Using Singular Value Decomposition forperforming image/video compression from scratch (without Python libraries)

Priya Jani
Bachelor of Technology
Ahmedabad University
Ahmedabad, India
priya.j@ahduni.edu.in

Priyanshu Pathak
Bachelor of Technology
Ahmedabad University
Ahmedabad, India
priyanshu.p@ahduni.edu.in

Yansi Memdani
Bachelor of Technology
Ahmedabad University
Ahmedabad, India
yansi.m@ahduni.edu.in

***Abstract*— The increasing usage of digital information also requires storage and transmission of images and videos. A technique called SVD (Singular Value Decomposition) is used to compress these images without affecting the quality of the image. This linear matrix transformation uses a product of three matrices and compresses the image by keeping necessary features. It finds the basic structure of the image by transformation into a series of linear approximations. The main idea behind this is to select some eigenvalues to compress and reconstruct the image. It deals with the rank of image and compression ratio.**

Keywords*— *Singular Value Decomposition, linear matrix transformation, linear approximations, eigenvalues, rank, compression ratio.

I. Introduction

The use of digital information is increasing day by day with the advancement in technology. Images or videos are used to store the media material. The issue here is storing and transmitting photos and videos. Compression is a clever method to store things in less memory. However, compression should be done in such a way that the content's quality is not compromised. By employing a linear transformation, SVD aids in this endeavour. Matrix A is an image matrix that may be expressed as the product of three matrices: U, S, and V, with S being a diagonal matrix containing singular values of matrix A.

II. Background

An image is basically a matrix of pixels containing different RGB values. This image in its raw form is very large and therefore storage and transmission are time-consuming. With large amounts of data becoming the norm, it is very crucial to transfer and store the said data efficiently. That is why compression of data, for its efficient storage and transfer, is extremely important. In this project, we will use Singular Value Decomposition (SVD) which is a very popular method used for image compression without compromising with the quality. The SVD was discovered over 100 years ago independently by Eugenio Beltrami (1835–1899) and Camille Jordan (1838–1921), James Joseph Sylvester (1814–1897), Erhard+ Schmidt (1876–1959), and Hermann Weyl (1885–1955).

III. Motivation

As previously said, it is critical to store and transfer data in such a way that vital information is retained. The primary purpose of creating this project report is to learn how to accomplish it. When compared to other compression algorithms, SVD produces good compression results with reduced computing complexity. Learning how to apply linear algebra in the realm of file compression can be extremely beneficial.

Also, it is important to learn about the various backend processes that take place while using libraries. Implementing programs from scratch will help in understanding linear algebraic concepts like Singular Value Decomposition.

There are 2 types of image compression techniques, i.e., lossy and lossless. SVD is a technique that uses fewer singular values to construct image and hence compresses image. The benefits of using SVD are:

1. SVD yields a good compression ratio compared to lossless techniques while also not critically affecting the image quality.
2. You have more control over the compression ratio as changing the value of k changes the amount of compression to be done.
3. Ringing and blocking artifacts can be prevented by using SVD compression.
4. One of the things that influences image compression is redundancy. There are three types of redundancies present while compressing an image: Coding redundancy, Inter-pixel redundancy, and psycho-visual redundancy. SVD tries to take advantage of these redundancies and ensures that the image size reduces while not affecting the quality

IV. Literature Survey

Image compression is the process of removing superfluous or extraneous data from an image. Removing redundancies is the same as lowering the number of bits necessary to represent a picture without sacrificing image quality. To achieve this, different image compression approaches use different ways or more properly coding algorithms. Only a few singular values are preserved after applying SVD to compress an image, while other singular values are destroyed. This is due to the fact that singular values on the diagonal of D are sorted in descending order, with the first singular value containing the most information and following singular values containing diminishing quantities of image information.

As a result, lower singular values carrying no or little information can be eliminated without causing severe visual distortion.

The application of SVD is not only compressing images but also in noise reduction, data science, face recognition, watermarking, and many more. The main general idea is choosing the appropriate value of k (number of eigenvalues used to compress and reconstruct an image). From a research paper by Miss Samruddhi Kahu and Ms. Reena Rahate, it can be observed that compression ratio and image quality go hand in hand. A smaller value of k will give a higher compression ratio but has more image degradation and vice versa. Thus, the appropriate value of k helps to compress the image without degrading the image quality to a certain extent.

Another research paper by Mrs. Rehna V.J and Mr. Abhranil Dasgupta shows jpeg compression using MATLAB R2010.

With the help of three principal types of data redundancies that are coding redundancy, spatial redundancy, and irrelevant information are used in the steps to compress an image. The step-by-step transformation of RGB components and reconstruction of the same is linked with applying approximation on matrices and removing singularity in order to reconstruct the image.

Concept of QR decomposition is also used in finding the eigenvalues and eigenvectors. Finding eigenvalues of matrices with dimensions greater than 2x2 becomes tedious, and that is why QR decomposition is used. In this method, we decompose a matrix A into an orthogonal matrix Q and an upper triangular matrix R.

V. Mathematical Model

Mathematical formulation for SVD is as follows:
Here A is a m x n matrix, U and V are orthogonal matrices. S is a diagonal matrix that contains the singular values of A. A can be written in the form of:

$A = USV^T$
Matrix U is an $m \times m$ orthogonal matrix

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij} = \begin{cases} 1,,, i = j \\ 0,,, i \neq j \end{cases}$$

Similar is matrix V.

S has singular values in decreasing order:
$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \sigma_4 .... \sigma_r \geq \sigma_{r+1} = ... = \sigma_p = 0$

$$S = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \sigma_{r+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & \sigma_n \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Multiplying them,
$A = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + \cdot\cdot\cdot + u_{n-1}\sigma_{n-1}v_{n-1}^T + u_n\sigma_n v_n^T$

$$A = \sum_{i=1}^{n} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$$

Although rank r of matrix contains the information: (r < m or r < n)
After "r" rest all are 0.
$A = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + \ldots + u_n\sigma_n v_n^T + 0u_{r+1}v_{r+1}^T + \ldots 0$
So, the product becomes:
$A = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + \ldots + \sigma_r u_r v_r^T$

For image compression instead of r, k is taken such that k < r:

$$A_{approx.} = \sum_{i=1}^{k} \sigma_i \mathbf{u_i} \mathbf{v_i}^T$$

For our project we have used QR decomposition to obtain eigenvalues and eigenvectors. QR decomposition decomposes matrix (A) into orthogonal (Q) and triangular matrix(R).

$A=Q*R$

Since Q is an orthogonal matrix, $Q^T*Q = Q*Q^T = 1$ and $Q^T = Q^{-1}$

We find $A_{i+1} = R_i Q_i$
$R_i = Q_i^{-1} A_i$
$R_i = Q_i^T A_i$
$A_{i+1} = Q_i^T A_i Q_I$

Compression ratio is measured to check the compression with respect to the original image.

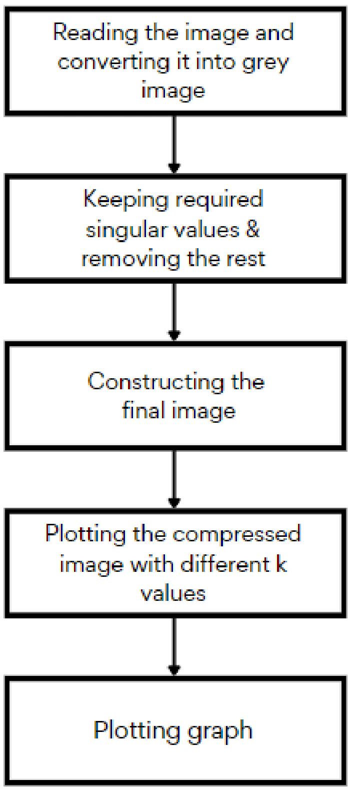
$Cr = m*n / (k(m + n + 1))$

And Mean Square error (MSE) is used to measure the quality of compressed images.

$$MSE = \frac{1}{mn} \sum_{y=1}^{m} \sum_{x=1}^{n} (f_A(x,y) - f_{A_k}(x,y))$$

VI. Coding Approach

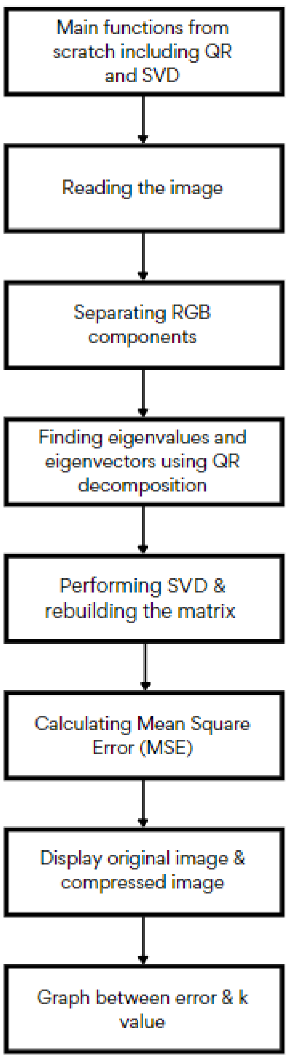
First for the greyscale images, we have used MATLAB. The flowchart is as follows:


Reading the image and converting it into grey image
Keeping required singular values & removing the rest
Constructing the final image
Plotting the compressed image with different k values
Plotting graph

For colored images, we have used python. Few important functions used are:

1. print_matrix: To print matrix
2. rows: To find rows
3. cols: to find columns
4. eye: returns identity matrix
5. pivot_index: Returns index of pivot in a row
6. pivot_value: returns value of pivot
7. swap: to swap rows
8. transpose: returns transpose of matrix
9. mat_multiply: To multiply two matrices
10. mat_splice: Returns a matrix with first r rows and first c columns of original matrix
11. is_diagonal: to check if matrix is diagonal
12. qr: returns the QR decomposition of matrix
13. qr_eig: to find eigenvalues and eigenvectors of matrix
14. svd: to perform SVD on matrix
15. reconstruct: to reconstruct the final compressed image
16. mean_square: to calculate mean square error between two matrices

The following is the pseudo flowchart that explains the procedure step by step.


Main functions from scratch including QR and SVD
Reading the image
Separating RGB components
Finding eigenvalues and eigenvectors using QR decomposition
Performing SVD & rebuilding the matrix
Calculating Mean Square Error (MSE)
Display original image & compressed image
Graph between error & k value

The entire code is divided into 5 main sections:

1. Utility functions: Basic linear algebra operations such as matrix multiplication, the function to find transpose, and other utility functions are included in this part and will be utilized later in the code.
Here functions are created from scratch such as transpose, dot product, scaling, clipping of matrices

2. Functions related to QR Eigenvalue Algorithm: This section offers functions that aid in determining the eigenvalues and eigenvectors of a symmetric matrix using the aforementioned approach.

3. Functions related to SVD: This section applies SVD to a matrix and returns the associated U, S, and V^T matrices. To be more specific, the function executes thin / reduced SVD on a matrix. Thin SVD only returns the elements of the matrix that have non-zero singular values.

4. The Main Section: This section includes the core image compressing code. To read an image file and transform it to a matrix, packages such as matplotlib and NumPy are used.

5. Analysis: This section contains information on the outcomes achieved. To compare accuracy, we use Mean

Square Error (MSE), which is determined by squaring the distance between each value of the original picture and each value of the compressed image.

VII. Numerical Results and Conclusion

From the graphs and images as outputs, the following can be concluded:

1. Small the K is, compression is more. Less singular values are present in matrix. As much of the data is lost and thus less storage will be required to store the image.

$$Cr = m*n/ (k (m + n + 1))$$

Compression ratio will be high.

2. MSE is the error that determines quality. As K becomes closer to the rank of matrix, better quality it would be as compared to the original matrix. The matrix that is reconstructed is approximately the same as MSE is small.

3. The MATLAB Grayscale approach showed that the k has an important role in compression. The following images show how drastic change can occur if we alter the value of "k".

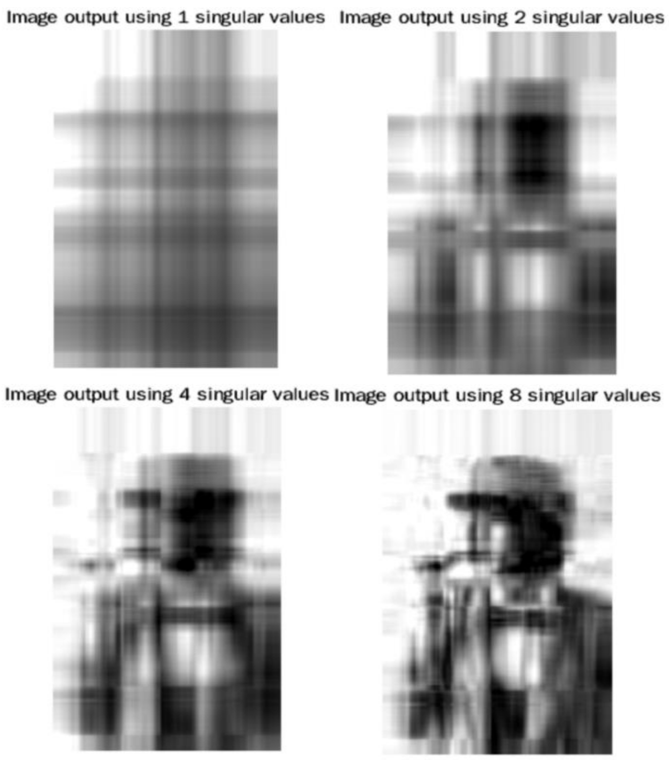


Image output using 1 singular values
Image output using 2 singular values
Image output using 4 singular values
Image output using 8 singular values

Image outputs using 1,2,4,8 singular values

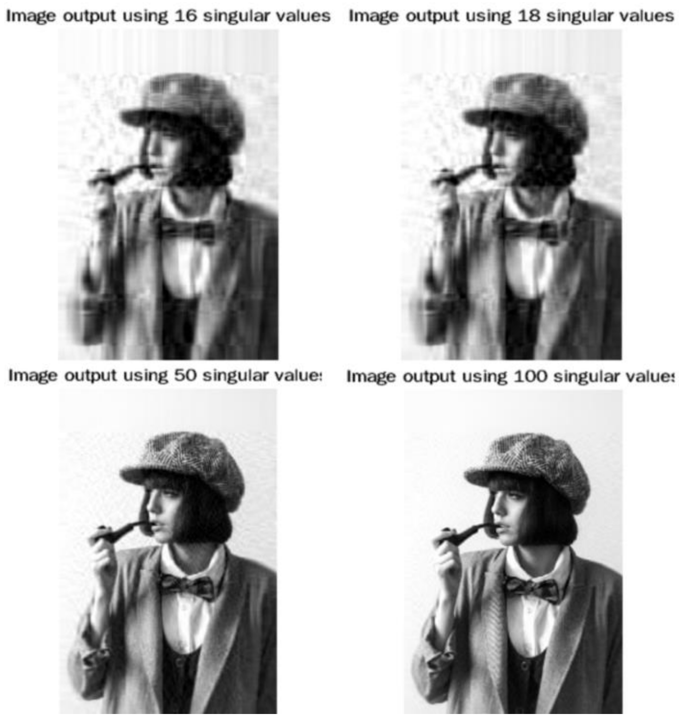
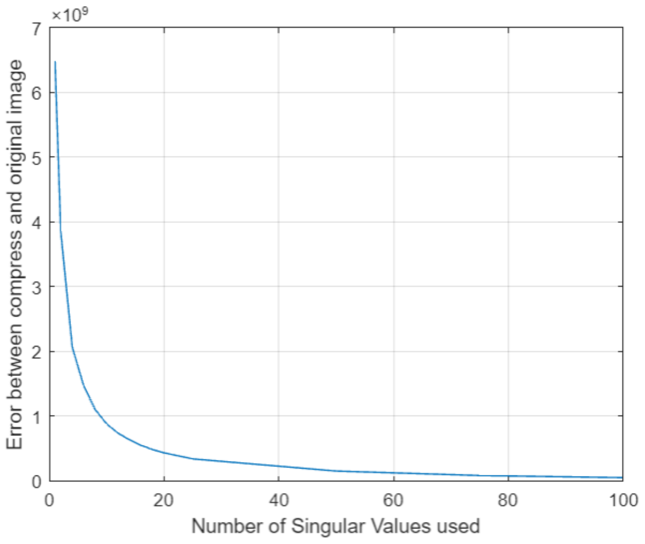


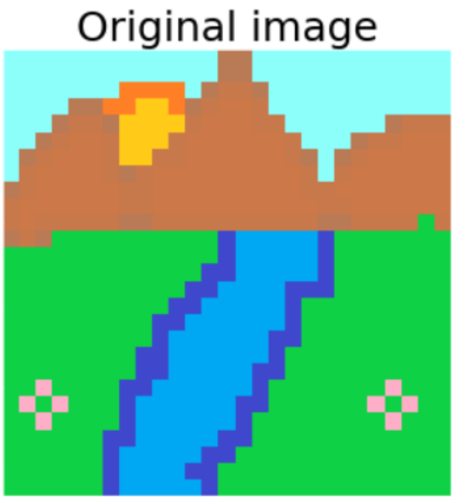
Image output using 16 singular values
Image output using 18 singular values
Image output using 50 singular values
Image output using 100 singular values

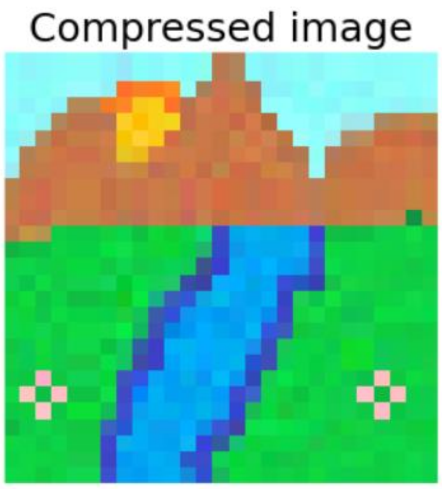
Image outputs using 16,18,50,100 singular values

4. The relation between error rate and number of singular values taken is shown below. It is clearly observed that the more singular values taken will relate more closely to the original matrix and hence error rate is less and vice versa.



5.The Colored Images are divided into 3 layers and each layer is compressed individually and then merged together as one. Compression does change the color and below is shown the same.

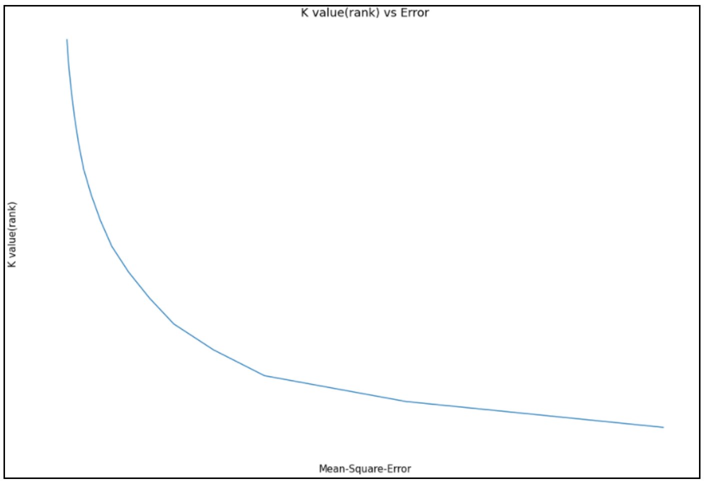




6. The error and number of singular values taken for colored images has same relation as grayscale. It shows same curve.

```
MEAN SQUARE ERRORS:

APPROACH        MEAN ERROR
-----------   ------------
SVD with QR      0.0034153
```



VIII. Contribution Pointwise

A. *Priya Jani*

- Research
- Literature Survey
- Mathematical Model
- Numerical Results
- Presentation
- Code
- Documentation

B. *Priyanshu Pathak*

- MATLAB Simulation
- Code
- Literature Survey
- Coding Approach
- Research
- Presentation

C. *Yansi Memdani*

- Research
- Background
- Motivation
- Literature Survey
- Code
- Presentation

IX. References

[1] Jameela, Rehna. (2011). JPEG Image Compression using Singular Value Decomposition. IOSR Journal of Electrical and Electronics Engineering. 1. 81 - 88.

[2] Kahu, S. and Rahate, R., 2013. Image Compression using Singular ValueDecomposition. [ebook] SciResPub., pp.244-248. Available at: <http://www.ijoart.org/docs/Image-Compression-using-Singular-Value-Decomposition.pdf> [Accessed 28 September 2021].

[3] Singular value decomposition applied to digital image ... (n.d.). Retrieved October 20, 2021, from https://sites.math.washington.edu/~morrow/498_13/svdphoto.pdf.

[4] Image compression using singular value ... - math. (n.d.). Retrieved October 20, 2021, from https://www.math.utah.edu/~goller/F15_M2270/BradyMathews_SVDImage.pdf.