Software - Assignment
Image Compression Using Truncated SVD

EE25BTECH11047 - RAVULA SHASHANK REDDY

November 8, 2025

0.1 Introduction

A greyscale image is compressed using truncated SVD by taking the top k(arbitrary) singular values.We can make a low-rank approximation by taking only the top k singular values.

Singular Value Decomposition of a matrix is

$$\mathbf{A} = \mathbf{U\Sigma V^T}$$

Truncated SVD is nothing but expressing the same using only top k singular values.

$$\mathbf{A}_k = \mathbf{U_k \Sigma_k V_k^T}$$

Summary of Strang's Video

Professor Gilbert Strang explains how to decompose a matrix using SVD.He made some computations and eliminate anyone of the matrix among **U** and **V**.He intentionlly multiplied A^TA and AA^T and proceeded in furthur computation.Large the singular value, dominant it contributes to that matrix.

Strang also tells that every matrix can be expressed into SVD decomposition.These points make us easy to compute the truncated SVD.He gave an intuition to start the project to compress the image by finding top k singular values.

0.2 Algorithms for Truncated SVD

1.Power Iteration with Deflation(Choosen Algorithm)

Explanation

Main Idea: $\mathbf{v} = \dfrac{A^T(A\mathbf{v})}{||A^T(A\mathbf{v})||} \implies$ Repeatedly apply this .

Initialize a random vector **v** and normalize it and multiply it with the matrix **A**.Find its norm and again new vector $\mathbf{v}_i = \mathbf{A}.\mathbf{v}$ .Now again normalize the new vector and multiply with A again.Now check the difference in value of the norm of the previous vector and present vector.Change decreases after multiple steps and at some step we get the vector mostly not changes .That is the top most right eigen vector and its norm gives the highest singular value.

$$\sigma_i = ||Av_i||_2$$
$$u_i = \frac{Av_i}{\sigma_i}$$

(u, σ, v) is the triplet required.

$$\text{Deflation} \implies A \to A - \sigma_i.u_i^T v_i$$

Now we have to remove the Top contributing part that is top most singular value part from the matrix(This is Deflation).Then continue the power iteration after deflation.Like this we can get the top k singular values and the right and left singular vectors.

Psuedo Code

```
Choose a random vector $v_i$ and normalize it
$\mathbf{v} \leftarrow \frac{A^T(A\mathbf{v})}{||A^T(A\mathbf{v})||}$
Repeat until the vector converge
$\sigma_i = ||Av_i||$
$u_i = \frac{Av_i}{\sigma_i}$
after one triplet$(\sigma_i, u_i, v_i)$
deflate the matrix by
$A \leftarrow A - \sigma_i u_i^T v_i$
Repeat from the start
```

2.Full SVD + Truncation

This is a classical method and takes much time and computations.

Compute the full SVD and find the top k singular values and its vectors .

Gives very accurate values but requires more computation.

3.Lanczos Bidiagonalization

This is the best way to be more efficient and more accurate .This algorithm works faster.But the disadvantage is it is very complicated to code implement.

Key Steps:

Reduces the matrix A to a bidiagonal matrix B_k. Matrix $A \in \mathbb{R}^{m\times n}$

Choose random vector v_1 such that $||v_1||_2 = 1$

$$\begin{aligned} u_i &= Av_i - \beta_{i-1}u_{i-1} \\ \alpha_i &= ||u_i||_2 \\ u_i &= u_i/\alpha_i \\ v_{i+1} &= A^T u_i - \alpha_i v_i \\ \beta_i &= ||v_{i+1}||_2 \\ v_{i+1} &= v_{i+1}/\beta_i \end{aligned}$$

by repeating these steps form B_k matrix and then perform SVD on this matrix. Construct U_k, V_k.

4.Randomized SVD

This algorithm runs very very faster.The main idea is to use projection of a random matrix on the given matrix.

Take a random matrix(G) and multiply it with the given matrix(A). Apply QR decomposition on this product of matrices.Perform SVD on a small matrix B

$$\begin{aligned} P &= AG \\ P &= QR \\ B &= Q^T A \\ A &\approx Q(SVD(B)) \end{aligned}$$

5.Block Power Method

This algorithm also uses the same logic of power iteration but uses multiple vectors at once.This method works faster since uses multiple vectors at a time.

It builds a subspace of L=[$Av, A^2v, A^3v, ...$].These are orthonormalised and again SVD is performed to this.

This algorithm works faster but difficult to code.

6.Jacobian SVD method

In this algorithm columns of given matrix(A) are repeatedly rotated by applying a Jacobi rotation matrix(G) to it.After multiple rotations of columns of A ,they become orthogonal.

The norms of these orthogonal columns are the singular values.

$$AG \to A$$

Repeat this until we get the orthogonal columns.This method is somewhat slower method. This algorithm gives higher accuracy.

Choosen Method

My choice of algorithm is ”Power Iteration using deflation”.

I chose this method because it is easy to implement.

We can code the algorithm easily in C .This algorithm gives medium accuracy and works at moderate speed.

0.3 Reconstructed Images

Einstein(k=5,20,40,60)

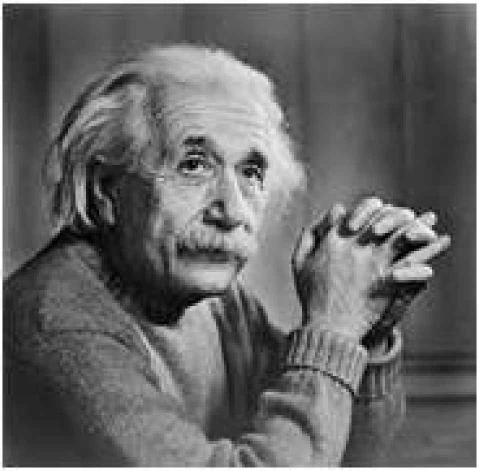

Figure 1: Original

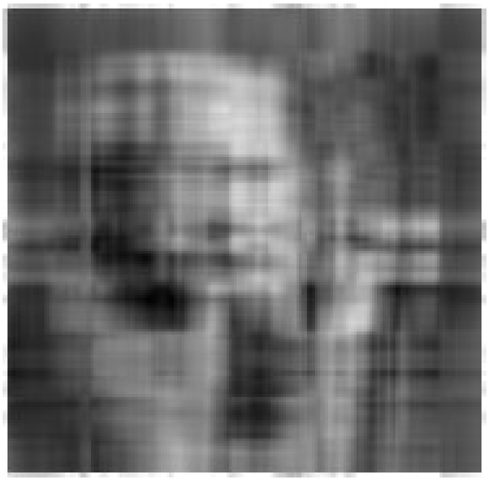

Figure 2: k=5

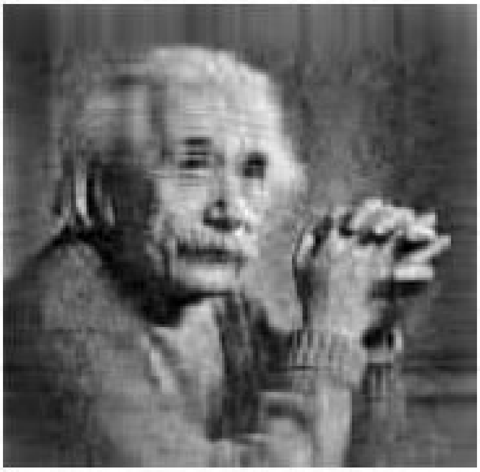

Figure 3: k=20

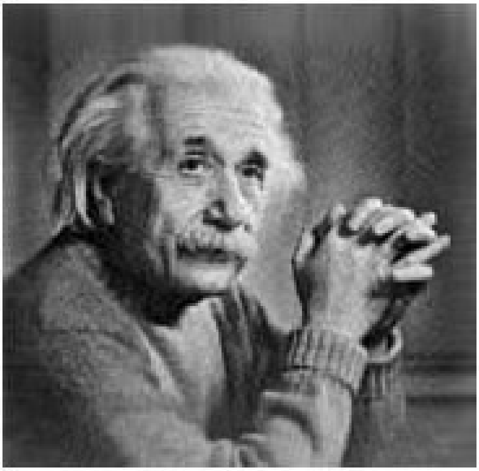

Figure 4: k=40

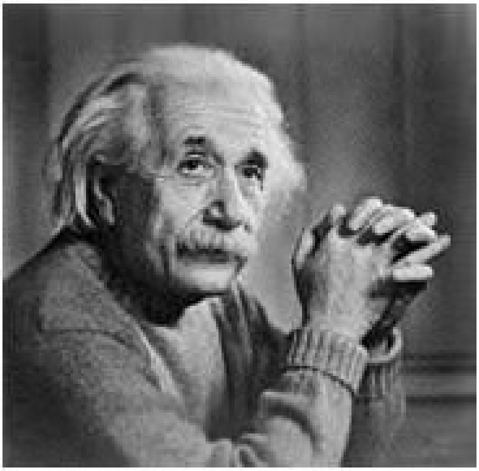

Figure 5: k=60

Globe(k=5,20,40,60)

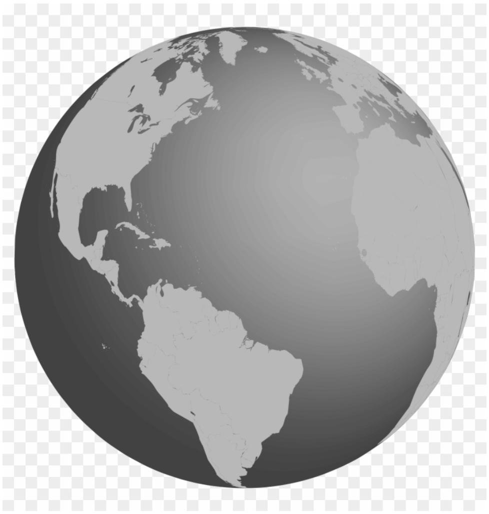

Figure 6: Original

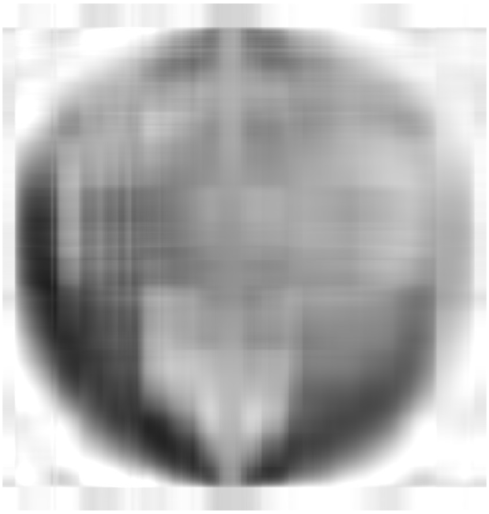

Figure 7: k=5

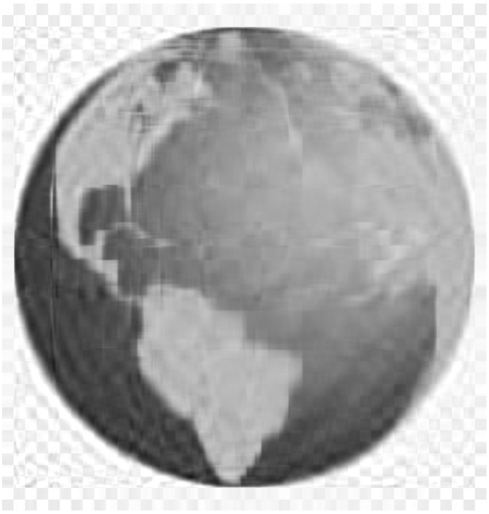
Figure 8: k=20

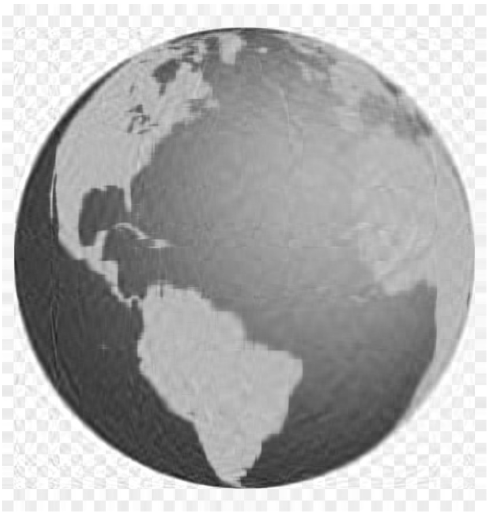
Figure 9: k=40

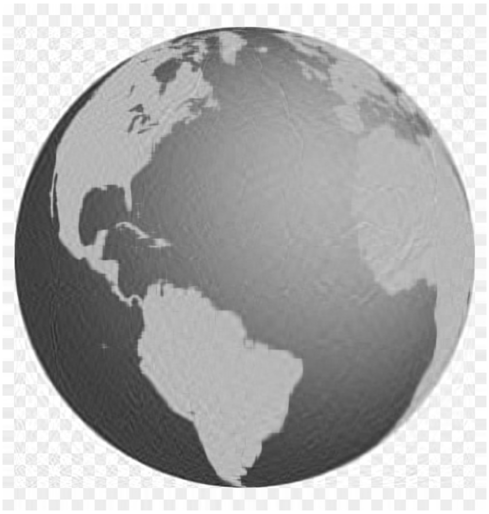

Figure 10: k=60

Greyscale(k=5,10,20,50)

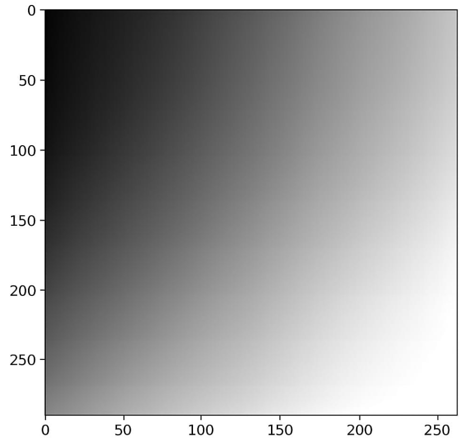

Figure 11: Original

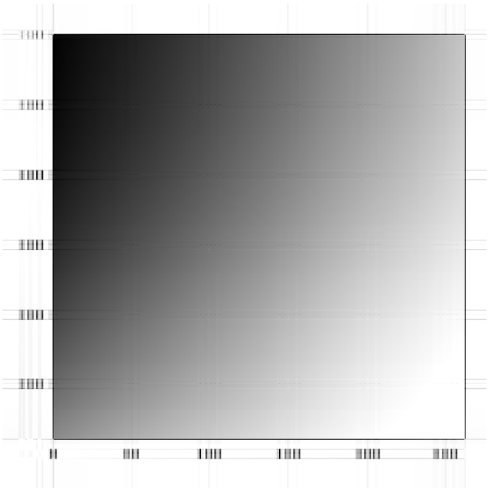
Figure 12: k=5

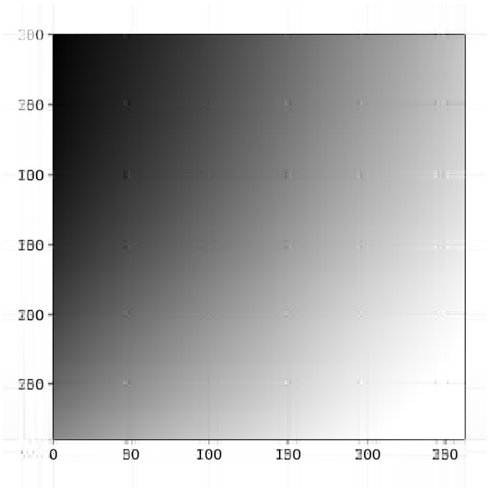

Figure 13: k=10

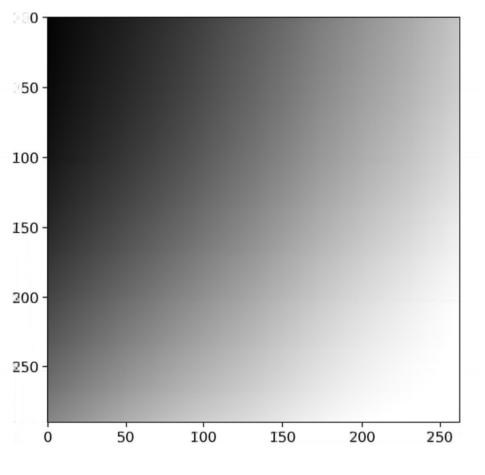

Figure 14: k=20

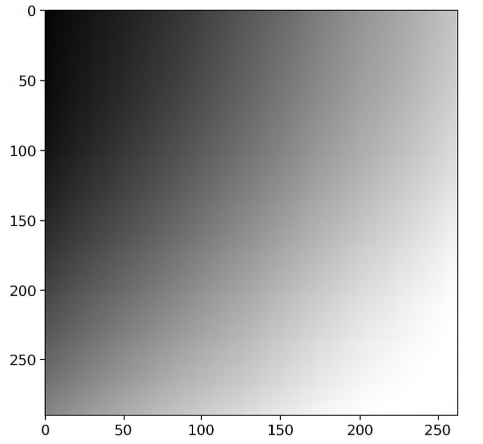

Figure 15: k=50

0.4 Error Analysis

Error is calculated by Frobenius Norm of Matrices.

$$\text{Relative error percentage} = \frac{||A_{original} - A_{truncated}||_F}{||A_{original}||_F} * 100$$

Image	k = 5	k = 10 / 20	k = 20 / 40	k = 60 / 50
Einstein	21.62%	9.75% (k = 20)	5.34% (k = 40)	3.00% (k = 60)
Globe	13.08%	6.72% (k = 20)	4.53% (k = 40)	3.44% (k = 60)
Greyscale	5.43%	3.43% (k = 10)	1.65% (k = 20)	0.44% (k = 50)

Table 0: Relative error percentages for different values of *k* in Truncated SVD.

0.5 Conclusion

”Image Compression using truncated SVD” for this project I have used ”Power Iteration with Deflation” algorithm.I have chosen this method because primarily it is an easy to understand algorithm.This algorithm easy to code and implement also.It does not require whole SVD decomposition to find top k values .

The major drawback is ,it works slower than other algorithms since it requires multiple steps to make vectors to converge to required vector.Somewhat accuracy is also moderate since it approximates the matrix.

By taking different values of k , I came to know that ”larger the k value less is the compression and more is the quality of the image and smaller the k value more is the compression and less is the quality of the image(Blurry image).Finally I understand the how dominant singular values contribute to image compressionand how to compress an image using truncated SVD.