REPORT

EE25BTECH11036 – M. Chanakya Srinivas

1 SUMMARY OF STRANG's VIDEO

Any matrix A can be expressed as a product of three simpler matrices:

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

V^T rotates the coordinate system; next, the diagonal matrix Σ compresses the data along new axes according to the singular values; and finally, U performs a second rotation to position the result in its final orientation.

The singular values in Σ indicate how much each direction contributes to the overall structure of the data. Larger singular values represent important patterns and brightness levels, while smaller ones capture finer details or noise. SVD works for all matrices—square or rectangular—and that keeping only the top k singular values produces the best possible rank-k approximation of the original matrix.

These insights form the foundation of this project: by letting us know the most significant singular components, we can effectively compress a grayscale image while exploring how different choices of k influence both image quality and compression efficiency.

2 Algorithms for Truncated SVD

Truncated Singular Value Decomposition (SVD) approximates a matrix $A \in \mathbb{R}^{m\times n}$ using only its top-k singular values and corresponding singular vectors. Several algorithms exist to compute or approximate this efficiently. Below, six commonly used algorithms are described.

2.1 1) Full SVD + Truncation

This method computes the full SVD of A:

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

and then retains only the top-k singular values and vectors:

$$U_k = U_{(:,1:k)}, \quad \Sigma_k = \Sigma_{(1:k,1:k)}, \quad V_k = V_{(:,1:k)}$$

The rank-k approximation is:

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

2.2 2) Power Iteration (Single Vector Iteration)

This iterative method estimates the dominant singular vector and value using repeated multiplication:

$$v_t = A^T u_{t-1}, \quad u_t = Av_t, \quad u_t \leftarrow \frac{u_t}{\|u_t\|_2}$$

Repeat until convergence. Then, the singular value is:

$$\sigma = \|A^T u_t\|_2, \quad v = \frac{A^T u_t}{\sigma}$$

2.3 3) Deflation Method

After computing a singular triplet (u_i, σ_i, v_i), remove its contribution:

$$A \leftarrow A - \sigma_i u_i v_i^T$$

Repeat the process to extract additional singular components.

2.4 4) Lanczos Bidiagonalization

This method reduces A to a smaller bidiagonal matrix B_k whose SVD approximates the dominant singular components:

$$A \approx U_k B_k V_k^T$$

It is particularly efficient for large and sparse matrices.

2.5 5) Randomized SVD

Randomized SVD approximates the range of A using a random matrix Ω:

$$Y = A\Omega$$

Optionally, apply q power iterations: $Y \leftarrow (AA^T)^q Y$. Compute an orthonormal basis Q of Y and project:

$$B = Q^T A, \quad B = \tilde{U}\Sigma V^T, \quad U_k = Q\tilde{U}_{(:,1:k)}$$

2.6 6) Krylov Subspace / Block Power Method

This method generalizes power iteration to a block of vectors, forming a Krylov subspace:

$$K = [A\Omega, (AA^T)A\Omega, \ldots, (AA^T)^{q-1}A\Omega]$$

Compute an orthonormal basis Q of K and project:

$$B = Q^T A, \quad B = \tilde{U}\Sigma V^T, \quad U_k = Q\tilde{U}_{(:,1:k)}$$

3 THE ALGORITHM USED

3.1 POWER ITERATION

This method computes the top-k singular values and vectors of a matrix $A \in \mathbb{R}^{m \times n}$ iteratively. It combines **Power Iteration** to estimate the dominant singular vector and **Deflation** to remove its effect, repeating the process to obtain multiple singular components.

4 PSEUDOCODE

Input: Grayscale image matrix $A \in \mathbb{R}^{m \times n}$, list of target ranks $k_1, k_2, \ldots, k_z$
Output: Compressed images for each k and Frobenius error

1) **Load Image:** Convert the input image to a grayscale matrix A.
2) **Compute Frobenius norm of original matrix:**

$$\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$$

3) **For each target rank k:**
   a) Initialize empty matrices $U \in \mathbb{R}^{m \times k}$, $V \in \mathbb{R}^{n \times k}$ and vector $S \in \mathbb{R}^k$.
   b) Make a copy of A as B (residual matrix).
   c) **For** $r = 1$ **to** k**:**
      i) Initialize vector u randomly (or sequentially as in your code).
      ii) **Power Iteration:**

$$\text{Repeat until convergence: } \begin{cases} v \leftarrow B^T u \\ u \leftarrow Bv \\ u \leftarrow u/\|u\|_2 \end{cases}$$

      iii) Compute singular value: $\sigma_r = \|B^T u\|_2$, normalize $v = B^T u/\sigma_r$.
      iv) Store u_r, v_r, σ_r in U, V, S.
      v) **Deflate matrix:** $B \leftarrow B - \sigma_r u_r v_r^T$
   d) **Reconstruct Approximate Matrix:**

$$\hat{A}_k = \sum_{r=1}^{k} \sigma_r u_r v_r^T$$

   e) **Compute Frobenius error:**

$$E = \frac{\|A - \hat{A}_k\|_F}{\|A\|_F} \times 100\%$$

   Print error.
   f) **Save compressed image:** Convert $\hat{A}_k$ to 8-bit grayscale and write to file.
4) Free all memory and exit.

4.1 WHY I CHOSE POWER ITERATION

Power iteration finds the strongest pattern in the matrix. Deflation removes that pattern, allowing the next strongest pattern to emerge.It is Easier than other methods while solving Truncated SVD.

4.2 RECONSTRUCTION OF A

The reconstructed matrix is:

$$\hat{A}_k = \sum_{r=1}^{k} s_r u_r v_r^\top$$

- s_r is the r-th **singular value** of A, with $s_1 \geq s_2 \geq \cdots \geq s_{\min(m,n)} \geq 0$.
- u_r is the r-th **left singular vector**, a column vector of length m.
- v_r is the r-th **right singular vector**, a column vector of length n.
- $v_r^\top$ is the transpose of v_r, so that $u_r v_r^\top$ is an $m \times n$ matrix (outer product).

5 ERROR ANALYSIS

Image	$k = 5$	$k = 20$	$k = 50$	$k = 100$
Einstein	21.61	9.75	4.04	0.76
Globe	13.08	6.72	3.91	2.32
Greyscale	5.70	1.93	0.56	0.18

As observed, the Frobenius error decreases with increasing rank k, this shows error depends on rank k.

the approximation error (Frobenius norm) is:

$$E = \frac{\|A - \hat{A}_k\|_F}{\|A\|_F} \times 100\%$$

6 RECONSTRUCTED IMAGES FOR DIFFERENT K

6.1 EINSTEIN.jpg

Fig. 1: k=5

Fig. 2: k=20

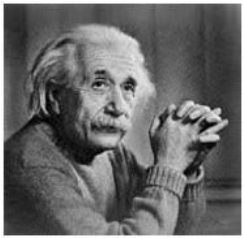


Fig. 3: k=50

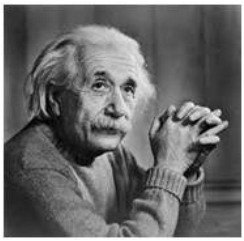


Fig. 4: k=100

6.2 ***GLOBE.jpg***

Fig. 5: k=5

Fig. 6: k=20

Fig. 7: k=50

Fig. 8: k=100

*6.3 **GREYSCALE.jpg***

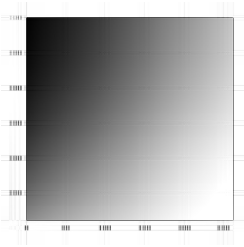
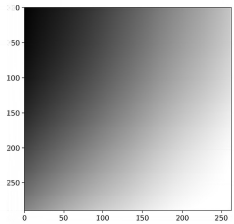


Fig. 9: k=5



H

Fig. 10: k=20

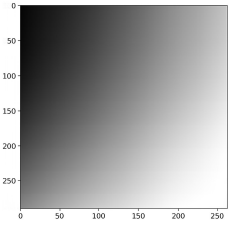


Fig. 11: k=50

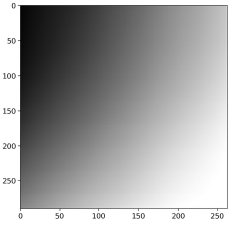


Fig. 12: k=100

Trade-off between k, Image Quality, and Compression

When compressing an image using truncated Singular Value Decomposition (SVD):

$$\hat{A}_k = \sum_{r=1}^{k} s_r u_r v_r^\top$$

Effect of k

As k increases, the error decreases, improving image quality.

Summary Table

k	Image Quality	Compression Ratio
Small	Low (blurry)	High
Moderate	Moderate-High	Moderate
Large	High (almost original)	Low