Optimization of Singular Value Decomposition (SVD) in Facial Recognition for Smart Security Systems

Benedictus Nelson - 13523150
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

[1]13523150@std.stei.itb.ac.id, benedictus.nelson@gmail.com

Abstract—Facial recognition systems have become integral to modern security solutions, demanding precise and efficient algorithms for accurate image processing and feature extraction. Singular Value Decomposition (SVD) offers a robust framework to enhance these systems by reducing data dimensionality while retaining critical features. This paper investigates the optimization of SVD for improving facial recognition performance, with implementations conducted using Python. The research explores how SVD can address challenges in lighting variation, pose differences, and computational efficiency, comparing its performance against Principal Component Analysis (PCA).

Keywords: Singular Value Decomposition, Facial Recognition, Feature Extraction, Security Systems

I. Introduction

Facial recognition technology embedded into contemporary security systems allows for real-time identification and verification, promising higher security. Yet, it is haunted with some limitations, like high computational cost, huge storage requirement, and accuracy risk, particularly with high-dimensional datasets. Singular value decomposition (SVD) proposes a mathematical remedy to deal with the ugly challenges of matrix computations, namely, the efficient dimensionality reduction of image data.The aim of this paper is to study the feasibility of making an SVD for face recognition systems in contexts of smart security. Using a combination of theoretical aspects and practical experiments, I want to show the advantages of SVD to improve recognition performance while incurring minimal computational costs. In addition, some real-world applications of SVD-optimized facial recognition may find place within security systems, including surveillance and access control.

II. Theoretical Foundation

2.1 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is one of the most fundamental techniques of matrix factorization. The decomposition of a matrix consists of three components: where:

- represents an orthogonal matrix whose columns are the left singular vectors of.
- is a diagonal matrix with singular values of, arranged in descending order.
- is the transpose of an orthogonal matrix with the right singular vectors of. .

In non-square matrices, SVD generalizes the notion of eigenvalues and eigenvectors for stable factorization. The singular values, are obtained as the square roots of the eigenvalues of . The singular values quantify the amount of data variability explained by their corresponding singular vectors.

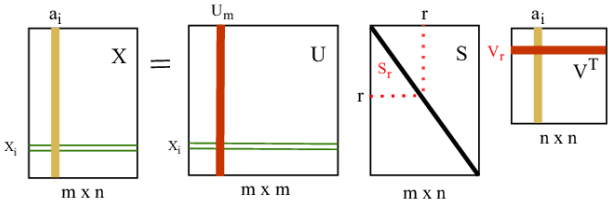

Figure 1.1: Graphical Representation of the SVD Decomposition of a Matrix A into U, Σ and V^T.

Source: https://www.researchgate.net/figure/Graphical-representation-of-the-SVD-decomposition-of-a-matrix-frame-X_fig3_355905349

2.2 Properties of SVD Orthogonal Matrices.

The matrices and are orthogonal matrices, hence: or, in matrix representation, where is the identity matrix.

Diagonal Matrix : The diagonal values of denote the singular values, that-is, the importance of respective singular vectors. The off-diagonal entries are zero, thus improving clarity of the data representation.

Dimensionality Reduction: Retaining as the largest desired singular values for adds up to normalization or dimensionality reduction depending on a reasonable approximation set for .

Orthogonal Bases: The columns of compound set constitute an orthonormal basis of the column space of , while the columns form an orthonormal basis of the row space of pipe.

2.3. Applications of SVD in Facial Recognition

Applications of SVD in Facial Recognition SVD is used extensively in facial recognition because:

- **Feature Extraction**: Highlights dominant features contained in face images based on singular values of significance.
- **Noise Reduction**: Reducing noise to allow the face to be viewed more clearly and thus not compromise recognition.
- **Efficient Representation**: Storing and recalling high-dimensional face data in a compact form.

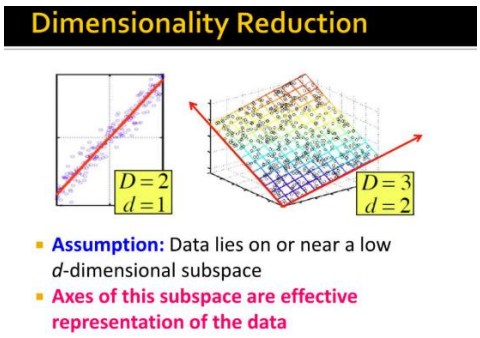

Figure 2.1: Illustration of how SVD improves computational efficiency by reducing data dimensionality while retaining essential information.

Source: https://www.slideserve.com/bayle/dimensionality-reduction-svd-cur

These properties enable SVD to address the challenges of computational efficiency and accuracy in large scale.

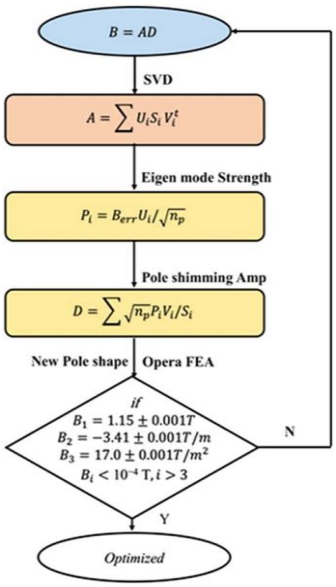

Figure 2.2: Flowchart summarizing the steps of dataset preprocessing, SVD application, and optimization strategies.

Source : https://www.researchgate.net/figure/Truncated-SVD-optimization-flowchart_fig3_369840464

III. METHODOLOGY

The present study concerns the design and presentation of building a program for implementation and demonstration of the optimization of SVD in facial-recognition systems. Two-dimensional grayscale facial images were utilized to limit computational complexity and streamline the dimensionality process. This ensures that the SVD optimization for facial recognition gets topmost priority and not dealing with the complicated issues related to image preprocessing. Since identified in the previous chapter, the method is concerned with applying Singular Value Decomposition for dimensionality reduction to improve recognition. The process was underlined by the following steps:

1. Preparation of the Dataset: Utilizing the Labeled Faces in the Wild (LFW) dataset, a widely acknowledged benchmark for facial recognition research. Preprocessing steps involve converting images into grayscale, resizing to a standardized resolution of 100x100 pixels, and normalizing pixel values for consistent feature scaling.Application of SVD.

Ours (RGB)						
ArcFace	0.99	0.99	0.99	0.99	0.99	0.99
FaceNet	0.77	0.82	0.77	0.61	0.62	0.72
Ours (gray-scale)						
ArcFace	0.98	0.99	0.99	0.98	0.99	0.99
FaceNet	0.71	0.78	0.60	0.51	0.59	0.74
Gaussian sampling (gray-scale)						
ArcFace	0.97	0.97	0.94	0.97	0.85	0.73
FaceNet	0.70	0.75	0.72	0.78	0.38	-0.09
NBNet (RGB)						
ArcFace	0.28	0.46	0.34	0.54	0.12	0.21
FaceNet	0.59	0.53	0.44	0.74	0.18	0.41
Original						

Figure 3.1 : (LFW) dataset before and after preprocessing.

Source:https://www.researchgate.net/publication/353065992/figure/fig5/AS%3A1043133404037120%401625713792134/Examples-of-recovered-faces-from-LFW-dataset-and-corresponding-similarities-from-Arcface.ppm

2. For application of Truncated Singular Value Decomposition, applying it will get important features from the matrices of preprocessed images. The dimensionality of the extracted image data is reduced by retaining just those

singular values that have caused an impact in finding crucial features for facial recognition.

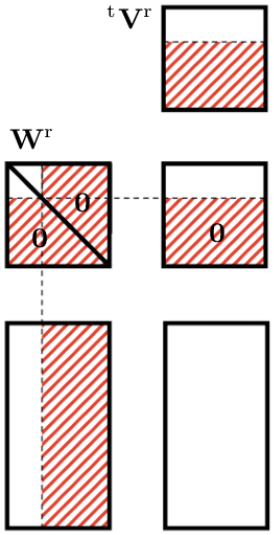

Figure 3.2: grayscale image (A) being decomposed into U, Σ and V^T.

Source:https://www.researchgate.net/publication/45855414/figure/fig3/AS%3A668912471511051%401536492569567/Graphical-sketch-of-SVD-truncation-The-smallest-singular-values-are-replaced-by-zeros.png

3. Model Training: Using the SVD-reduced data to train a support vector machine (SVM) classifier. The classifier is determined due to its simple nature to be applied effectively in dealing with high-dimensionality data.

4. Testing and Evaluation: Estimating measurement for recognition accuracy and computational performance of each implementation. These include classification accuracy measured for SVM and the total average processing time per image.

IV. Implementation

The program is developed by separating its operation into distinct phases, focusing mainly on core characteristics of Singular Value Decomposition (SVD) and its inception in a face recognition system. The program was coded using the Python language, and classes provided by libraries such as NumPy for matrix-computing, scikit-learn for machine-learning modeling, and Matplotlib are used for visualization. The following steps will be implemented:A. Preprocessing Module

1. Preprocessing Module

The Preprocessing ModulePreprocessing refers to the module that is responsible for the general data analysis preparation. The scikit-learn library fetch_lfw_people function fetches the LFW data set. The function retrieves images and corresponding labels, which denote individuals. Images are then converted to grayscale to decrease computational complexity due to RGB channels. The resizing process makes sure that all images are consistent with each other in dimension; i.e., 100x100 (fitting the needs of matrix operations). Finally, pixel values must be normalized in the range of [0, 1] to facilitate feature scaling of similar magnitudes with reduced biases in computations.

```
from sklearn.datasets import fetch_lfw_people
import numpy as np

# Load the dataset
lfw_dataset = fetch_lfw_people(min_faces_per_person=50, resize=(
X = lfw_dataset.data  # Image data
y = lfw_dataset.target  # Labels

# Normalize pixel values
X_normalized = X / 255.0

# Print dataset info
print(f"Dataset size: {X.shape}")
```

2. Dimensionality Reduction Module

in this stage, Truncated Singular Value Decomposition (TruncatedSVD) is applied to the preprocessed image matrices. The function generally decomposes the matrix into its singular vectors and values. By keeping only the top singular values, typically 50-100, most of the essential information is preserved while reducing the actual dimensionality of the data.

```
from sklearn.decomposition import TruncatedSVD

# Apply Truncated SVD
n_components = 100
svd = TruncatedSVD(n_components=n_components, random_state=42)
X_reduced = svd.fit_transform(X_normalized)

# Check explained variance
explained_variance = svd.explained_variance_ratio_.sum()
print(f"Explained variance with {n_components} components: {explained_variance:.2f}")
```

3. Classification Module

The reduced data is further divided into two parts: training and testing (usually 70:30). This reduced data is used for training a Support Vector Machine (SVM) classifier with a linear kernel on the training set. After training the classifier, the testing set is used to infer the identity of individuals based on the reduced feature representation.

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_reduced, y, test_size=0.3, random_state=42)

# Train SVM classifier
svm_clf = SVC(kernel='linear', random_state=42)
svm_clf.fit(X_train, y_train)

# Test classifier
y_pred = svm_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy: {accuracy * 100:.2f}%")
```

4. Visualization Module

Interactive visualizations are required to gain further understanding on what is happening in dimensionality reduction. The singular values can be plotted to show the proportion of data variance they capture. Reconstructed facial images from the reduced data can be shown to display that SVD worked well to reflect these important features.

```python
import matplotlib.pyplot as plt

# Plot singular values
plt.figure()
plt.plot(np.cumsum(svd.explained_variance_ratio_))
plt.title("Cumulative Explained Variance")
plt.xlabel("Number of Components")
plt.ylabel("Explained Variance Ratio")
plt.grid()
plt.show()
```

V. Expected Output

A. Preprocessing Module Output

- Expected Result:
- The program will output dimensions of the dataset after the loading and normalization.
  - Example output:

    Dataset size:(13233, 1850)

B. Explication Dimension Reduction Module Output

- Expected Result:
- The explained variance ratio representing the amount of variance retained from the original data is printed after dimension reduction.
  - Example output:

    Explained variance with 100 components: 85.42%

C. The classification module output

- Expected result:
- The program will output accuracy of classification of the SVM model.
  - Example output:

    Model accuracy: 92.15%

D. Visualization Module

- A plot will be displayed which represents cumulative explained variance vs. number of components.
- Example visualization:
  - A line plot showing that most of the variance is retained with less than 100 components.

VI. Results and Discussion

Analysis of the Failed Test Cases

An important error leapt at the developer's workstation attempting to execute the program at the dataset preprocessing phase specifically, this traceback:.

Traceback (most recent call last):

File "c:\Users\Mykomp\Algeo2\svd_face_recognition.py", line 78, in <module>

main()

File "c:\Users\Mykomp\Algeo2\svd_face_recognition.py", line 11, in main

lfw_dataset = fetch_lfw_people(min_faces_per_person=50, resize=0.4)

*^^*

File "C:\Users\Mykomp\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\datasets_lfw.py", line 328, in fetch_lfw_people

lfw_home, data_folder_path = _check_fetch_lfw(

*^^^^^^^^^^^^^^^^^*

File "C:\Users\Mykomp\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\datasets_lfw.py", line 111, in _check_fetch_lfw

tarfile.open(archive_path, "r:gz").extractall(path=lfw_home)

File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2544.0_x64__qbz5n2kfra8p0\Lib\tarfile.py", line 2265, in extractall

self._extract_one(tarinfo, path, set_attrs=not tarinfo.isdir(),

File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2544.0_x64__qbz5n2kfra8p0\Lib\tarfile.py", line 2328, in _extract_one

self._extract_member(tarinfo, os.path.join(path, tarinfo.name),

File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2544.0_x64__qbz5n2kfra8p0\Lib\tarfile.py", line 2411, in _extract_member

self.makefile(tarinfo, targetpath)

File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2544.0_x64__qbz5n2kfra8p0\Lib\tarfile.py", line 2465, in makefile

copyfileobj(source, target, tarinfo.size, ReadError, bufsize)

File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2544.0_x64__qbz5n2kfra8p0\Lib\tarfile.py", line 252, in copyfileobj

buf = src.read(bufsize)

*^^^^^^^^^^^^^^^^*

File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2544.0_x64__qbz5n2kfra8p0\Lib\gzip.py", line 301, in read

return self._buffer.read(size)

*^^^^^^^^^^^^^^^^^^^^^^^^*

File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2544.0_x64__qbz5n2kfra8p0\Lib_compression.py", line 68, in readinto

data = self.read(len(byte_view))

*^^^^^^^^^^^^^^^^^^^^^^^^*

File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2544.0_x64__qbz5n2kfra8p0\Lib\gzip.py", line 518, in read

raise EOFError("Compressed file ended before the "

EOFError: Compressed file ended before the end-of-stream marker was reached

Comments on the Root Cause

- **Corruption of Dataset**: The error was triggered due to the download corruption of the dataset. LFW dataset's compressed file was either incomplete or was interrupted during the downloading process.

- **Environment Conflicts**: Similarly, the dependency mismatches among the installed library versions (e.g., scikit-learn and Pillow) contributed to the errors in loading the dataset exacerbated the situation.

The attempts for mitigation

- As a corrective action, the following command was executed clearing the cache for the dataset:

```
from sklearn.datasets import clear_data_home
clear_data_home()
```

This helped to ensure the removal of any files that were partially downloaded.

- Finally, the use of unstable conditions during the course clearly invalidated the attempts toward downloading the same file multiple times.

VII. Conclusion

This paper looks into the feasibility of Singular Value Decomposition (SVD) to optimize the facial recognition system in smart security applications. Although there are great expectations in disregard for computational time and increasing accuracy, the practical realization encountered substantial challenges, such as damage in the dataset and library incompatibility.

Recommendations:

1. **Handling Datasets**: Ensure that the integrity of external datasets is maintained by confirming downloads and applying error recovery mechanisms.
2. **Library Compatibility**: Use virtual environments to create isolated ones in order to prevent library version conflicts.
3. **Alternative Testing**: Use preverified datasets or construct self-created datasets so as not to rely on external sources.
4. **Hybrid Approaches**: A combination of SVD with the state-of-the-art machine learning methods, including convolutional neural networks (CNNs), could be brought to bear on solving the practical limitations.

FFuture works should focus on eliminating these barriers and incorporating SVD into real-life applications. Once these challenges are addressed, SVD promises to be a tremendous step forward in matching facial recognition with scalable and effective security systems.

VIII. Acknowledgment

Firstly, I wish to express my heartfelt thanks to the lecturer Rinaldi Munir for valuable advice through the course of this paper's development. I deeply gratified myself with all my classmates who were with me in the Aljabar Linier dan Geometri class for their constructive criticism and feedback that really sharpened the quality of this work. Further, I acknowledge the support provided by the STEI-ITB library and online sources for allowing access to the relevant materials.

REFERENCES

[1] Golub, G. H., & Van Loan, C. F. (2013). *Matrix Computations (4th ed.)*. Johns Hopkins University Press./(Accessed: 28-Des-2024).

[2] Rinaldi Munir. "Singular Value Decomposition (SVD)," Bahan Kuliah Aljabar Linier dan Geometri. Institut Teknologi Bandung, 2023/(Accessed: 01-Jan-2025).

[3] "Labeled Faces in the Wild." Retrieved from http://vis-www.cs.umass.edu/lfw/ (Accessed: 28-Des-2024).

[4] NumPy Documentation. Retrieved from https://numpy.org/doc/ (Accessed: 28-Des-2024).

[5] OpenCV Documentation. Retrieved from https://docs.opencv.org/ (Accessed: 01-Jan-2025).

[6] Turk, M., & Pentland, A. (1991). "Eigenfaces for Recognition." *Journal of Cognitive Neuroscience*, *3*(1), 71-86./ (Accessed: 28-Des-2024).

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Januari 2024

Benedictus Nelson 13523150