Image Compression via SVD

Anirudh M Abhilash

November 8, 2025

Overview of SVD

The Singular Value Decomposition (SVD) expresses any real matrix A as

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

where U and V are orthogonal matrices and Σ is a diagonal matrix containing non-negative real numbers called singular values. Conceptually, this factorization represents a sequence of transformations: a rotation by V^T, followed by a stretching or compression along principal axes determined by Σ, and finally a rotation by U. [1]

Key Insights from the Concept

SVD provides a geometric and analytical view of how data or image matrices behave. The singular values, arranged in decreasing order, indicate the relative importance of each dimension or mode in the data. The first few singular values typically capture most of the matrix's "energy," which makes SVD particularly suitable for dimensionality reduction and image compression.

The columns of V correspond to the eigenvectors of A^TA, and those of U correspond to the eigenvectors of AA^T. The squares of the singular values are the eigenvalues of these symmetric matrices. This connection between SVD and eigen-decomposition gives a strong theoretical basis for analyzing images as structured data.

Relevance to Image Compression

When an image is represented as a matrix, applying SVD allows the separation of its structural components in order of significance. Retaining only the top k singular values and associated singular vectors produces the best possible rank-k approximation of the image in the Frobenius norm sense, as established by the Eckart–Young theorem. This truncation removes redundant or low-importance details while maintaining visual quality.

In our project, the Lanczos bidiagonalization algorithm followed by implicit QR iterations with Givens rotations accelerated by the Wilkinson shift was used to efficiently approximate the SVD for large matrices. This approach ensures numerical stability and computational efficiency, making it feasible for practical image compression tasks, including both grayscale and RGB images.

Lanczos Bidiagonalization

Intuition and Why It Works

Lanczos bidiagonalization compresses the action of a large matrix $A \in \mathbb{R}^{m\times n}$ into a smaller bidiagonal matrix B_k while preserving its dominant directions. Two orthonormal bases are built, V_k in

the input space and U_k in the output space, so that A can be represented in these bases as

$$AV_k = U_k B_k, \quad A^T U_k = V_k B_k^T.$$

Each iteration applies A and A^T to previous vectors, forming a *Krylov subspace*:

$$\mathcal{K}_k(A^T A, v_1) = \text{span}\{v_1, (A^T A)v_1, (A^T A)^2 v_1, \ldots, (A^T A)^{k-1} v_1\}.$$

This space becomes dominated by eigenvectors of $A^T A$ associated with the largest eigenvalues. When A is expressed in this subspace, the smaller bidiagonal matrix B_k captures the essential spectral properties of A. The singular values of B_k are called *Ritz approximations*—they estimate the largest singular values of A within the span of V_k. As k increases, these estimates converge to the true singular values.

Iterative Algorithm

Start with a random unit vector v_1 ($\|v_1\| = 1$), and set $u_0 = 0$, $\beta_1 = 0$. For $j = 1, 2, \ldots, k$:

$$\begin{aligned} u_j' &= Av_j - \beta_j u_{j-1}, \\ \alpha_j &= \|u_j'\|, \quad u_j = \frac{u_j'}{\alpha_j}, \\ v_{j+1}' &= A^T u_j - \alpha_j v_j, \\ \beta_{j+1} &= \|v_{j+1}'\|, \quad v_{j+1} = \frac{v_{j+1}'}{\beta_{j+1}}. \end{aligned}$$

After k iterations, form

$$U_k = [u_1, u_2, \ldots, u_k], \quad V_k = [v_1, v_2, \ldots, v_k],$$

and the bidiagonal matrix

$$B_k = \begin{pmatrix} \alpha_1 & \beta_2 & & \\ & \alpha_2 & \beta_3 & \\ & & \ddots & \ddots \\ & & & \alpha_k \end{pmatrix}.$$

By construction, the relation $AV_k = U_k B_k$ holds.

Approximate Singular Triplets

The small matrix B_k is decomposed as

$$B_k = \widehat{U}\, \Sigma_k\, \widehat{V}^T,$$

where $\Sigma_k = \text{diag}(\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_k)$. Approximate singular vectors of A are recovered as

$$\tilde{U}_k = U_k \widehat{U}, \quad \tilde{V}_k = V_k \widehat{V},$$

giving the rank-k approximation

$$\tilde{A}_k = \tilde{U}_k \Sigma_k \tilde{V}_k^T = U_k B_k V_k^T.$$

The diagonal elements $\tilde{\sigma}_j$ are the Ritz approximations of the true singular values σ_j.

Practical Remarks

In exact arithmetic, $AV_k = U_k B_k$ holds precisely, and the Ritz values converge rapidly to the largest singular values. In finite precision, small loss of orthogonality in U_k and V_k may cause repeated or spurious singular values. Reorthogonalization is used to correct this. The bidiagonal structure makes further computation efficient using implicit QR via Givens-based solvers, which is why this step is ideal before the QR phase in large-scale SVD compression.

Implicit QR on the Bidiagonal Matrix B_k

After Lanczos bidiagonalization, the matrix

$$B_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & 0 \\ & \alpha_2 & \beta_3 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{k-1} & \beta_k \\ 0 & & & & \alpha_k \end{pmatrix}$$

is upper bidiagonal. Our goal is to compute its singular values and the corresponding singular vectors.

Why the QR iteration is performed *implicitly on* B_k

The singular values of B_k are the square roots of the eigenvalues of

$$T = B_k^T B_k,$$

which is symmetric and tridiagonal. Forming T explicitly squares the condition number of B_k and doubles round-off errors. It also destroys the convenient bidiagonal structure.

Instead, we perform the QR process *implicitly on* B_k itself. This produces the same effect as a QR iteration on T without ever forming $T = B_k^T B_k$. The correctness of this approach follows from the **Implicit Q Theorem**.

The Implicit Q Theorem

Let $T = B_k^T B_k$ be tridiagonal and let μ be a shift. Suppose that $T - \mu I = QR$ is its QR factorization with orthogonal Q. Then the explicit shifted QR step gives

$$T^{(+)} = RQ + \mu I = Q^T T Q.$$

Both T and $T^{(+)}$ are symmetric and tridiagonal, sharing the same eigenvalues.

Implicit Q Theorem : *If a symmetric tridiagonal matrix T is transformed by an orthogonal sequence that preserves tridiagonality and introduces the same first subdiagonal as an explicit QR step with shift μ, then the resulting matrix is identical (up to signs) to the explicit $T^{(+)} = Q^T T Q$.*

Applying this to $T = B_k^T B_k$ means that any sequence of orthogonal transformations applied directly to B_k that induces the same similarity transformation on T will produce the same singular values. Hence, operating implicitly on B_k is fully equivalent to a QR step on T.

Bulge creation and chasing on B_k

To perform the shifted QR step implicitly, we begin with a shift μ (typically the Wilkinson shift) and form the leading vector

$$\begin{pmatrix} \alpha_1^2 - \mu \\ \alpha_1 \beta_2 \end{pmatrix}.$$

A right Givens rotation G_1 is chosen to annihilate the lower element. Applying this rotation to B_k from the right:

$$B_k \leftarrow B_k G_1,$$

destroys the first superdiagonal entry but introduces a nonzero element just below the diagonal—the so-called *bulge*. To restore bidiagonal form, a left Givens rotation P_1 is applied:

$$B_k \leftarrow P_1^T B_k.$$

This pair of rotations removes the bulge at the current position but creates a new one one step further down and to the right.

Repeating this process with successive pairs (P_i, G_i), the bulge is *chased* toward the bottom right corner:

$$B_k \leftarrow P_i^T B_k G_i, \quad i = 1, 2, \ldots, k-1.$$

When the bulge exits the matrix, B_k is bidiagonal again but with slightly modified diagonal and superdiagonal entries.

Connection to the Implicit Q Theorem

Each local transformation $B_k \leftarrow P_i^T B_k G_i$ induces an update on $T = B_k^T B_k$ of the form

$$T \leftarrow G_i^T T G_i,$$

exactly the same as one step in the explicit QR iteration. Therefore, the complete sequence of bulge-chasing rotations $(P_1, G_1), (P_2, G_2), \ldots, (P_{k-1}, G_{k-1})$ produces

$$T^{(+)} = Q^T T Q, \quad \text{where} \quad Q = G_1 G_2 \cdots G_{k-1}.$$

By the implicit Q theorem, this implicit process on B_k is mathematically identical to the explicit shifted QR iteration on T, but it is far more stable and efficient.

Formation of singular vectors

If the left and right rotations are accumulated,

$$P = P_1 P_2 \cdots P_{k-1}, \qquad Q = G_1 G_2 \cdots G_{k-1},$$

then after convergence we have

$$B_k \approx P^T(\Sigma) Q,$$

where Σ is diagonal with the singular values of B_k. These orthogonal accumulations directly give the left and right singular vector matrices:

$$\tilde{U} = P, \qquad \tilde{V} = Q.$$

Finally, lifting them through the Lanczos bases yields the approximate singular vectors of the original matrix A:

$$U_A = U_k \tilde{U}, \qquad V_A = V_k \tilde{V}.$$

Thus, the implicit QR algorithm on B_k combines the mathematical rigor of the QR iteration with the numerical efficiency of operating directly on the bidiagonal form.

Integrating the 2 steps to get the SVD of A

Bringing all steps together, the full algorithm computes an approximate SVD of the original matrix $A \in \mathbb{R}^{m \times n}$ through three structured stages:

1. **Lanczos Bidiagonalization:** The matrix A is reduced to an upper-bidiagonal form

   $$AV_k = U_k B_k,$$

   using k iterations of the Lanczos (Golub–Kahan) process. This step captures the dominant subspace of A in the orthonormal bases U_k and V_k.

2. **Implicit QR on B_k:** The small bidiagonal matrix B_k is diagonalized implicitly using the shifted QR method with Givens rotations and the Wilkinson shift. Each iteration applies local left and right rotations (P_i, G_i) that maintain bidiagonal form:

   $$B_k \;\leftarrow\; P_i^T B_k G_i.$$

   By the Implicit Q Theorem, this procedure is equivalent to performing a QR iteration on $B_k^T B_k$ but avoids forming it explicitly, ensuring numerical stability.

3. **Assembly of the SVD:** After convergence, B_k becomes diagonal:

   $$B_k \approx \tilde{U}^T \Sigma_k \tilde{V},$$

   where $\Sigma_k = \mathrm{diag}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_k)$ contains the leading singular values of A. The corresponding approximate singular vectors of A are obtained by

   $$U_A = U_k \tilde{U}, \qquad V_A = V_k \tilde{V}.$$

Because B_k is of size $k \times k$ with $k \ll n$, the Lanczos reduction dominates the computational cost at $O(kn^2)$, while the subsequent implicit QR iterations on B_k require only $O(k^3)$ operations. The overall complexity for obtaining the truncated SVD of A is therefore

$$\boxed{O(kn^2)}$$

with excellent numerical stability and high compression accuracy.

Algorithm 1 Lanczos bidiagonalization (Golub–Kahan)

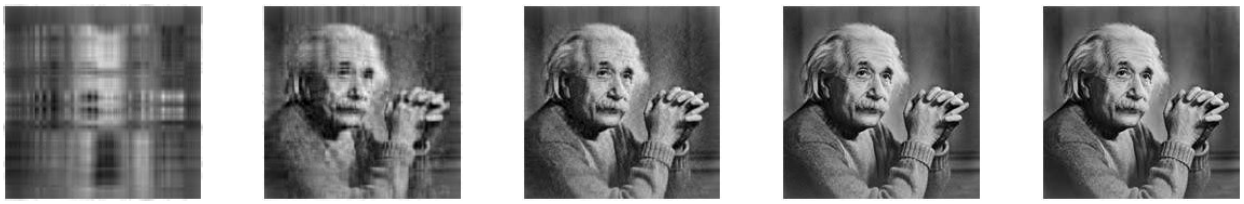
Require: $A \in \mathbb{R}^{m \times n}$, target rank k, initial unit vector $v_1 \in \mathbb{R}^n$
Ensure: Orthonormal bases $U_k \in \mathbb{R}^{m \times k}$, $V_k \in \mathbb{R}^{n \times k}$ and bidiagonal $B_k \in \mathbb{R}^{k \times k}$ such that $AV_k = U_k B_k$

$u_0 \leftarrow 0,\ \beta_1 \leftarrow 0$
for $j \leftarrow 1$ to k **do**
  $u'_j \leftarrow A v_j - \beta_j u_{j-1}$
  $\alpha_j \leftarrow \|u'_j\|$ ▷ break if $\alpha_j \approx 0$
  $u_j \leftarrow u'_j / \alpha_j$
  $v'_{j+1} \leftarrow A^T u_j - \alpha_j v_j$
  $\beta_{j+1} \leftarrow \|v'_{j+1}\|$ ▷ break if $\beta_{j+1} \approx 0$
  $v_{j+1} \leftarrow v'_{j+1} / \beta_{j+1}$
  store α_j on $\mathrm{diag}(B_k)$ and β_{j+1} on $\mathrm{superdiag}(B_k)$
  full reorthogonalization on u_j and/or v_{j+1}
end for
$U_k \leftarrow [u_1, \ldots, u_k],\quad V_k \leftarrow [v_1, \ldots, v_k],$ form B_k from $\{\alpha_j, \beta_{j+1}\}$

Algorithm 2 Implicit shifted QR on bidiagonal B_k (bulge chase, Wilkinson shift)

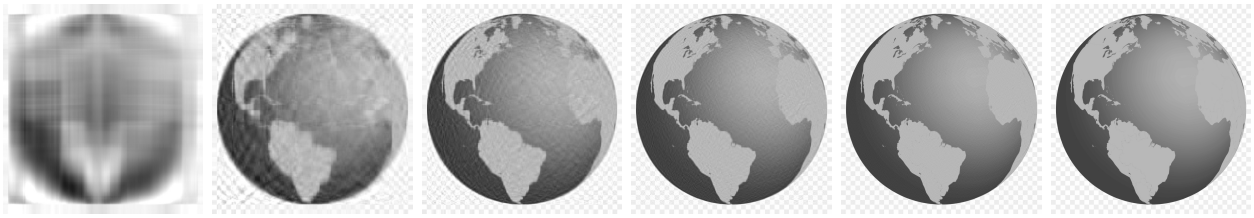
Require: Bidiagonal $B_k \in \mathbb{R}^{k \times k}$ with diagonals $\{\alpha_i\}$, superdiagonals $\{\beta_{i+1}\}$, tolerance `tol`
Ensure: Diagonal $\Sigma_k \approx \mathrm{diag}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_k)$ and accumulators $\tilde{U}, \tilde{V}$ (optional) so $B_k \approx \tilde{U}^T \Sigma_k \tilde{V}$

Initialize accumulators $\tilde{U} \leftarrow I_k,\ \tilde{V} \leftarrow I_k$
while active block size > 1 **do**
  form trailing 2×2 block of $T = B_k^T B_k$ and compute Wilkinson shift μ
  Create initial right rotation G_1 from vector $\left(\alpha_1^2 - \mu,\ \alpha_1 \beta_2\right)^T$
  $B_k \leftarrow B_k G_1$ ▷ introduces a bulge
  $P_1 \leftarrow$ left Givens to restore bidiagonal form; $B_k \leftarrow P_1^T B_k$
  if accumulate vectors **then**
    $\tilde{V} \leftarrow \tilde{V}\, G_1,\quad \tilde{U} \leftarrow \tilde{U}\, P_1$
  end if
  for $i \leftarrow 2$ to $k - 1$ **do** ▷ bulge chase step
    compute right rotation G_i to annihilate current bulge (columns $i, i+1$)
    $B_k \leftarrow B_k G_i$
    compute left rotation P_i to restore bidiagonal shape (rows $i, i+1$)
    $B_k \leftarrow P_i^T B_k$
    if accumulate vectors **then**
      $\tilde{V} \leftarrow \tilde{V}\, G_i,\quad \tilde{U} \leftarrow \tilde{U}\, P_i$
    end if
  end for
  Check deflation: if $|\beta_j| \le \mathtt{tol}(|\alpha_j| + |\alpha_{j+1}|)$ then set $\beta_j \leftarrow 0$ and split blocks
  If an eigenvalue converged within tolerance extract $\tilde{\sigma} \leftarrow$ corresponding diagonal entry
end while
return Σ_k from remaining diagonals; optionally $\tilde{U}, \tilde{V}$



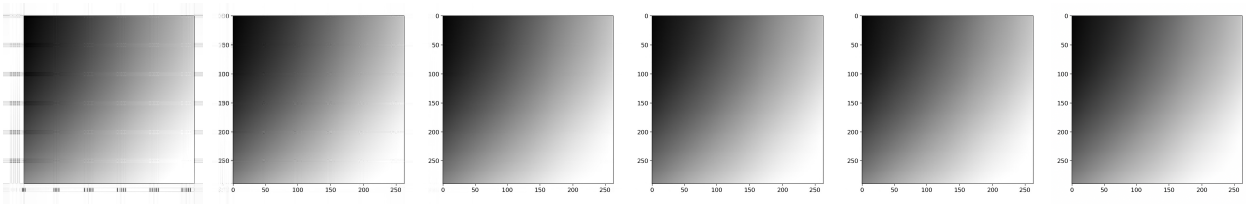
$k = 5$ $k = 20$ $k = 50$ $k = 100$ Original

Figure 1: Reconstruction of Einstein image at various truncation ranks k.



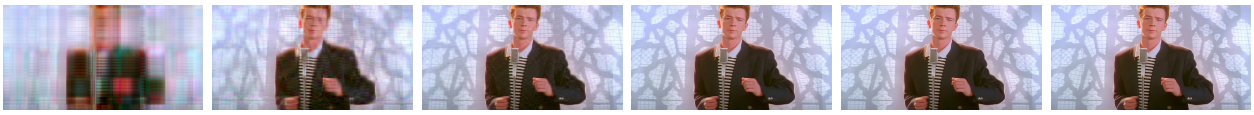
$k = 5$ $k = 20$ $k = 50$ $k = 100$ $k = 200$ Original

Figure 2: Reconstruction of Globe image at various truncation ranks k.



$k = 5$ $k = 20$ $k = 50$ $k = 100$ $k = 200$ Original

Figure 3: Reconstruction of Grayscale image at various truncation ranks k.



$k = 5$ $k = 20$ $k = 50$ $k = 100$ $k = 200$ Original

Figure 4: Reconstruction of Rick image at various truncation ranks k.

$k = 5$ $k = 20$ $k = 50$ $k = 100$ $k = 200$ Original

Figure 5: Reconstruction of Jon Snow image at various truncation ranks k.

Error Analysis

Frobenius Norm Errors

For a given rank-k approximation $\tilde{A}_k = U_k \Sigma_k V_k^T$ of A, the reconstruction error is measured using the Frobenius norm:

$$E_F(k) = \|A - \tilde{A}_k\|_F = \sqrt{\sum_{i,j} (A_{ij} - \tilde{A}_{ij})^2}.$$

This represents the total energy of the residual matrix not captured by the first k singular components.

The **absolute Frobenius error** is defined as

$$E_{\text{abs}}(k) = \|A - \tilde{A}_k\|_F.$$

The **relative Frobenius error** normalizes this quantity with respect to the total energy of the image:

$$E_{\text{rel}}(k) = \frac{\|A - \tilde{A}_k\|_F}{\|A\|_F} = \sqrt{1 - \frac{\sum_{i=1}^{k} \tilde{\sigma}_i^2}{\sum_{i=1}^{r} \sigma_i^2}},$$

where σ_i and $\tilde{\sigma}_i$ are the true and approximated singular values respectively.

Energy Captured and Loss

The fraction of the image's total energy captured by the first k singular values is:

$$\eta(k) = \frac{\sum_{i=1}^{k} \tilde{\sigma}_i^2}{\sum_{i=1}^{r} \sigma_i^2}.$$

It quantifies the amount of information preserved in the rank-k reconstruction.

The corresponding **energy loss** is:

$$\mathcal{L}(k) = 1 - \eta(k) = \frac{\sum_{i=k+1}^{r} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}.$$

A small $\mathcal{L}(k)$ implies that most of the image variance (energy) is retained even with a truncated SVD.

Interpretation

- As k increases, both E_{abs} and E_{rel} decrease monotonically.
- $\eta(k)$ increases rapidly at small k and saturates near 1, showing diminishing returns.
- In well-compressible images (smooth or structured), over 90% energy is often captured with less than 20% of total singular vectors.

Hence, the Frobenius error and energy loss jointly provide quantitative evidence of the trade-off between compression and reconstruction quality.

Table 1: Frobenius error and energy retention for Einstein image

Rank (k)	Absolute Frobenius Error	Relative Frobenius Error	Energy Captured (%)
5	5574.33	0.2556	93.47
20	2525.98	0.1158	98.66
50	1203.07	0.0552	99.70
100	275.30	0.0126	99.98

Table 2: Frobenius error and energy retention for Globe image

Rank (k)	Absolute Frobenius Error	Relative Frobenius Error	Energy Captured (%)
5	25276.10	0.1597	97.45
20	12298.49	0.0777	99.40
50	7279.02	0.0460	99.79
100	4379.68	0.0277	99.92
200	2247.24	0.0142	99.98

Table 3: Frobenius error and energy retention for Grayscale image

Rank (k)	Absolute Frobenius Error	Relative Frobenius Error	Energy Captured (%)
5	12758.70	0.0658	99.57
20	5308.79	0.0274	99.92
50	1491.40	0.0077	99.99
100	619.86	0.0032	99.99
200	533.72	0.0028	99.99

Table 4: Frobenius error and energy retention for Rick image

Rank (k)	Absolute Frobenius Error	Relative Frobenius Error	Energy Captured (%)
5	35257.90	0.1344	98.19
20	17655.10	0.0673	99.55
50	10169.60	0.0388	99.85
100	6041.35	0.0230	99.95
200	2975.02	0.0113	99.99

Table 5: Frobenius error and energy retention for Jon Snow image

Rank (k)	Absolute Frobenius Error	Relative Frobenius Error	Energy Captured (%)
5	75504.60	0.3559	87.33
20	41592.78	0.1961	96.16
50	20315.64	0.0958	99.08
100	9136.67	0.0431	99.81
200	3683.18	0.0174	99.97

Comparative Discussion: Algorithmic Trade-offs and Implementation Choice

Table 6: Comparison of SVD-based Image Compression Methods

Algorithm	Advantages	Disadvantages
Golub–Kahan SVD (Householder + Givens)	• Most numerically stable and accurate full SVD method. • Well-conditioned for both symmetric and non-symmetric matrices. • Produces exact orthogonal U, Σ, and V^T.	• Computationally expensive ($O(n^3)$) — impractical for large images. • Requires full reduction even when only top-k singular values are needed. • Memory-heavy due to dense transformations.
Lanczos Bidiagonalization + Implicit QR (My Implementation)	• Efficient truncated SVD: $O(kn^2 + k^3)$ with minimal accuracy loss. • Uses only matrix–vector products — memory-light and scalable. • Implicit QR with Wilkinson shift ensures fast, stable convergence. • Ideal for image compression where small approximation is acceptable.	• Sensitive to orthogonality loss if re-orthogonalization is skipped. • Slower convergence for tightly clustered singular values. • Slight numerical drift compared to full SVD, but visually negligible. • Yields only leading singular vectors, not a full decomposition.
Jacobi Algorithm	• Extremely accurate and numerically stable. • Produces well-orthogonalized singular vectors naturally. • Useful for small matrices or benchmark-level precision.	• Very slow convergence ($O(n^3)$). • Inefficient for high-resolution image matrices.
Power / Subspace Iteration	• Simple to implement conceptually. • Works well for computing dominant singular values. • Requires minimal memory.	• Converges slowly for clustered or similar singular values. • Needs multiple passes over data. • Yields only leading singular vectors, not a full decomposition.

Continued on next page

Table 6 – continued from previous page

Algorithm	Advantages	Disadvantages
Randomized SVD (rSVD)	• Extremely fast for very large or sparse datasets. • Reduces computation via random projection and oversampling.	• Complicated to implement from scratch, requires knowledge beyond numerical linear algebra. • Accuracy depends on data distribution; non-deterministic output.

Reflection on my Implementation Choice

In principle, the **Golub–Kahan SVD** remains the best algorithm for exact decompositions. However, for the purpose of image compression, such accuracy is unnecessary. My approach — **Lanczos bidiagonalization followed by implicit QR** — strategically trades a small amount of precision for a huge gain in speed and memory efficiency. Making it ideal for quick, high-quality image compression. While **Randomized SVD** scales better for massive datasets, it introduces stochastic errors and is much more complicated to implement from scratch and offers little to no advantage at our scale. Thus, the selected method achieves the best balance between computational efficiency, accuracy, and practical implementability.

Bibliography

[1] G. Strang, "Singular Value Decomposition," *MIT OpenCourseWare Lecture*, https://youtu.be/TX_vooSnhm8