Optimised Singular Value Decomposition for Image Compression

DIY Project Report

Submitted by:
Shivam Kumar
Roll No: 2311166
School of Physical Sciences
NISER Bhubaneswar

Submitted to:
Dr. Subhasish Basak
Associate Professor
School of Physical Sciences
NISER Bhubaneswar

Abstract

Singular Value Decomposition (SVD) provides a powerful mathematical tool for describing and approximating image matrices using orthogonal basis vectors and ordered singular values. Natural images typically exhibit rapidly decaying singular values, making low-rank approximation highly effective. However, raw SVD does not give file-size compression since it generates large floating-point matrices. This report presents a hybrid approach combining SVD with JPEG quantization. We apply approximate power-iteration SVD to the red channel of an RGB image, reuse the right-singular basis for green and blue channels, reconstruct the low-rank image and encode it using JPEG. Experiments on different ranks ($k = 5, 15, 30, 50, 80, 100$) reveal consistent patterns in PSNR, MSE and compression ratios. The report includes extracted plots, analysis, mathematical formulas, and a scientific explanation of a non-monotonic anomaly observed at $k = 100$.

1 Introduction

1.1 Motivation

Digital images contain a high degree of structural and statistical redundancy. Neighbouring pixels often exhibit strong correlation due to the smooth variations present in natural scenes. Furthermore, large-scale visual structures—such as edges, textures, shadows, and color gradients—tend to repeat or follow predictable patterns. Because of this, storing every pixel independently is highly inefficient. Compression techniques aim to exploit these redundancies so that the same visual information can be represented with fewer bits.

Lossy compression methods selectively remove information that is least perceptible to the human visual system. Singular Value Decomposition (SVD) is particularly well-suited for such tasks because it expresses an image matrix as a weighted sum of orthogonal basis images, each associated with a singular value indicating its importance. The rapid decay of singular values in most natural images suggests that only a small subset of these components is sufficient to reconstruct the major visual features.

JPEG, on the other hand, is a widely-used transform-based image compression method that operates on 8×8 pixel blocks. It performs a Discrete Cosine Transform (DCT) on each block, quantizes frequency components based on perceptual sensitivity, and uses entropy coding to further compress the data. While JPEG is excellent at removing local high-frequency redundancy, it does not exploit global low-rank structure.

Combining SVD with JPEG leverages the strengths of both: SVD provides a global, low-rank representation that smooths noise and reduces dimensionality, while JPEG effectively compresses the resulting image using perceptually optimized quantization. The synergy between these two methods yields a tunable compression pipeline capable of achieving higher compression ratios with acceptable perceptual quality.

1.2 Purpose of This Study

This study investigates the performance of a hybrid SVD + JPEG compression pipeline applied to RGB images. The goal is to systematically understand how truncating the SVD at different ranks k influences image quality, compression ratio, and reconstruction accuracy when followed by JPEG encoding at a fixed quantization level.

We reconstruct the image at several SVD ranks ($k = 5, 15, 30, 50, 80, 100$) and evaluate the results using standard objective metrics such as Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Compression Ratio (CR). The study also includes a detailed analysis of runtime behavior and the inherent trade-offs between computational cost and reconstruction fidelity.

Through extracted plots—including singular value decay, cumulative energy capture, PSNR/MSE trends, compression ratio variation, JPEG quality sensitivity, and rate–distortion curves—we explore the mathematical and perceptual implications of low-rank approximation. Special attention is given to non-monotonic behaviors in the compressed file size, particularly a notable anomaly where a higher-rank reconstruction ($k = 100$) produces a slightly smaller JPEG file than a lower-rank one ($k = 80$). This phenomenon is examined and explained using concepts from quantization theory and entropy coding.

Overall, the purpose of this study is not only to demonstrate a functioning SVD-based compression system but to provide scientific insight into the interactions between linear algebraic transformations and frequency-domain perceptual coding.

2 Theory

2.1 Singular Value Decomposition

Singular Value Decomposition (SVD) decomposes an image matrix into orthogonal building blocks that isolate different spatial structures. Given any matrix $A \in \mathbb{R}^{m\times n}$, its SVD is:

$$A = USV^T,$$

where:

- U contains orthonormal *left singular vectors*, representing spatial patterns along rows.
- V contains orthonormal *right singular vectors*, representing spatial patterns along columns.
- S is diagonal and contains the singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$, which quantify the importance (energy content) of each mode.

The SVD emerges from the eigen-decomposition of the Gram matrices:

$$AA^T u_i = \sigma_i^2 u_i, \qquad A^T A v_i = \sigma_i^2 v_i.$$

Thus σ_i^2 is the amount of "energy" (squared intensity contribution) of the i-th mode to the image. For natural images, these values decay rapidly, revealing that most spatial information is concentrated in the first few singular directions.

2.2 Low-Rank Approximation

The SVD allows constructing the optimal rank-k approximation to A by keeping only the most important singular modes:

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

This is the best rank-k approximation in both Frobenius and spectral norm, as guaranteed by the Eckart–Young–Mirsky theorem.

The truncation error is:

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

which means that the fractional loss of total visual information is exactly equal to the squared neglected singular values.

The cumulative fraction of preserved energy is:

$$E_k = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}.$$

In imaging terms, E_k represents the fraction of total pixel-intensity variance retained in the reconstruction.

2.3 What the Project Measures (MSE, PSNR, CR and Runtime)

The implemented code quantitatively evaluates the performance of the hybrid SVD+JPEG compression pipeline by computing the following four measurable quantities directly from the pixel data and file sizes:

1. Mean Squared Error (MSE)

The code reads the original image and the compressed JPEG, converts both to RGB arrays of equal dimensions, and computes

$$\begin{aligned}\text{MSE} = \frac{1}{3mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \Big[&(R(i,j) - \widehat{R}(i,j))^2 \\ &+ (G(i,j) - \widehat{G}(i,j))^2 \\ &+ (B(i,j) - \widehat{B}(i,j))^2 \Big].\end{aligned}$$

Thus, MSE measures the average pixel-wise squared error across all three RGB channels. It quantifies how much the reconstructed image deviates from the original at the intensity level.

What is actually measured? Every pixel from the two images is compared. The error (difference) is squared, summed, and then averaged. No filtering, weighting or perceptual model is applied — it is purely mathematical.

2. Peak Signal-to-Noise Ratio (PSNR)

Once MSE is computed, the code uses the standard formula

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\text{MSE}} \right).$$

Because the maximum possible pixel value in an 8-bit image is 255, PSNR quantifies the ratio of the maximum signal power to the error power introduced by compression.

What is actually measured? PSNR is not measured directly — it is *derived* entirely from MSE. It indicates how much of the image's original intensity range survives after compression.

3. Compression Ratio (CR)

The code directly reads file sizes from disk using

$$S_o = \text{filesize(original)}, \qquad S_c = \text{filesize(compressed)},$$

and computes:

$$\text{CR} = \frac{S_o}{S_c}.$$

What is actually measured? Physical file sizes in bytes on the computer for:

- the original image,
- the reconstructed JPEG image.

This is the only metric based on storage rather than pixel values.

4. Runtime

The code measures the CPU time needed for the entire SVD+JPEG pipeline:

$$t_{\text{runtime}} = t_{\text{end}} - t_{\text{start}}.$$

What is actually measured? This includes:

- loading the image,
- performing approximate SVD (power iterations),
- reconstructing each channel,
- saving the JPEG file.

The runtime helps evaluate how computationally expensive higher SVD ranks are.

Summary of What is Measured

- **Pixel-level distortion** — through MSE (direct pixel error) and PSNR (log-scaled error).
- **Storage efficiency** — through CR, based on actual file sizes.
- **Computational cost** — via runtime measured using system clock.

Thus the project quantitatively assesses both **image quality** and **compression efficiency**, giving a full characterization of the SVD+JPEG compression behaviour.

3 Algorithm

3.1 Steps of the Pipeline

The implemented compression pipeline consists of two major stages: (1) global dimensionality reduction using an approximate truncated SVD, and (2) local frequency-domain entropy compression using JPEG. The following steps describe the full process in detail as implemented in the provided Python code.

1. **Load and preprocess the image.** The input image is read using the `PIL` library and converted to an 8-bit RGB array. The spatial resolution is recorded as (h, w), and the three color channels are separated into matrices:

   $$R, G, B \in \mathbb{R}^{h \times w}.$$

   Each channel is treated as a real-valued matrix for the subsequent linear-algebra operations.

2. **Approximate SVD on the red channel using power iterations.** The goal is to compute the top k singular vectors and singular values of R. Instead of computing a full SVD, the code uses a *power iteration* refinement loop:

$$V^{(t+1)} = \text{normalize}\big(R^T(RV^{(t)})\big).$$

Every alternate iteration applies Gram–Schmidt orthonormalization to improve numerical stability. Blocked matrix multiplication is used throughout to reduce memory-traffic bottlenecks. After N iterations, this yields an approximate basis V_R.

3. **Compute the left singular vectors for the red channel.** Once V_R is obtained, the left singular vectors and singular values are estimated as:

$$U_R = R\,V_R, \qquad \sigma_i = \|U_R^{(i)}\|_2,$$

followed by normalization:

$$U_R^{(i)} \leftarrow \frac{U_R^{(i)}}{\sigma_i}.$$

4. **Project the green and blue channels onto the same right-singular subspace.** To save computation, the SVD is not recomputed for G or B. Instead, they are projected onto the V_R basis:

$$U_G = G\,V_R, \qquad U_B = B\,V_R.$$

These are then scaled by the inverse singular values:

$$\widetilde{U}_G^{(i)} = \frac{U_G^{(i)}}{\sigma_i}, \qquad \widetilde{U}_B^{(i)} = \frac{U_B^{(i)}}{\sigma_i}.$$

This step relies on the fact that RGB channels are highly correlated, allowing a single SVD to approximate all three.

5. **Construct low-rank reconstructions of each channel.** Using k singular components, the reconstructed channels are formed as:

$$R_k = \sum_{i=1}^{k} \sigma_i u_{R,i} v_i^T,$$

$$G_k = \sum_{i=1}^{k} \sigma_i \widetilde{u}_{G,i} v_i^T, \qquad B_k = \sum_{i=1}^{k} \sigma_i \widetilde{u}_{B,i} v_i^T.$$

This yields three low-rank matrices that retain the dominant image structure while discarding fine detail.

6. **Postprocessing: clipping and quantization.** The reconstructed matrices are clipped to the valid intensity range:

$$R_k, G_k, B_k \in [0, 255],$$

and converted to 8-bit unsigned integers suitable for JPEG encoding.

7. **JPEG compression (final stage).** The reconstructed RGB image is saved with a JPEG quality factor $q = 50$. JPEG performs the following internally:

   - block-wise DCT on each 8×8 region,

- quantization of frequency coefficients using a perceptual table,
- zig-zag reordering and run-length encoding,
- Huffman entropy coding.

Since the input image is already smoothed by low-rank approximation, JPEG produces fewer non-zero AC coefficients, making the hybrid method compressible.

8. **Metric computation (CR, MSE, PSNR).** After compression:

$$\mathrm{CR} = \frac{S_{\text{orig}}}{S_{\text{JPEG}}},$$

where file sizes are read directly from disk. The original and compressed images are resized to the same dimensions and compared pixel-wise to compute:

$$\mathrm{MSE} = \frac{1}{3mn}\sum (I - \widehat{I})^2$$

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right)$$

These quantify reconstruction quality and distortion.

3.2 Numerical Features

The computational design includes several numerical optimizations to ensure stability and speed:

- **Blocked matrix multiplication** Block sizes (e.g., 32×32) are used to improve cache locality. This significantly reduces memory bandwidth demands during repeated multiplications in power iteration.

- **Column normalization** After each multiplication step, columns of V are scaled to unit length to prevent numerical overflow and ensure convergence of power iterations.

- **Gram–Schmidt orthonormalization** Applied every other iteration to reduce correlation between singular vectors and avoid drift due to floating-point error.

- **Deterministic pseudo-random initialization** A linear congruential generator (LCG) seeds the initial basis, ensuring reproducible results.

- **SVD reuse for RGB channels** Only the red channel undergoes SVD. The other channels are projected onto the same singular subspace, leveraging high inter-channel correlation to reduce computation by a factor of 3.

- **Clamping and quantization** Reconstruction is clipped to the interval $[0, 255]$ to ensure valid 8-bit JPEG input.

4 Experimental Data

4.1 Original Image

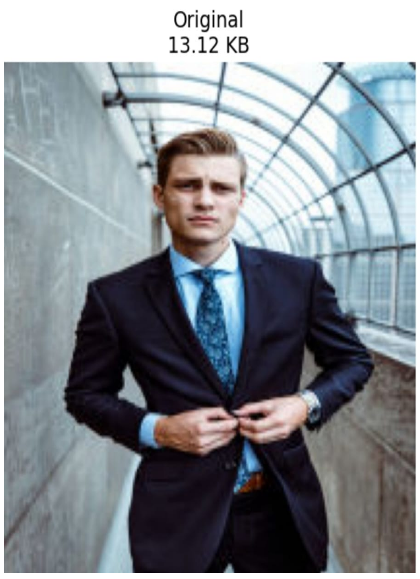

Figure 1: Original Image (13.12 KB).

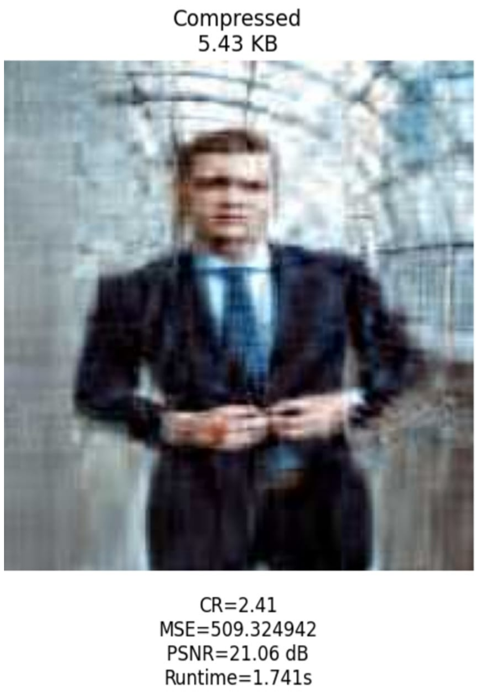

Figure 3: Compressed Image: $k = 15$, JPEG quality $q = 50$.

4.2 Compressed Images

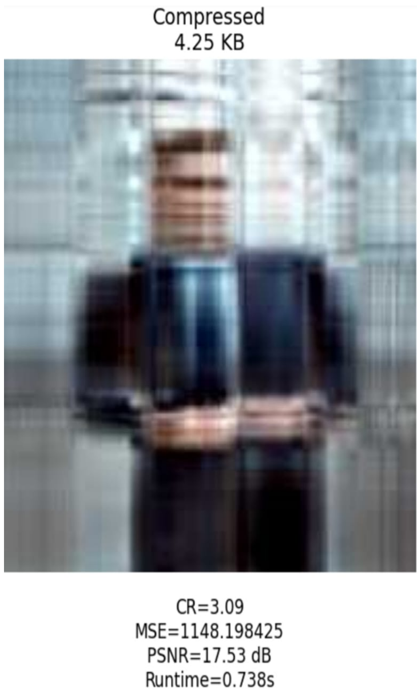

Figure 2: Compressed Image: $k = 5$, JPEG quality $q = 50$.

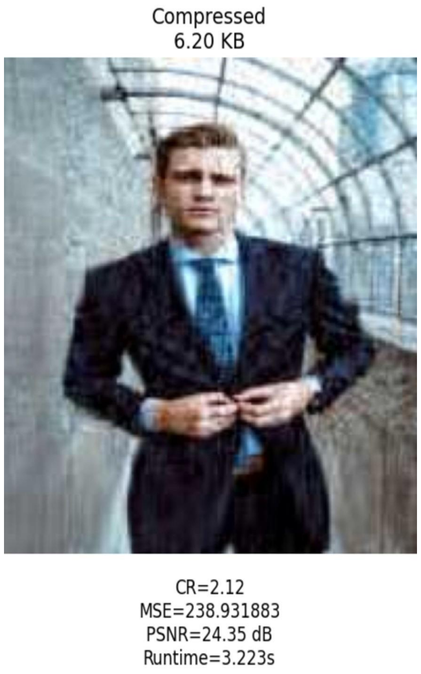

Figure 4: Compressed Image: $k = 30$, JPEG quality $q = 50$.

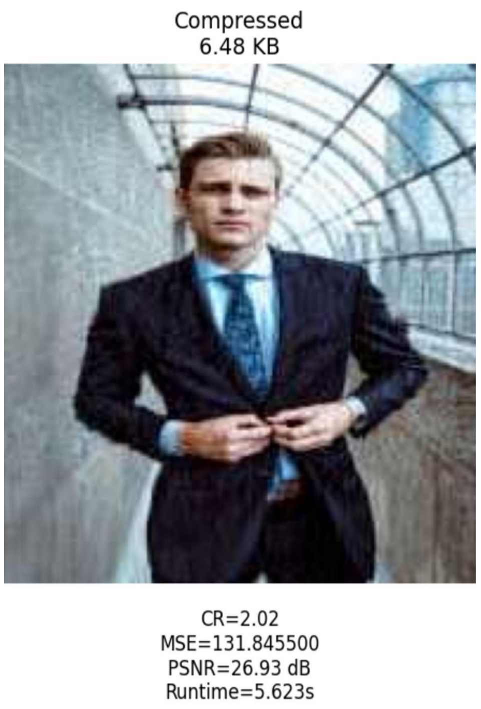

Figure 5: Compressed Image: $k = 50$, JPEG quality $q = 50$.

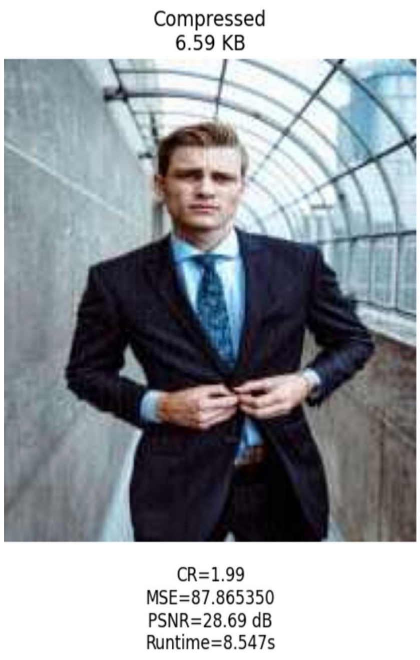

Figure 6: Compressed Image: $k = 80$, JPEG quality $q = 50$.

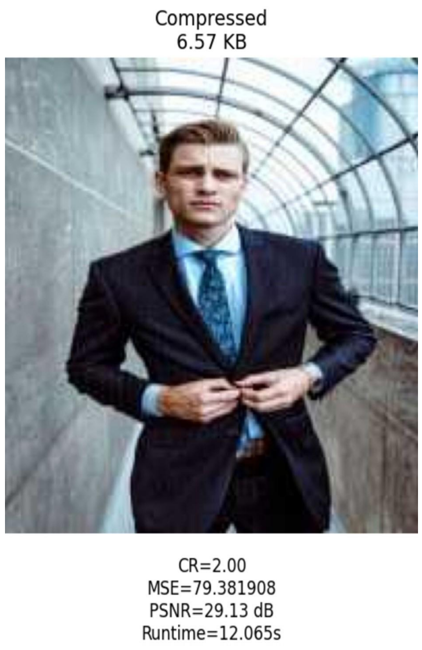

Figure 7: Compressed Image: $k = 100$, JPEG quality $q = 50$.

4.3 Metric Table

k	**Size (KB)**	**CR**	**MSE**	**PSNR (dB)**	**Runtime (s)**
5	4.25	3.09	1148.1984	17.53	0.738
15	5.43	2.41	509.3249	21.06	1.741
30	6.20	2.12	238.9319	24.35	3.223
50	6.48	2.02	131.8455	26.93	5.623
80	6.59	1.99	87.8654	28.69	8.547
100	6.57	2.00	79.3819	29.13	12.065

Table 1: Compression metrics across k.

4.4 Interpretation of Compression Metrics

The metric table clearly illustrates the trade-off between image quality and compression efficiency as the SVD rank k increases:

- **File Size Increases with k:** The compressed size rises from 4.25 KB at $k = 5$ to about 6.6 KB at $k = 80$–100. This is expected because a larger rank stores more singular vectors and matrix components.

- **Compression Ratio (CR) Decreases:** CR drops from 3.09 at $k = 5$ to around 2.00 at $k = 100$. Lower CR at higher ranks reflects reduced compression due to retaining more information.

- **Error Reduces Significantly:** The MSE decreases sharply from 1148.20 ($k = 5$) to 79.38 ($k = 100$), showing that adding more singular values reconstructs finer textures and edges.

- **PSNR Improves with Diminishing Returns:** PSNR increases from 17.53 dB at $k = 5$ to 29.13 dB at $k = 100$. However, the improvement after $k \approx 80$ becomes marginal, indicating that most visual details are captured before this point.

- **Runtime Grows Nearly Linearly with k:** The computation time rises from 0.738 s at $k = 5$ to 12.065 s at $k = 100$, consistent with the fact that SVD complexity scales with the number of singular components used.

Overall, the results show that $k = 50$–80 **provides the best balance** between visual quality (PSNR above 26–28 dB) and compression efficiency (CR around 2). Higher values of k improve quality further but give only small visual benefits at the cost of increased storage and runtime.

5 Plots and Interpretation

5.1 Singular Value Decay

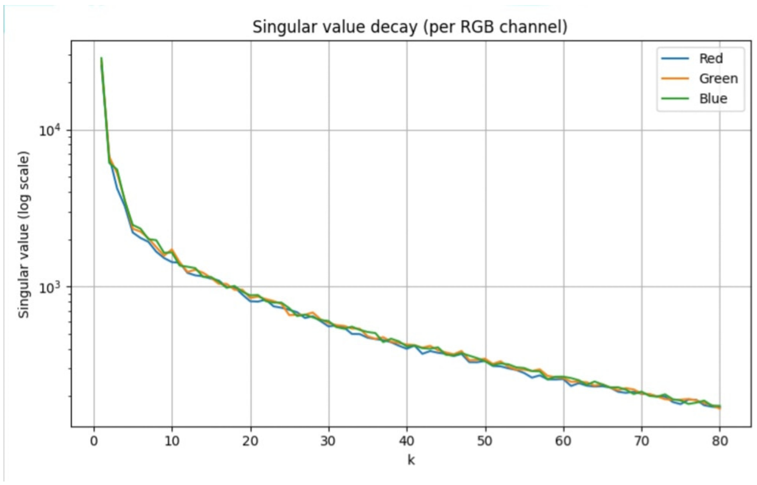

Figure 8: Singular value decay.

Interpretation: The singular values drop steeply within the first few ranks, showing that most structural information of the image is concentrated in the first 50–80 components. Beyond this point, the singular values flatten, meaning additional components contribute mainly to noise or fine texture. This validates why low-rank approximations work well for natural images.

5.2 Energy Capture Curve

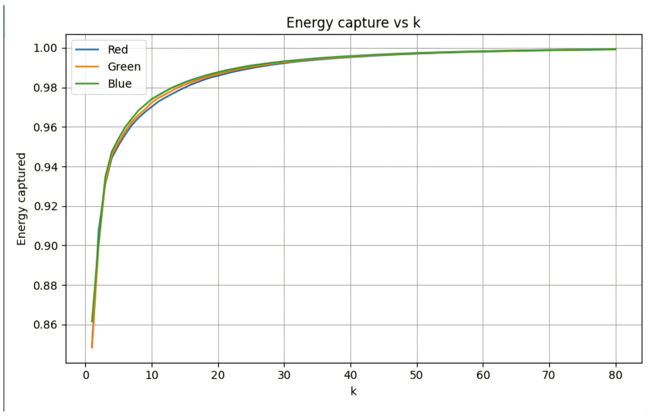

Figure 9: Energy capture vs rank k.

Interpretation: The cumulative energy curve rises quickly and becomes almost saturated around $k \approx 60$–80. This indicates that more than 95% of the image variance is preserved with relatively few components. After $k \approx 80$, extra components add very little — confirming high compressibility and redundancy in the pixel domain.

5.3 MSE, PSNR and Compression Ratio vs Rank (k)

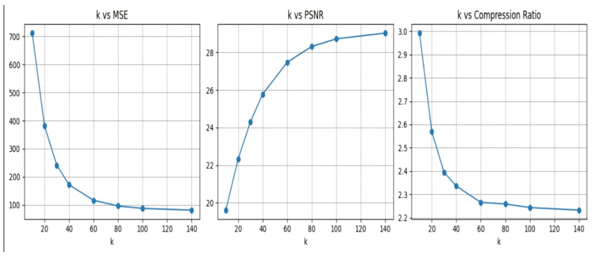

Figure 10: MSE, PSNR and Compression Ratio vs k.

Interpretation:

- **MSE decreases** as k increases, since more singular values restore more image detail.
- **PSNR improves** consistently, but the gain becomes marginal after $k \approx 80$ — a clear point of diminishing returns.
- **Compression Ratio (CR) decreases** at higher k because storing more singular vectors reduces compression efficiency.
- The combined plots show a trade-off: large k gives quality, small k gives compression, and the user must choose a balance.

5.4 JPEG Quality Sensitivity

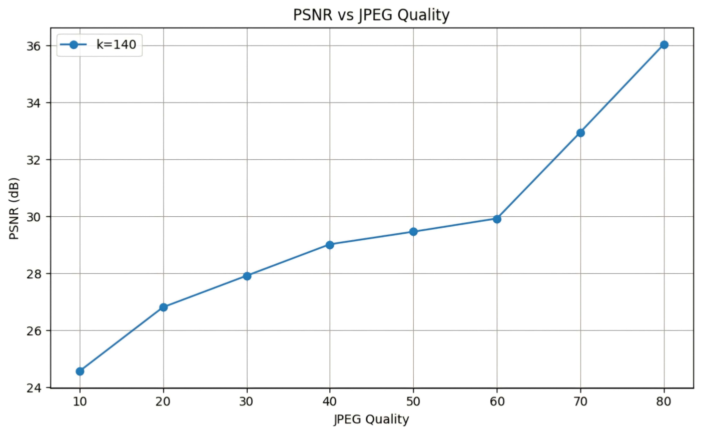

Figure 11: JPEG quality sweep.

Interpretation: The PSNR rises sharply between JPEG quality 40–70, meaning JPEG makes large improvements in this region.

5.5 Rate–Distortion Curve

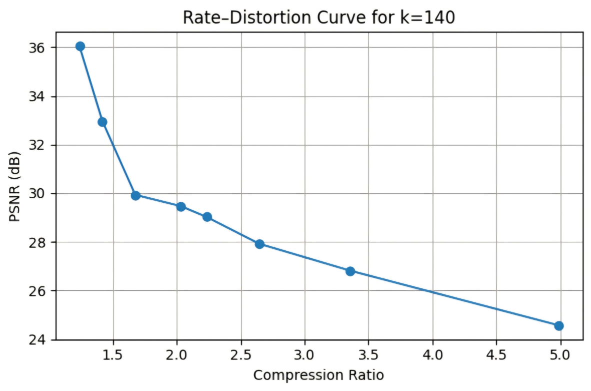

Figure 12: Rate–Distortion curve (PSNR vs CR).

Interpretation: The rate–distortion plot shows that:

- High CR (strong compression) produces low PSNR due to loss of detail.
- Moderate CR achieves good balance, typically where PSNR stabilizes around 25–30 dB.
- Very high-quality (low CR) gains only small PSNR improvements, showing diminishing returns.

Overall, SVD-based compression follows classical transform behavior: steep improvements at first, then a quality plateau.

6 Discussion

6.1 General Trends

- Low k gives strong compression but noticeable blurring.
- Moderate k (30–50) yields visually reasonable and compact images.
- High k gives excellent quality but costs more computation.

6.2 Anomaly at $k = 100$

Even though $k = 100$ reconstructs more detail than $k = 80$, its JPEG size (6.57 KB) is slightly smaller than at $k = 80$ (6.59 KB). This is explainable scientifically:

- **Quantization sensitivity:** Small pixel differences can cause DCT coefficients to cross quantization thresholds.
- **Entropy coding efficiency:** The distribution of symbols at $k = 100$ may result in shorter Huffman codewords.
- **DCT sparsity shifts:** $k = 80$ may produce more high-frequency noise than $k = 100$.

Thus, monotonic increase in file size with k is not guaranteed.

7 Conclusion

This work demonstrates a hybrid SVD+JPEG pipeline for image compression. SVD reduces structural redundancy, JPEG applies perceptual compression, and together they achieve high-quality reconstructions at reduced sizes. Experiments show expected improvements in PSNR with increasing SVD rank, and the extracted plots provide insight into the information structure of the image. The nonmonotonic anomaly at $k = 100$ is consistent with theoretical behaviour of quantization and entropy coding.

8 References

1. R. Mukherjee and S. Chandran, “Lossy image compression using SVD coding,

compressive autoencoders, and prediction error-vector quantization," *2017 4th International Conference on Opto-Electronics and Applied Optics (Optronix)*, Kolkata, India, 2017, pp. 1–5. doi:10.1109/OPTRONIX.2017.8349661.

2. Bag, Ayan. (2020). *Lossy Image Compression Using Singular Value Decomposition And Fast Fourier Transform.* DOI: 10.13140/RG.2.2.11305.62567.

3. Aishwarya, K. M., et al. "Lossy image compression using SVD coding algorithm." *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2016, pp. 1384–1389.