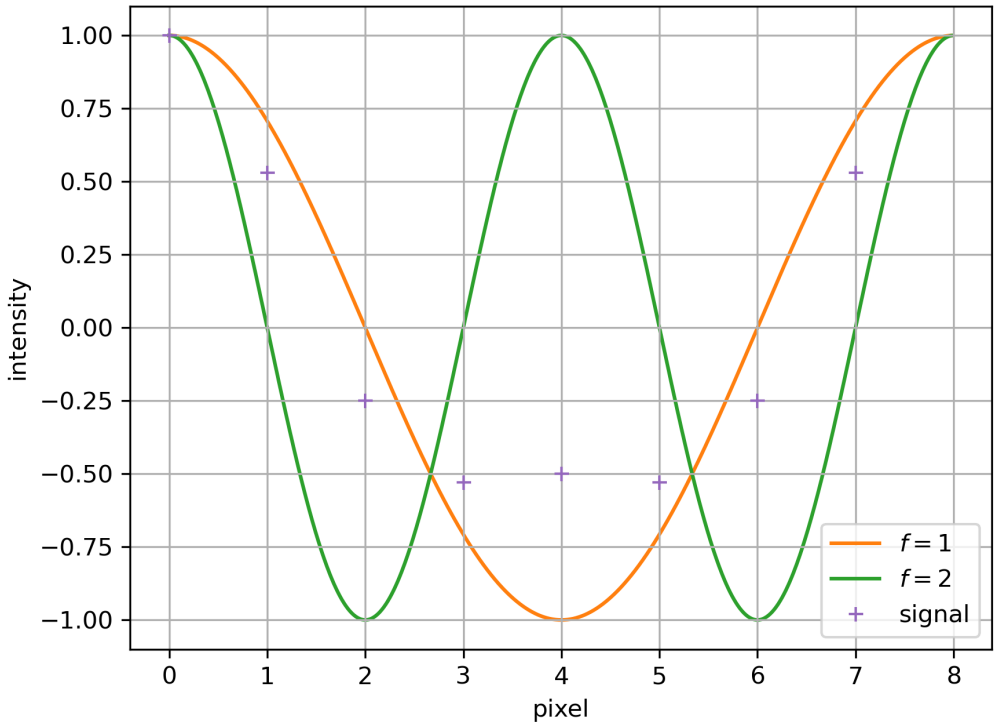

Figure 3: Weighted cosine curves

The higher the quantization factor, the lower the importance of that specific frequency block.

We can imagine the rounding process as a process of finding factors. By dividing the DCT coefficients with the quantization coefficients, we are forcefully factoring the DCT coefficients by the quantization coefficients. The smaller the quantization coefficient, the more accurate we can factor the DCT coefficients. By tuning the quantization coefficients, we can control how much data about a specific frequency to throw out.

3.4 Run Length Encoding

After quantization, the resulted matrix is compressed into a string. Due to the increased amount of quantization the further south east down the matrix, the quantized matrix often have a lot of similar values towards the south east side. In order to take advantage of this repetition, a pointer will traverse the matrix in a zig zag manner, compacting it into a string.

The result of this transformation is a 64 integer long string. Most of the integers with be repeated. JFIF performs run length encoding to losslessly compress the string. It does this by representing each integer as two values, the `integer` and the `amount of time it is repeated`.

$$\begin{bmatrix} 8 & 6 & 6 & 8 & 12 & 14 & 16 & 17 \\ 6 & 6 & 6 & 8 & 10 & 13 & 12 & 15 \\ 6 & 6 & 7 & 8 & 13 & 14 & 18 & 24 \\ 8 & 8 & 8 & 14 & 13 & 19 & 24 & 35 \\ 12 & 10 & 13 & 13 & 20 & 26 & 34 & 39 \\ 14 & 13 & 14 & 19 & 26 & 34 & 39 & 39 \\ 16 & 12 & 18 & 24 & 34 & 39 & 39 & 39 \\ 17 & 15 & 24 & 35 & 39 & 39 & 39 & 39 \end{bmatrix} \tag{2}$$

Figure 4: Luminance quantization table from Photoshop Save To Web 050 (Hass)

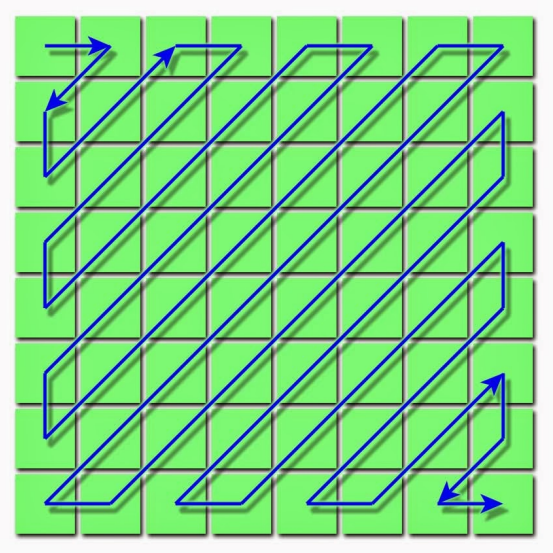

Figure 5: Path of the pointer (Programming Techniques)

For a string with many repetitions, run length encoding can reduce the space required to the data drastically.

3.5 Huffman Coding

To convert the string of data into binary, huffman coding is performed. Huffman coding furthers losslessly compresses the data by effectively assigning symbols to each element in the string. Firstly, an huffman tree is created from the string.

To create an huffman tree, the frequency of every element is counted.

Works Cited

Brailsford, David. "Elegant Compression in Text (The LZ 77 Method)". 1 Sept. 2013. <https://www.youtube.com/watch?v=goOa3DGezUA>.

Hass, Calvin. "JPEG Compression Quality from Quantization Tables". 2018. <https://www.impulseadventure.com/photo/jpeg-quantization.html>.

Johnston, Nick and David Minnen. "Image Compression with Neural Networks". *Google AI Blog* (29 Sept. 2016). <https://ai.googleblog.com/2016/09/image-compression-with-neural-networks.html>.

Joint Photographic Experts Group. "Overview of JPEG 1". 27 July 2021. <https://jpeg.org/jpeg/>.

LEARN MEDIA TECH. "Chroma Subsampling – 4:4:4 vs 4:2:2 vs 4:2:0". *Learn Media Tech* (2020). <https://learnmediatech.com/chroma-subsampling-444-vs-422-vs-420/>.

Pound, Mike. "JPEG 'files' & Colour (JPEG Pt1)- Computerphile". 22 Apr. 2015. <https://www.youtube.com/watch?v=n_uNPbdenRs>.

——."JPEG DCT, Discrete Cosine Transform (JPEG Pt2)- Computerphile". 22 May 2015. <https://www.youtube.com/watch?v=Q2aEzeMDHMA>.

Programming Techniques. "Discrete Cosine Transform and JPEG compression : Image Processing". *Following Tutorials* (7 Feb. 2014). <https://followtutorials.com/2014/02/discrete-cosine-transform-and-jpeg-compression-image-processing.html>.

Townsend, Alex. "JPEG: Image compression algorithm". Mar. 2017. <http://pi.math.cornell.edu/~web6140/TopTenAlgorithms/JPEG.html>.