Image Compression using Truncated SVD

AI25BTECH11012 - GARIGE UNNATHI

Summary of Strang's video :

In the video Gilbert Strang explains how the Singular Value Decomposition(SVD) can be used to factorise any matrix $\mathbf{A}$ as :

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$$

Where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and Σ is a diagonal matrix of singular values in descending order.
We find these matrices by :

$$\mathbf{A}^T\mathbf{A} = \mathbf{V}\sigma^T\sigma\mathbf{V}^T$$
$$\mathbf{A}\mathbf{A}^T = \mathbf{U}\sigma\sigma^T\mathbf{U}^T$$

By finding the eigen values and eigen vectors of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ we can find Σ , $\mathbf{V}$ and $\mathbf{U}$ cause both $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ are positive semi definite matrices. The square roots of the eigen values gives the singular values .

The key idea that we are using in this project is the fact for any matrix $\mathbf{A}$ SVD breaks up the original matrix into combinations of rank 1 matrices scaled by singular values

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

We are using this property in compression of an image which can be expressed as a huge matrix having each pixcel as an entry . The idea here is to find a lower and simpler matrix $\mathbf{A}_k$ where rank of $\mathbf{A}_k$ is k and :

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

Algorithm used:

As mentioned above I tried to impliment the SVD theorem by finding $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$. The eigenvalues of the matrix $\mathbf{A}^T\mathbf{A}$ or $\mathbf{A}\mathbf{A}^T$ correspond to squared singular value σ^2 and the eigen values of the matrices $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ correspond to columns and rows in $\mathbf{V}$ and $\mathbf{U}$ respectively .

To find the eigen vectors and eigen values for the matrix $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ we are using power iteration method .

Power iteration method :

For a symmetric matrix $\mathbf{A}$, power iteration method finds the dominant or the largest eigen pair (λ_i, v_i) . It works by repeatedly multipying a random vector by $\mathbf{A}$ and normalising the result until this resultant vector converges . This resultant vector is in turn the eigen vector cooresponding to the largest eigen value .

Proof for the above statement :
For any symmetric matrix , all the eigen values are real and all the eigen vectors are orthogonal therefore there exists n linearly independent eigen vectors which form the basis of a vector space .
From this we can say that if we taka any vector $\mathbf{v_0}$, we can express this vector as the combination of the eigen vectors .

$$\mathbf{v_0} = c_1x_1 + c_2x_2 + ... + c_nx_n$$

When we multiply this vector repeatedly

$$\mathbf{A}^k\mathbf{v_0} = \mathbf{A}^k(c_1x_1 + c_2x_2 + ... + c_nx_n)$$

The term with the largest eigen value grows faster and the vector aligns with that eigen vector.

We find the corresponding eigen value by using the formula :

$$\lambda = \mathbf{v}^T\mathbf{A}\mathbf{v}$$

For finding the other eigen values we repeat the some but after removing the infuence of the eigen vector which is already found from our original vector $\mathbf{A}$ .This is acheived by the equation :

$$\mathbf{A} = \mathbf{A} - \lambda_1\mathbf{v}_1\mathbf{v}_1^T$$

Thus by using this process we can find the eigen values and eigen vectors of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ which will help us find the matrices $\mathbf{U}$,$\mathbf{V}$ and Σ

Comparision with other Algorithms :

In this project, the Singular Value Decomposition (SVD) of an image matrix was implemented from scratch using the Power Iteration method for eigenvalue computation. The Power Iteration - based approach offers several advantages, particularly in enhancing the understanding of the underlying mathematical concepts, enabling implementation from first principles, and allowing clear visualization of how singular values influence image reconstruction and compression quality.

Algorithm	**Concept**	**Advantages**	**Disadvantages**
Power Iteration	finds the dominant eigen vector and eigen value by repeatedly multiplying A to a random vector and normalising it	Conceptual clarity; Simplicity; Computational efficiency for moderate sizes.	Finds only one eigenpair at a time; Limited numerical accuracy.
Jacobi	Iteratively diagonalizes a symmetric matrix by applying successive rotations to zero out off-diagonal elements.	Accurate for small to medium matrices; Provides all eigenpairs.	Computationally expensive for large matrices; Convergence can be slow.
Direct Library - SVD (NumPy)	Uses optimized, built-in algorithms	Fast, stable, and provides all singular values/vectors directly.	Less educational insight

TABLE 0

The main goal of the project was not only to compute the SVD, but also to understand and demonstrate the mathematical process of matrix factorization - especially how eigenvalues and singular values relate to each other.

The Power Iteration method was chosen because:

- Simplicity
- Helps visualize how singular values affect content in images.
- Computational Efficiency

Reconstructed Images

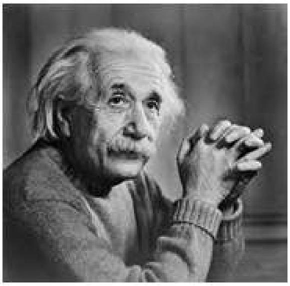

Fig. 0.1: Original Einstein image

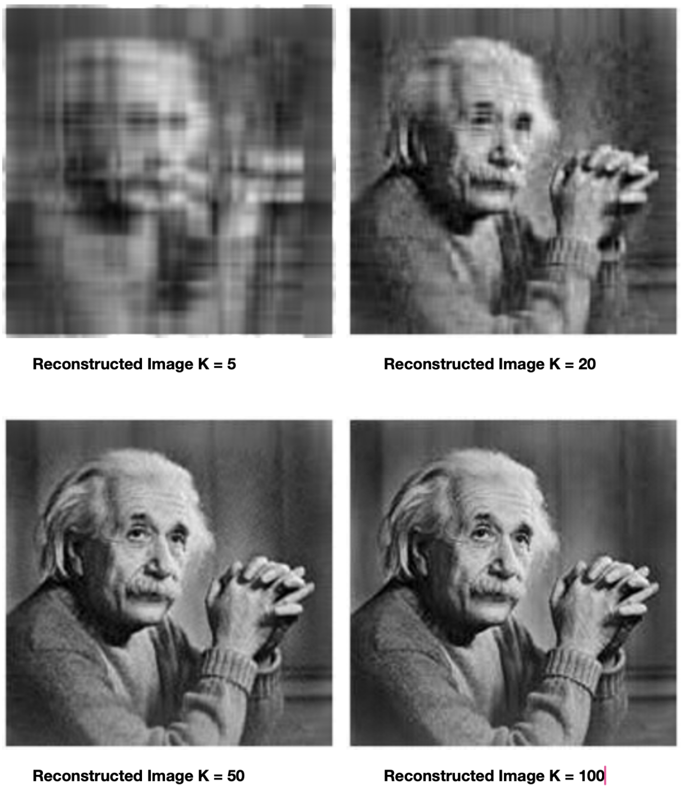
Reconstructed Image K = 5
Reconstructed Image K = 20
Reconstructed Image K = 50
Reconstructed Image K = 100

Fig. 0.2

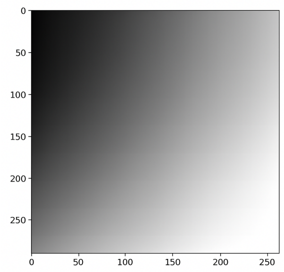

Fig. 0.3: Original Greyscale image

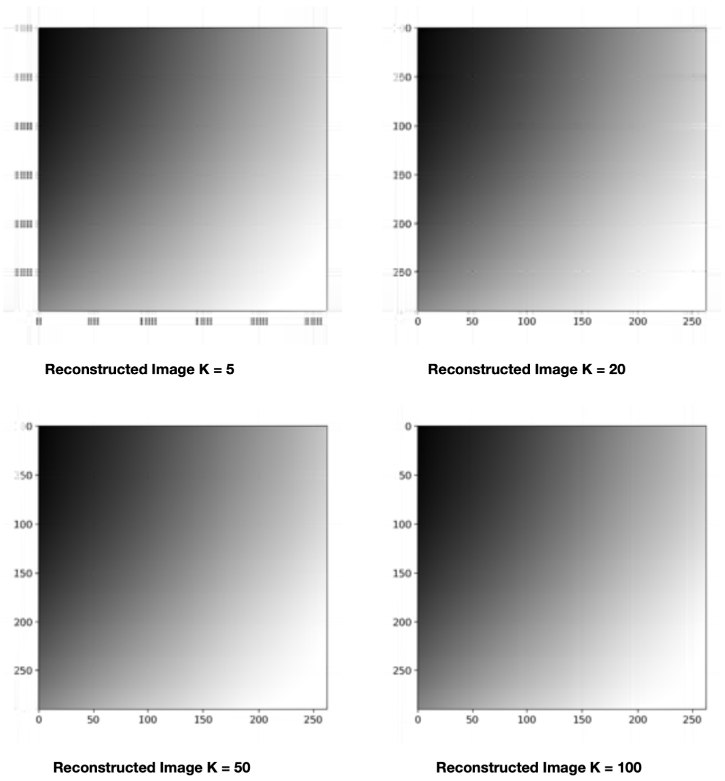

Fig. 0.4

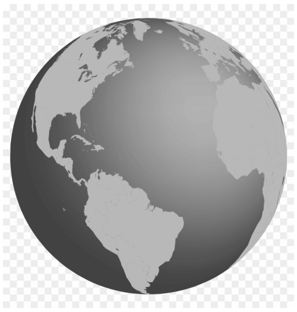

Fig. 0.5: Original Globe image

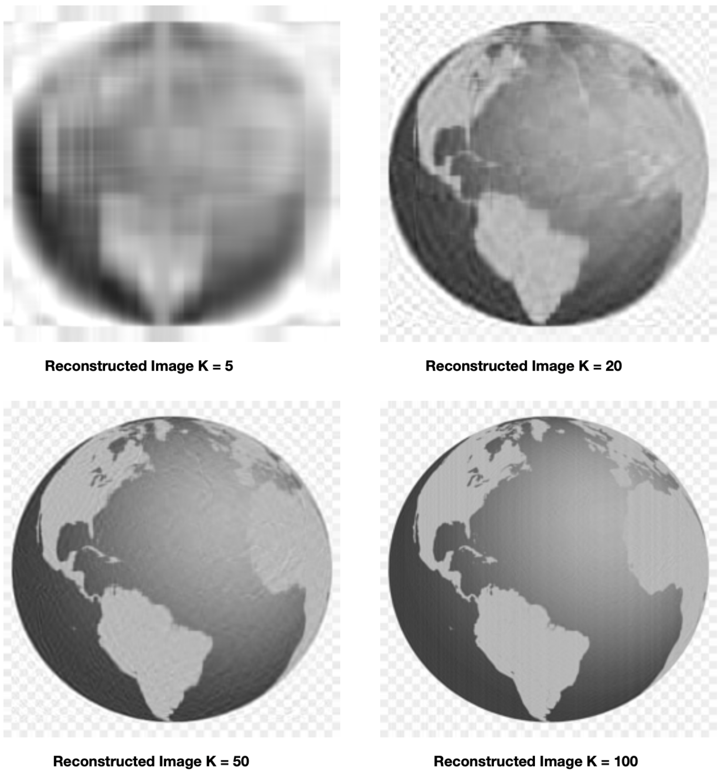
Reconstructed Image K = 5
Reconstructed Image K = 20
Reconstructed Image K = 50
Reconstructed Image K = 100

Fig. 0.6

Error Analysis :

In this project the quality of the reconstructed image A_k is evaluated by comapring it to the original image matrix A .
The Forbius norm is used to measure this error which is the overall difference between the matrices .

$$Error = \|\mathbf{A} - \mathbf{A}_k\|$$

Here are the tables that give us the Forbius error of each image at differnt k values :

K value	**Forbius Error**
$\boldsymbol{K = 5}$	6421.52
$\boldsymbol{K = 20}$	2802.71
$\boldsymbol{K = 50}$	1070.95
$\boldsymbol{K = 100}$	339.51

TABLE 0: Einstein

K value	**Forbius Error**
$\boldsymbol{K = 5}$	1608.59
$\boldsymbol{K = 20}$	900.41
$\boldsymbol{K = 50}$	503.59
$\boldsymbol{K = 100}$	275.67

TABLE 0: Greyscale

K value	**Forbius Error**
$\boldsymbol{K = 5}$	5950.07
$\boldsymbol{K = 20}$	2793.60
$\boldsymbol{K = 50}$	1315.09
$\boldsymbol{K = 100}$	602.15

TABLE 0: Globe

Conclusion :

Developing the SVD using the Power Iteration method provided a deeper understanding of the mathematical foundation behind matrix decomposition and image compression.

The Power Iteration - based SVD method used in this project was chosen for its simplicity and conceptual clarity, but it comes with several trade-offs :

- Its accuracy is lower compared to optimized methods
- computationally slower for large matrices

Through this project I successfully achieved the project's objectives :

- to compute the singular value decomposition manually
- reconstruct images at various compression levels
- analyze the effect of truncation on reconstruction accuracy