Software Project
Image Compression Using Truncated SVD

Shivam Sawarkar
AI25BTECH11031

November 8, 2025

1. Background

A grayscale image can be represented as a matrix $A \in \mathbb{R}^{m \times n}$, where each entry a_{ij} corresponds to the pixel intensity (0 = black, 255 = white).

The **Singular Value Decomposition (SVD)** expresses A as:

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

where:

- U and V are orthogonal matrices,
- Σ is a diagonal matrix with singular values in decreasing order.

A low-rank approximation of A can be obtained by keeping only the top k singular values:

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

This truncated version A_k preserves most of the important image content while using far fewer values, forming the basis of image compression.

2. Summary of Gilbert Strang's Video on SVD

The lecture explains the Singular Value Decomposition (SVD) of a matrix, highlighting its importance in transforming any matrix into a product of orthogonal matrices and a diagonal matrix.

In his lecture prof. Gilbert Strang says:

1. SVD is the best factorization of a matrix, consisting of two orthogonal matrices and a diagonal matrix.
2. The goal of SVD is to find orthonormal bases for the row and column spaces of a matrix, allowing for diagonalization.
3. The eigenvectors of A^TA provide the orthogonal matrix V in SVD, while the eigenvectors of AA^T provide the orthogonal matrix U.
4. The diagonal entries of the matrix Σ in SVD are the square roots of the eigenvalues from A^TA or AA^T.

3. Algorithm Explanation

Mathematical Basis

Steps

1. Read grayscale image in PGM format and store as matrix A (convert in PGM if it is JPG).
2. Compute A^TA and AA^T.
3. Use Power Iteration to find top eigenvalues and eigenvectors.
4. Construct U, Σ, V^T.

5. Truncate to top k singular values.
6. Reconstruct $A_k = U_k \Sigma_k V_k^T$.
7. Write A_k as new PGM file.
8. Give output as .pgm or .jpg

4. Pseudocode

Input: Grayscale image file (.pgm)
Output: Reconstructed compressed image (.pgm)

1. Read PGM image $\rightarrow$ matrix $A[m][n]$ (Convert to PGM first if the input is JPG)
2. Compute $A^T A$
3. For $i = 1, 2, \ldots, k$:
   Use power iteration on $A^T A$ to find top-k eigenvectors V
   Compute $\sigma_i = \sqrt{\lambda_i}$
4. Compute $U_i = \frac{AV_i}{\sigma_i}$
5. Form $A_k = U_k \Sigma_k V_k^T$
6. Write A_k to output PGM file (or JPG file)
7. Compute error $\|A - A_k\|_F$

5. Algorithm Comparison for Computing SVD

The below table compares various algorithms for calculating based on there speed, accuracy, and ease of coding

Algorithm	**Description**	**Speed**	**Accuracy**	**Ease of Coding**
Jacobi Method	Iterative Givens rotations to diagonalize the matrix; accurate but computationally heavy.	Slow	Very High	Moderate
Golub–Reinsch	Reduces the matrix to bidiagonal form using Householder reflections before SVD computation.	Fast	High	Complex
Power Iteration	Repeatedly estimates dominant eigenvectors to find top singular values (used for truncated SVD).	Medium	Moderate	Easy
Lanczos Method	Improves power iteration with orthogonalization, making it efficient for large or sparse matrices.	Very Fast	High	Complex
Randomized SVD	Uses random projections to approximate dominant singular subspace with less computation.	Medium	Moderate	Easy

Table 1: Algorithms for calculating SVD

Result: The Jacobi method provides the highest accuracy but is computationally difficult. The Golub–Reinsch algorithm, which reduces the matrix to a bidiagonal form before performing SVD.

For this project, I have chosen the Power Iteration method because:

- It computes only the top k singular values and vectors, which is sufficient for image compression.
- It is easy to implement from scratch in C without using external libraries.
- It is moderately accurate and easy to code.

6. Discussion

- Increasing k improves quality but takes more time.
- Computation time grows with image size and number of singular values computed.
- The first few singular values capture most of the image data. Beyond a certain k, additional singular values contribute very less to visual improvement but greatly increase computation.
- The Power Iteration method performs well for truncated SVD, though convergence slows for close or repeated singular values.

7. Error Analysis

The approximation accuracy is measured using the Frobenius norm:

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

As k increases, the error decreases because smaller singular values contribute less to the total image energy.

8. Conclusion

The project demonstrates image compression using truncated SVD implemented entirely in C. Results show that:

- Low-rank approximation reduces storage effectively.
- SVD captures the essential image features.
- There exists a clear trade-off between compression ratio and quality.