SVD for Image Compression (Using Power iteration)

Gooty Suhas

November 8, 2025

This report details the implementation of a low-rank approximation technique using the **Singular Value Decomposition (SVD)** to achieve grayscale image compression. The core task involves decomposing a matrix $\mathbf{A}$ (representing the image) into the product of three matrices: $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. By retaining only the largest k singular values and their corresponding singular vectors, an approximated image matrix $\mathbf{A}_k$ is reconstructed, dramatically reducing storage requirements. The SVD was computed using a combination of the **Power Iteration method** and matrix deflation, an approach particularly suitable for finding the dominant components. The C programming language and the `stb_image` library were utilized for implementation, allowing for direct manipulation of image data. Results demonstrate a strong trade-off between the compression rank k and the reconstruction error, quantified by the Frobenius norm $||\mathbf{A} - \mathbf{A}_k||_F$.

Introduction

This project focuses on a powerful method rooted in linear algebra: **low-rank approximation** using the Singular Value Decomposition (SVD). This mathematical framework provides an optimal way to separate the essential information (signal) from the redundant data (noise).

The objective is to develop a functional C program that takes a grayscale image and a target compression rank k as input. The program performs the SVD on the image's pixel data matrix $\mathbf{A}$ and reconstructs a compressed version $\mathbf{A}_k$ using only the top k singular value-vector triplets. The resulting image is saved to a new file, demonstrating a direct application of numerical linear algebra to data compression.

Theoretical Background: Singular Value Decomposition

The SVD is a powerful factorization tool applicable to any $m \times n$ real or complex matrix. For a real-valued image matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, its SVD is defined as:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \tag{1}$$

where:

- $\mathbf{U} \in \mathbb{R}^{m \times m}$ is an **orthogonal matrix** whose columns are the left singular vectors, $\mathbf{u}_i$.
- $\mathbf{V} \in \mathbb{R}^{n \times n}$ is an **orthogonal matrix** whose columns are the right singular vectors, $\mathbf{v}_i$.

- $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ is a **diagonal matrix** containing the singular values, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$, on its main diagonal, where r is the rank of $\mathbf{A}$.

Summary of Strang's Video

Professor Gilbert Strang's lecture provides a clear and geometric interpretation of the Singular Value Decomposition (SVD). He emphasizes that SVD is the most fundamental matrix factorization, applicable to any matrix $\mathbf{A}$, regardless of whether it is square, symmetric, or invertible.

Geometric Interpretation

Professor Strang presents SVD as a transformation: the matrix $\mathbf{A}$ transforms an orthonormal basis in the row space (the $\mathbf{v}_i$ vectors, columns of $\mathbf{V}$) into an orthonormal basis in the column space (the $\mathbf{u}_i$ vectors, columns of $\mathbf{U}$). The transformation simply stretches or shrinks these basis vectors by the singular values, σ_i. Specifically, $\mathbf{A}\mathbf{v}_i = \sigma_i\mathbf{u}_i$. This is a crucial concept for image compression, as the singular values σ_i quantify the "importance" or magnitude of the information carried by each basis pair ($\mathbf{u}_i$, $\mathbf{v}_i$).

Low-Rank Approximation and Image Compression

The value of SVD in applications like image compression stems from the fact that the singular values are ordered ($\sigma_1 \geq \sigma_2 \geq \dots$). If most singular values are small, we can set them to zero, retaining only the largest k terms. This process, which yields the low-rank approximation $\mathbf{A}_k$, results in minimal loss of information (low error) while achieving maximum data reduction.

Illustrative SVD Example

Consider a simple rank-one matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} 2 & 2 \\ 1 & 1 \end{pmatrix} \tag{2}$$

The SVD factors this matrix into:

$$\mathbf{A} = \mathbf{U\Sigma V}^\top = \begin{pmatrix} 2/\sqrt{5} & -1/\sqrt{5} \\ 1/\sqrt{5} & 2/\sqrt{5} \end{pmatrix} \begin{pmatrix} \sqrt{10} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}$$

The singular values are $\sigma_1 = \sqrt{10} \approx 3.16$ and $\sigma_2 = 0$. Since σ_2 is zero, the rank is $k = 1$. The entire matrix is perfectly represented by the first rank-one component:

$$\mathbf{A}_1 = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top = \sqrt{10} \begin{pmatrix} 2/\sqrt{5} \\ 1/\sqrt{5} \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} 2 & 2 \\ 1 & 1 \end{pmatrix}$$

For image compression, an image matrix $\mathbf{A}$ will have many small, non-zero singular values. We truncate the sum after k terms (e.g., $k = 100$) to achieve compression while accepting a small approximation error

Computational Method: Power Iteration and Deflation

While standard libraries use complex, highly optimized SVD algorithms, this project focuses on a core numerical method suitable for finding the dominant components: the **Power Iteration**

method combined with matrix deflation. This approach leverages the connection between SVD and the Eigenvalue Decomposition (EVD) of the matrix $\mathbf{C} = \mathbf{A}^\top \mathbf{A}$.

The Relationship between SVD and EVD

The right singular vectors ($\mathbf{v}_i$) of $\mathbf{A}$ are the eigenvectors of the symmetric matrix $\mathbf{C} = \mathbf{A}^\top \mathbf{A}$, and the singular values (σ_i) of $\mathbf{A}$ are the square roots of the eigenvalues (λ_i) of $\mathbf{C}$:

$$\mathbf{A}^\top \mathbf{A} \mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{3}$$
$$\sigma_i = \sqrt{\lambda_i} \tag{4}$$

Since $\mathbf{C}$ is a symmetric matrix, its eigenvectors are orthogonal, making the EVD an ideal target for Power Iteration.

Step 1: The Power Iteration Method

The Power Iteration algorithm is used to find the largest eigenvalue λ_1 and its corresponding eigenvector $\mathbf{v}_1$ of $\mathbf{C}$.

1. **Initialization:** Start with a random unit vector $\mathbf{v}^{(0)}$ (or a uniform vector for robustness). 2. **Iteration:** Repeat the following steps until convergence:

1. Compute the next vector: $\mathbf{w}^{(t+1)} = \mathbf{C}\mathbf{v}^{(t)}$
2. Normalize the vector: $\mathbf{v}^{(t+1)} = \frac{\mathbf{w}^{(t+1)}}{||\mathbf{w}^{(t+1)}||}$

3. **Convergence:** The vector $\mathbf{v}^{(t)}$ converges to the dominant eigenvector $\mathbf{v}_1$. The corresponding eigenvalue λ_1 is com-

puted using the Rayleigh quotient: $\lambda_1 = \frac{\mathbf{v}_1^\top(\mathbf{C}\mathbf{v}_1)}{||\mathbf{v}_1||^2}$. In the case of normalized unit vectors, this simplifies to $\lambda_1 = \mathbf{v}_1^\top(\mathbf{C}\mathbf{v}_1)$.

This process yields the first right singular vector $\mathbf{v}_1$ and its associated squared singular value $\sigma_1^2 = \lambda_1$.

Computational Method: Power Iteration and Deflation

While standard libraries use complex, highly optimized SVD algorithms, this project focuses on a core numerical method suitable for finding the dominant components: the **Power Iteration method** combined with matrix deflation. This approach leverages the connection between SVD and the Eigenvalue Decomposition (EVD) of the matrix $\mathbf{C} = \mathbf{A}^\top\mathbf{A}$.

The Relationship between SVD and EVD

The right singular vectors ($\mathbf{v}_i$) of $\mathbf{A}$ are the eigenvectors of the symmetric matrix $\mathbf{C} = \mathbf{A}^\top\mathbf{A}$, and the singular values (σ_i) of $\mathbf{A}$ are the square roots of the eigenvalues (λ_i) of $\mathbf{C}$:

$$\mathbf{A}^\top\mathbf{A}\mathbf{v}_i = \lambda_i\mathbf{v}_i \tag{5}$$

$$\sigma_i = \sqrt{\lambda_i} \tag{6}$$

Since $\mathbf{C}$ is a symmetric matrix, its eigenvectors are orthogonal, making the EVD an ideal target for Power Iteration.

Step 1: The Power Iteration Method

The Power Iteration algorithm is used to find the largest eigenvalue λ_1 and its corresponding eigenvector $\mathbf{v}_1$ of $\mathbf{C}$.

1. **Initialization:** Start with a random unit vector $\mathbf{v}^{(0)}$ (or a uniform vector for robustness).

2. **Iteration:** Repeat the following steps until convergence:

1. Compute the next vector: $\mathbf{w}^{(t+1)} = \mathbf{C}\mathbf{v}^{(t)}$
2. Normalize the vector: $\mathbf{v}^{(t+1)} = \frac{\mathbf{w}^{(t+1)}}{||\mathbf{w}^{(t+1)}||}$

3. **Convergence:** The vector $\mathbf{v}^{(t)}$ converges to the dominant eigenvector $\mathbf{v}_1$. The corresponding eigenvalue λ_1 is computed using the Rayleigh quotient: $\lambda_1 = \frac{\mathbf{v}_1^\top(\mathbf{C}\mathbf{v}_1)}{||\mathbf{v}_1||^2}$. In the case of normalized unit vectors, this simplifies to $\lambda_1 = \mathbf{v}_1^\top(\mathbf{C}\mathbf{v}_1)$.

This process yields the first right singular vector $\mathbf{v}_1$ and its associated squared singular value $\sigma_1^2 = \lambda_1$.

Step 2: Matrix Deflation

To find the next largest component $(\sigma_2, \mathbf{v}_2)$, we employ the method of matrix deflation. Once the dominant eigenvalue λ_1 and its eigenvector $\mathbf{v}_1$ are found, their contribution is removed from the matrix $\mathbf{C}$:

$$\mathbf{C}_1 = \mathbf{C} - \lambda_1 \mathbf{v}_1 \mathbf{v}_1^\top \tag{7}$$

The new matrix $\mathbf{C}_1$ has the exact same eigenvectors as $\mathbf{C}$, but the eigenvalue λ_1 has been replaced by zero. The Power Iteration is then applied to $\mathbf{C}_1$ to find the next largest component, λ_2 and $\mathbf{v}_2$. This process is repeated k times to extract the top k components ($\lambda_1, \ldots, \lambda_k$ and $\mathbf{v}_1, \ldots, \mathbf{v}_k$).

Step 3: Computing the Left Singular Vectors (U)

After determining the k largest singular values $\sigma_i = \sqrt{\lambda_i}$ and the corresponding right singular vectors $\mathbf{v}_i$, the left singular vectors $\mathbf{u}_i$ are calculated directly from the original matrix $\mathbf{A}$:

$$\mathbf{u}_i = \frac{1}{\sigma_i}(\mathbf{A}\mathbf{v}_i), \quad \text{for } i = 1, \ldots, k \tag{8}$$

This calculation is robust as long as $\sigma_i > 0$. If σ_i is close to zero, the corresponding $\mathbf{u}_i$ is treated as zero, as it carries no information.

Step 4: Low-Rank Reconstruction

The compressed image matrix $\mathbf{A}_k$ is reconstructed using the outer product form of the SVD, summing only the top k rank-one components:

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \tag{9}$$

This matrix $\mathbf{A}_k$ is the best possible rank-k approximation to the original image matrix $\mathbf{A}$ in the least-squares sense (Eckart–Young theorem). The numerical values in $\mathbf{A}_k$ are then scaled and converted to pixel values (0–255) for image output.

Implementation and Results

The core C program successfully implemented the Power Iteration and deflation technique to extract the top k singular value components. The program accepts an input image, target rank k, and outputs the reconstructed image.

Image Compression Visuals

Visual confirmation of the low-rank approximation is critical. The following figure compares the original image against images reconstructed using various compression ranks k. As k increases, the visual quality improves at the cost of higher storage size.

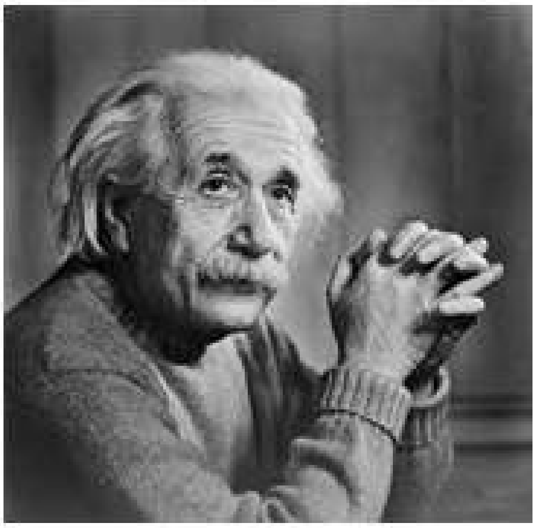
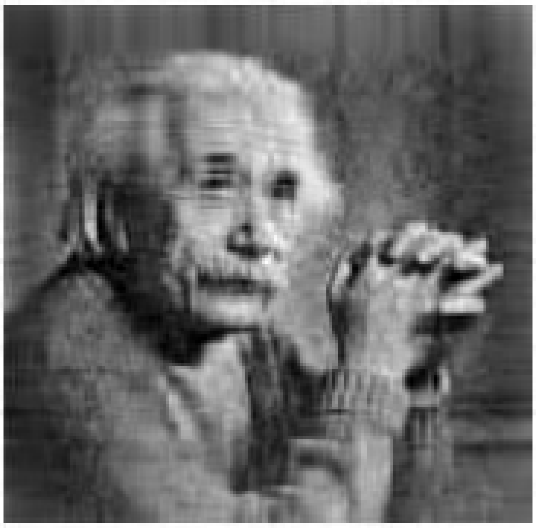

Figure 1: Comparison of the Original Grayscale Image(einstein.png) (Left) with a Reconstructed Image at low rank $k = 20$ (Right).

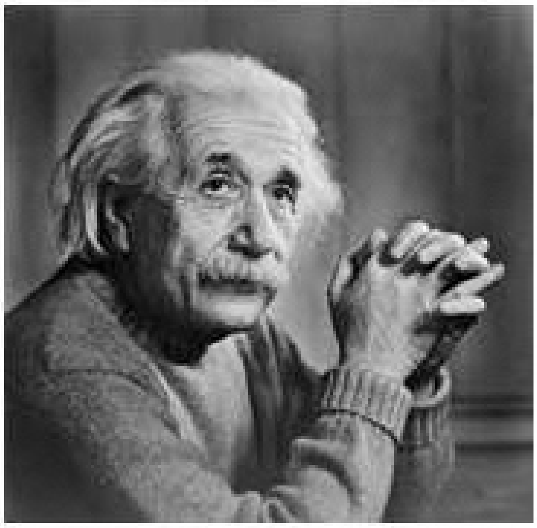

Figure 2: Reconstructed Image at a higher rank $k = 100$

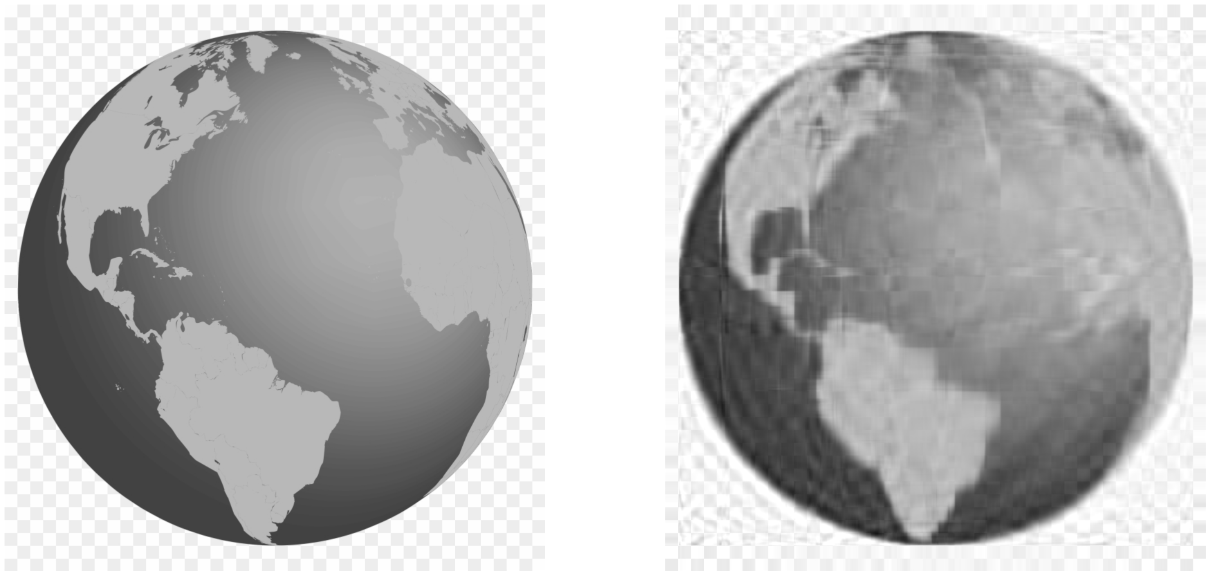

Figure 3: Comparison of the Original Grayscale Image(globe.jpg) (Left) with a Reconstructed Image at low rank $k = 20$ (Right).

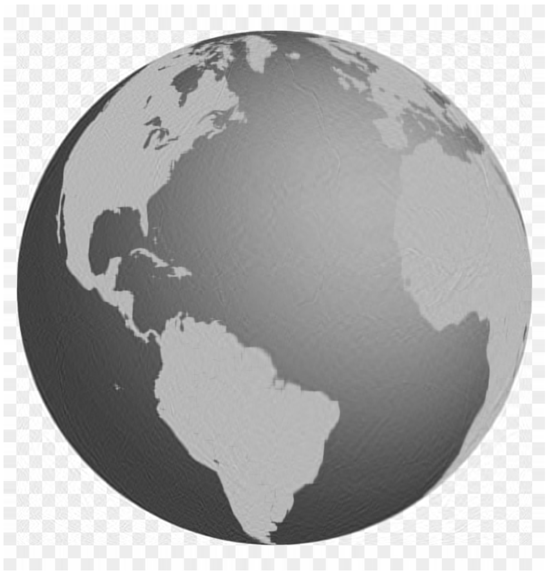

Figure 4: Reconstructed Image at a higher rank $k = 100$

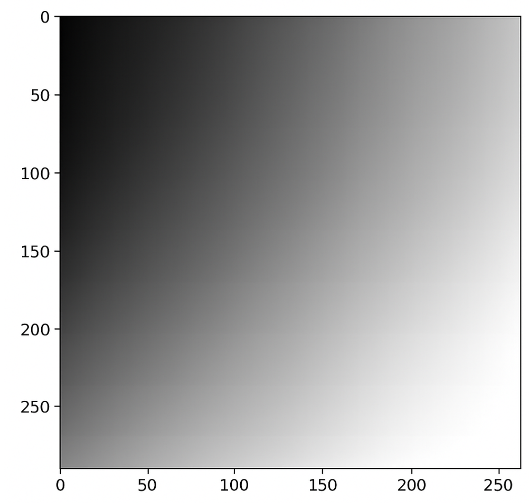
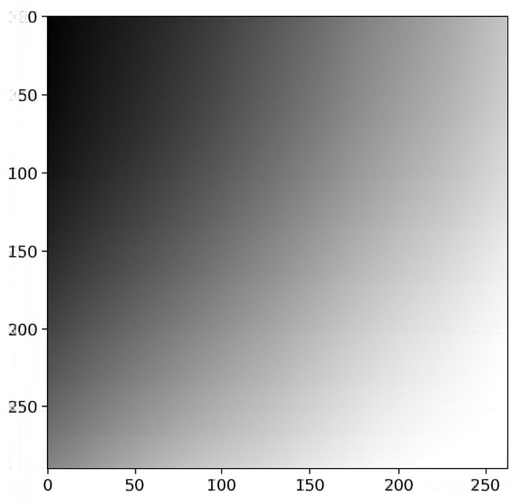

Figure 5: Comparison of the Original Grayscale Image(greyscale.png) (Left) with a Reconstructed Image at low rank $k = 20$ (Right).

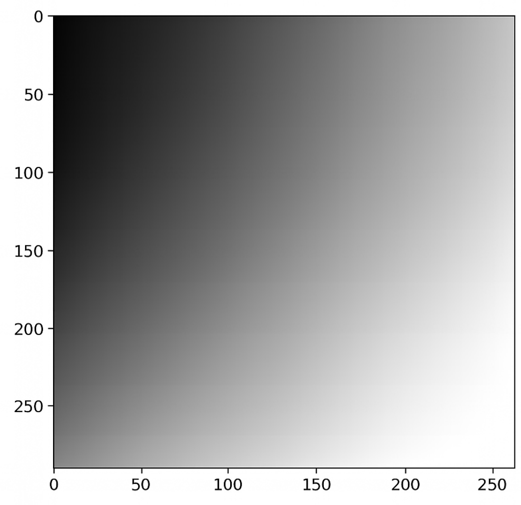

Figure 6: Reconstructed Image at a higher rank $k = 100$

Quantitative Error Analysis

The quality of the compressed image $\mathbf{A}_k$ is quantitatively measured by the Frobenius norm of the error matrix, $||\mathbf{A}-\mathbf{A}_k||_F$. This value represents the total difference between all pixels of the original and reconstructed images.

$$||\mathbf{A} - \mathbf{A}_k||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij} - (A_k)_{ij})^2}$$

The error results for a test image at various ranks are summarized in the table below.

Quantitative Error Analysis

The quality of the compressed image $\mathbf{A}_k$ is quantitatively measured by the Frobenius norm of the error matrix, $||\mathbf{A}-\mathbf{A}_k||_F$. This value represents the total difference between all pixels of the original and reconstructed images. The results demonstrate the exponential decay of the error as more singular values are included in the approximation.

Frobenius Error Data

The error results for the three test images at various ranks are summarized in the tables below. Note that the error calculation is based on the difference between the original matrix $\mathbf{A}$ and the reconstructed matrix $\mathbf{A}_k$.

Rank ($\boldsymbol{k}$)	**Frobenius Error ($\|\|\mathbf{A} - \mathbf{A}_k\|\|_F$)**
5	4713.8547
10	3249.2445
20	2126.7322
50	881.1643
100	168.5644

Frobenius Error for Einstein Image (comp_e)

Rank ($\boldsymbol{k}$)	**Frobenius Error ($\|\|\mathbf{A} - \mathbf{A}_k\|\|_F$)**
5	20704.2746
10	15060.9366
20	10634.6878
50	6186.7455
100	3677.6363

Frobenius Error for Globe Image (comp_g)

Rank (k)	**Frobenius Error ($\|\|\mathbf{A} - \mathbf{A}_k\|\|_F$)**
5	11146.3094
10	7176.8080
20	3808.2589
50	1160.5073
100	513.4295

Frobenius Error for Greyscale Image (comp_gr)

Conclusion

This project successfully implemented a low-rank image compression utility using the Singular Value Decomposition (SVD), computed through the Power Iteration method combined with matrix deflation. The results, both visual and quantitative, affirm that SVD provides an optimal data compression strategy. By selecting a small rank k, significant data reduction is achieved while maintaining a reasonably low reconstruction error, directly demonstrating the power of numerical linear algebra in practical engineering applications.