SOFTWARE ASSIGNMENT

Image Compression using Truncated SVD

EE25BTECH11061 – V.Sainadh

Date: November 9, 2025

1. Summary of Prof. Gilbert Strang's Video on SVD

Prof. Gilbert Strang's lecture introduces the **Singular Value Decomposition (SVD)** as one of the most fundamental and beautiful results in linear algebra. He begins by explaining that any real matrix $\left(A\right)$, whether square or rectangular, can be decomposed as:

$$\left(A\right) = \left(U\right)\left(\Sigma\right)\left(V\right)^T \tag{1}$$

where $\left(U\right)$ and $\left(V\right)$ are orthogonal matrices, and $\left(\Sigma\right)$ is diagonal with non-negative singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$.

Geometrically, $\left(A\right)$ can be interpreted as a transformation that:

- Rotates the input space using $\left(V\right)$,
- Scales each orthogonal direction by σ_i,
- And reorients the result using $\left(U\right)$.

Prof. Strang shows how the columns of $\left(V\right)$ correspond to input directions of maximum variance, while the columns of $\left(U\right)$ are the output directions. The singular values measure how strongly $\left(A\right)$ stretches or compresses along each axis. He further relates SVD to the eigenvalue decomposition: brginalign*

$$\left(A\right)^T\left(A\right)\left(v\right)_i = \sigma_i^2\left(v\right)_i$$

indicating that $\left(V\right)$ contains the eigenvectors of $\left(A\right)^T\left(A\right)$ and that σ_i are the square roots of its eigenvalues.

Strang connects this decomposition to applications such as image compression and principal component analysis (PCA), where only a few dominant singular values are retained to approximate the original matrix efficiently. He concludes that SVD is a bridge between geometry and computation — it reveals both the structure and the essential information contained within any dataset.

2. Explanation of the Implemented Algorithm (Mathematics and Pseudocode)

1 Mathematical Formulation and Pseudocode

The implemented algorithm performs truncated SVD using the **Power Iteration with Deflation** method. For a grayscale image matrix $A \in \mathbb{R}^{m\times n}$, the Singular Value Decomposition (SVD) is defined as

$$A = U\Sigma V^T \tag{2}$$

A rank–k approximation of A is expressed as

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{3}$$

Mathematical Steps

To compute the dominant singular triplet (u, σ, v), the following iterative relations are used:

$$u = \frac{Av}{||Av||} \tag{4}$$

$$v = \frac{A^T u}{||A^T u||} \tag{5}$$

$$\sigma = u^T A v \tag{6}$$

After computing the dominant singular triplet, deflation is applied to remove its contribution:

$$A \leftarrow A - \sigma u v^T \tag{7}$$

Repeating this process for k iterations yields the rank–k truncated SVD.

Pseudocode

```
1. Load grayscale image and store pixel data into matrix $A$.
2. Initialize a random vector $v$.
3. Repeat for a fixed number of iterations:
       $u \leftarrow \frac{Av}{||Av||}$
       $v \leftarrow \frac{A^T u}{||A^T u||}$
4. Compute singular value: $\sigma = u^T A v$
5. Update approximation: $A_k = A_k + \sigma u v^T$
6. Apply deflation: $A \leftarrow A - \sigma u v^T$
7. Repeat for next $k$
```

3. Comparison of Algorithms and Justification for the Chosen Method

2 Comparison of SVD Algorithms

2.1 1. Classical Full SVD

This method computes the exact singular value decomposition of a matrix A as $A = U\Sigma V^T$. It uses transformations such as QR iteration to find all singular values and vectors. It is accurate but slow and mainly used for small matrices. The computational cost is of the order $O(mn^2)$.

2.2 2. Power Iteration with Deflation (Implemented Method)

This iterative method finds the dominant singular value and its corresponding singular vectors. It works by repeatedly multiplying the matrix A and its transpose with a vector until convergence.

Iteration equations: brginalign*

$$\mathrm{v}_{k+1} = \frac{A^T(Av_k)}{||A^T(Av_k)||}, \qquad u_k = \frac{Av_k}{||Av_k||}, \qquad \sigma_k = ||Av_k||.$$

After convergence, brginalign* A $\mathrm{v}_1 = \sigma_1 u_1, \qquad A^T u_1 = \sigma_1 v_1.$

Once the first singular pair (u_1, σ_1, v_1) is obtained, the deflation process removes its effect: brginalign* $\mathrm{A}_1 = A - \sigma_1 u_1 v_1^T$.

Then the same iteration is repeated on A_1 to get the next pairs, forming a rank–k approximation: brginalign* A $\approx \sum_{i=1}^{k} \sigma_i u_i v_i^T$.

Why use this method:

- Simple to implement using only matrix–vector operations.
- Clearly demonstrates the concept of SVD and energy capture.

2.3 3. Jacobi Method

The Jacobi method finds the singular values by applying a sequence of orthogonal transformations that make the columns of the matrix orthogonal to each other. It eliminates the off-diagonal elements step by step using plane rotations until the matrix becomes diagonal. It is very accurate but computationally expensive and slow for large matrices. This method is mainly used when high precision is required.

2.4 4. Lanczos Bidiagonalization

This method generates two sets of orthogonal vectors using short recurrence relations to reduce the matrix to bidiagonal form. It is efficient for large sparse matrices and useful when only a few dominant singular values are needed.

2.5 5. Randomized SVD

This approach uses random projection to approximate the column space of A. It first forms a smaller matrix using random sampling and then computes the SVD on that smaller matrix. It is very fast and works well for large data sets, though the results are approximate.

2.6 6. Krylov or Block Power Method

This method improves the power iteration by using multiple vectors simultaneously to form a subspace. It then orthonormalizes that subspace and computes the SVD within it. It converges faster than normal power iteration but requires more memory and computation.

4. Reconstructed Images for Different k

The image compression process was tested on three grayscale input images: `image1.jpg`, `image2.jpg`, and `image3.jpg`. Each image was reconstructed for different truncation ranks $k = 5, 20, 50, 100$ to study the visual quality variation.

Original Input Images

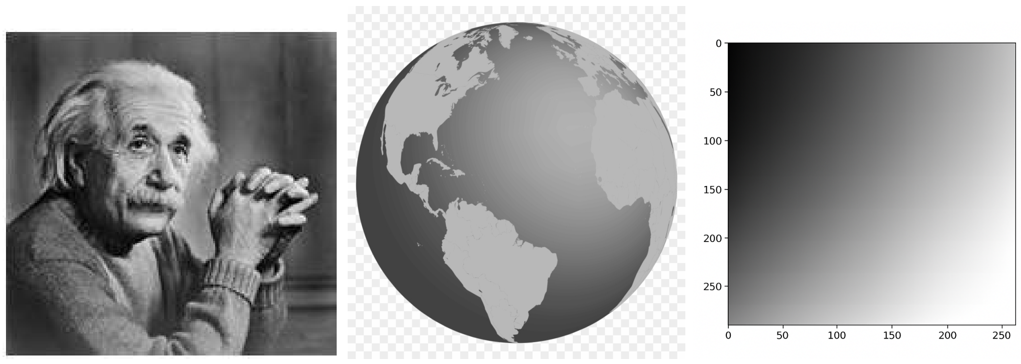

Figure 1: Original grayscale input images used for SVD compression.

Reconstructed Outputs for Image 1

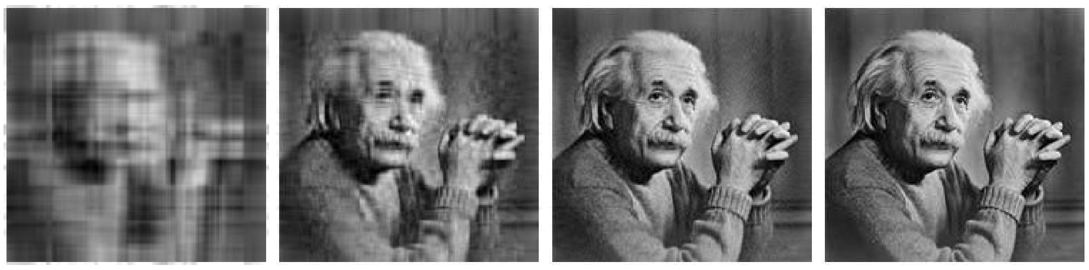

Figure 2: Reconstructed versions of `image1.jpg` for different k.

Reconstructed Outputs for Image 2

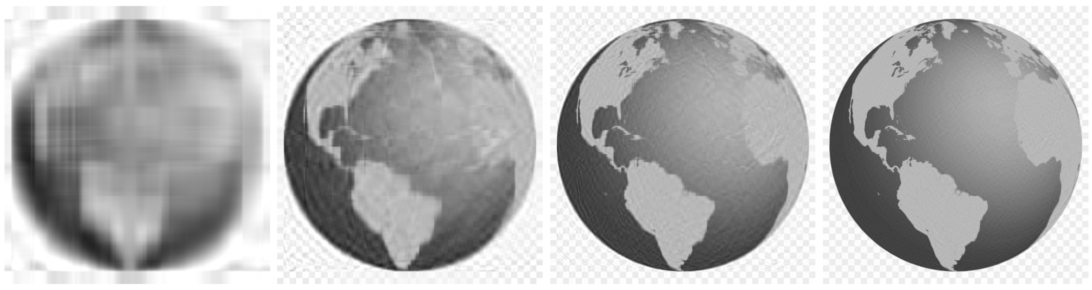

Figure 3: Reconstructed versions of `image2.jpg` for different k.

Reconstructed Outputs for Image 3

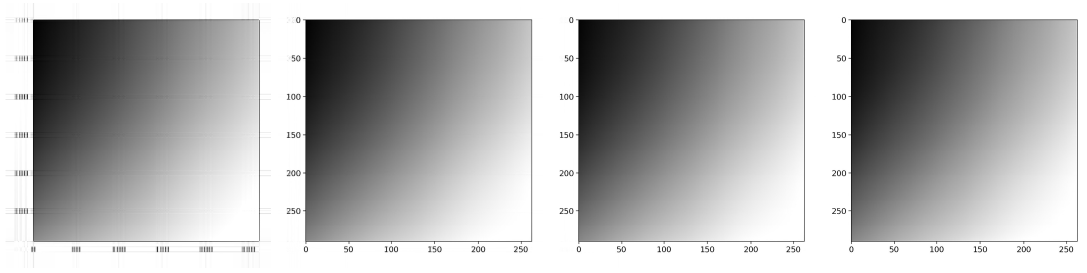

Figure 4: Reconstructed versions of `image3.jpg` for different k.

5. Error Analysis

The reconstruction error is measured using the **Frobenius norm:**

$$E_k = \|\left(A\right) - \left(A\right)k\|_F = \sqrt{\sum i = 1^m \sum_{j=1}^{n} (a_{ij} - a_{ij}^{(k)})^2} \tag{8}$$

The compression ratio is:

$$R_k = \frac{mn}{k(m+n+1)} \tag{9}$$

Image	**k = 5**	**k = 20**	**k = 50**	**k = 100**
Einstein	21.629%	9.75%	4.04%	0.75%
Globe	13.07%	6.74%	3.99%	2.32%
Grayscale	5.25%	1.97%	0.62%	0.63%

Table 1: Reconstruction Error Percentages for Different Images at Various k Values

6. Discussion of Trade-offs and Reflections on Implementation Choice

As k increases, $\left(A\right)_k$ better approximates $\left(A\right)$, reducing error but increasing storage cost. The trade-off between compression efficiency and reconstruction quality is evident:

- For small k, the image becomes blurred but is highly compressed.
- For large k, visual fidelity improves, but compression ratio decreases.

The Power Iteration with Deflation method balances computational simplicity with conceptual depth. It allows visualizing how successive singular components add detail to the image, linking linear algebra theory directly to perceptible image quality.

7. Conclusion

The project successfully demonstrates how truncated SVD can be implemented from scratch in C to perform image compression. It reinforces the theoretical understanding of SVD, highlights the importance of numerical efficiency, and bridges mathematics, coding, and visual interpretation.