MNIST DATASET DIMENSION REDUCTION USING SINGULAR VALUE DECOMPOSITION AND TRANSFORMATION TECHNIQUES

Roy Huang[1] *Vincent Cai*[2] *MingWei Xi*[3]

From the University of Iowa 1. Department of Electrical and Computer Engineering,
2. Department of Computer Science, 3. Department of Mathematics

This project investigates the impact of various data transformations on Singular Value Decomposition (SVD) for dimension reduction on the MNIST dataset. SVD is a powerful tool for analyzing high-dimensional data, but its effectiveness can vary based on how the data is preprocessed. By applying seven transformations—centering, random shift, separating each digit, deskewing, median filtering, max pooling, and binary thresholding—this study evaluates how these transformations influence singular value decay and reconstruction accuracy. Results indicate that centering and single-digit sub setting enhance SVD's ability to capture variance while reducing reconstruction error. These findings show the importance of data transformation in improving SVD's performance.

1. INTRODUCTION

A common belief is that most realistic data sets will lay closely to a manifold, a lower dimensional curved surface in high dimensional space. This induced the practice of dimension reduction. Dimension reduction is a technique commonly used to simplify high-dimensional data while retaining essential information. In this project, Singular Value Decomposition (SVD) is applied to the MNIST dataset, which consists of images of hand-drawn digits, to investigate how different transformations affect dimension reduction.

Data transformations play a critical role in pre-processing, influencing how well SVD can capture variance and structure in data. In our project, we will demonstrate how fast singular values decrease and provide reconstruction evaluation and visualization. We applied seven transformations to the MNIST dataset before performing SVD and image reconstructions: centering, random shifting, separating each digit, deskewing, median filtering, max pooling, and binary thresholding.

To demonstrate and compare the effectiveness of each transformation on dimension reduction, we plotted all singular values, calculated explained variance by the first 10 singular values, and calculated reconstruction errors. All implementations were done using python.

2. SINGULAR VALUE DECOMPOSITION AND IMAGE RECONSTRUCTION

Singular Value Decomposition (SVD): Singular Value Decomposition decomposes a matrix into three components:

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

Given matrix A with size m by n, to perform SVD, we first find AA^T and A^TA. Since these two matrices are symmetrical, we can find their corresponding orthogonal eigenvectors u_1, u_2, u_3 ... (left singular vectors) and v_1, v_2, v_3 ... (right singular vectors). [1]

After finding eigenvalues for left and right singular vectors, we sort the eigenvalues from the largest to the smallest. We realize that the left and right singular vectors share the same singular values if A is a square matrix. If A is not a square matrix, meaning m ≠ n, left and right singular vectors will still share the same eigenvalues from the beginning until one set of the eigenvalues runs out. The rest of the non-shared eigenvalues will all be 0. We call these shared eigenvalues—singular values.

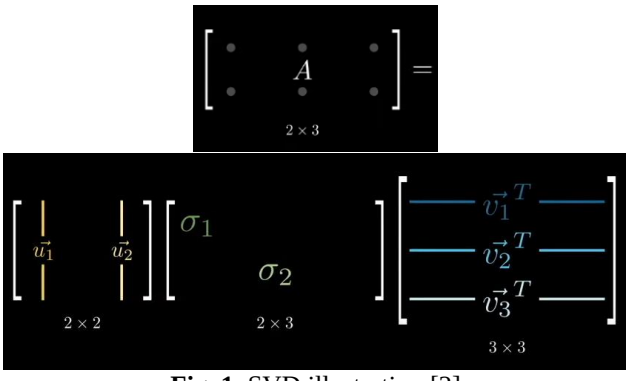

Fig. 1. SVD illustration [2]

A (m, n): The original matrix
U (m, m): Left singular vectors
∑ (m, n): Singular values as diagonal entries, 0 for the rest
V^T (n. n): Right singular vectors

In our project, we used 60,000 images in the MNIST training set. Each image was flattened from size (28, 28) to size (1,784). Basically, we treat each image as a vector in 784-dimensional space. Data matrix A is constructed of 60,000 rows of flattened images, resulting in size (60,000, 784).

```
# perform SVD
u, s, vh = np.linalg.svd(data_matrix, full_matrices=False)
```

Fig. 2. SVD in python

We performed SVD and obtained U, ∑, and V^T matrices in reduced formats. Reduced matrices removed all the 0 entries to save memory space. ∑ became a one-dimensional array.

A: (60,000, 784)
U: (60,000, 60,000) → (60,000, 784)
∑ (60,000, 784) → (784,)
V^T (784, 784) → (784, 784)

To see how fast the singular values decrease and to compare results between transformations, we plotted the singular values and calculated the variance explained (by the first 10 singular values):

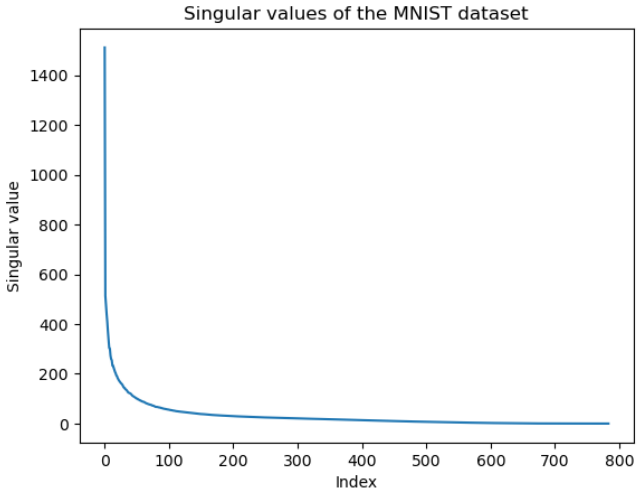

Fig. 3. Singular values of the original dataset

$$Total\ var = \sum \sigma^2$$

$$Sum\ of\ top\ 10\ var = \left(\sum_{i=1}^{10} {\sigma_i}^2\right)$$

$$Var\ explained = \frac{Sum\ of\ top\ 10\ var}{Total\ var} = 0.6916$$

This plot and variance explained serve as our baseline for comparison with other transformations. If any transformation improves the result of SVD, the variance explained should increase; vice versa if it degrades the result.

Image Reconstruction: Since the diagonal matrix Σ contains singular values sorted in descending order. These singular values represent the importance of each dimension in the data. To reconstruct the original image, we can use the top k singular vectors/values to capture the most significant information. This is known as low-rank approximation.

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

For example, with k=50, the dimensionality of the dataset is reduced from 784 to 50. If we look at the sizes of the matrices, we can see the computation burden is also reduced:

U_k: (60,000 * 50)
Σ_k: (50 * 50)
V_k^T: (50 * 784)
A_k: (60,000 * 784)

As demonstrated in the following figure, when k becomes larger, the image quality becomes better. With k = 784, the reconstructed images are identical to the original images since all singular vectors/values are used for reconstruction. It is worth mentioning that with k = 50, although the reconstructed images still lack sharpness and clarity, we can already visually identify the digits.

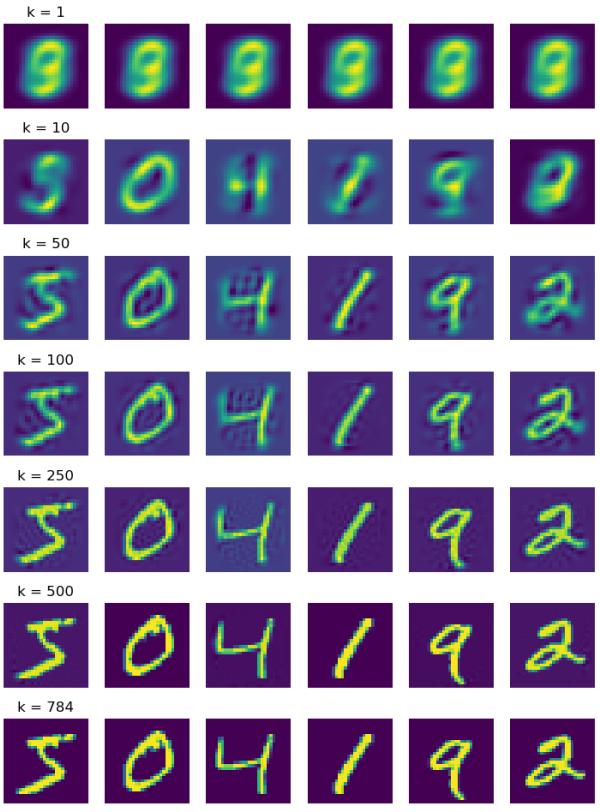

Fig. 4. Image reconstruction with different values of k

3. TRANSFORMATIONS

1. **Centering the data:** We calculated the center of weight (intensity) for each image and shifted each image so that the center of weight is at the center of the image, pixel value (14, 14). However, we realized this didn't change anything since the dataset was already centered.

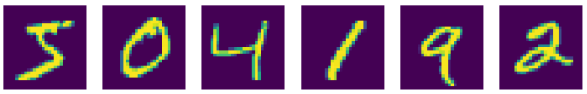

Fig. 5. Centering

2. **Random shifting:** Random shifting on x and y axis of each image.

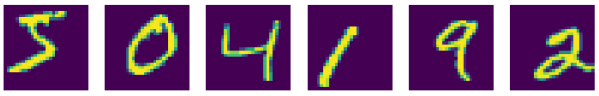

Fig. 6. Random shifting

3. **Separating each digit:** Dataset is separated into 10 subsets (0 to 9), each subset consists of the same digits.

Fig. 7. Separating each digit

4. **Deskewing:** Utilizing moments function in OpenCV, we can skew each image so that the digits look more standing upright. [3]

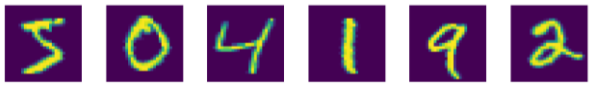

Fig. 8. Deskewing

5. **Median filtering:** Applying median filtering on each image. This transformation will remove high frequency noises.

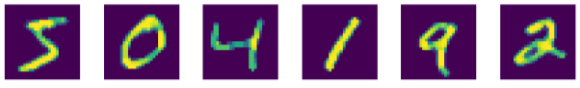

Fig. 9. Median filtering

6. **Max pooling:** A commonly used transformation technique in neural networks. In our case, we applied 2 by 2 max pooling with a stride of 2. Thus, the image size decreases from (28, 28) to (14, 14), resulting in only 1/4 of the original data size. Matrix A became size (60,000, 196).

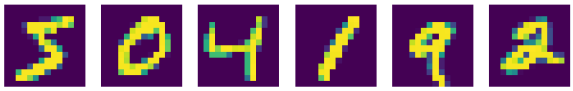

Fig. 10. Max pooling

7. **Binary thresholding:** We applied binary thresholding to each image, hoping to eliminate some variations. Our threshold was set to 0.5.

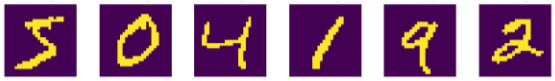

Fig. 11. Binary thresholding

4. RESULTS

After each transformation, we apply SVD on the transformed dataset, calculated the variance explained by the first 10 singular values (10 largest singular values), and visualized the reconstructed results with 10 singular vectors/values. For each image, we calculated L2 loss and took the mean of all images as reconstruction error.

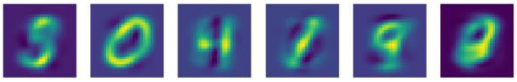

Fig. 12. Reconstruction (centered data)

1. **Centering the data:** Even though the dataset is already centered, this still provides us with a baseline comparison. Explained variance by first 10 components = 0.6916. Reconstruction error = 5.1306. Reconstruction results provide a rough outline of digits. "2" is barely recognizable.

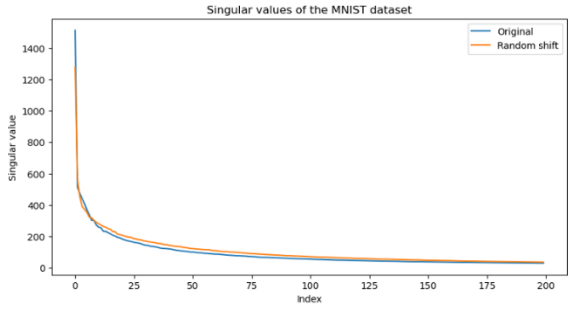

Fig. 13. Singular values (original/centered vs random shift)

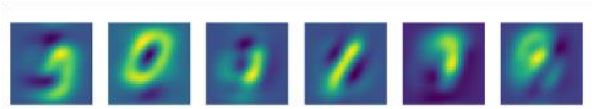

Fig. 14. Reconstruction (random shift)

2. **Random shifting:** Explained variance by first 10 components = 0.5607. Reconstruction error = 8.1188. This decrease in the explained variance can be visualized in the reconstructed digits. Digits are totally unrecognizable.

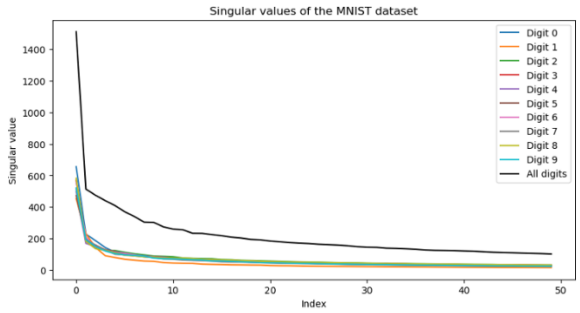

Fig. 15. Singular values (each digit)

Fig. 16. Reconstruction (each digit)

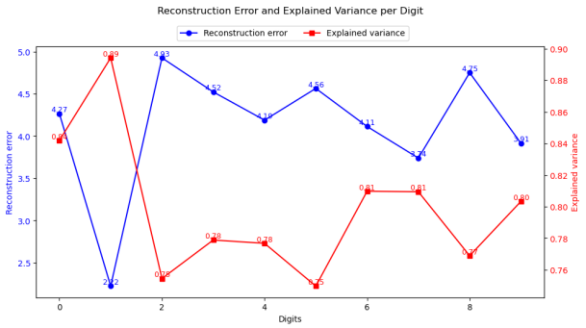

Fig. 17. Reconstruction error vs explained variance (each digit)

3. **Separating each digit:** After separating each digit, explained variance was calculated for each digit dataset, and reconstruction error was calculated for each digit dataset as well. Then we took the meaning of all digits, to get the average explained variance and average reconstruction error. We can see that when explained variance increases, reconstruction error decreases; vice versa when explained variance decreases, reconstruction error increases. Average explained variance by first 10 components = 0.7988. Average reconstruction error = 4.1203.

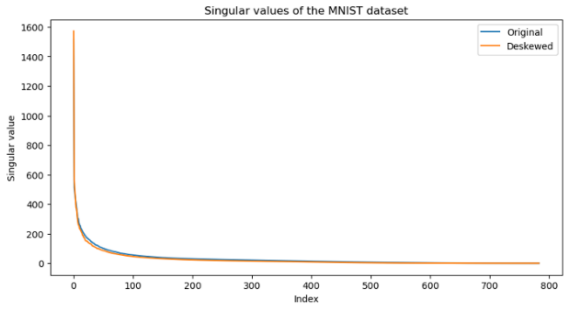

Fig. 18. Singular values (deskewing)

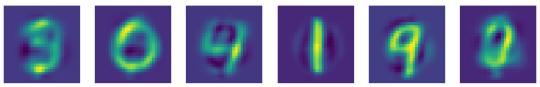

Fig. 19. Reconstruction (deskewing)

4. **Deskewing:** Explained variance by the first 10 principal components = 0.7642. This suggests that deskewing helps to capture more of the underlying structure of the data, leading to a more accurate and informative low-dimensional representation. Reconstruction error using deskewing = 5.8528. However, reconstruction error seems to be worse than centered dataset. The reason for this could be that after deskewing, the dataset is already manipulated compared to the original dataset.

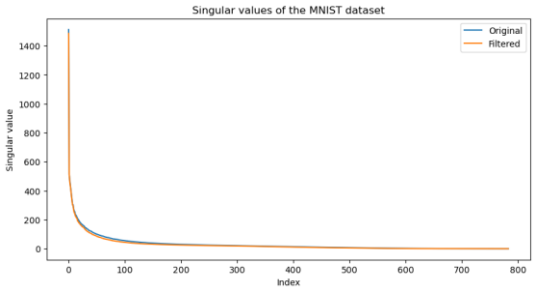

Fig. 20. Singular values (median filtering)

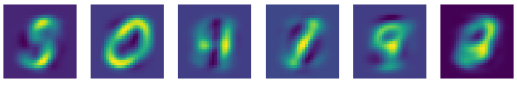

Fig. 21. Reconstruction (median filtering)

5. **Median filtering:** Explained variance by the first 10 principal components = 0.7328. This suggests that median filtering helps to capture more of the underlying structure of the data, leading to a more accurate and informative low-dimensional representation. Reconstruction error using median filtering = 5.1826. However, reconstruction error seems to be worse than centered dataset. The reason

for this could be that after median filtering, the dataset is already manipulated compared to the original dataset.

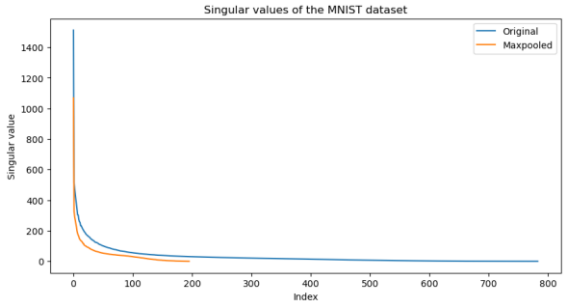

Fig. 22. Singular values (max pooling)

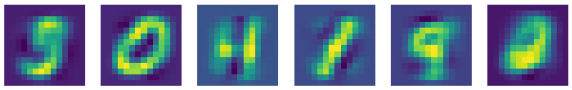

Fig. 23. Reconstruction (max pooling)

6. **Max pooling:** Since max pooling makes each image shrink its size, reconstruction error was calculated with respect to the already max pooled dataset. This might seem a little unfair. Explained variance by the first 10 principal components = 0.7923. Reconstruction error using max pooling = 2.6364.

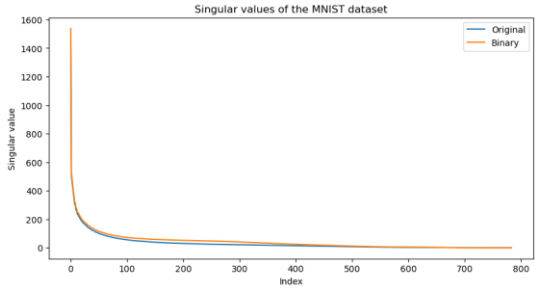

Fig. 24. Singular values (binary thresholding)

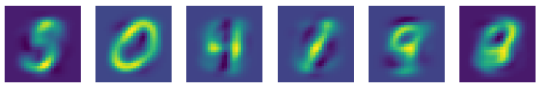

Fig. 25. Reconstruction (binary thresholding)

7. **Binary thresholding:** Explained variance by the first 10 principal components = 0.6163. Reconstruction error using binary thresholding = 5.1576. Binary thresholding performed worse than centering the dataset.

5. CONCLUSION

To compare the impact of all the transformations, we sorted the variance explained by the first 10 components and made plots.

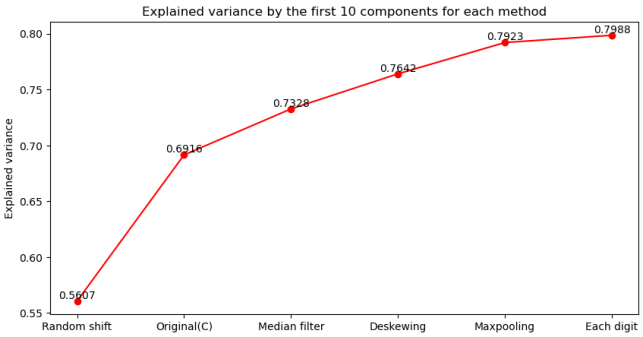

Fig. 26. Explained variance of all transformations

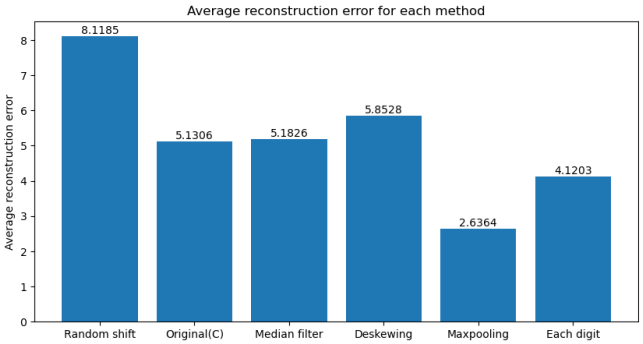

Fig. 27. Reconstruction error of all transformations

From these two plots, we can observe that random shifting produces the worst explained variance, separating each digit gives the best explained variance. This makes sense since random shifting will create more variance in the entire dataset, and separating each digit will give us the cleanest subsets due to less variance among images of the same digit. Max pooling gives us the second best explained variance since the total dimensionality has been reduced to 1/4 already. Deskewing also eliminates variance in skewing, minimizing the variance among rotation and y axis, since every digit will stand up right after deskewing. Median filtering eliminates the high frequency noises, thus giving us a better result of explained variance.

However, higher explained variance does not necessarily mean that lower reconstruction error was achieved. Even though we observed a pattern during separation of each digit. The increase in reconstruction error for median filtering and deskewing could be explained by some manipulation of the original data during the translation, causing a mismatch between the reconstruction and the original data.

Overall, separating each digit created the best explained variance and the lowest reconstruction error. But if we cannot create subsets of data, centering the data to the center of mass will also give us a low reconstruction error.

6. ACKNOWLEDGEMENTS

We gratefully acknowledge our professor, Dr. David Stewart, for his guidance and support during the semester.

7. REFERENCES

[1] Storrs, E. (2021). Explained: Singular value decomposition (SVD). Retrieved from https://storrs.io/svd/

[2] Visual Kernel. (2022, February 8). SVD visualized, singular value decomposition explained [Video]. YouTube. https://www.youtube.com/watch?v=vSczTbgc8Rc&t=53s

[3] Ghosh, D., & Wan, A. (n.d.). Deskewing. Retrieved from https://fsix.github.io/mnist/Deskewing.html