Optimization of Image Compression as performed in the JPEG standard and its Effectiveness

Rohit Ramabhadran

220121072

Engineering Physics

Abstract:
For the past few years, a joint commitee known as JPEG (Joint Photographic Experts Group) has been working to establish the first international compression standard for continuous-tone still images both grayscale and colour. To meet the differing needs of different applications, the JPEG standard uses a DCT-based method for "lossy" compression and a predictive method for "lossless" compression. This paper provides an overview of the JPEG standard, and focuses in detail on the the Discrete Cosine Transform Method and the use of fourier transform in image compression and the possible improvements that can be made.

Introduction:
So lets talk about how computers represents images. The standard color space that computers use is the RGB model. Every pixel of the image stores 3 values from 0 to 255 with higher values representing a larger weighting of the respective color. So if a image has, say 5 million pixels, based on an assumption that each pixel requires 3 bytes, then the total size of our image should be 15 megabytes. But with JPEG compression the actual file size is only about 0.8 megabytes: same number of pixels but 5% of the expected size. To get compression on the ~~ve~~ levels of 5% there's really no other option other than to actually discard some information from our original image i.e lossy compression. Now the fun question to ask is what sort of information from an image can we get rid of & How do we get rid of it.

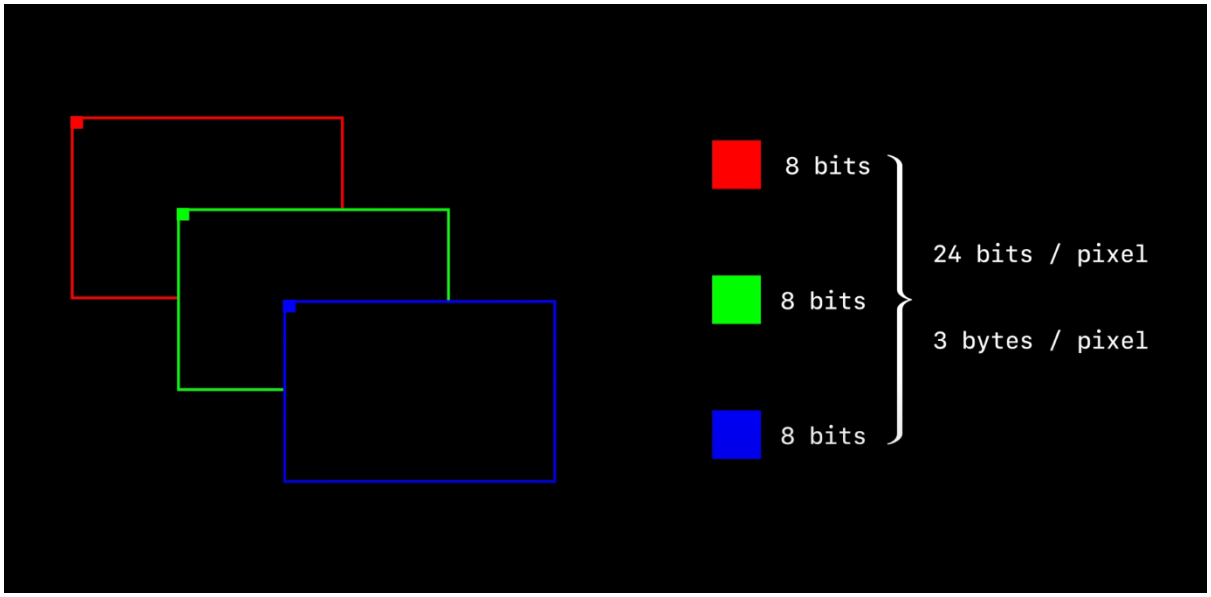


Figure 1 – Composition of each pixel in RGB model

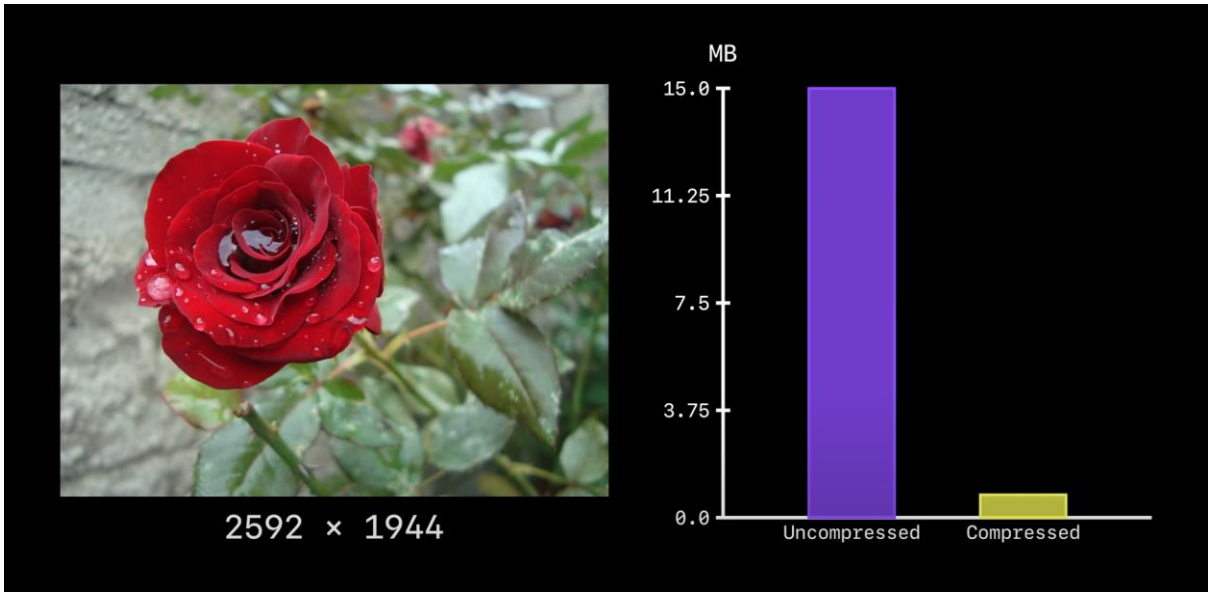
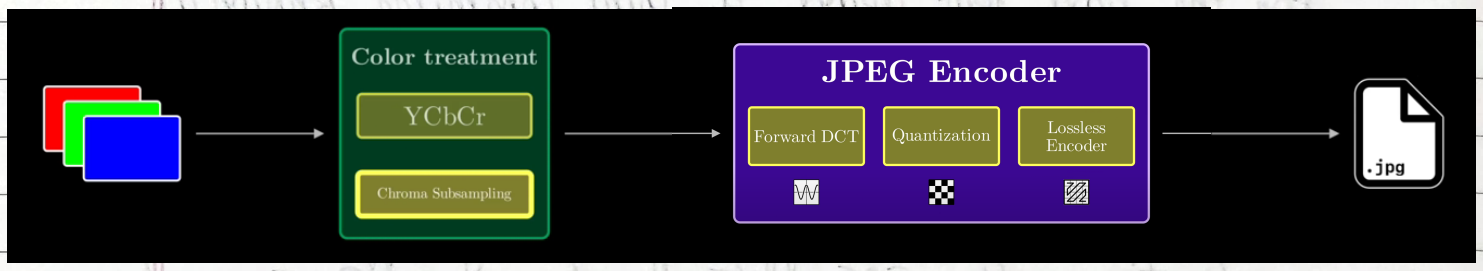


Figure 2. Comparison Size of a Reference image in Megabytes

Architecture of the Proposed Standard:



Part 1: Conversion of RGB to YCbCr model :-

In the ~~ret~~ realm of image compression, the human visual system model has been shaped by a fascinating finding - our eyes are more receptive to changes in brightness than to variation in colour. This knowledge has profound implications, especially concerning colour spaces.

In the RGB colour space; which combines red, green and blue components; each value is represented on a separate axis within a three dimensional space. All possible colours are visualized as points within this RGB cube. What's fascinating is that as you move diagonally from the origin to the colour (255, 255, 255), you encounter progressively brighter colours. The line connecting all these points defines all possible grayscale colours, which are a direct measure of brightness. The idea of separating brightness from colour is fundamental to another colour space called YCbCr. YCbCr stands for Luma (Y), Chroma blue (Cb) and Chroma Red (Cr). The Y component quantifies the image brightness, while the Cb and Cr components encode the colours. In the context of JPEG image compression, the YCbCr colour space allows for a technique known

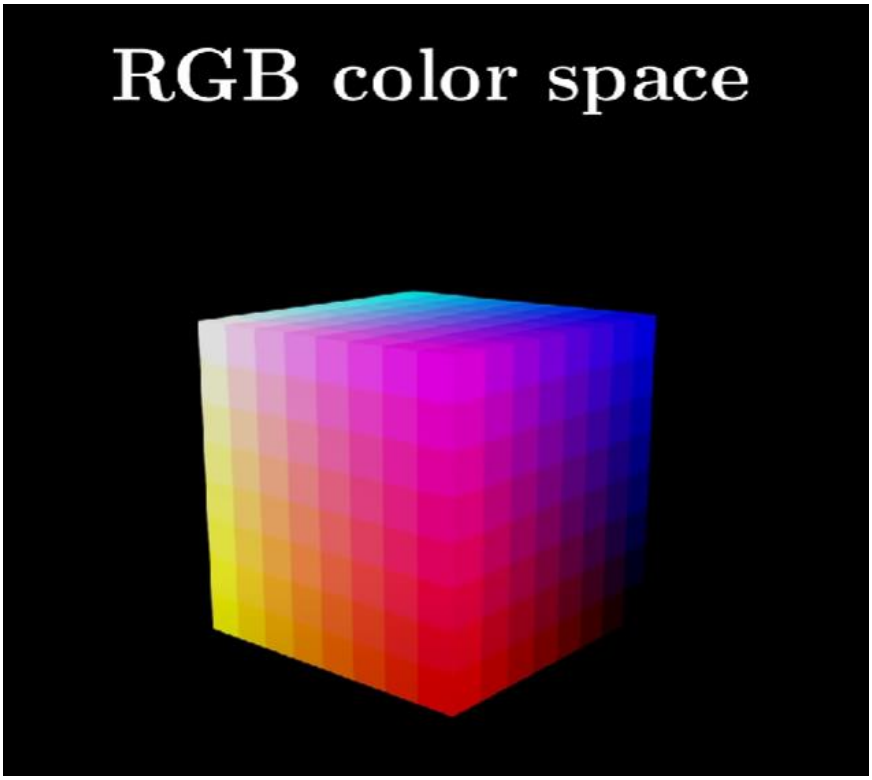


Figure 3. Illustration of the RGB colour space

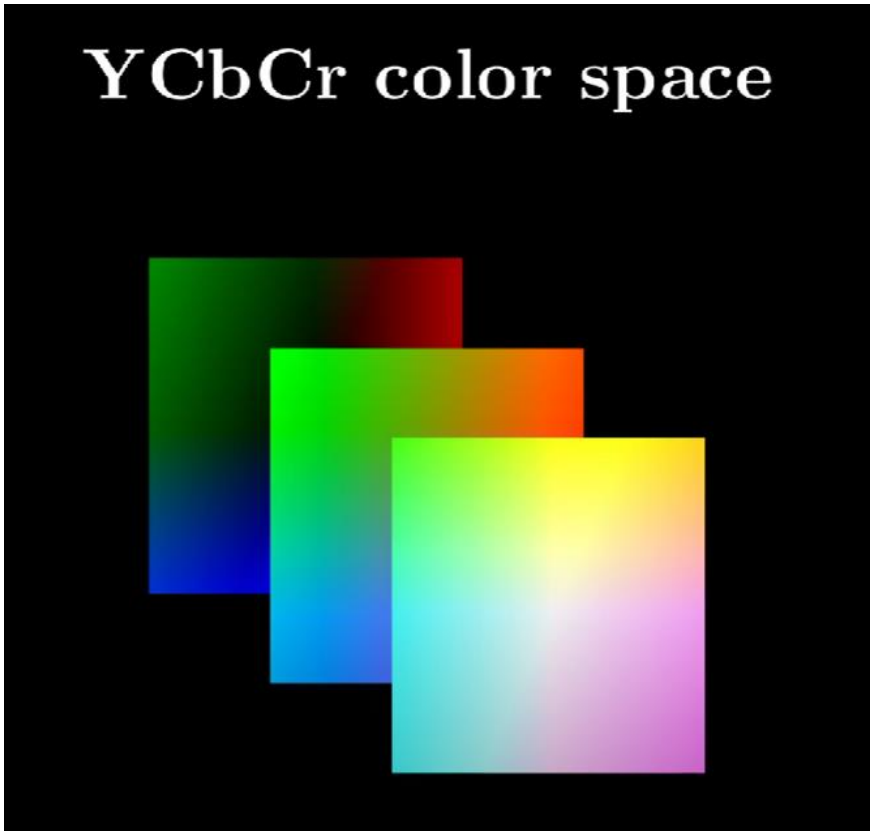
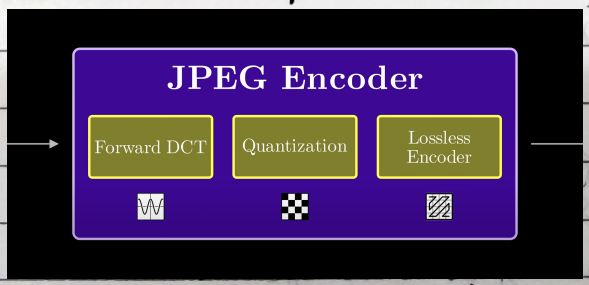


Figure 4. Illustration of the YCbCr colour space

as chroma subsampling. This involve sampling less from the Cb and Cr components since our eyes are less sensitive to colour changes. One common approach is 4:2:0 chroma subsampling scheme. In this scheme, 2x2 blocks of pixels in the original image are averaged, effectively reducing the resolution of the colour information. In more practical terms, it means that instead of retaining all 4 colour sample is chosen (typically the top-left pixel) to represent the entire 2x2 block. This results in a substantial reduction in data size while preserving image quality, especially in cases where the changes are imperceptible to the human eye. However, this method alone doesn't achieve the level of compression seen in advanced JPEG techniques, which exploit additional factors beyond human perception of brightness.

Part 2: JPEG Encoder



Lets focus on the y channel which essentially defines grayscale images. One way to think about images is treating them as signals. If I slice a particular row of an image, we essentially have a row of pixels each with some value between 0 and 255. If we plot these values we can get an approximation of a signal. ~~Visualization~~ Visualizing an image as a

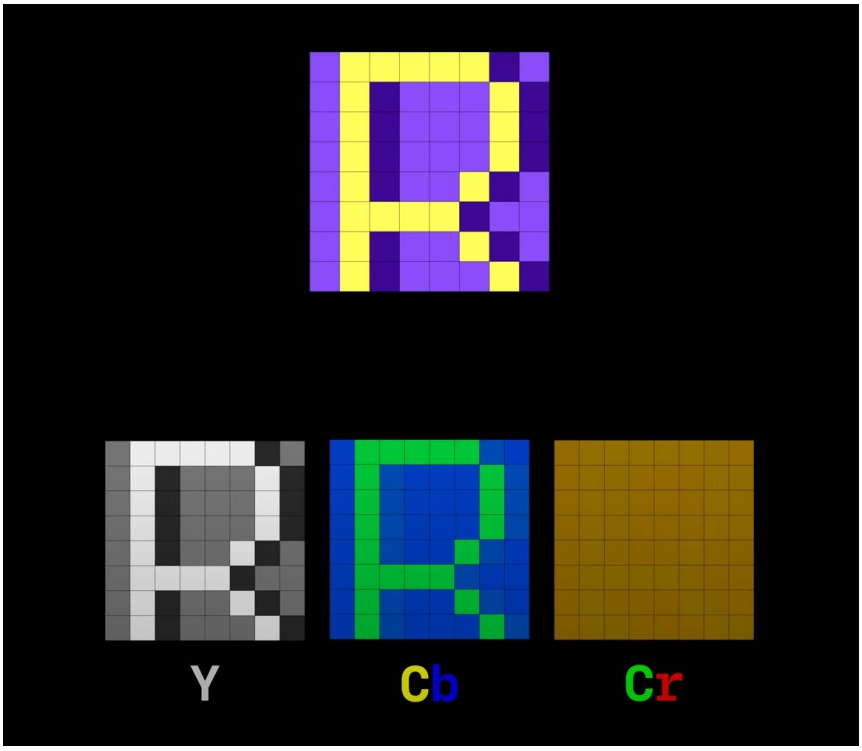


Figure 5. Representing a sample image in its corresponding Y, Cb and Cr components

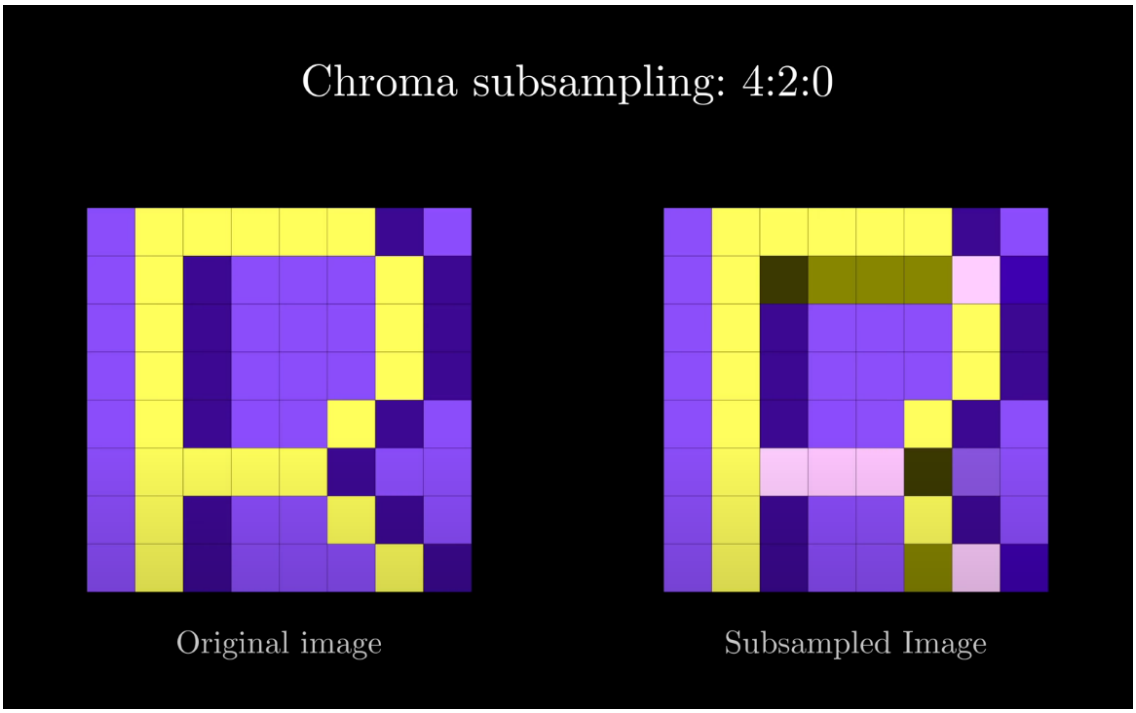


Figure 6. Showcasing how the 4:2:0 Chroma subsampling works in the pixel level

Figure 7. Showcasing how Chroma Subsampling can be used without much loss of details in a image as a whole

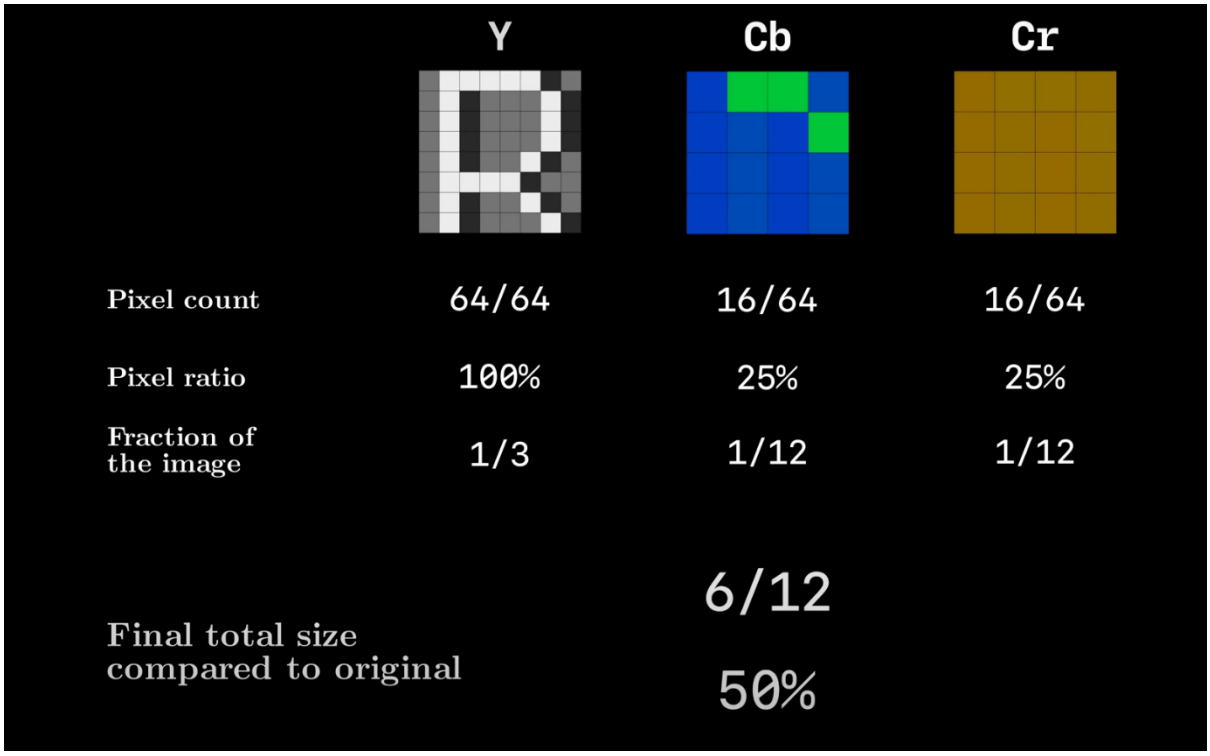


Figure 8. Showing how Chroma Subsampling on YCbCr images can reduce file size by 50%

signal allows us to talk about frequency components within an image. The higher frequency components correspond to rapid changes between pixels while lower frequency components are related to smoother changes between pixels. There are two key aspects of frequencies within image that are incredibly important to jpeg compression. The first is that a lot of real world images shot from cameras are mostly composed of lower frequency components. In other words if I take a random portion of a realistic image, its pretty likely that the pixels in a small area do not change that rapidly and the second key fact is that the human visual system is generally less sensitive to higher frequency detail in images. JPEG takes advantage of these ideas by strategically removing less important and less common higher frequency components from an image to achieve even more compression.

Part 3: Discrete Cosine Transform

The discrete Cosine Transform is very similar to the more familiar Discrete Fourier transform. In General we can think of a transform as representing a signal $x[n]$ in terms of a weighted sum of basis signals in linear algebra terms.

$$x[n] = \sum_{K=0}^{N-1} b_K C_K[n] \quad ; n = 0, 1, 2, \ldots, N$$

b_K : Coefficients $\quad$ $C_K[n]$ = basis signals.

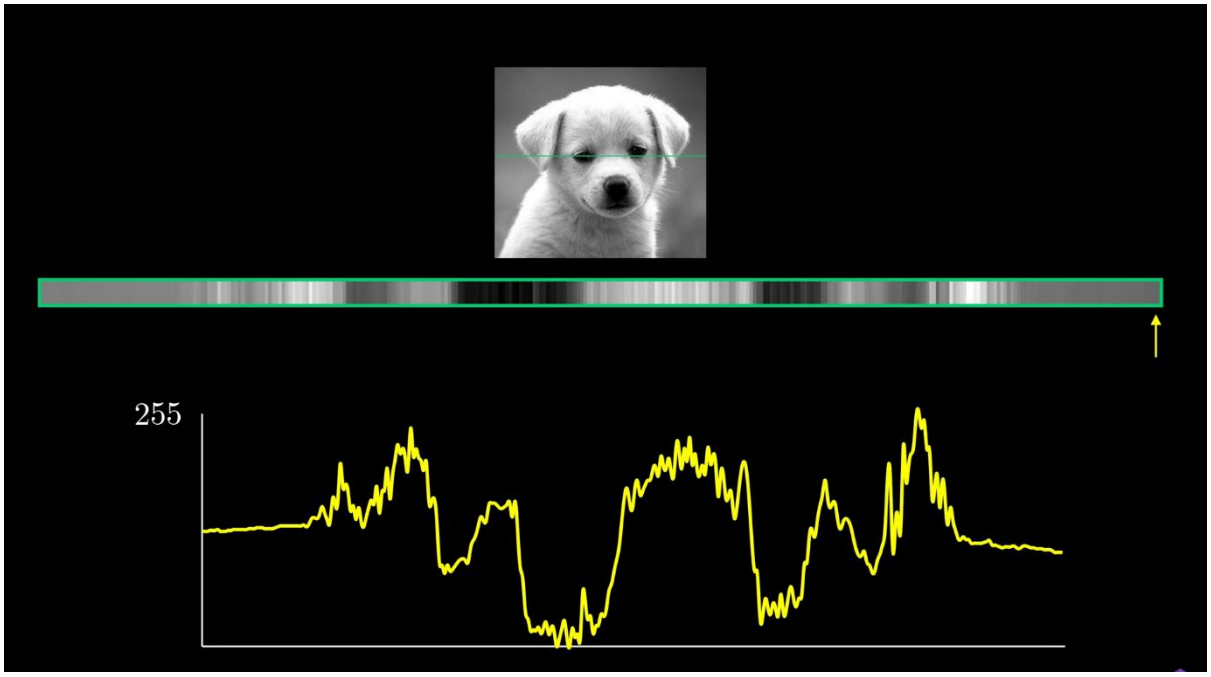


Figure 9. Taking a horizontal column of pixels in a grayscale image and plotting a graph between the brightness (Y axis – ranges from 0(black)-255(white)) and location (X axis)

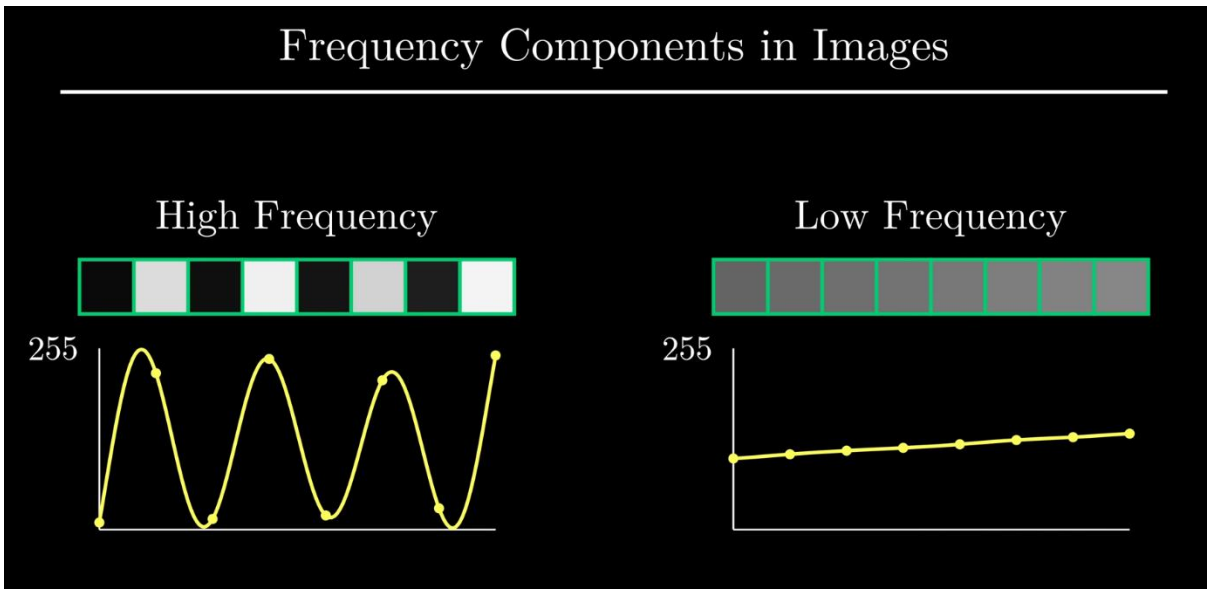


Figure 10. Illustrating how high frequency and low frequency images can be converted to graph in frequency domain

In Discrete Fourier Transform the $C_k[n]$ are complex sinusoids i.e.

$$C_k[n] = e^{j2\pi(\frac{k}{N})n}$$

where K = length of signal.

So why are complex sinusoids useful? Well the biggest reason is because how they interact with linear time-invariant systems

$$e^{j2\pi(\frac{k}{N})n} \longrightarrow H(\hat{f}) \longrightarrow H(\tfrac{k}{N})e^{j2\pi(\frac{k}{N})n}$$

So the action of the system is to multiply the input complex sinusoid by the frequency response of the system at that frequency.

$$\Rightarrow x[n] = \sum_{k=0}^{N-1} C_k e^{j2\pi(\frac{k}{N})n} \longrightarrow LTI$$

$$y[n] = \sum_{k=0}^{N-1} H(\tfrac{k}{N}) C_k e^{j2\pi(\frac{k}{N})n} \longleftarrow$$

So if we to expand an arbitrary signal $x[n]$ as a weighted sum of complex sinusoids and we put that into a linear time Invariant system, then the output takes a particularly simple form. It takes the same form as the input except the coefficients have been modified by the corresponding values of the frequency response. So the action of the system is one of multiplication of these

coefficients that we use to represent the signal. This leads to the important notion of Filtering because we can think of filtering as a multiplication effect in the frequency domain. However, if we are interested in other tasks such as performing compression of a signal then it is not necessarily the best choice.

Discrete Cosine Transform

The basic idea ~~between~~ behind compression is that we take our signal $x[N]$ and write it as a weighted sum of basis signals $c_k[N]$ which are known. Then we compress by setting the b_k that are insignificant to zero.

$$x[n] = \sum_{k=0}^{N-1} b_k c_k[n], \quad n = 0, 1, \ldots, N-1$$

if most $b_k \approx 0$, then we only need to store a few coefficients and we will have all the information that was in $x[n]$

$x[n]$: N values
but only L values of b_k are significant
then;

$$\text{Savefactor} = \frac{N}{L}$$

This is the basic idea used in JPEG, MPEG, MP3

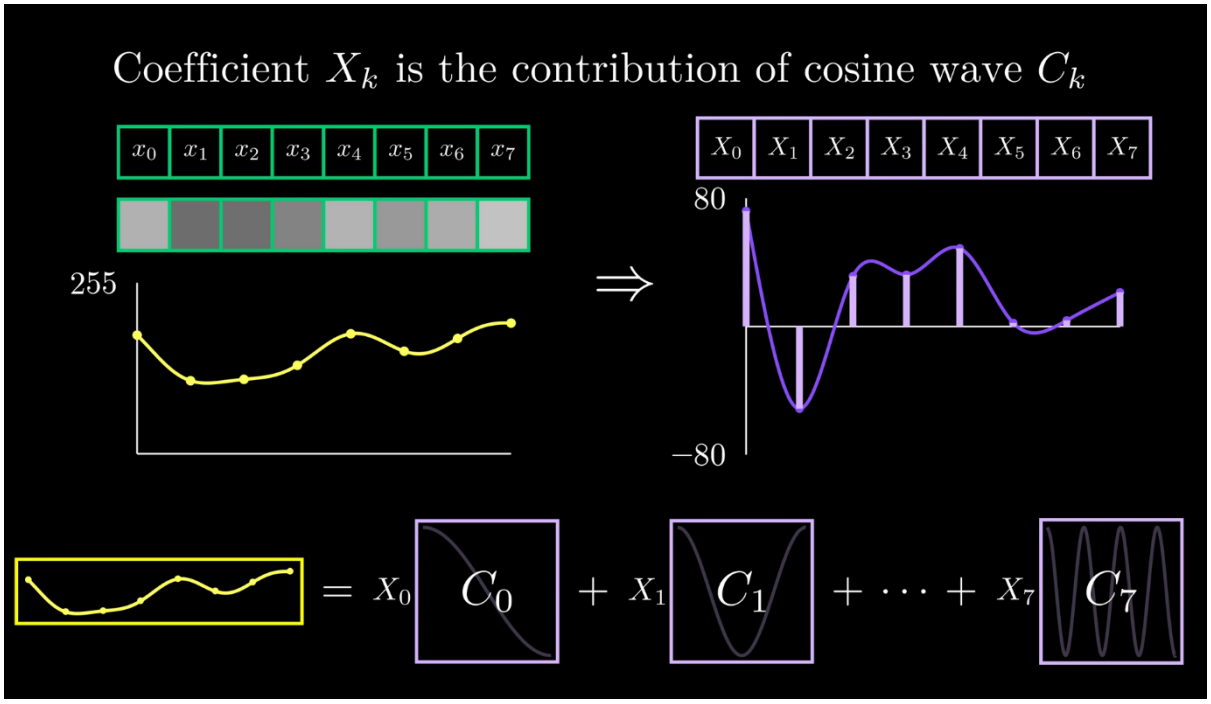


Figure 11. Illustration of an image signal is converted to frequency domain of a weighted sum of basis signals using Discrete Cosine Transform in 1 Dimension

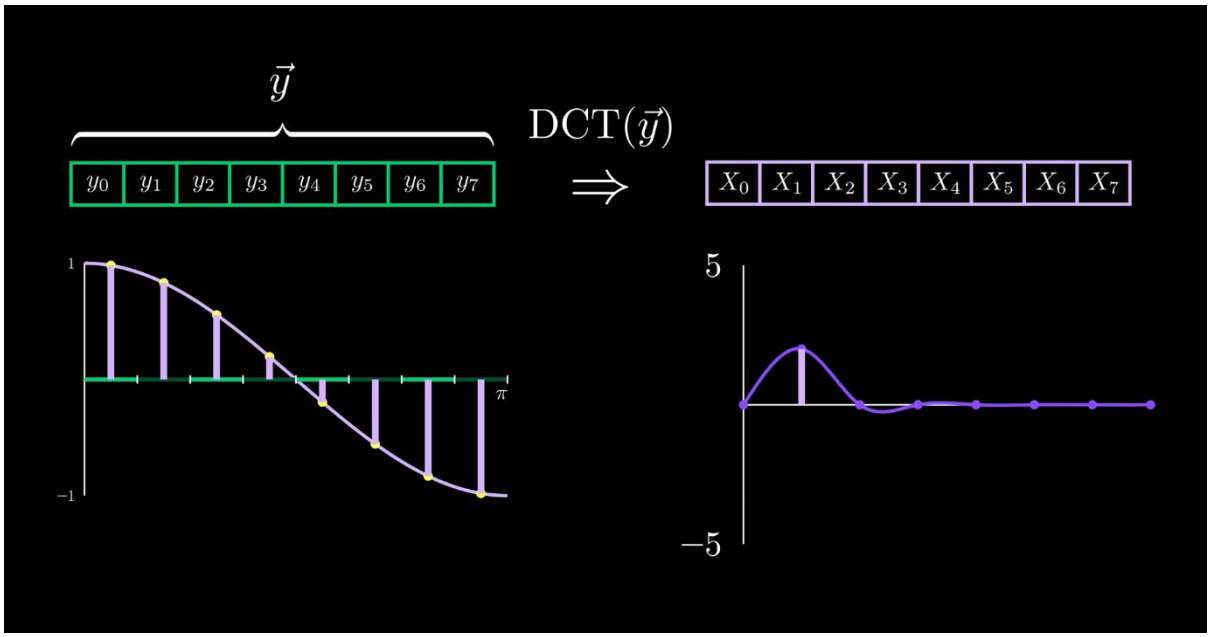


Figure 12. Illustration of cos x in frequency domain by performing Discrete Cosine Transform on it

Discrete Cosine Transform :-

$$x[n] = \frac{1}{N} X^c[0] + \frac{1}{N} \sum_{k=1}^{N-1} 2X^c[k] \cos\left(2\pi\left(\frac{k+1/2}{2N}\right)n\right)$$

$n = 0, 1, \cdots N-1$

where $b_0 = \frac{1}{N} X^c[0]$

$C_0 = 1$

$b_k = \frac{1}{N} 2X^c[k]$

$C_k[n] = \cos\left(2\pi \frac{(k+1/2)}{N} n\right)$

~~where~~ Hence we can say

$$X^c[k] = \sum_{n=0}^{N-1} x[n] \cdot \cos\left(2\pi\left(\frac{k}{2N}\right)(n+1/2)\right)$$

~~Since~~ This is different from fourier transform since everything in these expression is real ~~not~~ ~~valued~~ valued. Hence we do not have to work with complex sinusoids which occurs in DFT as cosines do not share the same property in Linear Time Invariant systems.

⇒ Due to this the out is not some factor multiplied times the cosine but rather the output has the magnitude of the frequency response in front of the cosine along with a phase factor thats inside the argument of the cosine.

$$\cos(2\pi \hat{f} n) \xrightarrow{DCT} \boxed{LTI} \rightarrow |H(\hat{f})| \cos(2\pi \hat{f} n + \phi(\hat{f}))$$
$$\neq A(\hat{f}) \cos(2\pi \hat{f} n)$$

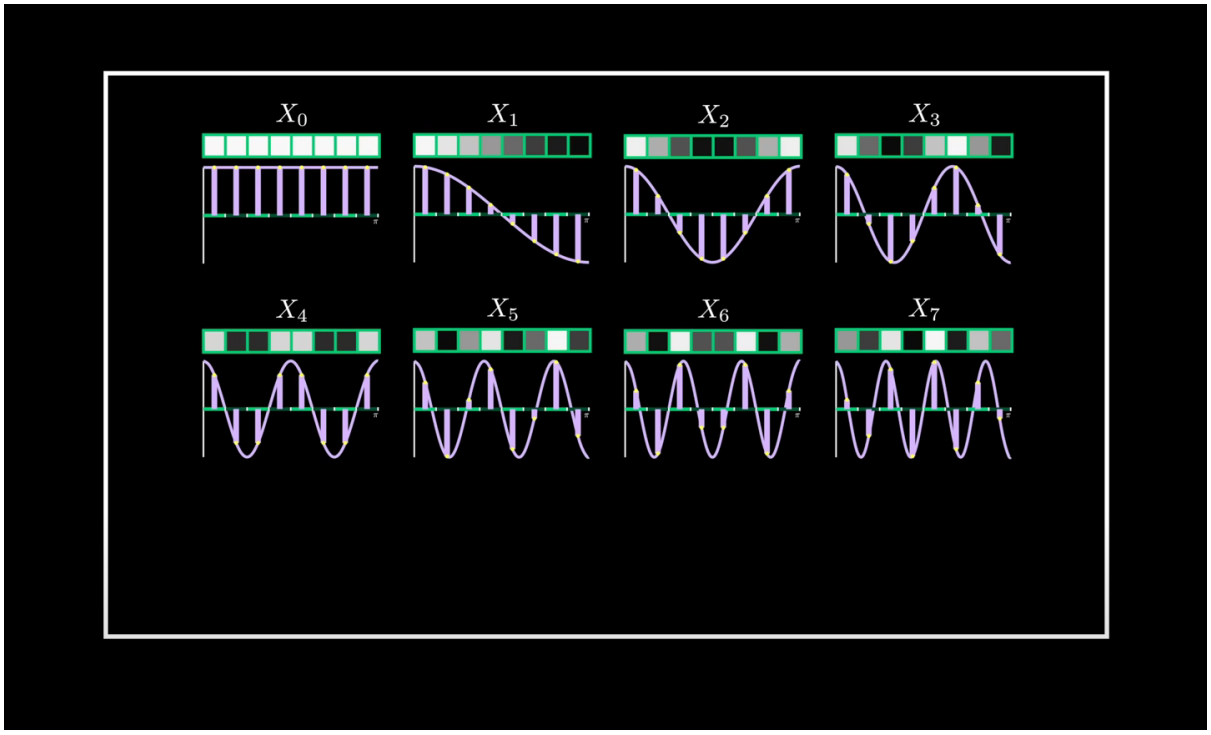


Figure 13. Illustration of cos kx for k ranging from 0 to 7 for different image input signals in frequency domain by performing Discrete Cosine Transform on it

This makes DCT very useful when working with compression because of the fact that these coefficients are real and for many signals of interest using cosines is an efficient way to represent the signal. But in order to use DCT for an image we require a 2-D Discrete Cosine Transform.

2-D Discrete Cosine Transform.

So the 2-D DCT expresses an image $x[m,n]$ as a multiplication of 2 individual signals that represent the length ($\rightarrow$) and height ($\uparrow$) of an image.

$$x[m,n] = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \underbrace{a_k a_l}_{a_i} X^c[k,l] \underbrace{\cos\left(2\pi \left(\frac{l}{2N}\right)(n+1/2)\right) \cos\left(2\pi \left(\frac{k}{2N}\right)(m+1/2)\right)}_{C_{k,l}[m,n]}$$

$$a_i = \begin{cases} 1 & i = 0 \\ \sqrt{2} & \text{otherwise.} \end{cases}$$

$$x[m,n] = \sum_{k=0}^{N-1} \sum_{l=0}^{N} b_{k,l} C_{k,l}[m,n]$$

weighted sum of "basis" images

where;

$C_{k,l}[m,n]$ = product of a vertical cosine of frequency $\left(\frac{k}{2N}\right)$ and a horizontal cosine of frequency $\left(\frac{l}{2N}\right)$

The discrete cosine transform

The DCT of an image f of size $N \times N$ is an image F of same size defined as:

$$F(u,v) = \frac{2}{N} C(u) C(v) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m,n) \cos\left[\frac{(2m+1)u\pi}{2N}\right] \cos\left[\frac{(2n+1)v\pi}{2N}\right]$$

where

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0, \\ 1 & \text{if } u > 0. \end{cases}$$

The DCT is similar to Fourier transform except that the complex exponentials are replaced by cosines.

The DCT decomposes an image into 2D cosines of different frequencies. Considering an image of size 8 × 8, the 64 possible cosines of the DCT are represented Fig. 37.

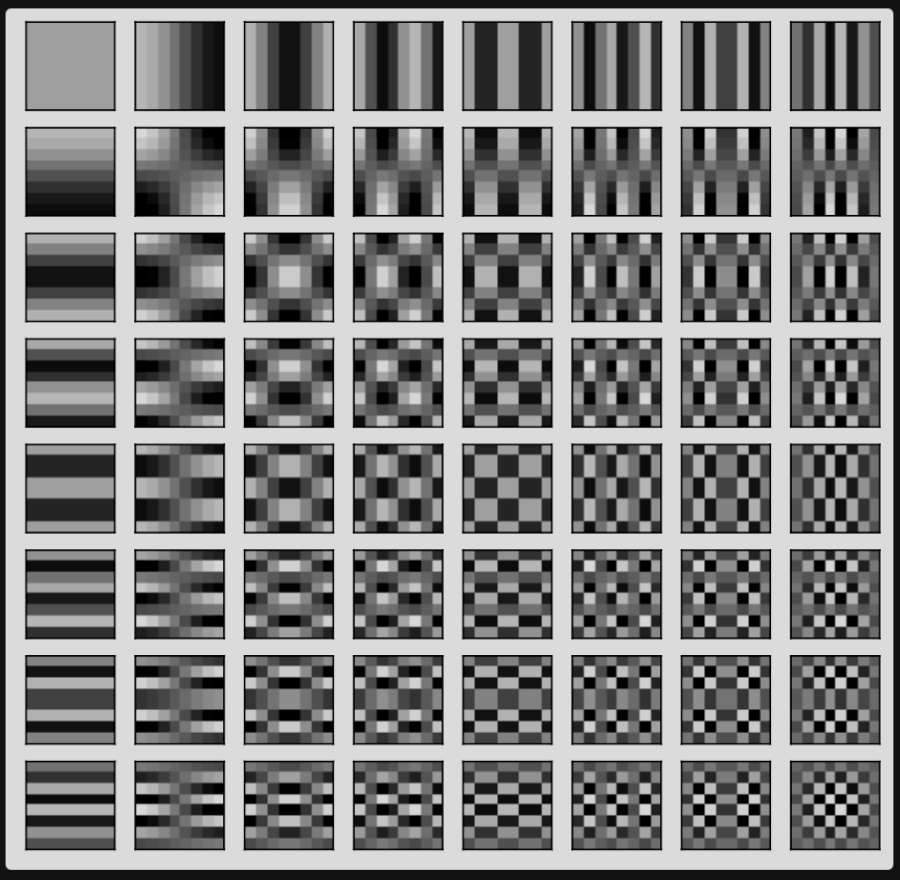


Fig. 37 The 64 coefficients of a DCT of size 8 × 8. #

Hence for $N=8$ we get a $C_{kl}[m,n]$ we get an 8×8 representation having 64 possible images. (See Image on next Page for further clarification)

So in the first row we are seeing the horizontal frequency cosines and in the ~~1st vertical~~ row we see the vertical frequency cosines. As we go to the bottom right hand corner we can see how the image would look having highest frequency of vertical and horizontal cosines.

So how does JPEG use this. JPEG takes an image and splits it into 8×8 blocks and then centres their values around 0 by subtracting 128. Then we take the block and apply the DCT to each row of the block giving us eight sets of dCT coefficients. We then apply the dCT to each column of the block. This process is what defines the two dimensional DCT. In the end we have 64 coefficients each of which are a weight on a specific 8×8 pattern. Hence we can build any 8×8 image using these 64 fundamental patterns. The weight of each coefficient shows how bright each pattern is.

Seeing this in action [☆ Please see the Video here] makes us realize that by the time we incorporate even a few coefficients our signal and image look pretty close to the original image.

A fun observation we can make is that if we map out the magnitude of the DCT coefficients, we see that most of the largest values are in the upper left section which correspond to lower frequency components. This is applicable for almost all of the set of pixels. This property of the DCT is commonly reffered to as Energy Compaction.

This is exactly the property that will allow us to aggressively compress images while still retaining high visual quality..

Quantization

But the next natural Question is how we actually eliminate the higher frequency components in JPEG. This is done by ~~Quatizadi~~ Quantization.
Quantization is a simple idea where given an 8x8 matrix of frequency components, we essentially divide each element by a scalar value and round it to an integer. These scalar values are defined by a quatization table which is given by the JPEG standard. Notice that larger values in the bottom right of the table lead to zero values in the higher frequency components. In the decoding stage of JPEG we will actually be multiplying this result by the same quantization matrix but the final matrix will be quite different from the original one.

Explanation of 2-D DCT.mp4

Watch Video 1 - Explanation of 2-D DCT

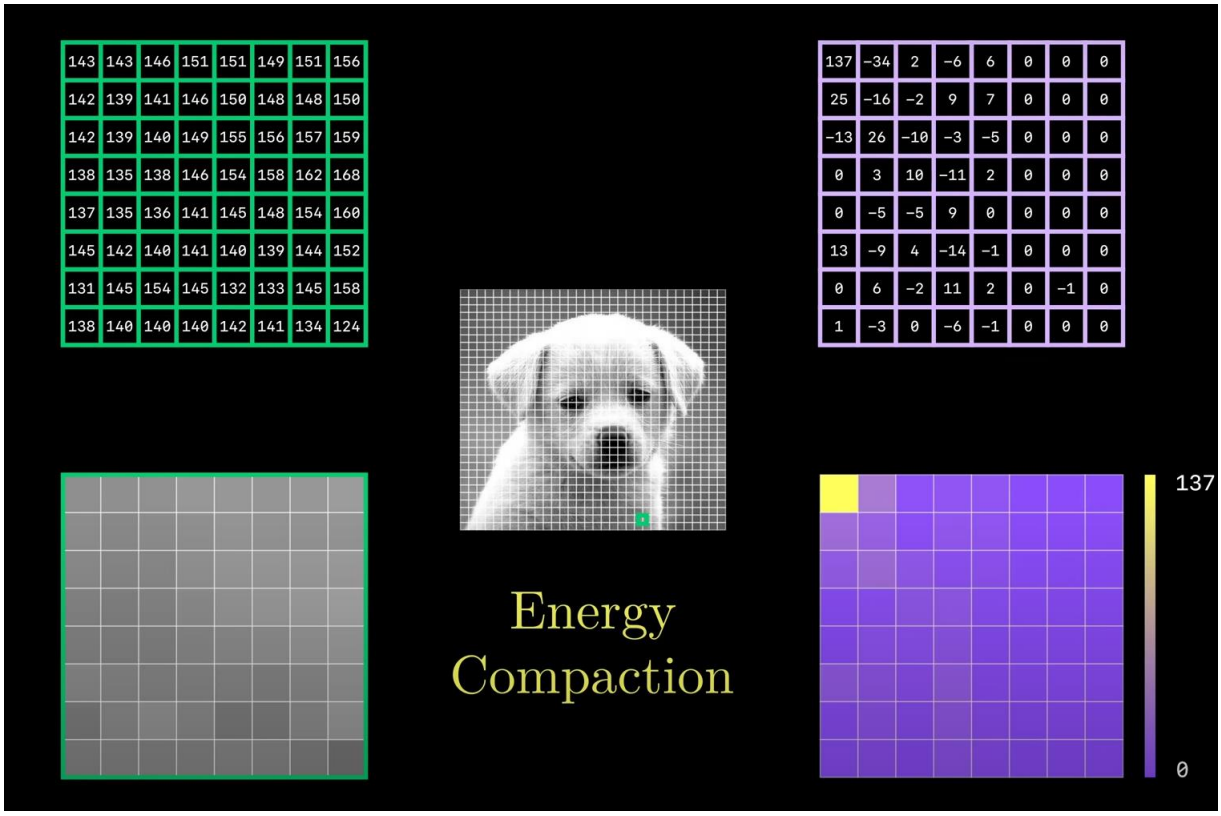


Figure 14. Illustration of Energy Compaction property of Discrete Cosine Transform

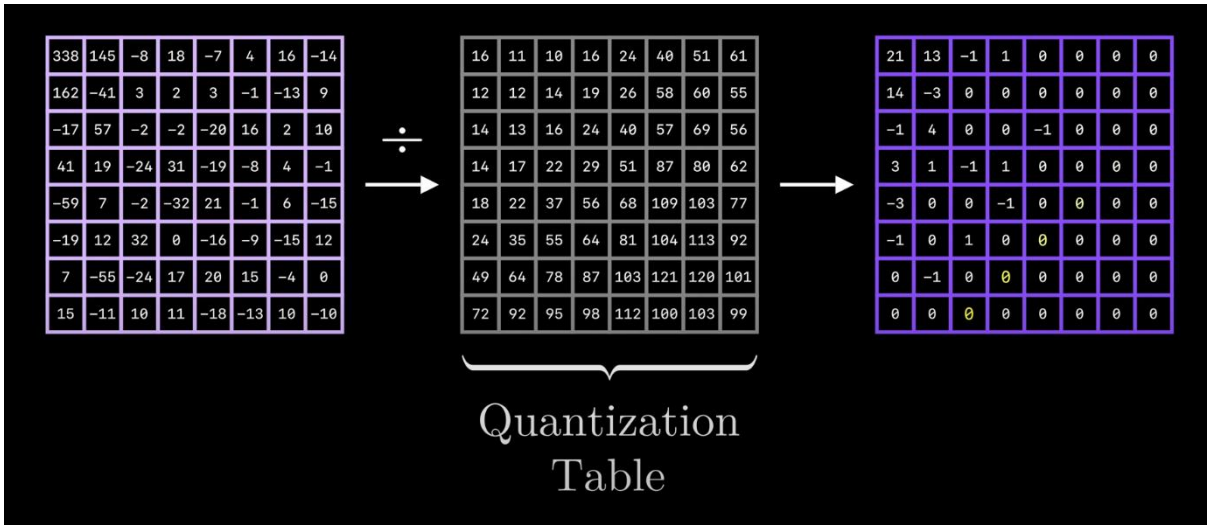


Figure 15. Illustration of the process of Quantization on the image matrix after it has been converted in frequency domain using 2D Discrete Cosine Transform and Energy Compaction has been done to get zeros in the high frequency region

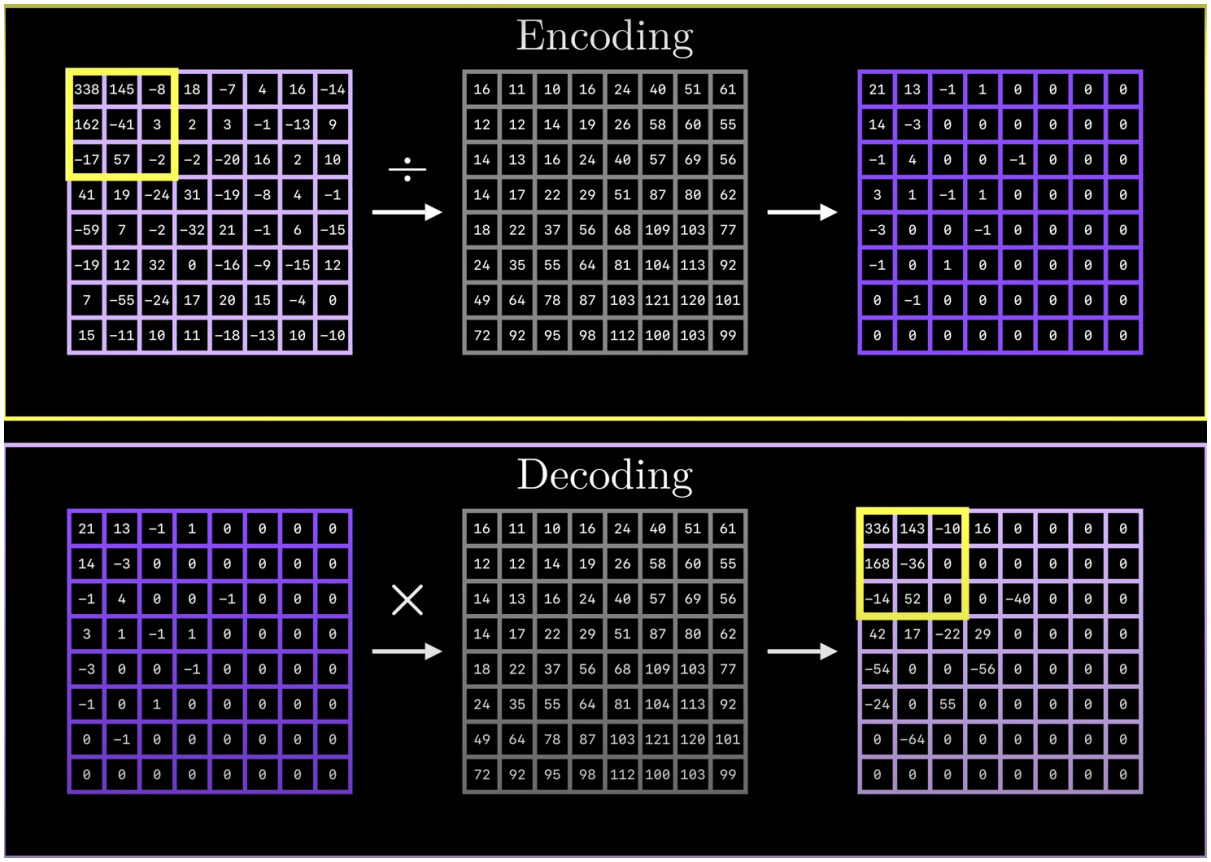


Figure 16. Illustration of the encoding and decoding of an image takes place and how the final image matrix after decoding is not the same as the initial image matrix hence showcasing the property of lossy compression.

Watch Video 1 – Output of using 2D DCT

Output of using 2D DCT.mp4

So what this means, is ~~were~~ we are purposely losing information, but most of the lower components will be retained. This is why energy compaction property of the DCT is so useful. When the largest values lie in the lowest frequencies, we will end up with a lot of zeros in the less important high frequency components.

These Quatization tables are provided by the JPEG standard from visual experiments and are the main way for jpeg to define quality of compression. In practice JPEG also defines a separate quantization table for both the luma and colour channels.

Lossless Encoders: Runlength & Huffman Encoding

The last part of jpeg encoding involves a combination of run length Encoding and Huffman Encoding. These encoders uses ~~&~~ a ~~deep~~ clever trick to store the matrix that we got from the previous step by exploiting the fact that a lot of values in the matrix are zeros

In Run length encoding, coefficients are strategically ordered in a zigzag pattern, maximizing sequences of zeros. This arrangement facilitates classic run length encoding, a technique that efficiently represents sequences of repeated values by encoding them as a count of occurrences.

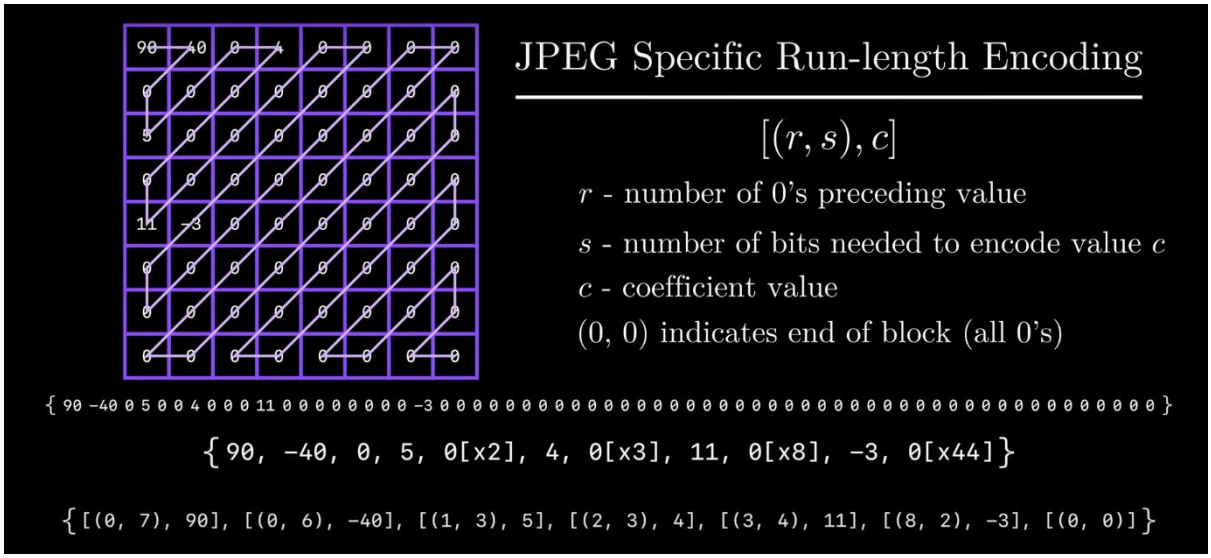


Figure 17. Illustration of JPEG Specific Run Length Encoding

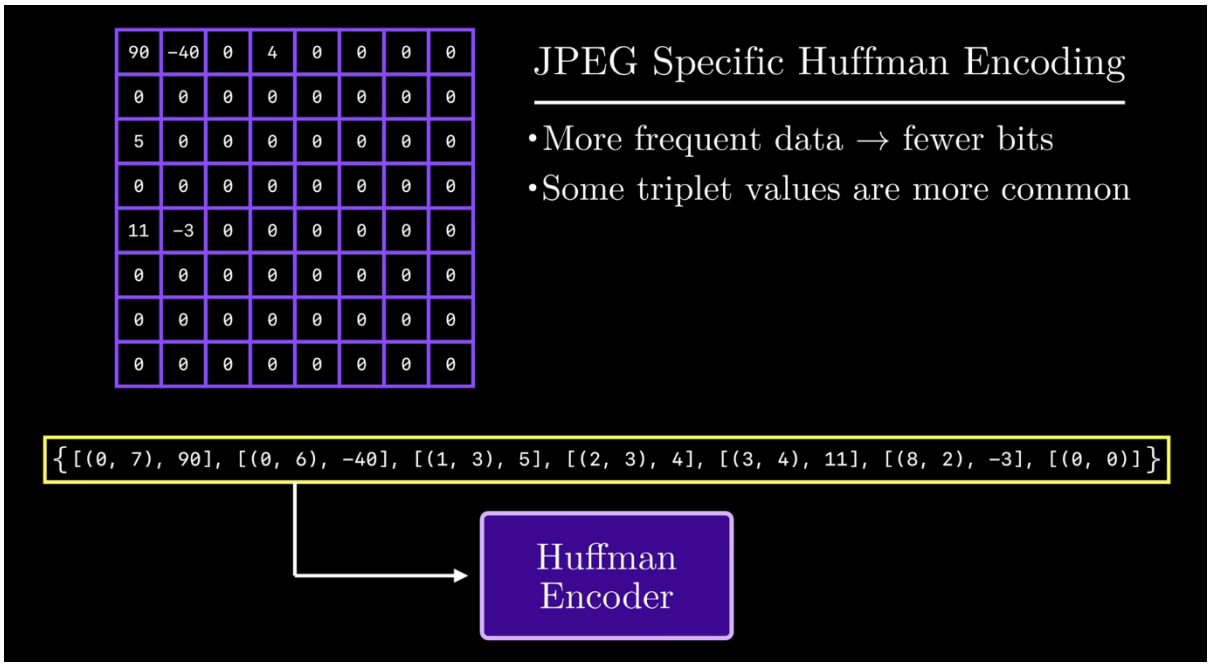


Figure 18. Illustration of Huffman Coding

Huffman encoding, a more sophisticated compression approach, assigns shorter codes to more frequently occuring values. This is done by employing triplets that represent: & the no. of preceding 0's, the bits required, the coefficient & value

The intricate dance between run-length encoding and Huffman encoding is a key aspect of achieving significant data compression.

By doing all this we are finally able to compress an image by more than 90% of its initial size while maintaining its quality and by getting rid of redundant data that is not used. JPEG compression is extremely efficient at storing images and makes use of highly effective techniques to preserve the maximum Quality.

The innovative foundations of JPEG compression emerge from experiments and a profound understanding of human visual systems. Experiments reveal that human eyes are less sensitive to colour & higher frequencies, leading to a strategic removal of such information without perceptually significant consequences.

Results of JPEG Compression

Conclusion

The principles underlying JPEG compression find resonance in audio and video compression, where similar techniques leverage perceptions of sound and motion to remove less relevant data. The adaptibility and effectiveness of the Discrete Cosine Transform and quantization extend across different domains, showcasing a unified approach to achieving efficient compression in diverse digital realms.

In conclusion, the JPEG algorithm, though intricate and challenging to implement, stands as a testament to the power of leveraging redundancy and understanding human perceptual limitations. The delicate balance between compression and visual fidelity in JPEG underscores its significance in the digital landscape, offering an insightful journey into the realm of efficient data compression.

Bibliography:

1. https://web.stanford.edu/class/ee398a/handouts/papers/Wallace%20-%20JPEG%20-%201992.pdf
2. https://vincmazet.github.io/bip/compression/lossy.html
3. https://web.archive.org/web/20050830153441/http://www.site.uottawa.ca:80/~remi/Cornell_JPEG2K.PDF
4. https://nautil.us/the-math-trick-behind-mp3s-jpegs-and-homer-simpsons-face-234629/
5. https://arxiv.org/ftp/arxiv/papers/1405/1405.6147.pdf
6. https://www.researchgate.net/publication/268523100_THE_JPEG_IMAGE_COMPRESSION_ALGORITHM/link/549f251f0cf267bdb8fdbb89/download
7. https://www.youtube.com/watch?v=n_uNPbdenRs
8. https://www.youtube.com/watch?v=Ba89cI9eIg8
9. https://www.youtube.com/watch?v=Kv1Hiv3ox8I&t=963s
10. https://www.youtube.com/watch?v=DS8N8cFVd-E&t=181s
11. https://www.youtube.com/watch?v=0me3guauqOU
12. https://www.youtube.com/watch?v=Aq7YgjbVrPg
13. https://www.youtube.com/watch?v=Iz6C1ny-F2Q