Images, pre-processing and Dimensionality Reduction (PCA)

Francesco Villi
francesco.villi@edu.unifi.it

Abstract

This project investigates the impact of image preprocessing and Principal Component Analysis (PCA) on image retrieval and prediction performance using the VGGFace dataset. We explore feature extraction with and without preprocessing, subsequently applying PCA for dimensionality reduction.
The core research question is whether preprocessing and PCA-based dimensionality reduction enhance accuracy in image retrieval and prediction tasks.

1. Introduction

The data set used is "Labeled Faces in the Wild", after extracting the colored images.
Therefore I resized the images and then I compared the application of normalization and without std according to the data given by the test set and applied also to the test set. To extract features I used VGGFace. I used Faiss library for similarity research.
I'm going to compare retrieval without reduction and a reduction maintaining the first 20, 50, 100, 396 components as we can see from images 1 and 1 some tests without the first 1-2-3 components due to these components being focused on low frequencies and I expect that they will not have a huge impact in discriminating images.

1.1. Principal Component Analysis

Principal component analysis (PCA) is a linear dimensionality reduction technique; the data is linearly transformed into a new coordinate system such that the directions (principal components) capturing the largest variation in the data can be easily identified.

The principal components of a collection of points in a real coordinate space are a sequence of *p* unit vectors, where the *i-th* vector is the direction of a line that best fits the data while being orthogonal to the first *i-1* vectors.

The process of PCA involves several steps, including standardizing the data, computing the covariance matrix, calculating the eigenvalues and eigenvectors of this matrix,

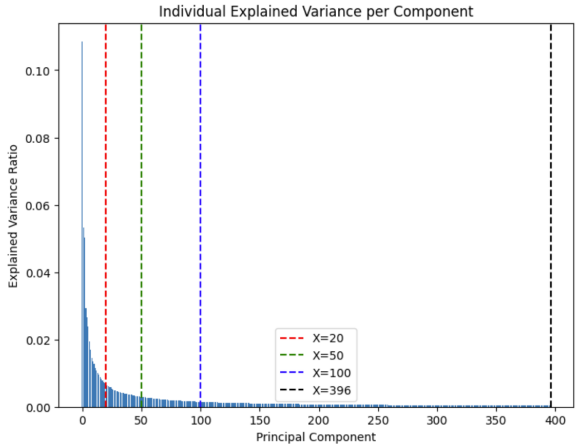

Figure 1. Individual explained variance per component (training set)

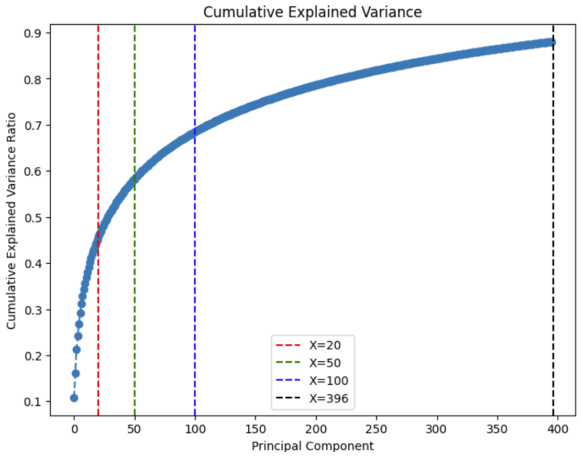

Figure 2. Cumulative variance to reconstruct (training set)

and finally, transforming the original data into the new coordinate system defined by the principal components. Practically the steps are:

- **Train**: Given sample

$$D = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^n$$

- Compute:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$$

- Compute eigenvalues and eigenvectors of Σ, where:

$$\Sigma = \Phi\Lambda\Phi^T,$$

$$\Lambda = diag(\sigma_1^2, \ldots, \sigma_n^2),$$

$$\Phi^T\Phi = I$$

- Order eigenvalues $\sigma_1^2 > \ldots > \sigma_n^2$
- Select K eigenvalues and eigenvectors

- **Test**: Given principal components $\phi_i, i \in 1, \ldots, k$ and test sample $T = \{t_1, \ldots, t_n\} \in \mathbb{R}^d$
  - Subtract mean from each point $t_i' = t_i - \hat{\mu}$
  - Project onto eigenvector space $y_i = At_i'$ where

$$A = \begin{pmatrix} \phi_1^T \\ \vdots \\ \phi_k^T \end{pmatrix}$$

  - Use $T' = \{y_1, \ldots, y_n\}$

1.2. Faiss library

Faiss is a powerful and efficient library designed for similarity search and clustering of high-dimensional vectors. It supports various distance metrics like L2 (Euclidean distance) and IP (inner product):

- **L2 distance** (Euclidean distance) measures the straight-line distance between two vectors in the vector space avoiding the square root due to speed reasons; so the result is an approximation:

$$A, B \in R^n$$

$$\text{L2 Dist.} = \sum_{i=1}^{n}(A_i - B_i)^2$$

- **Inner product distance** (IP), a metric commonly used in machine learning and information retrieval tasks. The norm of the query vectors does not affect the ranking of results (of course the norm of the database vectors does matter).In our case, this is cosine similarity, due to the vectors are normalized as you can see from proposition 3.

$$A, B \in R^n$$

$$\text{IP Dist.} = \mathbf{A} \cdot \mathbf{B} = \cos\theta$$

2. Project structure

- **Get face images**: Using LFW datasets from sklearn I downloaded all images in RGB where the lowest number of samples was 55, this dataset has 9 different faces.
- **Balancing data**: I addressed the dataset imbalance by implementing a balancing technique to mitigate the disparity between the underrepresented and overrepresented classes by selecting the minimum number of available samples for the underrepresented class.
- **Split dataset**: I partitioned the dataset into training and testing sets. Subsequently, I computed the mean and standard deviation of the training set.
- **Pixel standardization**: I scaled pixel values to have a zero mean and unit variance using the mean and std computed at the previous step
- **Extracting features**: I used VGGFace without the 3 fully connected layers at the top of the network. So for each sample, I got a tensor with shape (7, 7, 512).
- **Reduction and Testing**: I applied PCA reductions to the features extracted from VGGFace. I conducted tests comparing the original data to datasets subjected to PCA reductions, with preservation of 20, 50, and 100 components. Additionally, variations were explored by excluding 1, 2, or 3 of the first components. Then I tested the accuracy in recognition task.

3. Equivalence of Cosine Similarity and L2 Distance for Normalized Vectors

Let $\mathbf{a}$ and $\mathbf{b}$ be normalized vectors, i.e., $\|\mathbf{a}\| = \|\mathbf{b}\| = 1$.

Cosine Similarity with Normalized Vectors:

$$\text{cosine\_similarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} = \mathbf{a} \cdot \mathbf{b}$$

L2 Distance with Normalized Vectors:

$$\begin{aligned} \text{L2\_distance}(\mathbf{a}, \mathbf{b}) &= \|\mathbf{a} - \mathbf{b}\|_2 \\ &= \sqrt{(\mathbf{a} - \mathbf{b}) \cdot (\mathbf{a} - \mathbf{b})} \\ &= \sqrt{\mathbf{a} \cdot \mathbf{a} - 2\mathbf{a} \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{b}} \end{aligned}$$

3.1. Substitute $\|\mathbf{a}\| = \|\mathbf{b}\| = 1$:

$$\begin{aligned} &= \sqrt{1 - 2(\mathbf{a} \cdot \mathbf{b}) + 1} \\ &= \sqrt{2 - 2(\mathbf{a} \cdot \mathbf{b})} \end{aligned}$$

$$\text{cosine\_similarity}(\mathbf{a}, \mathbf{b}) = \mathbf{a} \cdot \mathbf{b}$$

$$\text{L2\_distance}(\mathbf{a}, \mathbf{b}) = \sqrt{2 - 2(\mathbf{a} \cdot \mathbf{b})}$$

We observe that both expressions are equivalent. Hence, for normalized vectors, cosine similarity is indeed equivalent to L2 distance.

4. Results and Discussion

4.1. Retrieval task

I'm going to compare results with varying numbers of components and removing 1,2,3 first eigenvectors. I'll use only IP distance due to the prop. 3.

$ID\ Image$	$P@10 - STD$				
	RAW	20	50	100	396
0	**0.97**	0.96	0.96	0.96	0.96
1	**0.97**	0.92	0.93	0.93	0.93
2	**0.94**	0.93	0.92	0.91	0.91
3	**0.92**	0.85	0.85	0.85	0.84
4	**0.86**	0.83	0.82	0.80	0.79
5	**0.95**	0.94	**0.95**	**0.95**	0.94
6	**0.91**	0.86	0.86	0.86	0.87
7	**0.99**	0.98	0.98	0.98	0.98
8	**0.86**	0.84	0.84	0.85	0.84
AVG	**0.93**	0.90	0.90	0.90	0.90

Table 1. Table displaying P@10 scores over 9 runs with different split test-train. Comparing PCA with 20,50,100,396 and 7x7x512 components with image standardization.

$ID\ Image$	$P@10$				
	RAW	20	50	100	396
0	**0.96**	0.86	0.87	0.87	0.86
1	**0.93**	0.86	0.86	0.87	0.86
2	**0.87**	0.83	0.83	0.84	0.83
3	**0.81**	0.75	0.75	0.75	0.75
4	**0.79**	0.66	0.68	0.70	0.67
5	0.85	0.85	**0.86**	**0.86**	0.84
6	**0.85**	0.82	0.83	0.84	0.84
7	**0.98**	0.97	0.96	0.96	0.96
8	**0.82**	0.77	0.77	0.76	0.76
AVG	**0.87**	0.82	0.82	0.83	0.82

Table 2. Table displaying P@10 scores over 9 runs with different split test-train. Comparing PCA with 20,50,100,396, 7x7x512 components without image standardization

Upon comparing tables 1 and 2, it becomes evident that employing image standardization yields notably improved outcomes. Notably the algorithm demonstrates superior performance in both tables where ALL components are maintained.
However, it's worth highlighting that even in scenarios where resource constraints such as limited memory and CPU capacity are prevalent, good results can be achieved by applying PCA.

Then I decided, after applying PCA, to remove the first 1/2/3 components out of 20/50/396 as we can see in tables 10 and 11. Here the best results are archived with image standardization and removing the first 1-2 components, but they under-perform the previous approach (table 2 and 1).

STD PCA red.	P@5	P@10	P@20	P@ALL
20	0.94	0.90	0.84	**0.33**
50	0.94	0.90	0.82	0.28
100	0.94	0.90	0.82	0.27
396	0.94	0.90	0.81	0.27
0	**0.96**	**0.93**	**0.83**	0.14
-1/20	0.93	0.88	0.78	0.28
-2/20	0.92	0.88	0.79	0.29
-3/20	0.91	0.87	0.77	0.25
-1/50	0.93	0.88	0.76	0.24
-2/50	0.93	0.88	0.77	0.23
-3/50	0.92	0.86	0.74	0.20
-1/396	0.93	0.87	0.74	0.23
-2/396	0.93	0.87	0.74	0.22
-3/396	0.92	0.86	0.70	0.19

Table 3. Feature extracted has been standardized before to extract features and apply PCA.Precision as a mean value between all classes at different cut-off values for PCA components. Mean computed over 9 runs. 0 means no reduction

PCA red.	P@5	P@10	P@20	P@ALL
20	0.87	0.82	0.73	0.22
50	0.89	0.82	0.72	0.20
100	0.89	0.83	0.71	0.20
396	0.89	0.82	0.70	0.20
0	**0.93**	**0.87**	0.72	0.12
-1/20	0.88	0.84	**0.76**	**0.23**
-2/20	0.88	0.82	0.71	0.21
-3/20	0.87	0.82	0.71	0.21
-1/50	0.89	0.85	0.74	0.21
-2/50	0.89	0.83	0.69	0.18
-3/50	0.88	0.82	0.67	0.18
-1/396	0.90	0.84	0.72	0.20
-2/396	0.90	0.83	0.66	0.17
-3/396	0.89	0.82	0.64	0.17

Table 4. Precision as a mean value between all classes at different cut-off values for PCA components. Mean computed over 9 runs. 0 means no reduction.

In tables 3 and 4 there is a comparison between all previous methods with different precisions (5,10,20, RAW). We notice that the best results are achieved without PCA reduction.

We can see that removing the first component in table 4 has better results than keeping all components.

4.2. Image recognition task

I analyzed face recognition using K-Nearest Neighbors (KNN) alongside Principal Component Analysis (PCA). For KNN, a parameter of $k = 9$ is utilized, and in the event of a tie, the selection was based on the summation of distances grouped by label, with preference given to the greatest similarity.

Using a balanced dataset, we can observe the results presented in Table 5 and 6, it is evident that the highest accuracy is attained without PCA. Moreover, it is noteworthy that standardized images outperform non-standardized images.

Also using unbalanced dataset results are a little bit better than the previous case as we can see from table 7.

Eventually, I tested the algorithm setting $min\_faces\_per\_person = 30$, balancing the dataset and applying standardization to the images.

In this case, we have 34 different faces instead of 8. The accuracy gets a little bit worse wrt. the previous case but we achieved satisfactory results, in particular without PCA as we can notice from table 8.

ID test	Accuracy w/ n components				
	RAW	20	50	100	396
test 1	**0.985**	0.971	0.964	0.967	0.975
test 2	**0.989**	0.960	0.964	0.975	0.978
test 3	**0.978**	0.971	0.971	0.964	0.971
test 4	**0.989**	0.946	0.953	0.957	0.967
test 5	**0.982**	0.943	0.950	0.960	0.960
test 6	**0.982**	0.950	0.957	0.950	0.957
test 7	**0.982**	0.946	0.953	0.960	0.957
test 8	**0.982**	0.957	0.960	0.967	0.971
test 9	**0.989**	0.960	0.971	0.975	0.978
AVG	**0.984**	0.956	0.960	0.964	0.968

Table 5. Accuracy in face recognition with different train-test splits and using KKN with k=9

5. Conclusion

In this study, we explored the extraction of features using VGGFace and the application of Principal Component Analysis (PCA) to reduce dimensionality. Through our experiments, we observed varying outcomes dependent on different configurations and techniques applied.

Firstly, employing image standardization notably enhanced the performance of our algorithm, as evidenced by the comparison between tables 1 and 2. Maintaining all components consistently demonstrated superior results across different scenarios.

ID test	Accuracy w/ n components STD				
	RAW	20	50	100	396
test 1	0.979	**0.989**	**0.989**	0.969	0.979
test 2	**0.989**	0.949	0.939	0.959	0.969
test 3	**1.000**	0.989	0.989	**1.000**	0.989
test 4	0.989	0.989	0.989	**1.000**	0.979
test 5	**1.000**	0.979	0.989	0.989	**1.000**
test 6	**0.989**	0.949	0.959	0.959	0.969
test 7	**0.979**	0.969	0.969	0.969	0.969
test 8	0.989	**1.000**	0.979	0.989	**1.000**
test 9	**0.979**	**0.979**	**0.979**	**0.979**	0.969
AVG	**0.988**	0.977	0.976	0.979	0.980

Table 6. Accuracy in face recognition with different train-test splits and using KKN with k=9. All images have been pre-processed with standardization and a balanced dataset

ID test	Accuracy w/ n components STD Unb.				
	RAW	20	50	100	396
test 1	**0.996**	0.971	0.978	0.975	0.978
test 2	0.989	0.985	0.989	**0.992**	**0.992**
test 3	0.992	**1.000**	**1.000**	0.996	**1.000**
test 4	0.992	0.989	**0.996**	**0.996**	**0.996**
test 5	**0.992**	0.975	0.978	0.978	0.982
test 6	**0.992**	0.967	0.978	0.982	0.989
test 7	**0.989**	0.967	0.975	0.975	0.978
test 8	**0.989**	0.975	0.978	0.978	0.982
test 9	**0.989**	0.971	0.975	0.975	0.985
AVG	**0.991**	0.978	0.983	0.983	0.987

Table 7. Accuracy in face recognition with different train-test splits and using KKN with k=9. All images have been pre-processed with standardization using an unbalanced dataset

ID test	Accuracy w/ n components STD 30				
	RAW	20	50	100	396
test 1	0.965	0.901	0.941	0.960	0.960
test 2	0.970	0.911	0.926	0.936	0.950
test 3	0.990	0.931	0.960	0.960	0.970
test 4	0.980	0.931	0.950	0.960	0.965
test 5	0.975	0.955	0.965	0.960	0.960
test 6	0.985	0.965	0.960	0.965	0.980
test 7	0.980	0.926	0.975	0.980	0.985
test 8	0.990	0.936	0.965	0.970	0.980
test 9	0.980	0.960	0.955	0.960	0.975
AVG	**0.979**	0.934	0.955	0.961	0.970

Table 8. Accuracy in face recognition with different train-test splits and using KKN with k=9. All images have been pre-processed w standardization and used a balanced dataset where min_faces_per_person=30

However, when resource constraints such as limited memory and CPU capacity were taken into consideration, PCA offered a viable solution.

Further experimentation involved the removal of the

ID test	Accuracy w/ n components Unb. STD 30				
	RAW	20	50	100	396
test 1	0.970	0.924	0.945	0.959	0.959
test 2	0.983	0.949	0.966	0.978	0.983
test 3	0.978	0.926	0.953	0.959	0.962
test 4	0.978	0.917	0.947	0.962	0.964
test 5	0.978	0.932	0.953	0.968	0.970
test 6	0.987	0.945	0.955	0.968	0.968
test 7	0.985	0.947	0.949	0.955	0.970
test 8	0.976	0.928	0.951	0.953	0.964
test 9	0.966	0.907	0.938	0.949	0.951
AVG	**0.978**	0.930	0.951	0.961	0.966

Table 9. Accuracy in face recognition with different train-test splits and using KKN with k=9. All images have been preprocessed w standardization and used an unbalanced dataset where min_faces_per_person=30

first 1/2/3 components out of 20/50/396 after applying PCA. Interestingly, the best results were achieved without image standardization and by removing the first 1 or 2 components. However, it's noteworthy that this approach underperformed compared to the previous method, as observed in tables 10 and 11.
This suggests that while removing certain components can reduce noise, it can also lead to a degradation in overall performance.

Moreover, our comparison of different precision values (5, 10, 20, and all) across various PCA configurations highlighted the effectiveness of employing fewer components in achieving satisfactory results while conserving computational resources. In tables 3 and 4, a comparison between all previous methods with different precisions (5,10,20,ALL) further confirms these findings; the best results are generally achieved without PCA reduction.

Considering only PCA, the best results are achieved by removing the first component where non-standardized images are used to extract features; this statement becomes false when we apply standardization.

In addition to feature extraction, I analyzed face recognition using K-Nearest Neighbors (KNN) alongside PCA. With a parameter of $k = 9$ for KNN and tie-breaking based on the summation of distances grouped by label.
Also for recognition tasks standardized images generally outperformed non-standardized images, as indicated by the results in Table 5 and 6. With an unbalanced dataset, the results were slightly improved, as seen in Table 7.
Furthermore, balancing the dataset and applying standardization to the images with $min\_faces\_per\_person = 30$ resulted in 34 different faces instead of 8, with a slight decrease in accuracy compared to the previous case. However, satisfactory results were achieved for both balancing and unbalancing the data set, as evident from Table 8 and 9. In conclusion, these findings underscore the importance of considering both computational efficiency and performance trade-offs when applying feature extraction techniques such as PCA in facial recognition systems. Additionally, the impact of preprocessing steps such as image standardization can significantly influence algorithm performance.

$ID\ Image$	$STD - P@10$								
	$-1/20$	$-1/50$	$-1/396$	$-2/20$	$-2/50$	$-2/396$	$-3/20$	$-3/50$	$-3/396$
0	0.95	**0.96**	**0.96**	0.93	0.94	0.94	0.94	0.88	0.88
1	0.87	**0.88**	0.87	0.84	0.86	0.84	0.84	0.84	0.83
2	0.92	0.91	0.91	**0.94**	0.93	0.93	0.93	**0.94**	**0.94**
3	**0.81**	**0.81**	0.80	0.80	**0.81**	0.79	0.79	0.78	0.76
4	**0.82**	0.80	0.79	**0.82**	0.81	0.80	0.80	**0.82**	**0.82**
5	**0.92**	**0.92**	0.91	0.91	0.91	0.90	0.90	0.91	0.90
6	0.85	0.86	0.87	**0.88**	0.87	**0.88**	**0.88**	0.85	0.83
7	**0.98**	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.96
8	0.80	0.80	0.80	**0.82**	0.81	0.79	0.79	0.81	0.79
AVG	**0.88**	**0.88**	0.87	**0.88**	**0.88**	0.87	0.87	0.86	0.86

Table 10. Table displaying rounded Precision@10 scores for different PCA settings; "-X/K" denotes the removal of the first X out of K components in PCA reconstruction. Mean computed over 9 runs.

$ID\ Image$	$P@10$								
	$-1/20$	$-1/50$	$-1/396$	$-2/20$	$-2/50$	$-2/396$	$-3/20$	$-3/50$	$-3/396$
0	0.87	**0.88**	**0.88**	0.83	0.84	0.84	0.84	0.84	0.85
1	0.84	**0.85**	0.85	0.83	**0.85**	0.84	0.80	0.81	0.79
2	0.86	0.86	0.85	0.87	0.87	0.87	**0.91**	**0.91**	0.90
3	0.80	0.78	0.79	**0.82**	0.79	0.81	0.75	0.68	0.71
4	0.71	0.73	0.69	0.71	0.72	0.69	0.73	**0.74**	0.73
5	0.87	**0.88**	**0.88**	0.85	0.86	0.86	0.86	0.86	0.84
6	0.86	**0.89**	0.88	0.83	0.86	0.85	0.86	**0.89**	0.86
7	**0.97**	**0.97**	0.96	0.96	0.96	0.96	0.95	0.95	0.96
8	0.80	**0.81**	**0.81**	0.70	0.71	0.72	0.70	0.69	0.70
AVG	0.84	**0.85**	0.84	0.82	0.83	0.83	0.82	0.82	0.82

Table 11. Feature extracted has been standardized before to extract features and applying PCA. "-X/K" denotes the removal of the first X out of K components in PCA reconstruction. Mean computed over 9 runs.