# Matrix Theory Project Report

## Image Compression using Singular Value Decomposition (SVD)

**Name: Megha Shyam**
Roll No: AI25BTECH11005

# 1. Summary of Strang's Video

From Prof. Gilbert Strang's lecture on the Singular Value Decomposition, I understood that any matrix can be broken down into three parts:

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

The columns of $U$ and $V$ represent important directions of the matrix, and $\Sigma$ contains singular values showing how much the matrix stretches along each direction. For images, this means we can keep only a few large singular values to recreate the image almost perfectly, which results in compression. The idea is simple: most of an image's information lies in a few dominant components.

# 2. Explanation of the Implemented Algorithm

The project was implemented completely in Python using only `numpy` and `matplotlib`. No built-in SVD or `linalg` functions were used.

## Mathematical Idea

For a given image matrix $A$:

$$B = A^T A$$

We find eigenvalues and eigenvectors of $B$:

$$Bv_i = \lambda_i v_i$$

The singular values are:

$$\sigma_i = \sqrt{\lambda_i}, \qquad u_i = \frac{Av_i}{\sigma_i}$$

Then the approximate image is reconstructed as:

$$A_k = U_k \Sigma_k V_k^T$$

## 3.Pseudocode

```
Input: Image A, number of components k

1. Convert image to grayscale
2. Compute B = A^T * A
```

```
3. For i in 1..k:
     a. Find largest eigenvalue & eigenvector of B (power
        iteration)
     b. Remove its effect (deflation)
     c. Apply Gram-Schmidt to keep vectors orthogonal
4. Compute singular values sigma = sqrt(lambda)
5. Find U = (A * V) / sigma
6. Reconstruct A_k = (U * Sigma) * V^T
7. Display the compressed image
```

# 4. Reconstructed Images for Different k

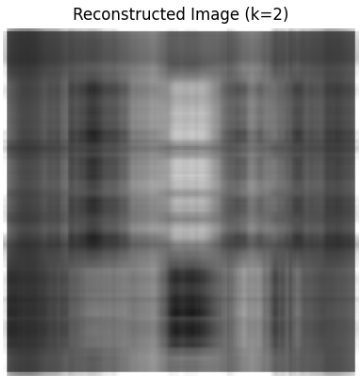


Figure 1: k = 2

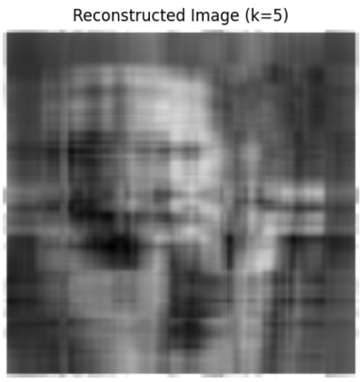


Figure 2: k=5

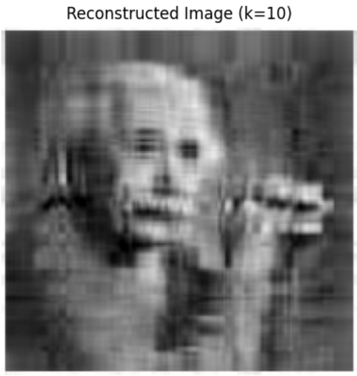


Figure 3: k=10

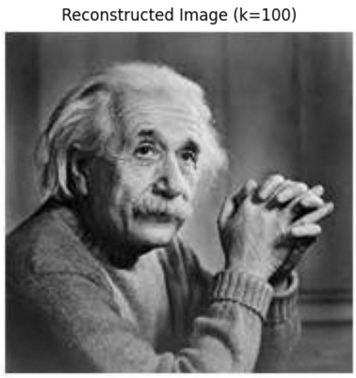


Figure 4: k=100


Figure 5: k=100

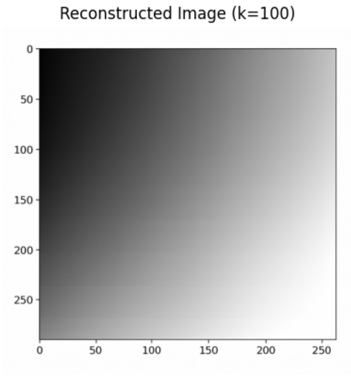


Figure 6: k = 100

# 5. Error Analysis

error calculating formula

$$\|A - A_k\|_F$$

where $\| \cdot \|_F$ is the Frobenius norm.

Below are the observed errors for different values of $k$ for the image einstein:

| **k** | **Reconstruction Error** |
|---|---|
| 2 | 6231 |
| 5 | 4250 |
| 10 | 3239 |
| 100 | 164.79 |

error tables for globe image:

| **k** | **Reconstruction Error** |
|---|---|
| 2 | 31946 |
| 5 | 20499 |
| 10 | 14917 |
| 100 | 3622 |

error table for greyscale image:

| **k** | **Reconstruction Error** |
|---|---|
| 2 | 16656 |
| 5 | 10992 |
| 10 | 6829.89 |
| 100 | 439.52 |

As $k$ increases, the error decreases and the image becomes clearer, but computation time also increases.

# 6. Discussion and Reflections

Implementing SVD manually helped me truly understand how eigenvalues and singular values are connected. When $k$ is small, the image looks blurry but is highly compressed. When $k$ is large, the image quality improves, but compression is reduced. This shows the trade-off between accuracy and storage. The Gram–Schmidt step also made a big difference in getting a stable and clean image. Overall, the project was a great way to connect linear algebra theory with a real-world application.