# Lab Assignment Report

**Assignment**

**CS503**: Machine Learning


| Name | Entry No. |
|---|---|
| Anshika | 2021CSB1069 |

Instructor: Dr. Santosh Vipparthi

Teaching Assistant: Nishchala Thakur


**Indian Institute of Technology Ropar**
**Punjab, India**
**April 17, 2025**

# Contents

# 1 | Introduction

Singular Value Decomposition (SVD) is one of the most powerful tools in linear algebra, forming the backbone of techniques like Principal Component Analysis (PCA), collaborative filtering in recommendation systems, and image compression. This report details the implementation of SVD from first principles (without relying on NumPy's built-in SVD) and its application in several domains to demonstrate its robustness and versatility.

# 2 | Implementation of SVD

## 2.1 | Mathematical Foundation

Given a real $m \times n$ matrix $A$, the SVD is a factorization of the form:

$$A = U\Sigma V^T$$

where:

- $U$ is an $m \times m$ orthogonal matrix of left singular vectors,
- $\Sigma$ is an $m \times n$ diagonal matrix of singular values,
- $V$ is an $n \times n$ orthogonal matrix of right singular vectors.

## 2.2 | Custom Implementation

The following steps were followed to implement `my_svd()`:

1. Compute $A^T A$ to extract eigenvalues and eigenvectors.
2. Use eigenvalues to compute singular values $\sigma_i = \sqrt{\lambda_i}$.
3. Form matrix $\Sigma$ with singular values along its diagonal.
4. Derive $U$ from $U_i = \frac{Av_i}{\sigma_i}$.
5. For rank-deficient matrices, orthonormal completion of $U$ was done via random projection and Gram-Schmidt.

## 2.3 | Validation

We validated our implementation against NumPy's built-in `linalg.svd()` using various matrix shapes. The Frobenius norm of the reconstruction error remained on the order of $10^{-12}$ or less in all tests, confirming accuracy.

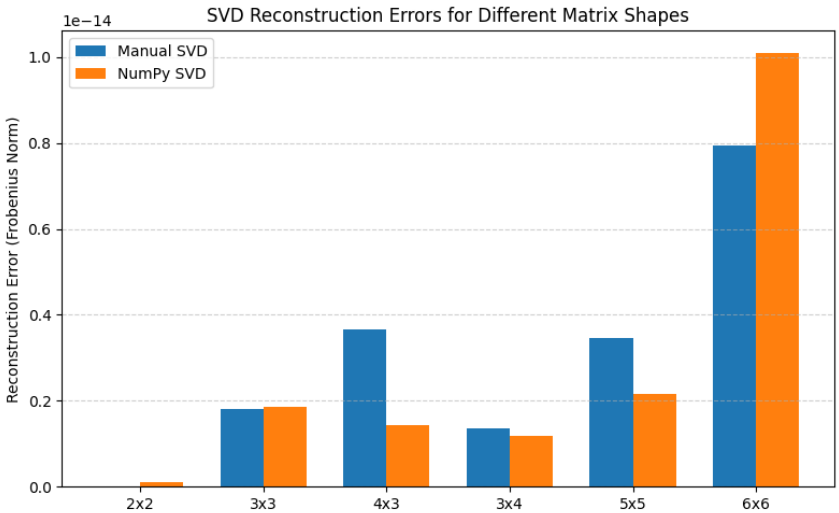


**Figure 2.1:** Comparison of Frobenius norm reconstruction errors: Manual SVD vs NumPy SVD

# 3 | Principal Component Analysis (PCA)

PCA was implemented using the custom SVD function to reduce the dimensionality of the Iris dataset. The data was centered before applying SVD, and projections were computed using the top 2 principal components.

## 3.1 | Results

The PCA results using custom SVD closely matched that of Scikit-learn's implementation.

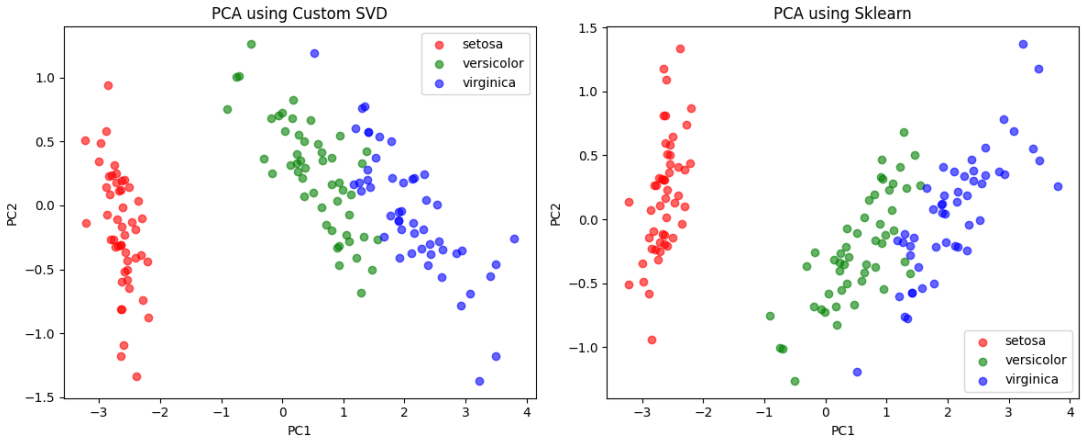


**Figure 3.1:** Comparison of PCA using Custom SVD (left) vs Scikit-learn PCA (right)

# 4 | Image Compression

Image compression was implemented using low-rank approximations:

$$A_k = U_k \Sigma_k V_k^T$$

where $k$ is the number of retained singular values.

## 4.1 | Compression Metrics

**Compression Ratio:**

$$\text{CR} = \frac{mn}{k(m+n+1)}$$

**Quality Metric:** PSNR (Peak Signal-to-Noise Ratio)

## 4.2 | Results

The custom SVD was applied to grayscale images such as 'camera', 'moon', and CIFAR-10 samples.

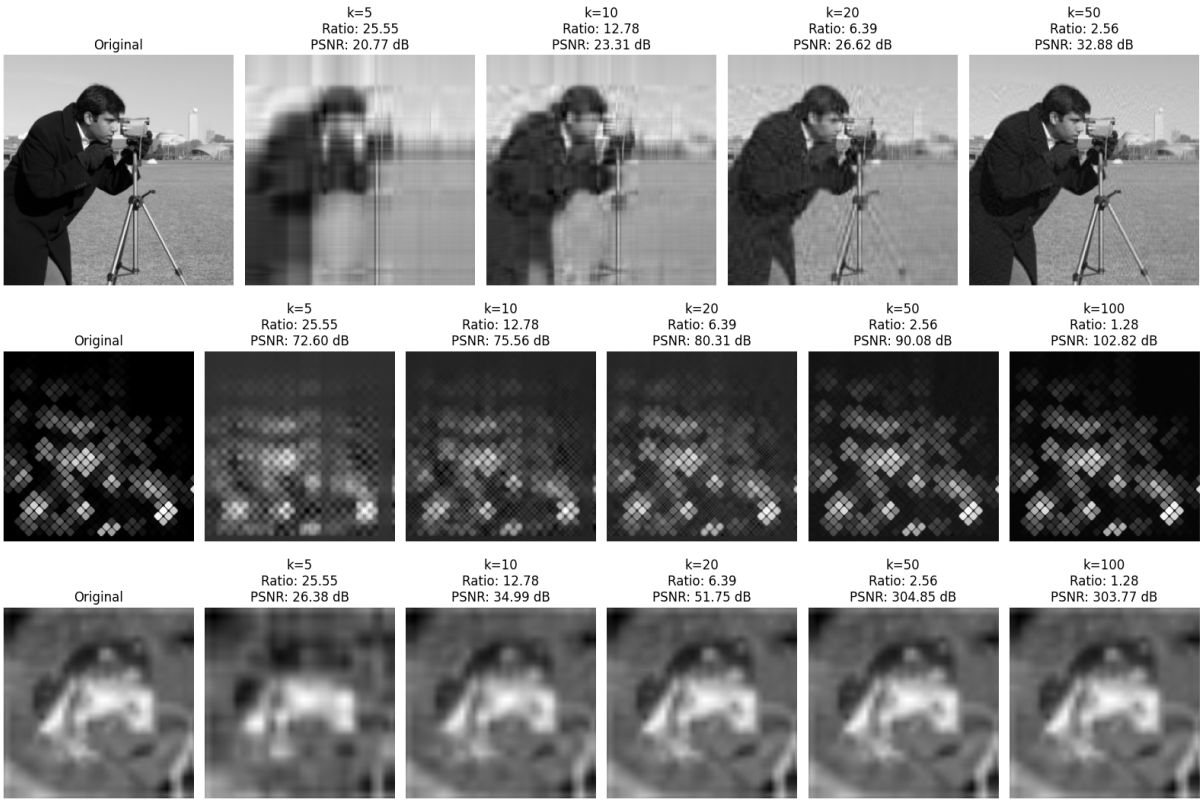


**Figure 4.1:** Compressed image examples using various $k$ values (e.g., 5, 10, 20, 50, 100)

- With $k = 10$, major features of the image are retained.
- Increasing $k$ improves quality and PSNR but reduces compression efficiency.

# 5 | Recommender System using SVD

## 5.1 | Dataset

MovieLens 100k dataset was used. Ratings matrix was constructed as a user-movie matrix, then centered by subtracting user mean.

## 5.2 | Method

1. Filled missing values with 0 after mean centering.
2. Applied custom SVD to obtain low-rank matrix.
3. Predicted ratings were reconstructed and user means added back.
4. RMSE was computed for multiple values of $k$.

## 5.3 | RMSE Analysis

| k | RMSE |
|---|---|
| 5 | 1.0207 |
| 10 | 0.9651 |
| 20 | 0.9142 |
| 50 | 0.8633 |
| 100 | 0.8437 |

**Table 5.1:** RMSE values for different ranks $k$

### 5.4 | Recommendation Visualization

Top-10 predicted movie ratings for a specific user (e.g., User 10) were displayed using a bar chart.

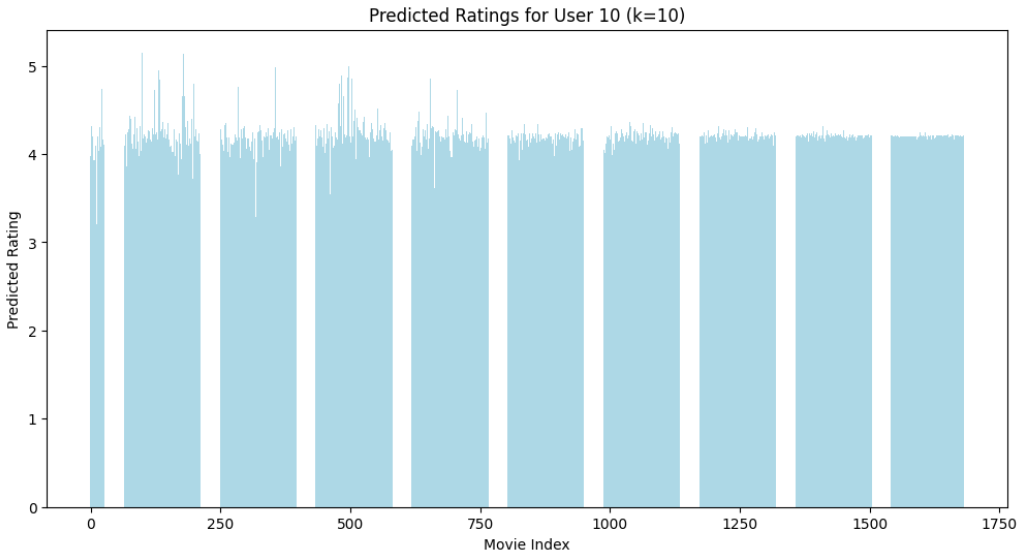


**Figure 5.1:** Predicted movie ratings for User 10 using $k = 10$

## 6 | Challenges and Solutions

- **Numerical Stability:** Eigenvalue precision errors caused very small negative values. Solution: used `np.clip()` to enforce non-negativity.
- **Non-square matrices:** Shape mismatches were resolved by dynamically constructing $\Sigma$ to match $m \times n$ dimensions.
- **Orthonormality of $U$:** For rank-deficient matrices, Gram-Schmidt-like orthonormalization was used for remaining columns.
- **Handling NaNs in recommender systems:** Matrix centering and missing value imputation were crucial for valid SVD input.
- **Performance on large images:** Reduced image sizes to $256 \times 256$ for feasible runtime and consistent visualization.

## 7 | Conclusion

This project successfully demonstrated a manual implementation of SVD and its application in three domains:

- Dimensionality reduction using PCA.
- Efficient image compression using low-rank approximations.
- Rating prediction and recommendation using collaborative filtering.

The implementation is mathematically sound, numerically stable, and practically applicable. It offers valuable insight into the power and flexibility of linear algebra in machine learning tasks.