Image Compression Using Truncated SVD

K. Sai Sreevallabh - EE25BTECH11031

November 8, 2025

Overview

Grayscale images are made up of pixels whose pixel intensity values range from 0 (black) to 255 (white). The grayscale image can be represented as a matrix $A \in \mathbb{R}^{m \times n}$ where each entry of the matrix corresponds to the pixel intensity value.

A low-rank approximation can be obtained by only keeping the top-k singular values of A as given below:

$$A_k = U_k \Sigma_k V_k$$

The above expression corresponds to a truncated Singular Value Decomposition of A and it forms the basis for image compression.

Singular Value Decomposition

[A Summary of Gilbert Strang's Lecture]

1. Introduction

The idea behind the Singular Value Decomposition is the mapping of an orthogonal basis in the row-space of any rectangular matrix $\mathbf{A} \in \mathbb{R}$ to an orthogonal basis of its column-space. We find a special set of orthogonal input axes that A transforms into an orthogonal output axes.

Given a rectangular matrix $A \in \mathbb{R}^{m \times n}$ of rank r, consider a unit vector $\mathbf{v_1}$ to be in its row-space. There exists a unit vector $\mathbf{u_1}$ in the column-space of $\mathbf{A}$ such that

$$\mathbf{A v_1} = \sigma_1 \mathbf{u_1}$$

Considering another unit vector $\mathbf{v_2}$, orthogonal to $\mathbf{v_1}$ in the row-space of $\mathbf{A}$, the vector $\mathbf{u_2}$ is given by

$$\mathbf{A v_2} = \sigma_2 \mathbf{u_2}$$

and is orthogonal to $\mathbf{u_1}$.

Following such a process we can find r linearly independent vectors in the row-space and the column-space and represent it in matrix form as

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

where

$$\mathbf{V} = \begin{pmatrix} \mathbf{v_1} & \mathbf{v_2} & \dots & \mathbf{v_r} \end{pmatrix} \quad \mathbf{U} = \begin{pmatrix} \mathbf{u_1} & \mathbf{u_2} & \dots & \mathbf{u_r} \end{pmatrix} \quad \text{and} \quad \mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{pmatrix}$$

However, we can also include the vectors present in the null-space into this. It will not affect the orthogonality of $\mathbf{V}$ because the null-space is always perpendicular to the row-space. All the corresponding σ values become zero and we can consider any set of $m - r$ vectors in $\mathbf{U}$ to make it an $m \times m$ matrix. Typically, we consider the null-space of $\mathbf{A}^\top$. Then, we get

$$\mathbf{V} = \begin{pmatrix} \mathbf{v_1} & \mathbf{v_2} & \dots & \mathbf{v_n} \end{pmatrix} \quad \mathbf{U} = \begin{pmatrix} \mathbf{u_1} & \mathbf{u_2} & \dots & \mathbf{u_n} \end{pmatrix} \quad \text{and} \quad \mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n \end{pmatrix}$$

2. Finding the SVD

For computing the singular value decomposition of a matrix, two orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ must be found first. To find $\mathbf{V}$, $\mathbf{U}$ is eliminated by doing the following

$$\begin{aligned} \mathbf{A}^\top\mathbf{A} &= \mathbf{V\Sigma}^\top\mathbf{U}^\top\mathbf{U\Sigma V}^\top \\ &= \mathbf{V}(\mathbf{\Sigma}^\top\mathbf{\Sigma})\mathbf{V}^\top \quad \text{(since } \mathbf{U} \text{ is orthogonal)} \\ &= \mathbf{V}(\mathbf{\Sigma}^2)\mathbf{V}^T \quad \text{(since } \mathbf{\Sigma} \text{ is diagonal)} \end{aligned}$$

The above is the Spectral Decomposition of the symmetric matrix $\mathbf{A}^\top\mathbf{A}$. Eigenvalues can be calculated by reducing $\mathbf{A}^\top\mathbf{A}$ into an upper triangular matrix and for each eigenvalue, the corresponding eigenvector (in normalised form) becomes an entry in the matrix $\mathbf{V}$.

$\sigma_1^2, \sigma_2^2, \dots \sigma_n^2$ are the eigenvalues of $\mathbf{A}^\top\mathbf{A}$. Out of these only the first r are non-zero.

Similarly, for finding $\mathbf{U}$, we eliminate $\mathbf{V}$, giving us

$$\mathbf{AA}^\top = \mathbf{U}(\mathbf{\Sigma}^2)\mathbf{U}^T$$

Again, we get the same non-zero eigenvalues, with the rest $m - r$ being non-zero.

After finding the eigenvalues, the corresponding eigenvectors can also be computed, yielding $\mathbf{U}$ and $\mathbf{V}$.

$$\therefore \mathbf{A} = \mathbf{U\Sigma V}^\top$$

Background About Algorithms

The algorithms used to compute the Singular Value Decomposition do not compute it in the manner as stated above due to the following reason:

- Computing $\mathbf{A}^\top\mathbf{A}$ or $\mathbf{A}\mathbf{A}^\top$ leads to numerical instability and loss of precision because it squares the condition number (which measures how sensitive the matrix is to small errors).

There are mainly two classes of algorithms used for the computation of the Singular Value Decomposition of a Matrix:

1. **Iterative Methods:**
    - Only the required number of singular values are computed, directly resulting in the truncated SVD.
    - The results are approximated progressively by convergence over iterations.
    - Block Power Method is an example of iterative methods.
2. **Direct Methods**
    - Computes all the singular values and hence gives us the total SVD.
    - A finite sequence of operations are executed, leading to the full SVD.
    - Golub-Reinsch Method and Jacobi Method are two examples of direct methods.

.
In the context of Image Compression:
To compress an image, we compute the truncated SVD of an input matrix, upto k singular values. Below are a few points of comparison between direct and iterative methods in this sense:

- Direct methods calculate the total SVD and then truncation must be done. While iterative methods compute the truncated SVD directly. For $k << min\{m, n\}$ iterative methods are much faster than direct methods.
- Direct methods are known to be more accurate as compared to iterative methods. However, the difference in accuracy is negligible in this context. There is very little distinguishable difference.

So, overall, iterative methods have an edge when it comes to image compression.

Algorithm: Block SVD Power Method

Name and structure of the algorithm has been referred from **Source**

This algorithm is different from the usual Block Power method for computing SVD.

1. Outline of Algorithm:

$\mathbf{A} \in \mathbb{R}$ is the input matrix, which is obtained from the image. We need to get the truncated SVD upto k singular values $\mathbf{A_k} = \mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^\top$.

Consider the initial assumption of $\mathbf{V_k}$ to be $\mathbf{V_0}$, which is a random $n \times k$ matrix.

$$\mathbf{Y} = \mathbf{AV_0}$$

Upon QR decomposition of $\mathbf{Y}$, we get $\mathbf{Q_1}$ and $\mathbf{R_1}$.

Fix our first assumption of $\mathbf{U_k}$ to be $\mathbf{U_0} = \mathbf{Q_1}$.

Now,

$$\mathbf{Z} = \mathbf{A}^\top\mathbf{U}$$

Upon doing QR decomposition of $\mathbf{Z}$, we get $\mathbf{Q_2}$ and $\mathbf{R_2}$.

Our second assumption of $\mathbf{V_k}$ becomes $\mathbf{Q_2}$. And $\mathbf{R_2}$ contains our current approximation of the singular values along its diagonal.

Now the process is again followed from the initial step, but with $\mathbf{V_1}$ in place of $\mathbf{V_0}$.

This process takes place as long as the convergence condition is not met or the maximum number of iterations is not crossed.

For the convergence condition the sum of absolute difference between the elements of $\mathbf{R}$ (obtained from QR decomposition of $\mathbf{Z}$) and the corresponding elements of the $\mathbf{R}$ calculated in the previous iteration is found. The absolute sum of elements of the current $\mathbf{R}$ is also calculated.

Once their division goes below a particular value, the loop stops running and our algorithm converges.

If the convergence criterion is not met before the maximum iterations are over, the loop stops nonetheless. This is to ensure that the code doesn't run for too long.

This however doesn't affect the image quality to a great deal, as verified through trial runs with my code.

NOTE:
For QR decomposition, the method that i have employed is the Modified Gram Schmidt Orthogonalisation process. It is considered to be more numerically stable than the Classical Gram Schmidt Orthogonalisation process.

2. Pseudocode

The main function for the Block SVD Power Method, takes the input of the image matrix $\mathbf{A}$, which already has certain values. It takes the dimensions, m and n also as input. The empty matrices $\mathbf{U}, \mathbf{V}, \mathbf{S}$ are the ones which are filled in this function. The function is a void function and hence doesnt return any value.

Algorithm 1: Block SVD Power Method

Input: Matrix $A_{m\times n}$, target rank k, Matrices U, S, V

$A^T \leftarrow$ transpose(A);
$R \leftarrow$ zero matrix of size $k \times k$;
$C \leftarrow$ zero matrix of size $k \times k$;

`/* Initialize` V `randomly between` -1 `and` 1 `*/`

$V \leftarrow$ new matrix of size $n \times k$;
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** k **do**
    $V_{ij} \leftarrow 2 \times \text{rand}() - 1$;
  end
end

$err \leftarrow 1$;
$val \leftarrow 1$;
$iter \leftarrow 0$;

while $err/val > 10^{-6}$ ***and*** $iter < 20$ **do**
  $Y \leftarrow A \times V$;
  $[U, R] \leftarrow QR(Y)$;
  $Z \leftarrow A^T \times U$;
  $[V, R] \leftarrow QR(Z)$;
  $err \leftarrow \sum |R - C|$;
  $val \leftarrow \sum |R|$;
  $C \leftarrow R$;
  $iter \leftarrow iter + 1$;
end

$S \leftarrow R$;

3. Mathematical Proof of Algorithm

Source

Let s be an integer and q too such that $r = qs$ where r is the rank of $\mathbf{A}$ and

$$\sigma_1 \geq \cdots \geq \sigma_s > \sigma_{s+1} \geq \cdots \geq \sigma_{qs} > 0$$

the singular values of $\mathbf{A}$.

We are considering q blocks of size s each and splitting our total matrix into these blocks as

$$\mathbf{A} = \sum_{i=1}^{q} \mathbf{U}_i \mathbf{\Sigma}_i \mathbf{V}_i^T$$

where $\mathbf{\Sigma}_i$ is a diagonal matrix with nonzero, monotonically decreasing diagonal $\sigma_{(i-1)s+1} \geq \sigma_{(i-1)s+2} \geq \cdots \geq \sigma_{is} > 0$. $\mathbf{U}_i$ and $\mathbf{V}_i$ are the orthogonal matrices whose columns are respectively the corresponding left and right singular vectors.

Let $\mathbf{V}_0 \in \mathbf{R}^{m\times s}$, is our first assumption of the matrix $\mathbf{V}$.
$\mathbf{V}_0 = \sum_{i=1}^{q} \mathbf{V}_i \mathbf{X}_i + \mathbf{V}_0^*$, where $\mathbf{V}_0^*$ corresponds to the null-space of $\mathbf{A}$.

We have

$$\mathbf{W}_0 = \mathbf{A}\mathbf{V}_0 = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{X}_1 + \sum_{i=2}^{q} \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{X}_i.$$

Suppose that the component $\mathbf{X}_1 = \mathbf{I}_s$.

$$\mathbf{A}\mathbf{V}_0 = \mathbf{U}_1\mathbf{R}_1 \quad \text{(QR factorization)}$$

$$\implies \mathbf{U}_1\mathbf{R}_1 = \mathbf{U}_1\mathbf{\Sigma}_1 + \sum_{i=2}^{q} \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{X}_i$$

$$\mathbf{U}_1 = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{R}_1^{-1} + \sum_{i=2}^{q} \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{X}_i\mathbf{R}_1^{-1}$$

Now,

$$\mathbf{A}^T\mathbf{U}_1 = \mathbf{V}_1\mathbf{R}_2 \quad \text{(QR factorization)}$$

We know $\mathbf{A}^\top\mathbf{U_1} = \mathbf{V_1}\mathbf{\Sigma}$

$$\implies \mathbf{V}_1\mathbf{R}_2 = \mathbf{V}_1\mathbf{\Sigma}_1^2\mathbf{R}_1^{-1} + \sum_{i=2}^{q} \mathbf{V}_i\mathbf{\Sigma}_i^2\mathbf{X}_i\mathbf{R}_1^{-1}$$

$$\mathbf{V}_1 = \mathbf{V}_1\mathbf{\Sigma}_1^2\mathbf{R}_1^{-1}\mathbf{R}_2^{-1} + \sum_{i=2}^{q} \mathbf{V}_i\mathbf{\Sigma}_i^2\mathbf{X}_i\mathbf{R}_1^{-1}\mathbf{R}_2^{-1}$$

and so on, if we note $\mathbf{N}_t = \mathbf{R}_1^{-1}\mathbf{R}_2^{-1}\ldots\mathbf{R}_t^{-1}$, at step k we have

$$\mathbf{U}_k = \mathbf{U}_1\mathbf{\Sigma}_1^{2k-1}\mathbf{N}_{2k-1} + \sum_{i=2}^{q} \mathbf{U}_i\mathbf{\Sigma}_i^{2k-1}\mathbf{X}_i\mathbf{N}_{2k-1}$$

and

$$\mathbf{V}_k = \mathbf{V}_1\mathbf{\Sigma}_1^{2k}\mathbf{N}_{2k} + \sum_{i=2}^{q} \mathbf{V}_i\mathbf{\Sigma}_i^{2k}\mathbf{X}_i\mathbf{N}_{2k}$$

$\mathbf{U}_k$ and $\mathbf{V}_k$ are orthogonal matrices, then

$$\mathbf{I}_s = (\mathbf{U}_k)^T\mathbf{U}_k = \mathbf{N}_{2k-1}^T\mathbf{\Sigma}_1^{4k-2}\mathbf{N}_{2k-1} + \sum_{i=2}^{q} \mathbf{N}_{2k-1}^T\mathbf{X}_i^T\mathbf{\Sigma}_i^{4k-2}\mathbf{X}_i\mathbf{N}_{2k-1}$$

$$\mathbf{I}_s = (\mathbf{V}_k)^T\mathbf{V}_k = \mathbf{N}_{2k}^T\mathbf{\Sigma}_1^{4k}\mathbf{N}_{2k} + \sum_{i=2}^{q} \mathbf{N}_{2k}^T\mathbf{X}_i^T\mathbf{\Sigma}_i^{4k}\mathbf{X}_i\mathbf{N}_{2k}$$

by left and right-factoring, we obtain

$$\mathbf{I}_s = \mathbf{N}_{2k-1}^T\mathbf{\Sigma}_1^{2k-1}\left(\mathbf{I}_s + \sum_{i=2}^{q} \mathbf{\Sigma}_1^{-2k+1}\mathbf{X}_i^T\mathbf{\Sigma}_i^{4k-2}\mathbf{X}_i\mathbf{\Sigma}_1^{-2k+1}\right)\mathbf{\Sigma}_1^{2k-1}\mathbf{N}_{2k-1}$$

$$\mathbf{I}_s = \mathbf{N}_{2k}^T\mathbf{\Sigma}_1^{2k}\left(\mathbf{I}_s + \sum_{i=2}^{q} \mathbf{\Sigma}_1^{-2k}\mathbf{X}_i^T\mathbf{\Sigma}_i^{4k}\mathbf{X}_i\mathbf{\Sigma}_1^{-2k}\right)\mathbf{\Sigma}_1^{2k}\mathbf{N}_{2k}$$

Since $\|\mathbf{\Sigma}_s^{-1}\| = \frac{1}{\sigma_s}$ and $\|\mathbf{\Sigma}_i\| = \sigma_{(i-1)s+1}$ then,

$$\begin{aligned}\left\|\mathbf{\Sigma}_1^{-p}\mathbf{X}_i^T\mathbf{\Sigma}_i^{2p}\mathbf{X}_i\mathbf{\Sigma}_1^{-p}\right\| &\leq \|\mathbf{\Sigma}_i\|^{2p}\|\mathbf{\Sigma}_1^{-1}\|^{2p}\|\mathbf{X}_i\|^2 \\ &\leq \left(\frac{\sigma_{(i-1)s+1}}{\sigma_s}\right)^{2p}\|\mathbf{X}_i\|^2 \xrightarrow{p\to\infty} 0\end{aligned}$$

Thus

$$\lim_{p\to\infty}(\mathbf{N}_p^T\mathbf{\Sigma}_1^p)(\mathbf{\Sigma}_1^p\mathbf{N}_p) = \lim_{p\to\infty}(\mathbf{\Sigma}_1^p\mathbf{N}_p)^T(\mathbf{\Sigma}_1^p\mathbf{N}_p) = \mathbf{I}_s.$$

Moreover, the matrix $\mathbf{\Sigma}_1^p\mathbf{N}_p$ is triangular with positive diagonal entries, then $\lim_{p\to\infty}\mathbf{\Sigma}_1^p\mathbf{N}_p = \lim_{p\to\infty}\mathbf{N}_p^{-1}\mathbf{\Sigma}_1^{-p} = \mathbf{I}_s$. Otherwise

$$\begin{aligned}\mathbf{A}^T\mathbf{U}_k\left(\mathbf{N}_{2k-1}^{-1}\mathbf{\Sigma}_1^{-(2k-1)}\right)\mathbf{\Sigma}_1^{-1} &= \mathbf{A}^T\mathbf{U}_k\mathbf{R}_{2k}^{-1}\left(\mathbf{N}_{2k}^{-1}\mathbf{\Sigma}_1^{-2k}\right) \\ &= \mathbf{V}_k\left(\mathbf{N}_{2k}^{-1}\mathbf{\Sigma}_1^{-2k}\right) \\ &= \mathbf{V}_1 + \sum_{i=2}^{q}\mathbf{V}_i\mathbf{\Sigma}_i^{2k}\mathbf{X}_i\mathbf{\Sigma}_1^{-2k} \xrightarrow{k\to\infty} \mathbf{V}_1\end{aligned}$$

$$\begin{aligned}\mathbf{A}\mathbf{V}_k\left(\mathbf{N}_{2k}^{-1}\mathbf{\Sigma}_1^{-2k}\right)\mathbf{\Sigma}_1^{-1} &= \mathbf{A}\mathbf{V}_k\mathbf{R}_{2k+1}^{-1}\left(\mathbf{N}_{2k+1}^{-1}\mathbf{\Sigma}_1^{-(2k+1)}\right) \\ &= \mathbf{U}_{k+1}\left(\mathbf{N}_{2k+1}^{-1}\mathbf{\Sigma}_1^{-(2k+1)}\right) \\ &= \mathbf{U}_1 + \sum_{i=2}^{q}\mathbf{U}_i\mathbf{\Sigma}_i^{2k+1}\mathbf{X}_i\mathbf{\Sigma}_1^{-(2k+1)} \xrightarrow{k\to\infty} \mathbf{U}_1\end{aligned}$$

Comparing With Different Algorithms

1. Golub-Reinsch Algorithm

It is a direct method for the computation of SVD.

Outline of the process:

- Matrix $\mathbf{A}$ is converted into bidiagonal matrix $\mathbf{B}$(only the main diagonal and one superdiagonal contain non-zero elements), using Householder Transformations on the left and right. It results in
$$\mathbf{A} = \mathbf{U_1}\mathbf{B}\mathbf{V_1^\top}$$
- The bidiagonal matrix is converted to a diagonal matrix (the required Σ) using QR iterations resulting in:
$$\mathbf{B} = \mathbf{U_2}\mathbf{\Sigma}\mathbf{V_2^\top}$$
- The Singular Value decomposition is given by
$$\mathbf{A} = \mathbf{U_1}\mathbf{U_2}\mathbf{\Sigma}\mathbf{V_1^\top}\mathbf{V_2^\top}$$
- This is then brought down to the truncated form by considering only the first k singular values and their corresponding right and left singular vectors.

Comparison:

- Block SVD Power Method directly gives the truncated SVD, upto k singular values, while the Golub-Reinsch Algorithm calculates the full SVD and then truncates it.
- The Block SVD Power Method is much faster compared to Golub-Reinsch when the value of $k << min\{m, n\}$.
- When the value of k is comparable to m, n, the Golub-Reinsch and the Block SVD Power Method take roughly the same amount of time to execute.
- The Block SVD Power Method is more straightforward, including only matrix multiplications, while the Golub-Reinsch Algorithm includes advanced methods like Householder transforms.

2. Block Power Method:

- The Block SVD Power Method has greater numerical precision than the usual Block Power Method.
- The Block SVD Power Method computes both $\mathbf{U}$ and $\mathbf{V}$, while the Block Power Method computes only one and then calculates for the other.
- The Block SVD Power Method involves two QR decompositions per iteration and hence can be more taxing than the usual method.

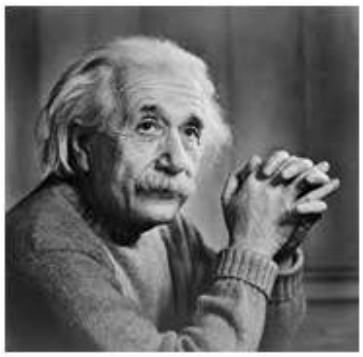
3. Power Iteration + Deflation:

- Block SVD Power Method computes all k required singular values at once, while power iteration computes them one by one.
- This makes the former more efficient and fast.

Reconstructed Images

1. Einstein

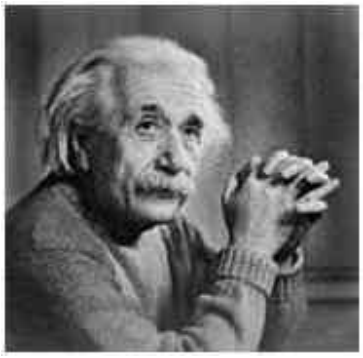
Original Image



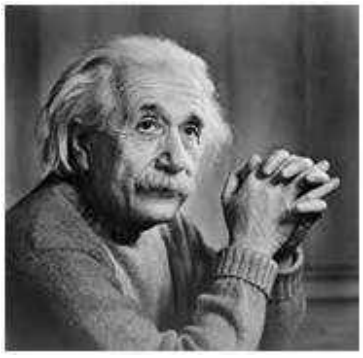
Reconstructed Images:

k=5

k=20



k=50

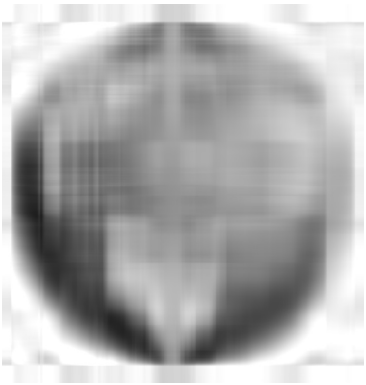


k=100

2. Globe

Original Image

Reconstructed Images:



k=5

k=20

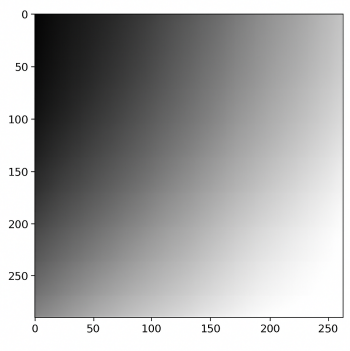
k=50

k=100

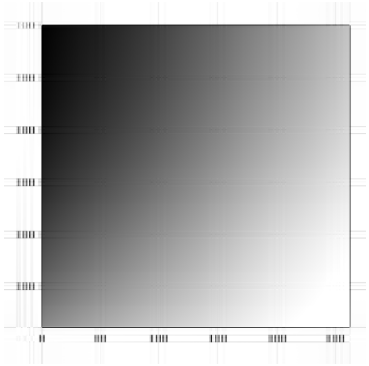
k=250

3. Greyscale

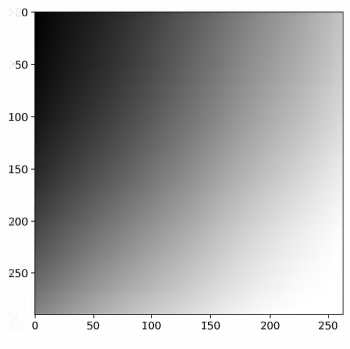
Original Image



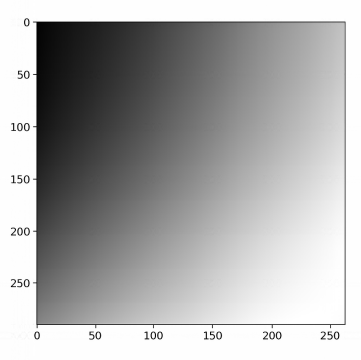
Reconstructed Images:



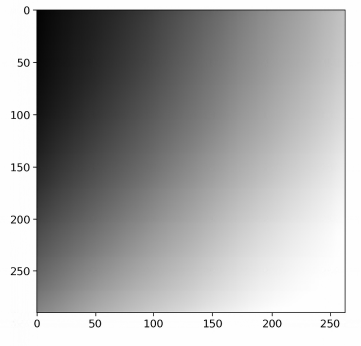
k=5



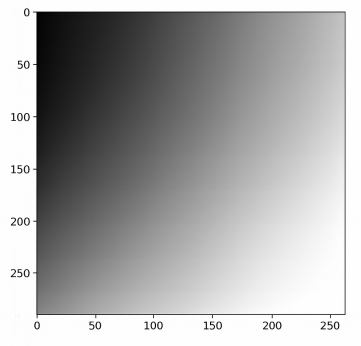
k=20



k=50



k=100



k=400

Error Analysis

1. Einstein:

k	**Absolute Error** ($\|\|A - A_k\|\|_F$)	**Percentage error**
5	4632.953	21.243
20	2130.275	9.767
50	887.268	4.068
100	222.108	1.018

2. Globe:

K	**Absolute error** ($\|\|A - A_k\|\|$)	**Percentage error**
5	20500.153	12.94
20	10,554.123	6.66
50	6157.929	3.889
100	5298.580	3.346
250	1531.013	0.967

3. Greyscale:

k	**Absolute Error** ($\|\|A - A_k\|\|_F$)	**Percentage error**
5	10,994.109	5.679
20	3538.740	1.828
50	1671.791	0.554
100	537.845	0.277
400	310.128	0.160