Ukrainian Catholic University

Faculty of Applied Sciences

Business Analytics & Computer Science Programmes

Use of Singular Value Decomposition and Discrete Cosine Transformation in Image Compression

Linear Algebra Second interim report

Authors:
Danyil Hutsul
Vladyslav Razoronov

12 May 2021

Contents

Abstract

A big problem for any internet platform is the size of transmitted data. Though an image that is transmitted in its raw state will have high fidelity and retain almost all of its quality, this comes at an expence of storage space, transmition time and bandwith. Due to this most images must be compressed to allow for efficient use of resources. Singular Value Decomposition and Discrete Cosine Transform are two widely known techniques frequently used in data compression. In this report SVD and DCT are implemented using python's numpy and scipy libraries and tested for their effectivness in image compression. The code can be found here

Keywords: Singular Value Decomposition, Discrete Cosine Transform, Image Compression, rank, Eigenvalues, Eigenvectors, two-dimensional DCT

1 Introduction

A digital image can be represented as 2D matrix, where each item $a_{i,j}$ of it represents either brightness or three color channels of pixelf on i-th row and j-th column. The compression of digital images is performed by reducing the dimensions of said matrix while trying to preserve the information contained inside. Singular Value Decomposition is one of the ways to compress a matrix. It is done by decomposing the matrix into 3 different matrices: U, S and V where $A = USV$ and A is our given matrix. Discrete Cosine Transform can also be used for compressing an image by representing its matrix as a sum of sinusoids of varying magnitutes and frequencies.

In this report we will analyze quality and efficiency of Singular Value Decomposition and Discrete Cosine Transform in order to determine their effectiveness in image compression. Both approaches will be tested using differently sized images. The main goal of this research is to determine the best use cases as well as the overall best approach.

2 Overview of Approaches

2.1 Singular Value Decomposition

Singular Value Decomposition (SVD for short) is a factorization of a real or complex matrix that generalizes the eigendecomposition of a square normal matrix to any $m \times n$ matrix. $A_{m,n}$ is an orthogonal matrix denoted by the following equation: $A = USV^T$ $U_{m,m}$ and $V_{m,m}$ are orthogonal matrices with columns composed of eigenvectors of AA^T and A^TA respectively $S_{m,n}$ is a diagonal matrix of the following form:

$$S_{m,n} = \begin{pmatrix} o_1 & 0 & \cdots & 0 \\ 0 & \vdots & \cdots & 0 \\ \vdots & \vdots & o_r & \vdots \end{pmatrix}$$

Where o are square roots of eigenvalues of A^TA called singular values of A with $o_1 >= o_2 >= \cdots >= o_r >= 0$ and $r = rank(A)$

2.2 Discrete Cosine Transform

The Discrete Cosine Transform expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCT takes N data-points as its input and returns N data-points as its output while ensuring that most of the input information is concentrated in only a few of the N output data-points. DCT is used

in most types of digital media, such as images, video and audio. Many different media formats use a modified DCT to compress their size, the most famous of which is .jpeg image format.

For a One-Dimensional DCT of a sequence of length N is defined as
$C_{(u)} = a_{(u)} \sum_{x=0}^{N-1} f(x) cos(\frac{\pi(2x+1)u}{2N})$, where $u = 0, 1, 2, ..., N-1$
An inverse transformation is declared in a similar manner as
$f(x) = \sum_{u=0}^{N-1} a_{(u)} C_{(u)} cos(\frac{\pi(2x+1)u}{2N})$, where $x = 0, 1, 2, ..., N-1$
For both equations the same $a_{(u)}$ holds:

$$\begin{cases} a_{(u)} = \sqrt{\frac{1}{N}}, \text{ for u = 0} \\ a_{(u)} = \sqrt{\frac{2}{N}}, \text{ for u != 0} \end{cases} . \tag{1}$$

When working with digital images Two-dimensional DCT is used. The two-dimensional transform is equivalent to a one-dimensional DCT performed along a single dimension followed by a one-dimensional DCT in the other dimension and is defined as:
$C_{(u,v)} = a_{(u)} a_{(v)} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos(\frac{\pi(2x+1)u}{2N}) cos(\frac{\pi(2y+1)v}{2N})$,
where $u = 0, 1, 2, ..., N-1$
$f(x) = \sum_{u=0}^{N-1} \sum_{y=0}^{N-1} a_{(u)} a_{(v)} C_{(u)} cos(\frac{\pi(2x+1)u}{2N}) cos(\frac{\pi(2y+1)v}{2N})$, where $x = 0, 1, 2, ..., N-1$

2.3 Lossy vs Lossless compression

Image compression can be categorized into two categories:

- Lossy compression
- Lossless compression

As per name, data transformed by a Lossy compression will be impossible to fully recover, while the Lossless algorithm allows for the complete restoration of the original data. When compressing images, a Lossless method would be preferable in situations where said images are frequently downloaded and copied, to properly preserve the data. However Lossy algorithms are noticeably more efficient space-wise, allowing for huge decrease in size and load time in exchange for a relatively small drop in quality. See Figure: 1 for comparison of Lossless and Lossy image formats. PNG - a Lossless image format takes up 52.4 KB. JPEG - a Lossy image format, takes up a noticeably smaller 31.3 KB. The images remain very similar though the png image takes up almost twice the size of JPEG one. This reduced size makes Lossy compression algorithms a very good choice for internet websites.

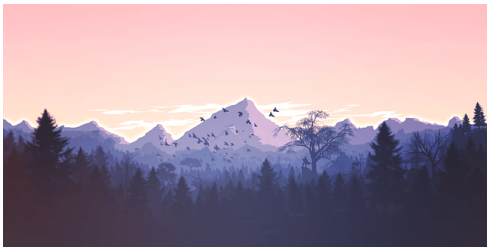
Singular Value Decomposition is classified as a lossy compression due to the fact that this involves reducing the number of singular values used. Removed singular values are only recoverable from the original picture.

A 2D Discrete Cosine Transform is lossless by itself. Though it is used in some lossy formats like JPEG, the loss does not result from the transformation itself. However, irrational numbers can sometimes occur which lead to round-off errors.

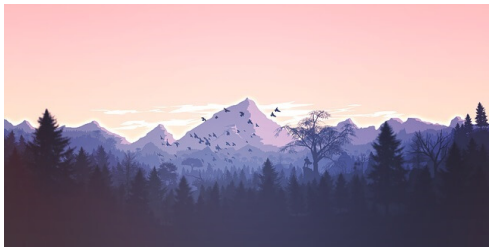
2.4 Application of SVD and DCT to digital images

2.4.1 Singular Value Decomposition

Let's take a look at an aforementioned representation of SVD as a sum of rank one matrices:



(a) PNG



(b) JPEG

Figure 1: Comparison of the same image in different formats

$A = \sum_{i=1}^{r} o_i u_i v_i^T$

By setting r to the number of singular values of A, i.e. $rank(A)$, we can fully reconstruct matrix A.

But by reducing r, i.e. reducing the number of singular values used, (while taking into consideration that $1 <= r <= rank(A)$) we obtain an approximation of our intial matrix A. If we take into account the fact that $o_1 >= o_2 >= \cdots >= o_r >= 0$, we can see that the columns (u_i and v_i^T) are chosen in a descending order of importance (i.e. more important columns are accounted for first). As such the received approximation will be fairly accurate.

By transforming an image into a matrix (Black-and-White) or 3 matrices (RGB) and applying SVD while reducing the value of r we get a compressed image.

2.4.2 Discrete Cosine Transform

The algorithm used for creating output image $B_{p,q}$ obtained from input image $A_{M,N}$ can be represented by the following formula:

$B_{p,q} = a_p a_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} cos\frac{\pi(2m+1)p}{2M} cos\frac{\pi(2n+1)q}{2N}$, where

$0 <= p <= M-1$ and $0 <= q <= N-1$

$$\begin{cases} a_p = \frac{1}{\sqrt{(M)}}, \text{ for p} = 0 \\ a_p = \sqrt{\frac{2}{M}}, \text{ for } 1 <= \text{p} <= \text{M}-1 \end{cases}. \tag{2}$$

$$\begin{cases} a_q = \frac{1}{\sqrt{(N)}}, \text{ for Q} = 0 \\ a_p = \sqrt{\frac{2}{N}}, \text{ for } 1 <= \text{q} <= \text{N}-1 \end{cases}. \tag{3}$$

(a)

(b)

Figure 2: Images used

3 Experiments

3.1 Implementation

To test the approaches images showcased in Figure 2 will be used. The images chosen are of different dimensions: 524px∗429px and 2800px∗1862px for images a) and b) respectively. The images will be used in grayscale as color images follow the same process of compression, by using three (R, G, B) matrices instead of a single black-and-white matrix. The analysis will be done by compressing the images to 5%, 15%, 25%, 50% and 75% of initial image dimensions.

The SVD approach is realized using numpy.linalg.svd method to split the image matrix into U S and V matrices. The image is then recreated using the following formula $A = USV^T$, while only accounting for N singular vales.

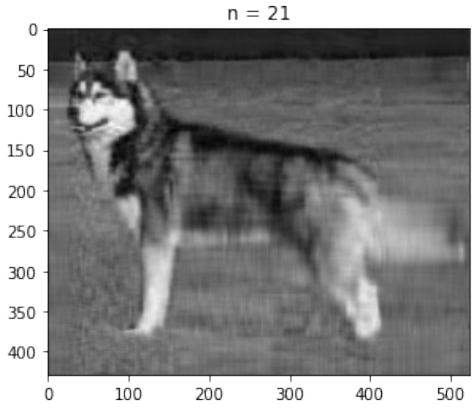
The DCT approach is realized by utilizing the scipy.fftpack library. The image matrix is transformed using a 2D Cosine Transform, than all but first n% of rows and columns are set to 0 and finally the compressed image is recreated using 2D Inverse Cosine Transform.

Considering the similar complexity level of both approaches it is fair to assume that the resulting compressed images will be of a similar quality.

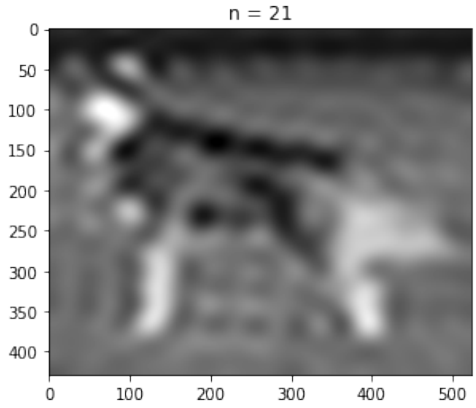
3.2 Results

The images showcased here show both approaches side to side. Image a) showcases the SVD approach, b) - DCT.

As we can see from Figures 3 and 8 the quality of image b) is much worse then image a). However, all other images are almost identical to each other.

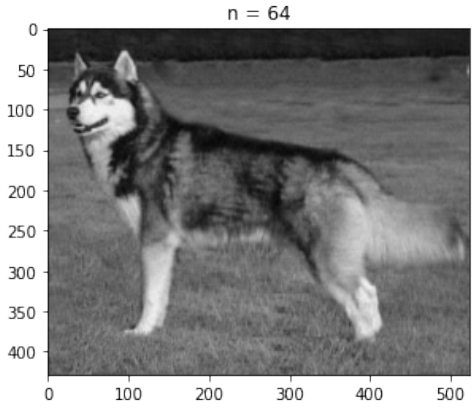


(a)

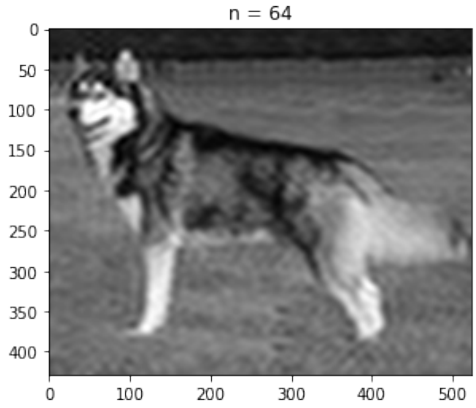


(b)

Figure 3: 5%



(a)

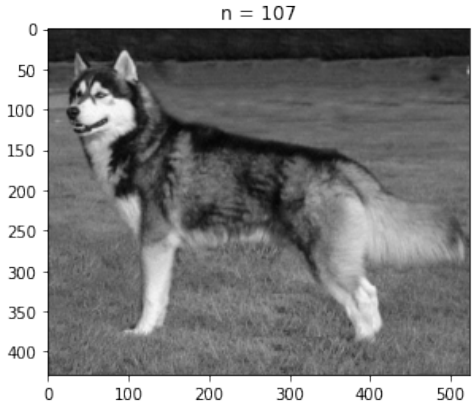


(b)

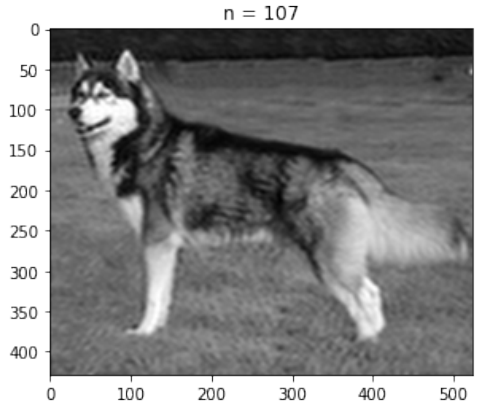
Figure 4: 15%

Both approaches show that the increase of desired size past 15% (Figures 4 and 9) of original dimensions leads to diminishing returns, drastically increasing size for an unnoticable amount of quality gain.

In terms of efficiency the DCT approach proves to be a far better option. Though for image A the time difference in negligible, for image B SVD approach is around 9 times slower.

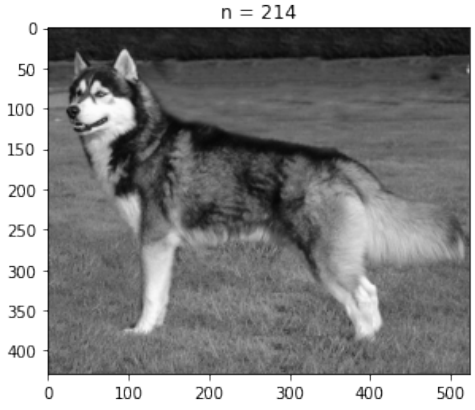


(a)

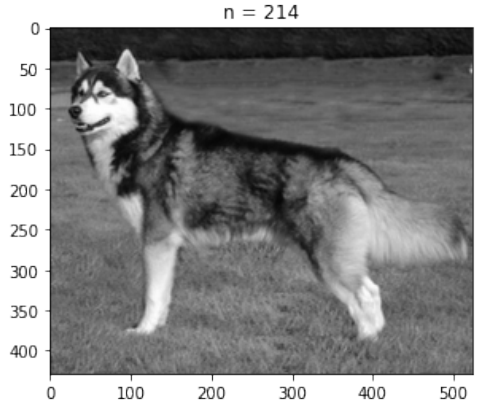


(b)

Figure 5: 25%

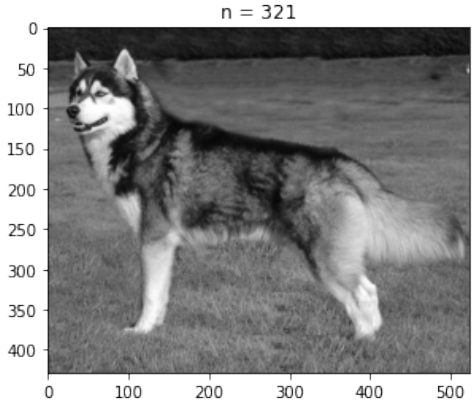


(a)

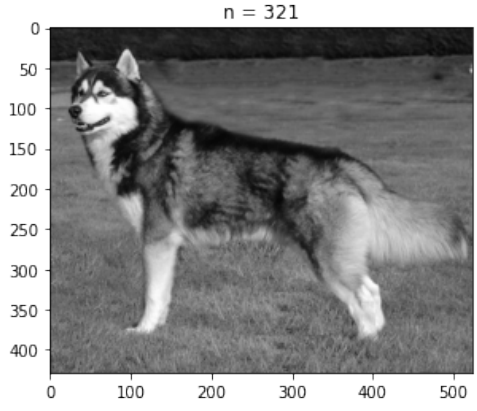


(b)

Figure 6: 50%



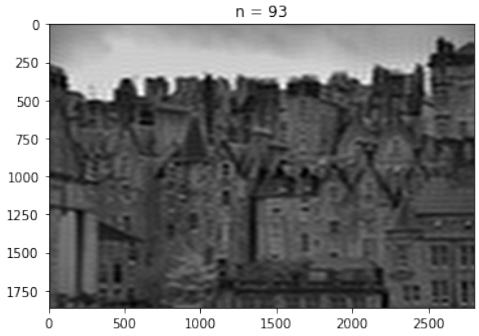
(a)



(b)

Figure 7: 75%

(a)



(b)

Figure 8: 5%

(a)

(b)

Figure 9: 15%

(a)

(b)

Figure 10: 25%

(a)

(b)

Figure 11: 50%

(a)

(b)

Figure 12: 75%

4 Conclusions

In this report we have compared different approaches to digital image compression. After analyzing the results we have determined that DCT is the overall best approach. Due to DCT's speed and efficiency there is little reason to use SVD. Though SVD offers better performance for high ratio compression and easier implementation, it is lossy and also massively underperforms in speed compared to DCT. Thus, SVD should not be used in real applications in favor of more efficient approaches.

References

[1] John Hopcroft , Ravi Kannan, *Computer Science Theory for the Information Age*, 2012

[2] Frank Cleary, *Singular Value Decomposition and Applications*, `http://github.com/linear-algebra`

[3] `https://www.mathworks.com/help/images/discrete-cosine-transform.html`

[4] SciPy Python Library Documentation `https://docs.scipy.org/doc/scipy/reference/index.htm`

[5] NumPy Python Library Documentation `https://numpy.org/doc/stable/contents.html`