Singular Value Decomposition

Maziar Kosarifar

October 2019

Introduction

This project is meant to study the use of Singular Value Decomposition in compressing images. In this project for simplification all images have been turned into black-and-white images. The same method can be used to compress images with color (The process for Red, Green, and Blue).
We use three given images, with different level of complexity, and and try to find the level of compression would keep the integrity of the image, while reducing the size, through SVD compression.
The program is written in Python, and the files can be found in my GitHub Account.

Study of the method

A matrix M can be represented as multiple of $M = USV^T$.
Where S is the square matrix with all elements zero and the diagonals equal to square root of eigenvalues of the matrix AA^T. Which is actually an important factor in why this method of compression is very effective, since we don't need to keep the zero elements in the memory, and can turn the S into a vector.
Now using the features of SVD, and knowing that if M is of dimension (m, n), U of dimension (m, p), S (p, p), and V^T (p, n), we can change the value of p, and still get an image of the same size. We only need to figure out how small p of a p would keep the integrity of the image.
To compute the amount of memory saved, we know the memory used to keep U, S, and V^T is

$$m * p + p + p * n$$

The original picture used $m \times n$ memory slots. Therefore by using a compression rate of "C" we can compress the image by the rate of:

$$\frac{m \times \frac{p}{C} + \frac{p}{C} + n\frac{p}{C}}{m \times p + p + p \times n} = (m + 1 + n)\frac{1}{C}$$

So we have linear compression rate.

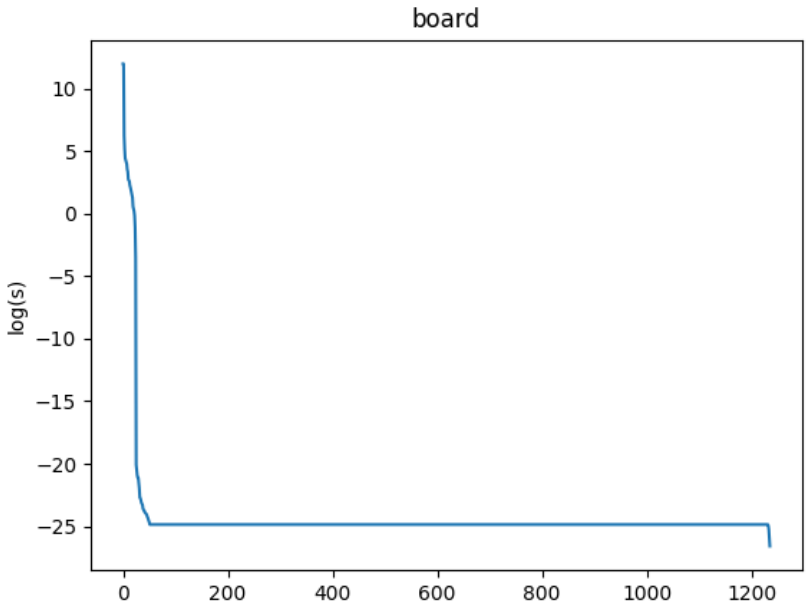
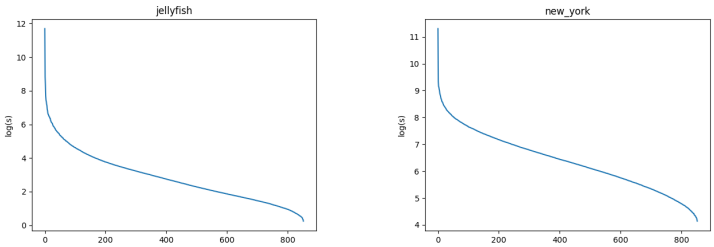


Figure 1: Chess board

Eigenvalues and eigenvectors can represent the very features of the original matrix(Here the image), and since they are stored in S in decreasing order, (as shows in the following plots), we can ignore the eigenvalues and their representing eigenvectors, and still not lose too much of the main picture.
This can be especially obvious in the case of the chess board, so much that except the very early few eigenvalues all are smaller than 1.
While in case of the Jellyfish and the New York landmark, the eigenvalues are larger, which means that they cannot be as efficiently compressed as the chess board.

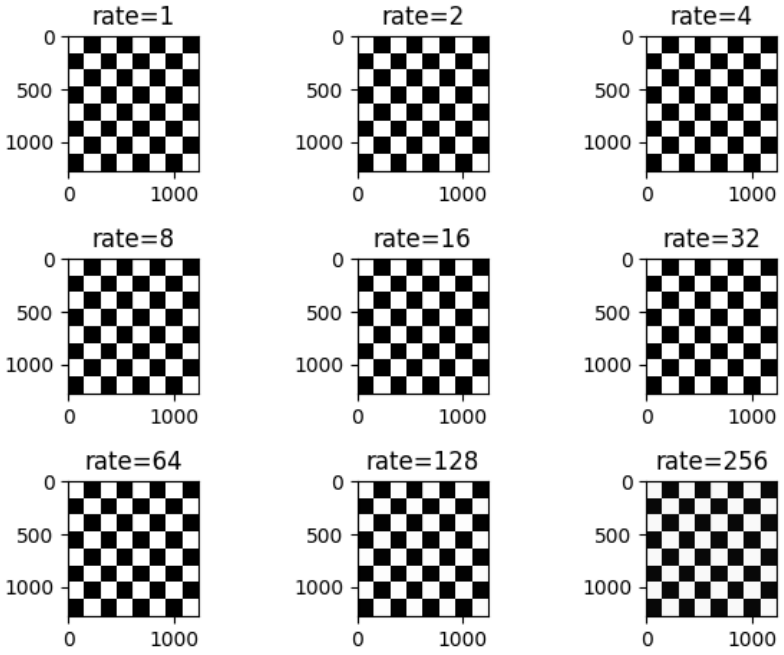


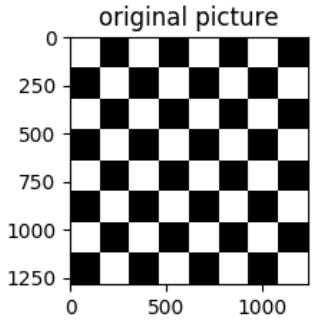
Finding the the optimal compression rate

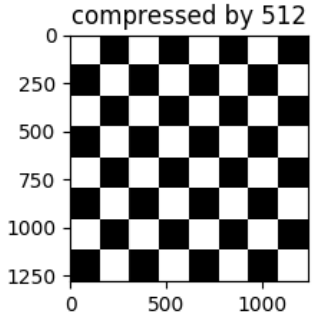
I have used the idea of a binary search with no upper bound to be able to find the at which rate the picture would still be recognizable.
So I started by plotting the original picture in black-and-white, and then compressing the image size by compression rate of 2, then 4, and until 2^8.

Chess Board

Plotting the different constructed images with the compression rates for chess board, shows that for a simplistic image this method of compression can be really useful. And so we can reduce the memory required by 512 times.

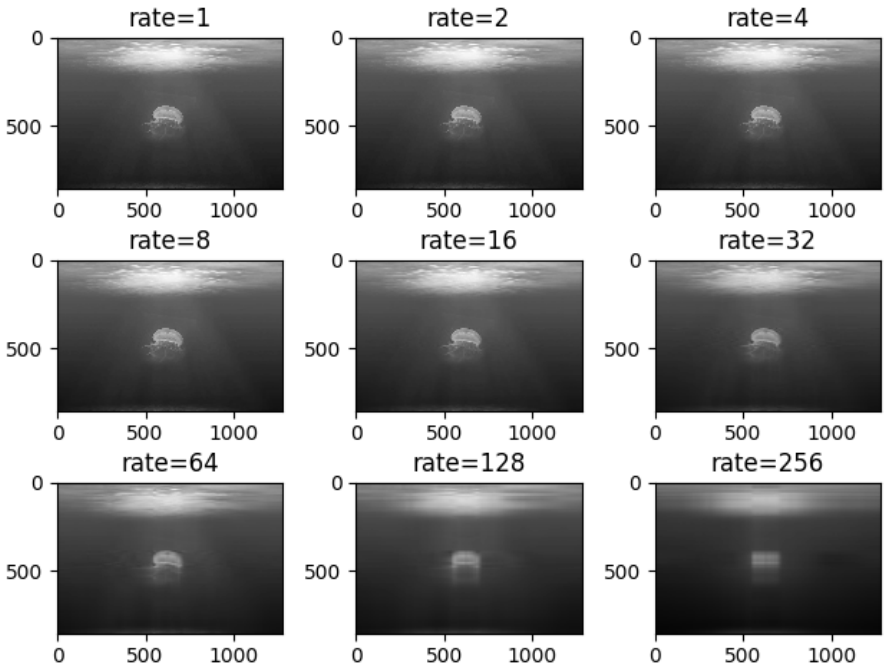


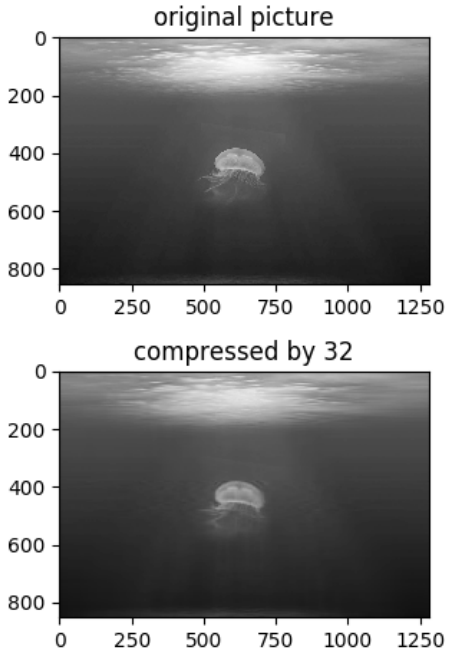

original picture
0
250
500
750
1000
1250
0
500
1000


compressed by 512
0
250
500
750
1000
1250
0
500
1000

Jellyfish

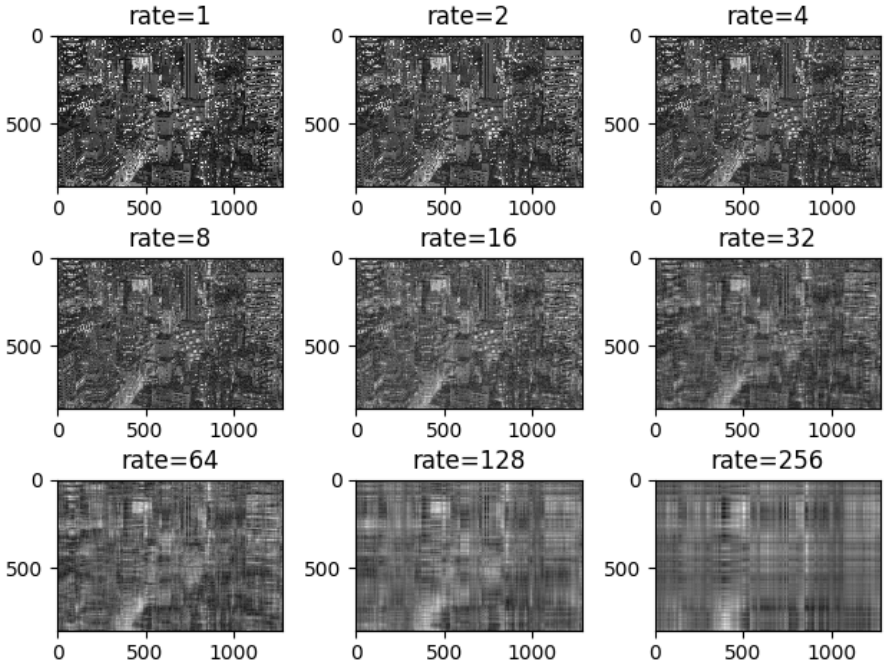
Since the eigenvalues in S for the jellyfish picture are mainly larger than the one for the chessboard the image cannot be as efficiently compressed, but we can still have an image with compression rate of 32 times.

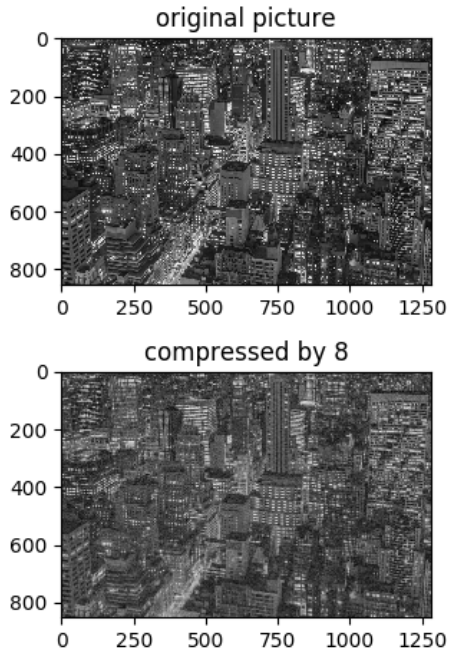



original picture
0
200
400
600
800
0
250
500
750
1000
1250
compressed by 32
0
200
400
600
800
0
250
500
750
1000
1250

New York

This is the most complex image among the three and hardest to compress.





Conclusion

In this assignment we studied the effectiveness of SVD in compressing images, and how effective it can be when the original image is rather simple. This method can also be used for images with color, since they are simply 3, 2d arrays, so the process will be 3 times slower, but just as effective.

The better way of finding the best compression rate is to study the log of eigenvalue plots, and find the number of eigenvalues and find the plot points, and used them to determine how many of the eigenvalues we can lose without losing the integrity of the picture.