SVD and DCT Image Compression

Connor Kuhn

April 28, 2016

1 Introduction

Digital images proliferate through every aspect of our live today. This means that being able to compress them efficiently is very important. Two linear algebra techniques, low-rank approximation from a single-value decomposition and discrete cosine transformations, for compressing an image will be explored today.

A gray-scale image can stored as a matrix, with each entry representing a pixel in the image. Each entry is a value from 0 to 255 that represents the amount of black and white in that pixel. A 255 value represents 100% white and a 0 value represents 100% black. If only whole numbers are used, then there are 256 possible values for a given pixel. This is known as an 8-bit image.

The Python programming language will be used in order to explore the two compression techniques above. A 512px by 512px, 8-bit, grayscale image that is included in the scipy Python package. The original image can be seen below.

2 SVD Compression

Image compression can be done using a low-rank approximation on the single-value decomposition. The single-value composition takes the form

$$D = U\Sigma V^T$$

where the columns of U and V are the left and right singular vectors, respectively, and Σ) has diagonal entries that are the singular values of D. That is

$$\Sigma = diag(\sigma_1, ..., \sigma_m)$$

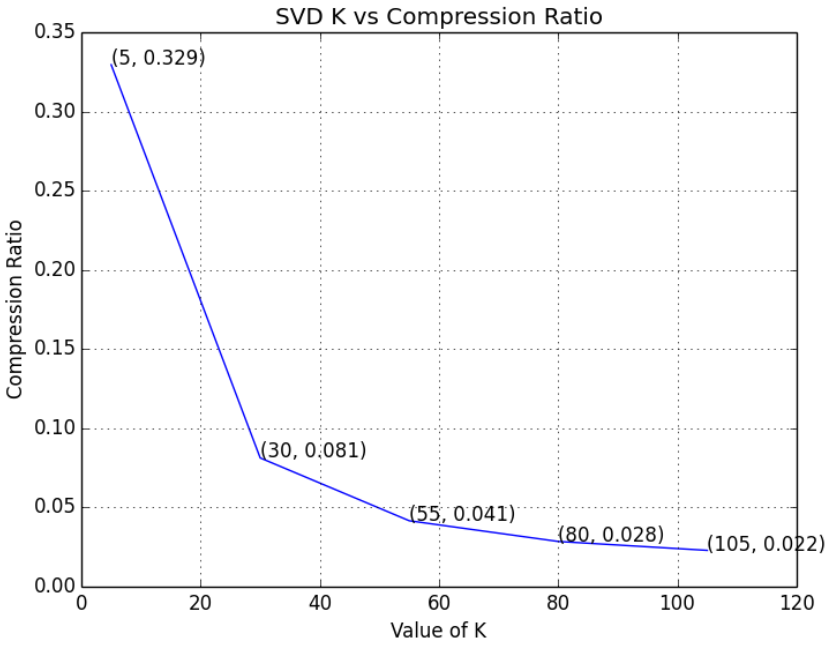
The low-rank approximation is then done by zeroing out the values in Σ where $r > k$, where r is the row index and k is a real number, and $0 <= k <= rank(\Sigma)$. This can be viewed as $\Sigma_k = \Sigma - diag(0, ..., \sigma_k, ..., \sigma_m)$. So to find the low-rank approximation for k calculate

$$D_k = U(\Sigma - \Sigma_k)V^t$$

An image can be compressed using the above formula. The lower the value of k, the lower the quality of the compressed image will be and the smaller the size of the image will be. Conversely, the higher the value of k as it approaches the rank of the image matrix, the higher the quality of the image will be and the larger the size will be.

This algorithm was applied to the image above, for $k = 5, ..., 105, k = k+25$, to generate a scale of images from high compression to low compression. The image below contains the resulting approximations, i.e $k = 5, k = 30, k = 55, k = 80, k = 105$ plus the original appended last.

As can be seen in the above image, it is difficult to tell a difference between the last two images. They are close to the same quality for normal viewing sizes. If the image is magnified then it is far easier to see a difference in quality. This led me to determine that $k = 80$ was the minimum number of values that must be kept for acceptable image quality for SVD low-rank approximation. The compression ratio for this k value is 2.8%, which is not a high amount of compression.



3 DCT Compression

Compression of an image can also be done via a discrete cosine transformation. This algorithm is used in some of the most widely used image and audio formats, like JPEG and MP3 [**?**, wikipedia] A DCT transforms an image from the spatial domain to the frequency domain, separating the parts of the image into areas of differing importance. It is similar to a discrete Fourier transformation.

The process to get the approximation to the process used for the low-rank approximation. First a DCT is found by the following equation [2].

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i,j) cos\left[\frac{\pi u}{2N}(2i+1)\right] cos\left[\frac{\pi v}{2M}(2j+1)\right] f(i,j) \tag{1}$$

where

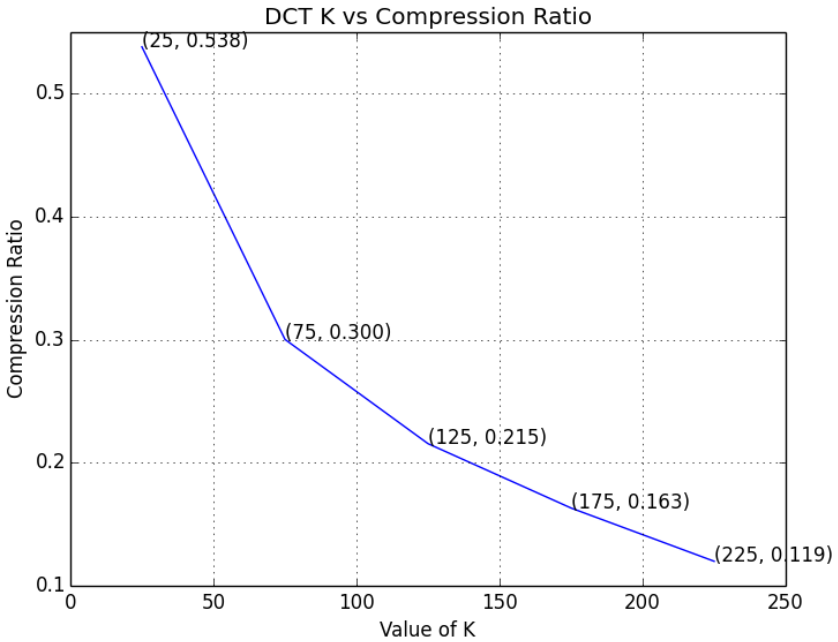
$$\Lambda(i) \begin{cases} \frac{1}{\sqrt{2}} & for \varepsilon = 0 \\ 1 & otherwise \end{cases} \tag{2}$$

Once a DCT D is found, an approximation is found by zeroing out the values in D where $r > k$, where r is the row index and k is a real number, and $0 <= k <= rank(D)$. This can be viewed as $D_k = D - diag(0, ..., D_k, ..., D_m)$. So to find the low-rank approximation for k calculate

$$F^{-}1(D - D_k)$$

As before, a lower value of k represents higher compression and lower quality, while a higher value of k indicates a lower compression but higher quality.

This was applied to the original image for $k = 25, ..., 225, k = k + 50$ to generate a series of approximations. These images are displayed in the below scale, with the original appended last. As can be seen there is not much difference between the last two images. I have concluded that $k = 175$ is the least amount of value that must be kept for an acceptable image quality. This value has a compression ratio of 16.3% which is significantly higher compression than that of the SVD compression with similar image quality. This can be seen in the graph below.

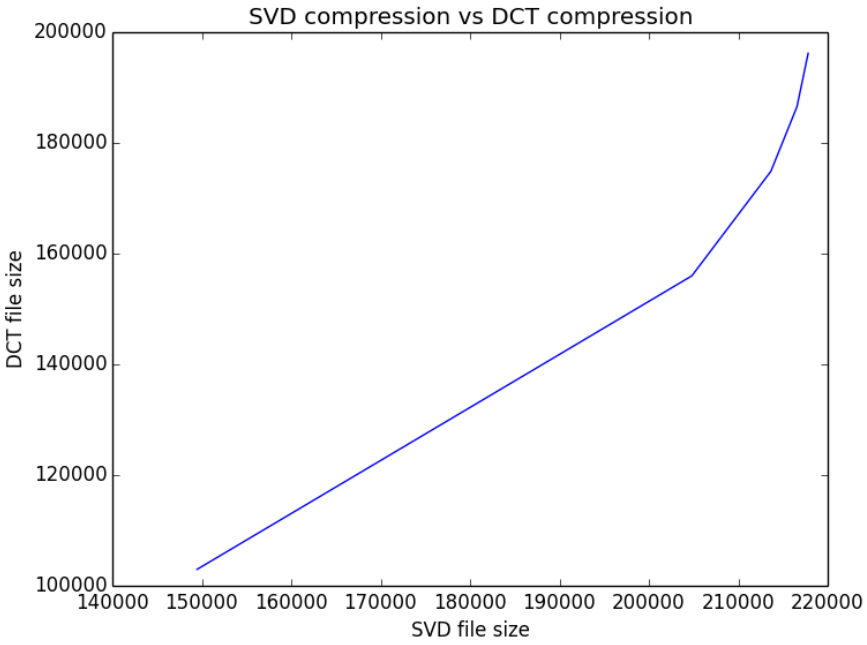


4 Conclusion

One of the more interesting parts of this was comparing how the decrease in image quality manifested itself between the two image compression techniques.

There is a clear difference in how the low quality SVD compressed images are distorted versus how the low quality DCT compressed images were distorted.

Another interesting difference was how DCT had a much higher compression ratio than the SVD compression, with the result images having similar image quality. The final compression ratio for DCT was about 8 times higher. The graph below shows a comparison between SVD and DCT for the various k values tested.



While the techniques for this project were explored using a grayscale image, all of the above could easily be applied to a RGB or RGBA image to further explore the topic. It would be interesting to see how the additional dimension of color affects the compression algorithms and compression ratios.

References

[1] Wikipedia, *Discrete Cosine Transformation.*, https://en.wikipedia.org/wiki/Discrete_cosine_transform

[2] Dave Marshall, *The Discrete Cosine Transform (DCT)*, https://www.cs.cf.ac.uk/Dave/Multimedia/node231.html