Software Assignment-Image Compression Using Truncated SVD

AI25BTECH11004-B.JASWANTH

Introduction

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:

$$A = U\Sigma V^T \tag{0.1}$$

where:

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

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

$$A_k = U_k\Sigma_k V_k^T \tag{0.2}$$

This approximation often preserves most of the image content while using far fewer numbers, which is the basis of **image compression**.

Strang's Video Summary

The Singular Value Decomposition (SVD) of a real matrix $A \in \mathbb{R}^{m\times n}$ is given by:

$$A = U\,\Sigma\,V^T \tag{0.3}$$

where each term has a specific meaning:

- A : The original data or image matrix of size $m \times n$.
- U : An orthogonal matrix of size $m \times m$ (or $m \times r$ if rank $r < \min(m, n)$), whose columns are called the *left singular vectors* of A. They form an orthonormal basis for the **column space** of A.

$$U^T U = I_m \tag{0.4}$$

- V : An orthogonal matrix of size $n\times n$ (or $n\times r$), whose columns are the *right singular vectors* of A. They form an orthonormal basis for the **row space** of A.

$$V^T V = I_n \tag{0.5}$$

- Σ : A diagonal (or rectangular diagonal) matrix of size $m \times n$ whose nonzero entries $\sigma_1, \sigma_2, \ldots, \sigma_r$ are called the *singular values* of A. These are always non-negative and are arranged in decreasing order:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0 \tag{0.6}$$

Each σ_i measures how much A stretches the corresponding right singular vector v_i.

- **Rank-r structure:** If A has rank r, then only the first r singular values are nonzero, and the SVD can be written as:

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{0.7}$$

where u_i and v_i are the ith columns of U and V, respectively.
The SVD can be viewed as a sequence of transformations:

Obtaining the Intensity matrix

First we have to obtain the intensity matrix from the given input figure by the c code.It can be obtained by using the *stb_image.h* and find the intensity matrix.Now we have to use truncated svd so that we should get an approaximated intensity matrix using jacobi iteration.The concept in the jacobi method is :

Theory: Randomized SVD and Low-Rank Approximation with Jacobi Iteration

The goal of the randomized Singular Value Decomposition (rSVD) algorithm is to compute a low-rank approximation of a large matrix $A \in \mathbb{R}^{m\times n}$ efficiently. Instead of performing a full SVD, which is computationally expensive for large matrices, rSVD projects A onto a lower-dimensional random subspace that approximately spans the column space of A.

Random Sampling

We begin by generating a random Gaussian matrix G,

$$\Omega \in \mathbb{R}^{n\times k}, \tag{0.8}$$

whose entries are drawn independently from a standard normal distribution

$$\Omega_{ij} \sim \mathcal{N}(0, 1). \tag{0.9}$$

Multiplying A with Ω produces the sample matrix

$$Y = A\Omega, \tag{0.10}$$

where $Y \in \mathbb{R}^{m\times k}$ captures the dominant column space of A with high probability. This step effectively compresses A while preserving its essential features.

Orthonormal Basis Computation

We compute an orthonormal basis for the range of Y using, for example, the Gram-Schmidt process:

$$Y = QR, \tag{0.11}$$

where $Q \in \mathbb{R}^{m\times k}$ has orthonormal columns and $R \in \mathbb{R}^{k\times k}$ is upper-triangular. The matrix Q forms an approximate orthonormal basis for the column space of A.

Projection to a Smaller Matrix

Next, we project A onto the subspace spanned by Q:

$$B = Q^T A, \tag{0.12}$$

where $B \in \mathbb{R}^{k\times n}$ is a much smaller matrix than A. Because Q approximately spans the column space of A, we have

$$A \approx QQ^T A = QB. \tag{0.13}$$

Small SVD via Eigen-Decomposition

Since B is small, we can efficiently compute its singular value decomposition:

$$B = \tilde{U}\Sigma V^T, \tag{0.14}$$

where $\tilde{U} \in \mathbb{R}^{k\times k}$, $\Sigma \in \mathbb{R}^{k\times k}$, and $V \in \mathbb{R}^{n\times k}$. In the program, this SVD is computed via the eigen-decomposition of the symmetric matrix

$$BB^T = \tilde{U}\Sigma^2\tilde{U}^T, \tag{0.15}$$

using a Jacobi eigen-solver.

Mapping Back to Original Space

The left singular vectors of A are obtained by mapping back using Q:

$$U = Q\tilde{U}. \tag{0.16}$$

Thus, the approximate singular value decomposition of A is

$$A \approx U\Sigma V^T. \tag{0.17}$$

Rank-k Approximation

Truncating to the top k singular values and vectors gives the low-rank approximation

$$A_k = U_k\Sigma_k V_k^T, \tag{0.18}$$

where U_k, Σ_k, and V_k contain only the first k columns or entries of the respective matrices.

This method achieves significant computational savings because the heavy operations (matrix multiplications and eigen-decomposition) are performed on small matrices of size $m \times k$ or $k \times n$, rather than the full $m \times n$ matrix.

Comparision with other algorithms

Jacobi have high numerical stability and it maintains orthogonality of eigenvectors well. It only uses basic arithmetic and trignometric operations.

QR algorithm is fast for large matrices.
It can compute singular values of large matrices also.

Power Iterations has low memory reqirement.
It directly gives the larger singular value.

Why to use Jacobi Iteration?

After random projection, the small matrix BB^T has size $k \times k$ where k is typically much smaller (example, 50, 100). The Jacobi algorithm is computationally efficient for such small matrices. Jacobi ensures orthogonal eigenvectors and is highly stable for symmetric matrices, which BB^T always is. For small matrices, Jacobi gives nearly the same accuracy as QR-based SVD but with simpler logic and fewer numerical edge cases.Jacobi can be implemented using only basic math, which makes your program fully self-contained.

Error analysis using Frobenius Norm

The accuracy of the low-rank approximation $A_k = U_k \Sigma_k V_k^T$ and $A = \sum_{i=1}^{r} \sigma_i \, u_i v_i^T$,where r is rank of A, can be measured using the Frobenius norm of the residual matrix:

$$E = A - A_k \tag{0.19}$$

$$||A - A_k||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij} - (A_k)_{ij})^2} \tag{0.20}$$

In terms of the singular values σ_i of A, the optimal rank-k approximation error is:

$$||A - A_k||_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2} \tag{0.21}$$

For comparison across matrices, the relative error is defined as:

$$\text{Relative Error} = \frac{||A - A_k||_F}{||A||_F} \tag{0.22}$$

Intensity Matrix to Image

The above obtained matrix should be converted into image using first k singular values.There is a c code to transform the matrix using *stb_image_write.h* .It gives the output as a figure with the given k value.

Implementation

Einstein

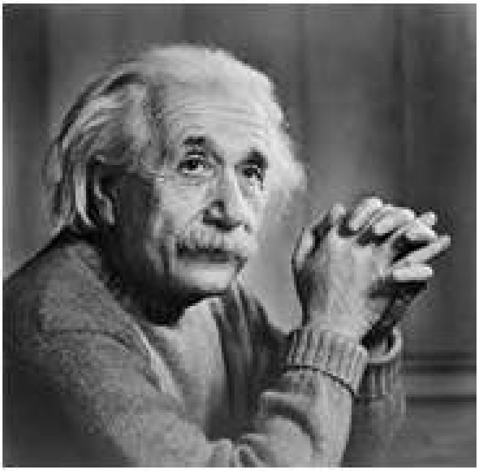

Fig. 0: Original image

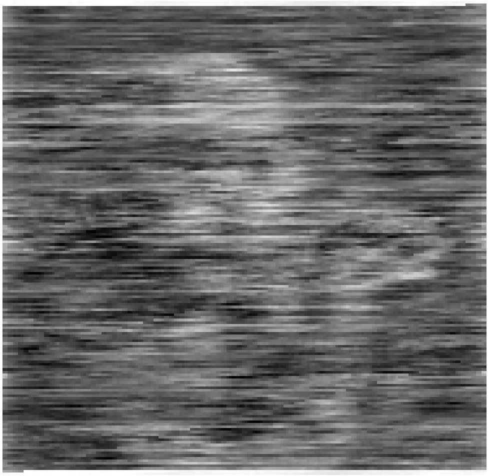

Fig. 0: Image at k=10

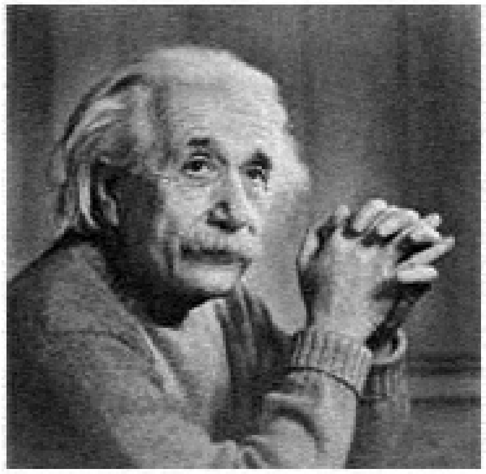

Fig. 0: Image at k=100

Its original intensity matrix size was 182×186.So,that for value of k=100,the image came is almost similar.Greater the value of k,greater the sharpness and Quality of the picture.

Error with different k values

Name	**Value of k**	**Frobenius norm**
einstein_10	10	8061.238837
einstein_50	50	3767.087054
einstein_100	100	1365.972045
einstein_150	150	402.006749
einstein_180	180	42.358696

TABLE 0: Relation between k and frobenius norm

Globe

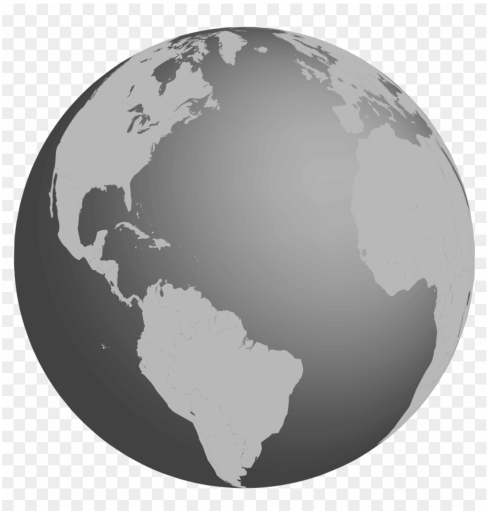
Fig. 0: Original Image

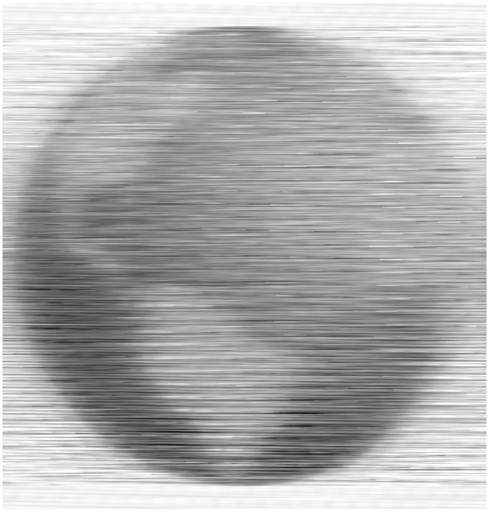
Fig. 0: Image at k=10

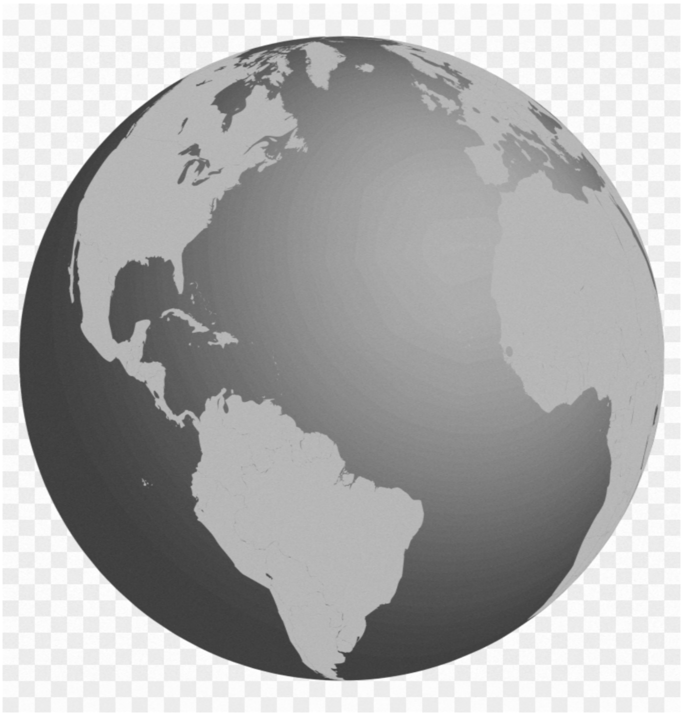
Fig. 0: Image at k=500

Error with different k values

Name	**Value of k**	**Frobenius norm**
globe_10	10	32728.746414
globe_100	100	9448.077475
globe_500	500	1496.871018
globe_800	800	32.221910

TABLE 0: Relation between k and frobenius norm

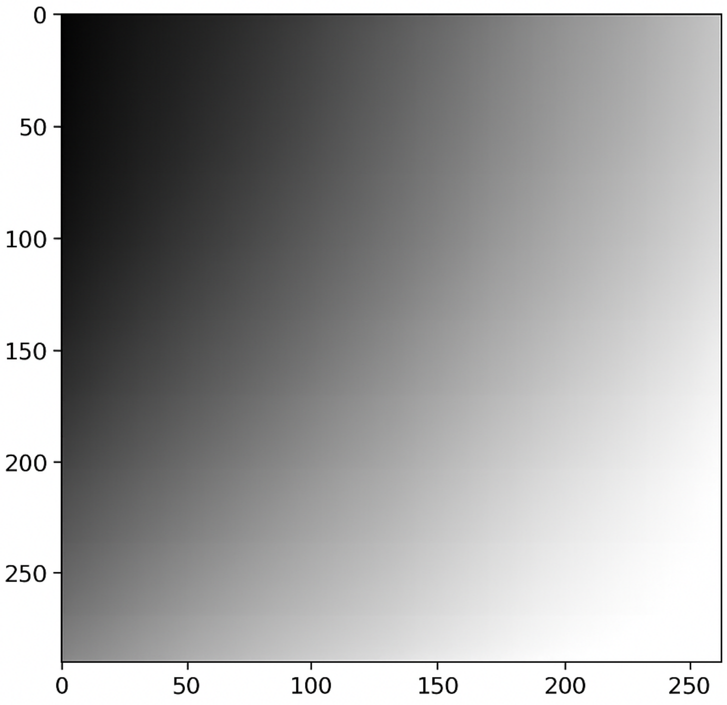

Fig. 0: Original Image

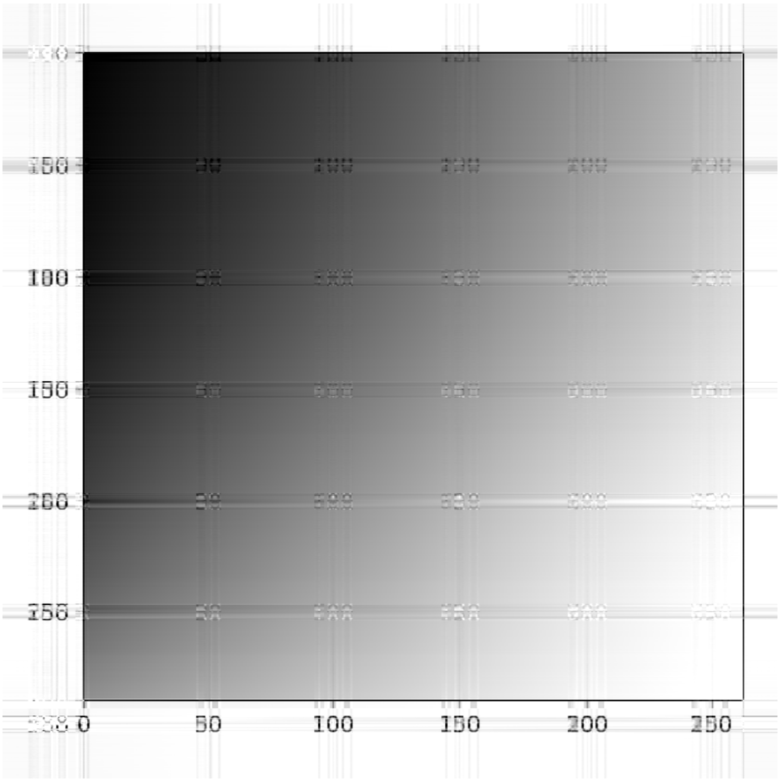

Fig. 0: Image at k=10

Error with different k values

Name	Value of k	Frobenius norm
greyscale_10	10	10867.079753
greyscale_100	100	1022.218584
greyscale_500	500	316.357178
greyscale_800	800	126.479341
greyscale_1000	1000	13.574708

TABLE 0: Relation between k and frobenius norm

Trade-off between k,image Quality,and compression

In SVD-based image compression, the parameter k controls the rank of the approxi-

mation.
A smaller k yields higher compression since fewer singular values and vectors are stored, but results in loss of fine details and higher reconstruction error.
A larger k improves image quality by preserving more singular components but increases the storage cost.
Thus,k represents a tradeoff between compression efficiency and visual fidelity.