Image Compression Using Singular Value Decomposition

Matrix Theory (EE1030)

J.Navya sri
Roll No: EE25BTECH11028

November 2025

1 Summary of strang's video

The Singular Value Decomposition (SVD) factorizes any matrix A as

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

where U and V are orthogonal matrices and Σ holds the singular values. It generalizes eigenvalue decomposition and works for any rectangular matrix. Geometrically, SVD represents a rotation, scaling, and another rotation. Large singular values capture the main information or energy of the matrix. By keeping only the top k singular values, we get the best low-rank approximation. SVD is widely used in image compression, noise reduction, and data analysis.

2 Implementation of SVD (svd.c)

The file `svd.c` implements the core mathematical functions used for Singular Value Decomposition (SVD) in image compression.

1. `power_iteration()`

This function finds the dominant eigenvalues and eigenvectors of the matrix A^TA using the **Power Iteration algorithm**. It initializes a random vector b and repeatedly applies the update:

$$b_{new} = \frac{A^TAb}{\|A^TAb\|}$$

until convergence. The corresponding eigenvalue is estimated as:

$$\lambda = b^T(A^TA)b$$

After each dominant eigenpair is found, deflation is applied to remove its influence:

$$W = W - \lambda\, bb^T$$

This process repeats for the top k components.

2. `compute_svd()`

This function performs the overall SVD process:

1. Computes A^T and $A^T A$.
2. Calls `power_iteration()` to get eigenvalues and eigenvectors.
3. Computes singular values as $S_i = \sqrt{|\lambda_i|}$.
4. Calculates left singular vectors $U_i = (AV_i)/S_i$.

The result gives the top k singular components used for compression.

3. `reconstruct()`

This function reconstructs the image using the top k singular components:

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

This produces a compressed version of the image while preserving most of the important visual features.

3 Power Iteration Algorithm

In this project, the **Power Iteration algorithm** is used inside the `compute_svd()` function to find the largest eigenvalues and eigenvectors of the matrix $A^T A$. These are required to calculate the top singular values for the SVD. Instead of computing all eigenvalues, the algorithm iteratively multiplies a random vector by $A^T A$ until it converges to the dominant direction. This method is simple, efficient, and suitable for large image matrices. It allows the program to compute only the top k singular values, which are later used for image compression.

4 Results

The algorithm was tested on three images (Einstein, Globe, Grayscale) for different k values. Below are example reconstructions:

Outputs for Image 1:

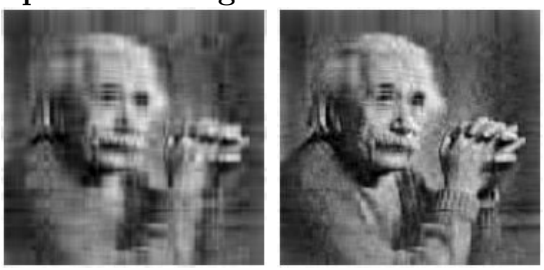

Figure 1: $k10$, $k20$

Outputs for Image 2:

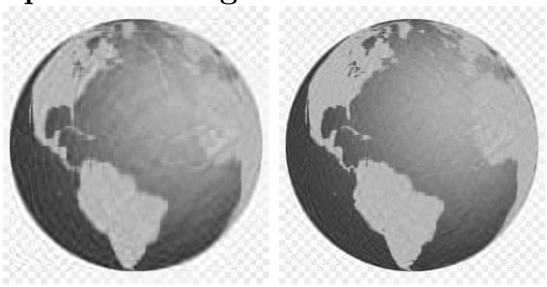

Figure 2: $k25$, $k40$

Outputs for Image 3:

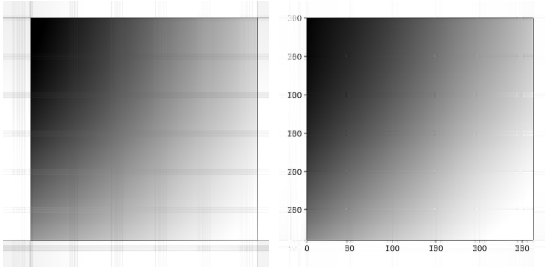

Figure 3: $k2$, $k10$

5 Error Analysis

The approximation error is computed using the Frobenius norm:

$$\|A - A_k\|_F = \sqrt{\sum_{i,j} (A_{ij} - (A_k)_{ij})^2}$$

This metric quantifies the loss of information as k decreases. The results show that the error decreases rapidly for small k, confirming that the leading singular values carry most of the image energy.

6 Pros and Cons of Power Iteration Algorithm

The **Power Iteration algorithm** is simple, efficient, and requires very little memory, making it suitable for finding the largest eigenvalue and eigenvector of large matrices like $A^T A$. It works well for image compression where only the top singular values are needed. However, it finds only one eigenvalue at a time, converges slowly when eigenvalues are close, and depends on the initial random vector. Despite these limitations, it is ideal for truncated SVD applications due to its simplicity and low computational cost.