Course Project(*Software*): Image Compression Using Truncated SVD

EE25BTECH11020 - Darsh Pankaj Gajare

I. Summary of Strang's Video

The concept of Singular Value Decompositions states that any matrix $\mathbf{A}$ as $\mathbf{U\Sigma V}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices and $\mathbf{\Sigma}$ is a diagonal matrix. A vector $\mathbf{v_1}$ in rowspace corresponding to vector $\mathbf{u_1}$ in column space can be written as

$$\sigma\mathbf{u_1} = \mathbf{Av_1} \tag{1}$$

Similarly for $\mathbf{v_2}, \mathbf{v_3}$... and $\mathbf{u_2}, \mathbf{u_3}$... this can be written. These row vectors and column vectors should be orthonormal to each other.

$$\mathbf{A}\begin{pmatrix}\mathbf{v_1} & \mathbf{v_2} & ... & \mathbf{v_r}\end{pmatrix} = \begin{pmatrix}\mathbf{u_1} & \mathbf{u_2} & ... & \mathbf{v_r}\end{pmatrix}\begin{pmatrix}\sigma_1 & \sigma_2 & ... & \sigma_r\end{pmatrix}_{diag} \tag{2}$$

$$\mathbf{AV} = \mathbf{U\Sigma} \tag{3}$$

As $\mathbf{V}$ is orthogonal,

$$\mathbf{A} = \mathbf{A}as\mathbf{U\Sigma V}^\top \tag{4}$$

Computing $\mathbf{A}^\top\mathbf{A}$ we get,

$$\mathbf{A}^\top\mathbf{A} = \mathbf{V\Sigma}^\top\mathbf{\Sigma V}^\top \tag{5}$$

$$\mathbf{A}^\top\mathbf{A} = \mathbf{V}\begin{pmatrix}\sigma_1^2 & \sigma_2^2 & ... & \sigma_r^2\end{pmatrix}_{diag}\mathbf{V}^\top \tag{6}$$

Here $\mathbf{V}$ are eigenvectors of $\mathbf{A}^\top\mathbf{A}$ and$\mathbf{V}\begin{pmatrix}\sigma_1^2 & \sigma_2^2 & ... & \sigma_r^2\end{pmatrix}_{diag}$ are the eigen values of $\mathbf{A}^\top\mathbf{A}$
For $\mathbf{U}$

$$\mathbf{AA}^\top = \mathbf{U\Sigma\Sigma}^\top\mathbf{U}^\top \tag{7}$$

Here $\mathbf{U}$ are the eigenvectors of $\mathbf{AA}^\top$
Interpretation:
$\mathbf{V}$: Basis for input (*row*) space
$\mathbf{U}$: Basis for output (*column*) space
$\mathbf{\Sigma}$: Scaling Transformation

II. Explanation of the implemented algorithm

A. Power Iteration with Deflation for SVD

The code implements Singular Value Decomposition (SVD) using power iteration combined with deflation to compute the top k singular values and vectors of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$.

B. Algorithm Overview

For each rank $r = 1, 2, \ldots, k$, the algorithm performs:

Power Iteration: Starting with a random vector $\mathbf{v}^{(0)}$, iteratively compute:

$$\mathbf{u}^{(t)} = \mathbf{A}\mathbf{v}^{(t-1)} \tag{8}$$

$$\mathbf{v}^{(t)} = \mathbf{A}^T\mathbf{u}^{(t)} \tag{9}$$

$$\mathbf{v}^{(t)} = \frac{\mathbf{v}^{(t)}}{||\mathbf{v}^{(t)}||} \tag{10}$$

After convergence (50 iterations), compute the singular triplet:

$$\mathbf{u}_r = \mathbf{A}\mathbf{v}_r \tag{11}$$

$$\sigma_r = ||\mathbf{u}_r|| \tag{12}$$

$$\mathbf{u}_r = \frac{\mathbf{u}_r}{\sigma_r} \tag{13}$$

Deflation: Remove the rank-1 component from $\mathbf{A}$:

$$\mathbf{A} \leftarrow \mathbf{A} - \sigma_r\mathbf{u}_r\mathbf{v}_r^T \tag{14}$$

C. Reconstruction

The image is reconstructed using the rank-k approximation:

$$\mathbf{Y} = \sum_{r=1}^{k} \sigma_r\mathbf{u}_r\mathbf{v}_r^T \tag{15}$$

III. Comparison between different algorithms

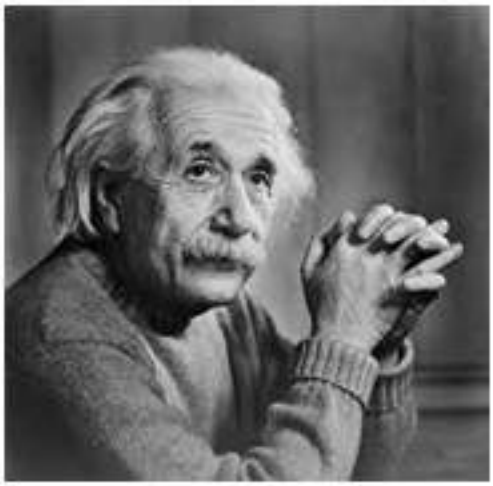
TABLE I: Comparison of different SVD algorithms

Algorithm	Advantages	Disadvantages
Power Iteration + Deflation (Implemented)	• Simple to implement • Memory efficient: $O(mn + k(m+n))$ • Computes only top-k components • Works well for dominant singular values • No external libraries required	• Slow convergence for close singular values • Approximate solution • Accumulation of deflation errors • Random initialization affects results • Fixed iteration count (50) may be suboptimal
Jacobi SVD	• High numerical accuracy • Stable and reliable • Parallelizable	• Slow for large matrices: $O(m^2n)$ • Not suitable for sparse matrices • Computes full SVD (inefficient for top-k)
Divide & Conquer SVD	• Faster than classical methods • High accuracy • Efficient for medium-sized matrices • Standard in LAPACK	• Complex implementation • High memory usage • Not optimal for very large matrices • Computes full SVD
Golub-Reinsch (LAPACK)	• Robust and well-tested • Industry standard • High precision • Handles edge cases well	• $O(mn^2)$ complexity • Requires external libraries • Computes all singular values • Overkill for top-k approximation
Randomized SVD	• Very fast: $O(mn \log k)$ • Excellent for large matrices • Scalable to big data • State-of-the-art for top-k	• Probabilistic accuracy • Requires tuning of oversampling • May fail for difficult spectra • More complex than power iteration
Lanczos Algorithm	• Fast convergence • Memory efficient • Ideal for sparse matrices • Computes extreme eigenvalues quickly	• Numerical instability (loss of orthogonality) • Requires reorthogonalization • More complex implementation • Sensitive to rounding errors

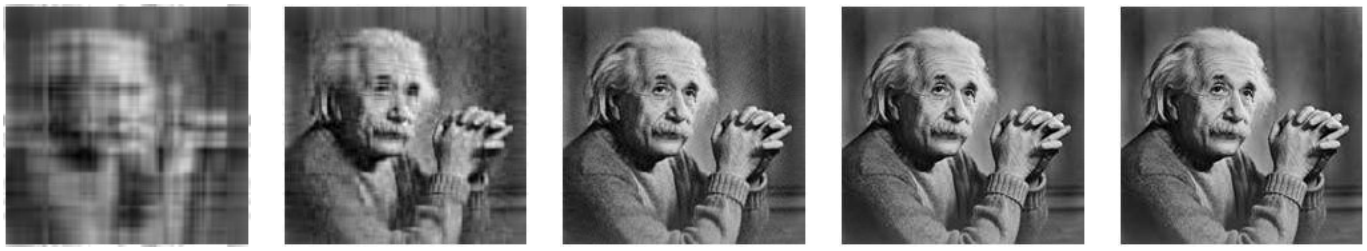
This method was chosen because it directly targets the top-k singular values through iterative refinement, avoiding the $O(mn^2)$ complexity of full SVD algorithms. For image compression, where only the dominant spectral components contribute significantly to visual quality, makes Power Iteration with Deflation an efficient choice.

IV. Reconstructed images

Fig. 1: Original



A. Einstein Image



k = 5, 0.05s | k = 20, 0.17s | k = 50, 0.42s | k = 100, 0.83s | k = 150, 1.25s

Fig. 2: Einstein image reconstruction with varying rank k and execution time

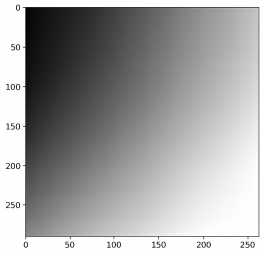
B. Globe Image

Fig. 3: Original

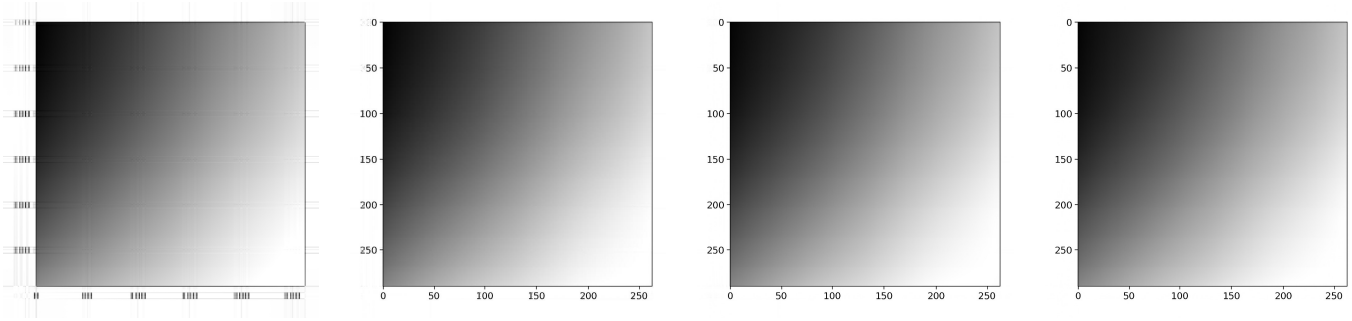
$k = 5$, 0.95s $k = 20$, 3.83s $k = 50$, 9.30s $k = 100$, 19.13s $k = 200$, 37.59s

Fig. 4: Globe image reconstruction with varying rank k and execution time

Fig. 5: Original



C. Gray Image



$k = 5$, 1.38s $k = 20$, 5.48s $k = 50$, 13.72s $k = 100$, 26.10s

Fig. 6: Gray image reconstruction with varying rank k and execution time

V. Error analysis

A. Einstein Image

TABLE II: Reconstruction errors for Einstein image

k	Relative Error (%)	Frobenius Error
5	21.61	4713.6
20	9.75	2126.6
50	4.04	880.5
100	0.755	164.84
150	0.0436	9.5268

B. Globe Image

TABLE III: Reconstruction errors for Globe image

k	Relative Error (%)	Frobenius Error
5	13.078	20704.27
20	6.717	10634.6
50	3.907	6185.7
100	2.3205	3673.78
200	1.129	1787.59

C. Gray Image

TABLE IV: Reconstruction errors for Gray image

k	Relative Error (%)	Frobenius Error
5	5.758	11146.3
20	1.967	3808.19
50	0.599	1160.13
100	0.264	512.34

VI. Discussion and Reflections

The code is easy to debug and understand. Results show it works well: Einstein image gets 0.04% error at $k = 150$, which is pretty good quality.

A. What Could Be Better

The main problem is it's slow when singular values are similar. Using 50 iteration for all images is too good for some while its not enough for larger ones. Also, errors add up as we compute more ranks - Globe has 1.13% error even at $k = 200$.

B. Other Methods I Considered

Golub-Kahan Bidiagonalization: This is the standard method used in libraries like LAPACK. It's more accurate and stable. But it is harder to implement. **Jacobi** This method is very easy to implement but does not give good results as Power iteration.

C. Conclusion

Time increases with k and error drops quickly as we add more ranks