EE69205: Signal Processing System Design
Indian Institute of Technology, Kharagpur

Image Compression using Singular Value Decomposition

Anirvan Krishna | Roll no. 21EE38002

1. Objective

The objective of this experiment is to compress a gray-scale image using the Singular Value Decomposition (SVD) algorithm. Additionally, the experiment aims to evaluate how the quality of these compressed images is affected by retaining only a subset of singular value components of the original image and to analyze the compressed image quality by measuring the mean squared error (MSE), peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) between the original and compressed images. Also the aim is to analyze the performance when image compression is performed patch-wise.

2. Singular Value Decomposition for image compression

Singular Value Decomposition (SVD) is a matrix factorization technique that decomposes a matrix A into three matrices: U, Σ, and V^T. Mathematically,

$$A = U\Sigma V^T \tag{1}$$

where U and V are orthogonal matrices, and Σ is a diagonal matrix of singular values.

To determine the matrices U and V in the SVD equation, we perform the following steps:

(1) **Compute the covariance matrix:** For a given matrix A, compute the covariance matrix $C = A^TA$.
(2) **Eigenvalue decomposition:** Perform eigenvalue decomposition on the covariance matrix C to obtain the eigenvalues and eigenvectors.
(3) **Form the matrices:**
    (1) The matrix V is formed by the eigenvectors of C.
    (2) The matrix U is formed by the normalized projections of A onto the eigenvectors of C.

Mathematically, this can be represented as:

$$C = A^TA = V\Lambda V^T \tag{2}$$

where Λ is the diagonal matrix of eigenvalues. The columns of V are the eigenvectors of C. The matrix U is then given by:

$$U = AV\Sigma^{-1} \tag{3}$$

where Σ is the diagonal matrix of singular values, which are the square roots of the eigenvalues in Λ.

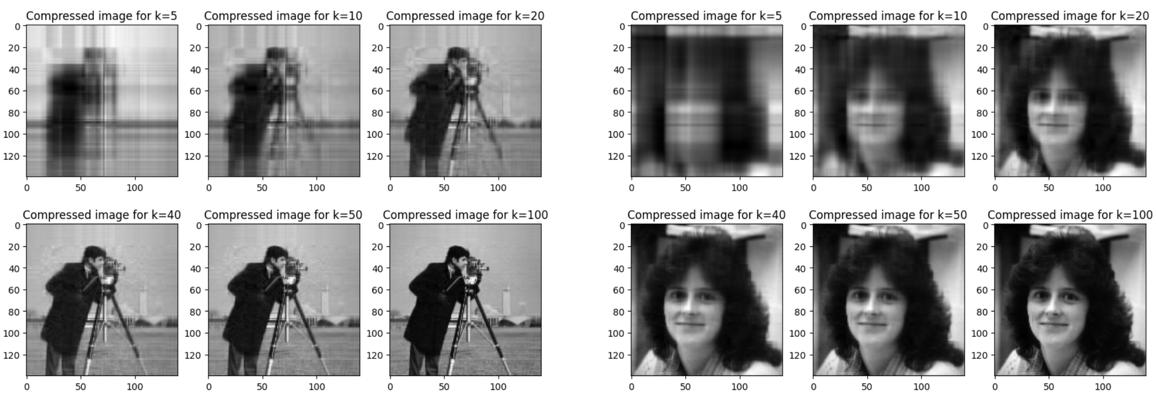
For any value of K, the image can be reconstructed as:

$$A' = \sum_{k=1}^{K} U_{(:,k)} \Sigma_{(k)} V_{(k,:)}^{T} \tag{4}$$

Compression Ratio (CR) for any value of K is defined as:

$$CR = \frac{K(1+H+W)}{HW} \tag{5}$$

In this experiment we are using an image of size 140×140. So, we have a total of 140 singular values. Let k denote the number of singular values retained during the compression.



Cameraman image — Face image

Fig. 1. SVD compressed images for different proportion of retained singular values

Evaluation Metrics: To evaluate the quality of the compressed image in comparision with the original image, we used the following metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure SSIM and Mean Squared Error (MSE).

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2 \tag{6}$$

where x and y are the two 1D signals being compared, N is the length of the signals.

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}_x^2}{\text{MSE}} \right) \tag{7}$$

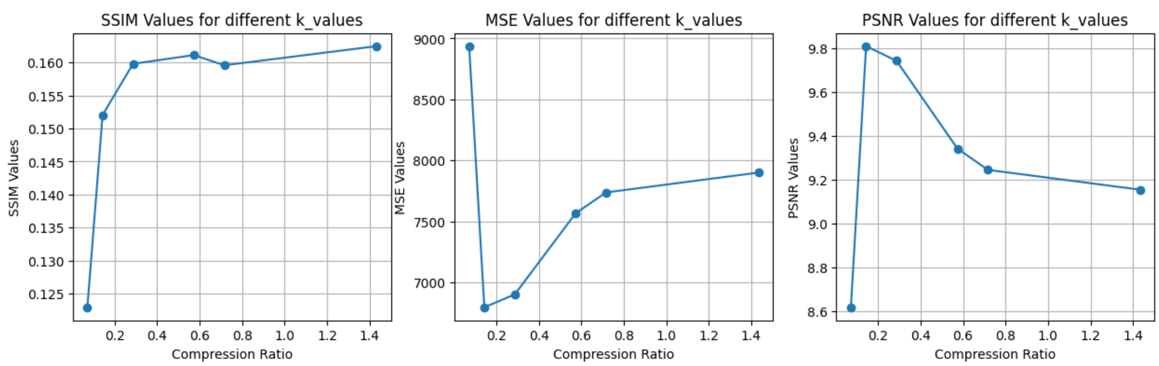
where MAX_x is the maximum possible value in the signal x, MSE is the Mean Squared Error between signals x and y.

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{8}$$

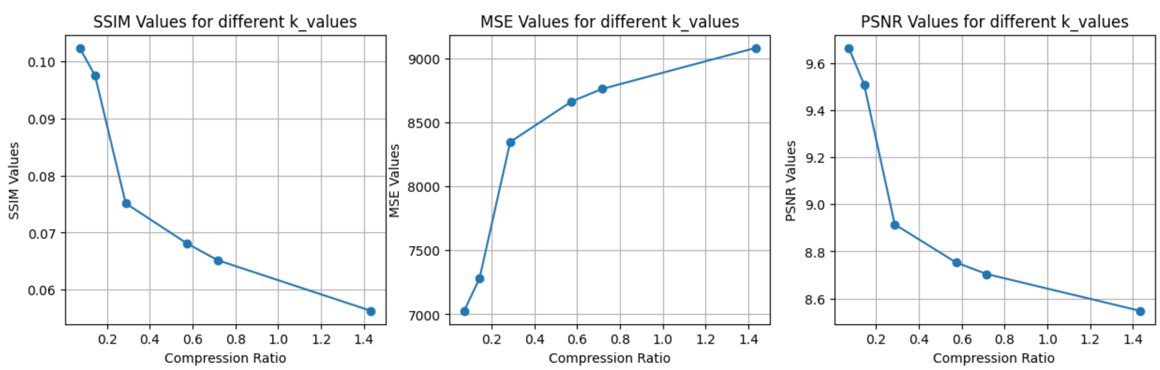
where μ_x and μ_y are the mean values of x and y, σ_x^2 and σ_y^2 are the variances of x and y, σ_{xy} is the covariance between x and y, C_1 and C_2 are small constants to avoid division by zero, typically set relative to the dynamic range of the signals.

k	Cameraman Image			Face Image		
	PSNR	**SSIM**	**MSE**	**PSNR**	**SSIM**	**MSE**
5	8.6190	0.1229	8936.7979	9.6629	0.1024	7027.3229
10	9.8082	0.1521	6796.0765	9.5082	0.0976	7282.1534
20	9.7413	0.1598	6901.5918	8.9146	0.0751	8348.7889
40	9.3416	0.1611	7566.9646	8.7529	0.0681	8665.3825
50	9.2447	0.1596	7737.5985	8.7040	0.0652	8763.6281
100	9.1545	0.1625	7900.0332	8.5479	0.0563	9084.2568

Table 1. SVD Compression Results for Different Values of k for both images



Metrics for cameraman image



Metrics for face image

Fig. 2. Metric comparision for SVD-based compression-reconstruction

3. Patch-wise SVD for compression

Need for Patch-wise Compression: Patch-wise compression improves efficiency and quality by targeting specific characteristics of smaller image regions. Key benefits include:

(1) **Localized Detail Handling:** Compresses high-detail patches with less loss, preserving quality, while applying stronger compression to low-detail areas without visible degradation.

(2) **Error Localization:** Artifacts or errors are confined to small regions, reducing their visibility, which is beneficial for applications like video streaming.
(3) **Parallel Processing:** Allows independent processing of patches, enabling faster compression on multi-core or distributed systems.
(4) **Improved Compression Ratios:** Algorithms like JPEG can better compress repetitive patterns by dividing images into optimal patch sizes, enhancing compression ratios with minimal quality loss.
(5) **Adaptability to Compression Methods:** Different patches can use different algorithms or levels, such as lossy for textures and lossless for precise details like text or faces.
(6) **Machine Learning Compatibility:** Patch-wise processing aligns with neural network architectures, as seen in models like VAEs or transformers, which handle patches for local features and aggregate global information.
(7) **Applications:** Used in CODEC (HEVC, AV1) and neural compression for streaming, teleconferencing, and real-time applications, where quality and bandwidth efficiency are critical.

Patch-wise compression optimally balances quality and efficiency by adapting compression to localized image characteristics, making it essential in modern compression algorithms.

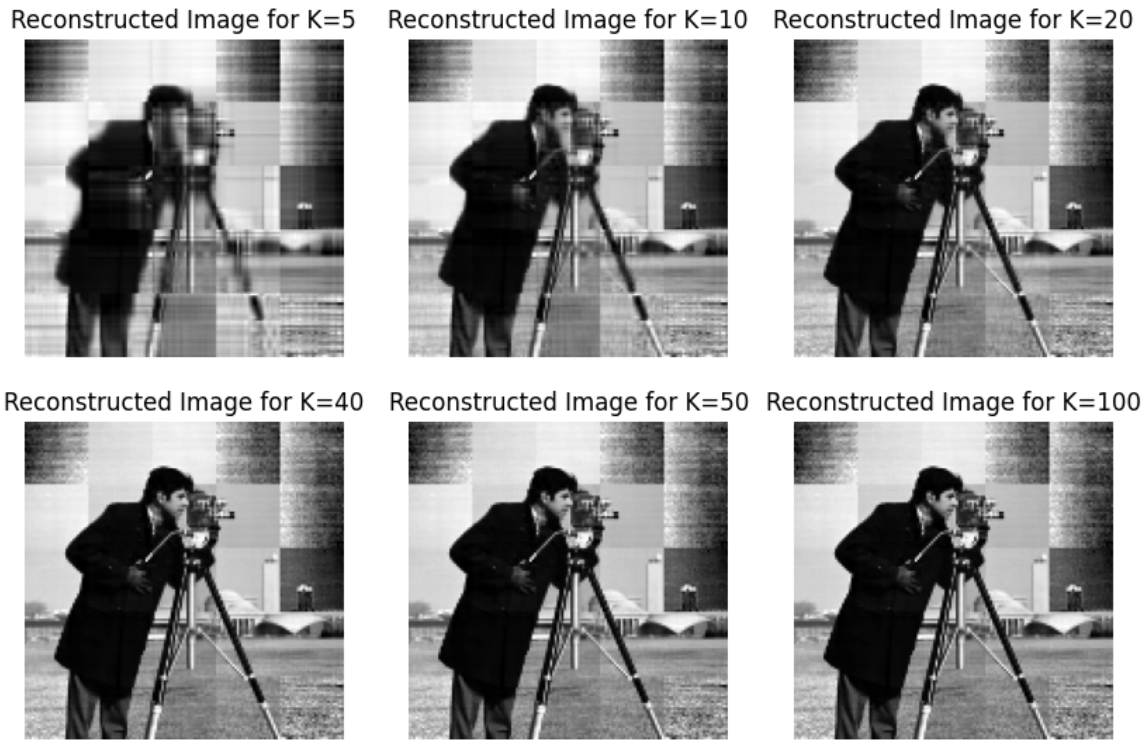
To implement patchwise compression and reconstruction, we first extract image patches based

Cameraman Image

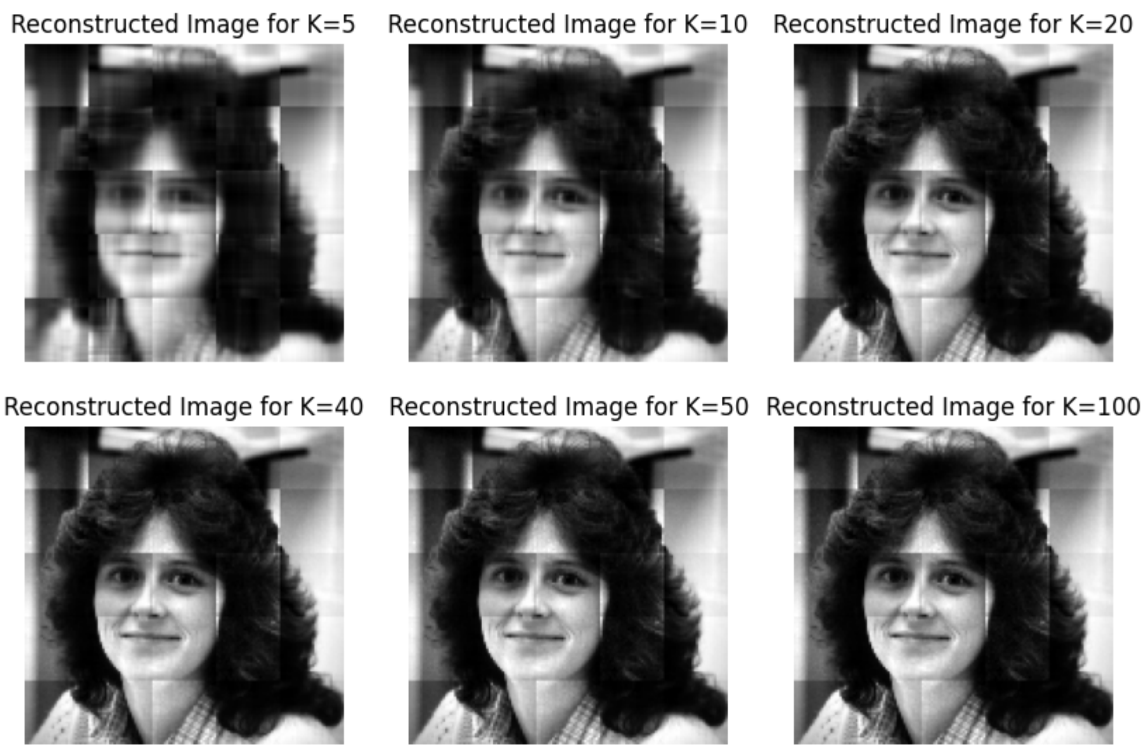
Facial Image

Fig. 3. Patch-wise decomposition of the given the given 140×140 images into 25 images of size 28×28

on each k-value, where each set of 25 patches represents a specific k. For each k, these patches are merged into a 5x5 grid using `montage`, forming a reconstructed image. We store each reconstructed image and compute its PSNR, MSE, and SSIM against the original image. This patchwise approach enables detailed compression by reconstructing sections individually, while `montage` seamlessly assembles the patches to recompose the total image with minimal artifacts.

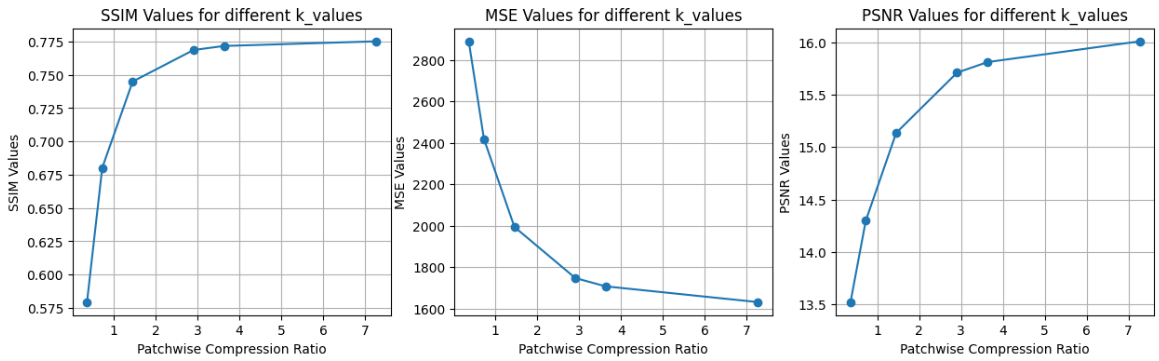


Patchwise compressed and reconstructed cameraman image

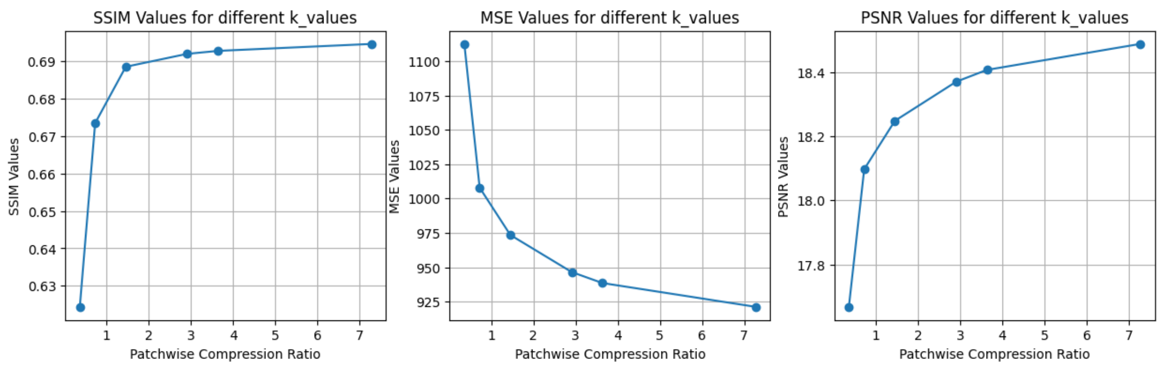


Patchwise compressed and reconstructed face image

Fig. 4. Patch-wise compressed and reconstructed images for different number of retained singular values



Metrics for patch-wise compression of cameraman image



Metrics for patch-wise compression of face image

Fig. 5. Patch-wise compressed and reconstructed images for different number of retained singular values

k	Face Image				Cameraman Image			
	CR	PSNR	SSIM	MSE	CR	PSNR	SSIM	MSE
5	0.3635	17.6678	0.6244	1112.5071	0.3635	13.5231	0.5793	2889.1329
10	0.7270	18.0967	0.6735	1007.8813	0.7270	14.2976	0.6802	2417.2244
20	1.4541	18.2476	0.6885	973.4563	1.4541	15.1352	0.7450	1993.2215
40	2.9082	18.3696	0.6920	946.4906	2.9082	15.7088	0.7687	1746.6089
50	3.6352	18.4065	0.6928	938.5002	3.6352	15.8096	0.7717	1706.5537
100	7.2704	18.4875	0.6946	921.1590	7.2704	16.0072	0.7752	1630.6340

Table 2. Patch-wise compression-reconstruction of face and cameraman images

Space and Time Complexity: The time and space complexity of *normal SVD* is $O(mn^2)$, where $m \times n$ is the size of the matrix being decomposed. This involves computing the singular values and vectors of the entire matrix, which can be computationally expensive for large datasets. In contrast, *patchwise SVD* performs SVD on smaller blocks (patches) of the image or data matrix, which reduces both time and space complexity. For a block of size $k \times k$, the complexity is $O(k^3)$ per block. Thus, patchwise SVD is more efficient, especially when the data is sparse or has a block structure, allowing for faster computations with reduced memory usage.

4. Adaptive patch-wise compression

Need for Adaptive Patchwise Compression: Adaptive patchwise compression is essential for efficiently compressing images by adjusting the block size and the number of singular values used based on the local complexity of the image. Simple regions, such as uniform or low-variance areas, can be compressed more aggressively, reducing both storage and computational costs. On the other hand, complex regions with high variance (e.g., edges or textures) require greater fidelity, so smaller block sizes and more singular values are preserved. This approach ensures that important image details are maintained while minimizing the overall data size.

Algorithm 1 Adaptive Block Size SVD Compression

Input: Image I, Initial block size $B = 16$, Threshold $\tau = 0.1$
Convert the image I to grayscale and normalize it to the range $[0,1]$
Get the dimensions of the image: $height, width = I.shape$
Initialize an empty matrix $compressed\_image$ of size $(height, width)$
for $i = 0$ **to** $height$, step B **do**
    for $j = 0$ **to** $width$, step B **do**
        Extract the block $block = I[i : i+B, j : j+B]$
        Compute the variance $\text{var}(block)$
        if $\text{var}(block) > \tau$ **then**
            Set $block\_size = B$ ▷ High complexity, use smaller block
            Set $k = B//2$ ▷ Higher rank for detailed representation
        else
            Set $block\_size = B$ ▷ Low complexity, use larger block
            Set $k = B//3$ ▷ Lower rank for compression
        end if
        Apply Singular Value Decomposition (SVD) on the block:
        $U, \Sigma, Vt = \text{SVD}(block)$
        Keep the top-k singular values: $U_k = U[:, :k]$, $\Sigma_k = \Sigma[:k]$, $Vt_k = Vt[:k, :]$
        Reconstruct the compressed block: $compressed\_block = U_k \cdot \Sigma_k \cdot Vt_k$
        Store the compressed block in the corresponding location of $compressed\_image$:
        $compressed\_image[i : i + block\_size, j : j + block\_size] = compressed\_block$
    end for
end for
Scale $compressed\_image$ back to 8-bit format: $compressed\_image = \text{clip}(compressed\_image \times 255, 0, 255)$
Output: Compressed image

Time Complexity: The *adaptive patch-wise compression algorithm* involves dividing the image into smaller blocks, performing Singular Value Decomposition (SVD) on each block, and adjusting block sizes based on the variance of each block. The time complexity of the algorithm is dominated by the SVD operation, which for a block of size $k \times k$ takes $O(k^3)$ time. Since the image is divided into $\frac{h}{B} \times \frac{w}{B}$ blocks (where h and w are the image dimensions, and B is the block size), the overall time complexity is $O\left(\frac{h}{B} \times \frac{w}{B} \times k^3\right)$, where k is the rank used in SVD, which depends on the complexity of each block.

Table 3. Compression of face image with varying block sizes and thresholds

Block Size	Variance Threshold	PSNR (in dB)	SSIM	MSE
4	0.0100	35.6314	0.9665	17.7803
	0.0500	33.3163	0.9592	30.3003
	0.1000	32.8964	0.9582	33.3765
	0.2000	32.8964	0.9582	33.3765
8	0.0100	38.8792	0.9773	8.4170
	0.0500	35.1922	0.9673	19.6727
	0.1000	34.2089	0.9649	24.6713
	0.2000	34.2089	0.9649	24.6713
16	0.0100	41.9770	0.9878	4.1245
	0.0500	39.2620	0.9827	7.7070
	0.1000	37.8695	0.9804	10.6201
	0.2000	37.5404	0.9795	11.4561
32	0.0100	45.6251	0.9919	1.7806
	0.0500	41.0673	0.9845	5.0857
	0.1000	39.8254	0.9792	6.7692
	0.2000	39.8254	0.9792	6.7692

Table 4. Compression of cameraman image with varying block sizes and thresholds

Block Size	Variance Threshold	PSNR (in dB)	SSIM	MSE
4	0.0100	33.1969	0.9539	31.1448
	0.0500	28.7251	0.9298	87.2110
	0.1000	27.2966	0.9213	121.1772
	0.2000	27.2136	0.9206	123.5145
8	0.0100	34.7812	0.9643	21.6252
	0.0500	30.2617	0.9413	61.2225
	0.1000	28.2310	0.9273	97.7187
	0.2000	28.2197	0.9272	97.9745
16	0.0100	36.4612	0.9747	14.6879
	0.0500	33.1292	0.9579	31.6343
	0.1000	31.6063	0.9511	44.9212
	0.2000	31.1087	0.9482	50.3741
32	0.0100	38.8362	0.9803	8.5008
	0.0500	35.2991	0.9564	19.1943
	0.1000	32.1902	0.9428	39.2702
	0.2000	32.1902	0.9428	39.2702

Fig. 6. Adaptive patch-wise compression-reconstruction of cameraman images

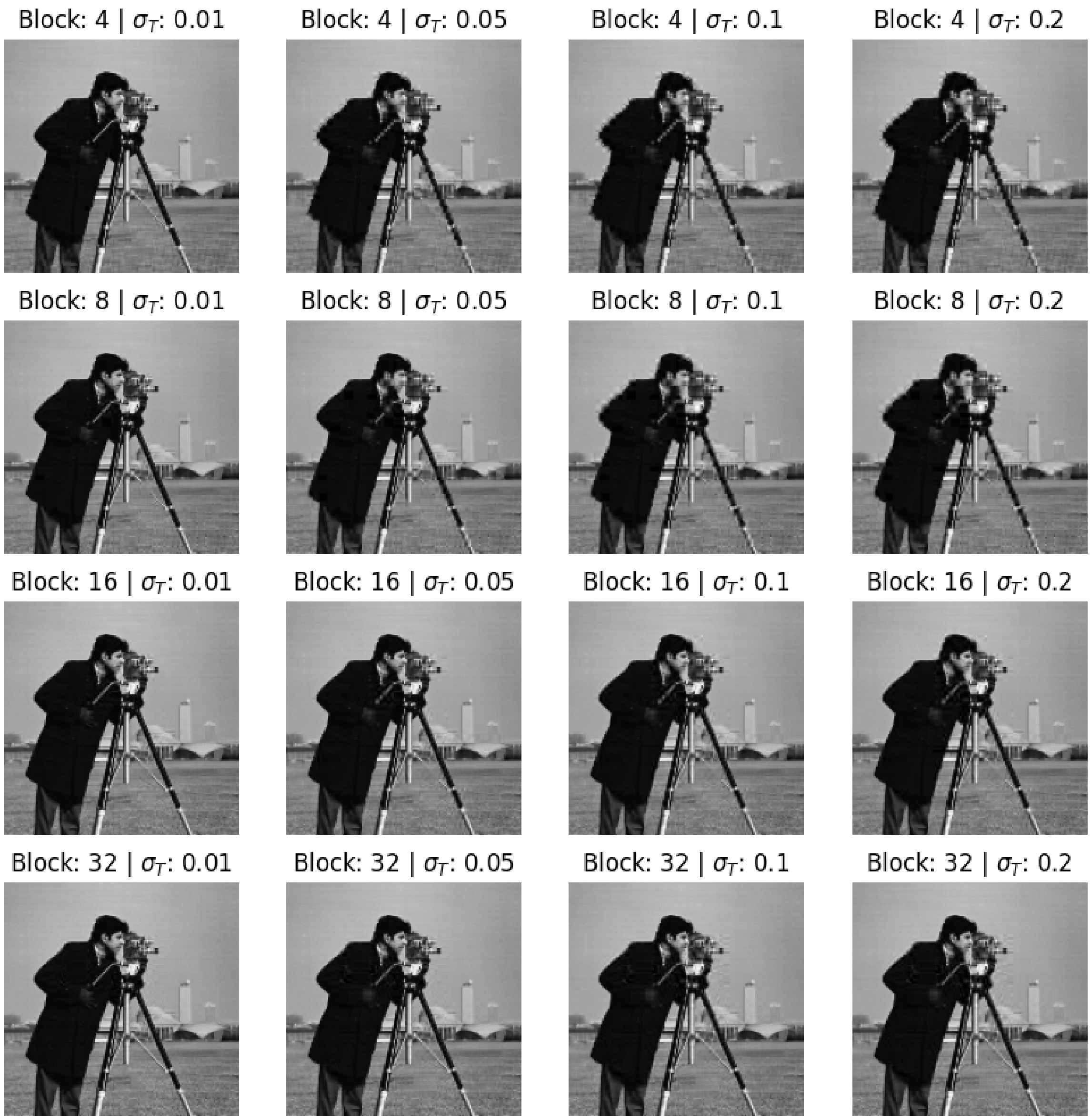


Fig. 7. Adaptive patch-wise compression-reconstruction of cameraman images

5. Observation and conclusion

In this experiment, image compression using Singular Value Decomposition (SVD) was applied to grayscale images (Cameraman and Face). Compression quality was assessed based on Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Mean Squared Error (MSE) for varying singular values (k). As k decreased, PSNR and SSIM dropped, and MSE increased. The Face image showed lower PSNR and SSIM than the Cameraman, indicating higher sensitivity to compression. Patch-wise SVD improved compression efficiency by reducing error in less detailed areas, maintaining quality in complex regions. It also provided better compression ratios (CR) compared to regular SVD.