IMAGE COMPRESSION METHODS USING PYTHON

Vishwas Shivakumar, Saurav Subash Prasad

ABSTRACT

Image compression is the art of reducing the size of data that comprises a digital image while maintaining image quality. An image requires a large amount of digital data to be stored. Despite technical advancements in the storage space necessary to keep data, the demand for storage capabilities is not being met by availability. Image compression may be the sole option in such a situation. In this paper, we attempt to provide an overview of several image compressing algorithms, we will use four different linear image compression techniques: K-means, Principal component analysis (PCA), Singular Value Decomposition (SVD), and Non-Negative Matrix Factorization (NMF) and evaluate the benefits and drawbacks of each.

INTRODUCTION

Image compression is in high demand as the need to store and transmit images is growing faster than the current capacity to analyze all the data. Image compression aims to decrease the number of bits required to represent an image, as well as the relevance and redundancy of image data, and to efficiently store or transmit data. Image compression is generally of two types, “Lossy” and “Lossless”. "Lossy" compression is a type of data encoding that compresses data by deleting a portion of it. Different compression schemes utilize different algorithms to determine how to successfully discard the data while maintaining the image within an acceptable level of quality as specified by the user's needs, but once the data is lost under "lossy" compression techniques, it's gone for good. When it comes to protecting digital information, however, we generally prefer to avoid compressing the data unless we can compress it "losslessly." The term "lossless" compression refers to the ability to reduce the size of any arbitrary piece of digital content while also restoring it to its original size without losing any information in the process.

As content creators, we deal with a lot of photographs captured with a DSLR. These images are stored in either RAW format, which contains all information of an image, or JPEG format, which is a compressed format. Unfortunately, JPEGs are a type of "lossy" compression. Instagram is now the most popular platform for content creation, and the maximum size of an Instagram post is 1080px x 1920px with 256-bit colors and 72dpi, with an image size of 1.9 MB. When our very high-quality photographs, which were normally 30 MB in size, were compressed to 2MB with virtually no loss of clarity when uploaded to Instagram, we were intrigued. This led us to perform additional image compression research utilizing techniques we learnt throughout the semester. We proposed four methods we could use: PCA, K-means, SVD, and NMF.

RELATED WORK

1) Image Compression: Review and Comparative Analysis [4]

This paper discusses importance of image compression along with various image compression approaches and provides a comparison of Huffman encoding, Arithmetic coding, run length coding, Transform coding, and Wavelet coding.

2) The Essential Guide to Image Processing, Academic Press [6]

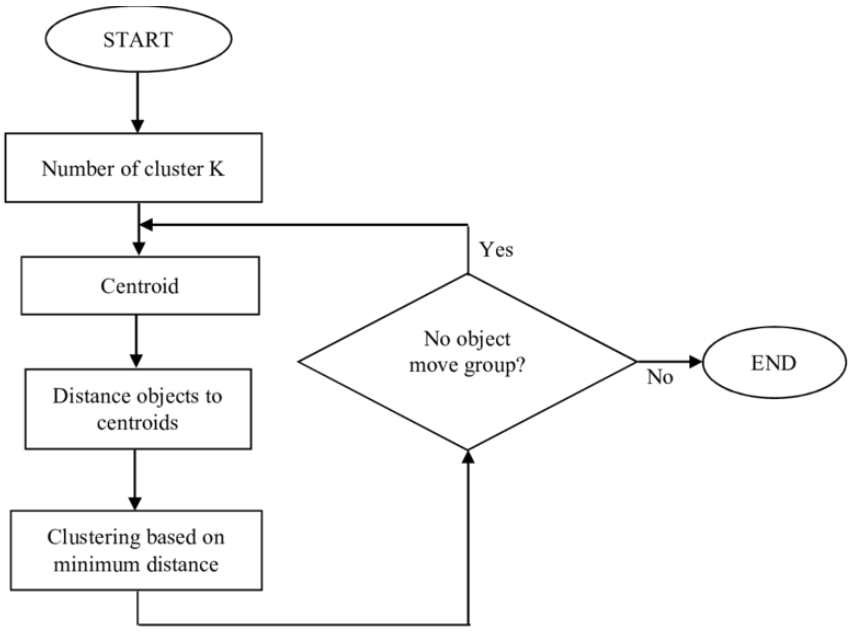
This book provides a comprehensive and cutting-edge approach to image processing with a complete introduction to major applications such as picture watermarking, fingerprint recognition, face recognition, iris recognition, and medical imaging.

3) Application of K-means Algorithm in Image Compression [7]
This report underlines the issues that should be addressed when employing clustering. It also includes, a realistic example of image compression using K-Means.

METHODS

K-MEANS Clustering

The iterative K-means algorithm divides the dataset into K distinct, non-overlapping subgroups (clusters), each with a single data point. While keeping the clusters as distinct (far) apart as possible, it aims to make the intra-cluster data points as comparable as possible. To minimize the sum of the squared distances between the data points and the cluster centroid, which represents the average of the data points in the cluster, data points are distributed to clusters in a specific way. As the degree of variance inside the cluster drops, the homogeneity (similarity) of the data points within the cluster rises.



To identify k colors that best represent its Similar colors, we shall use k-Means clustering. These k-colors will represent the algorithm's centroid spots. The centroid points of each pixel will then be used in place of their values. Compared to the whole color combination, the color combination created with only k values will be relatively small. We'll experiment with various k values and look at the final image. The original image cannot be recovered from the compressed image because K-means utilizes lossy compression, which is different from lossless compression.

PRINCIPAL COMPONENT ANALYSIS (PCA)

To maintain as much of the original data's variability as possible, Principal Component Analysis (PCA), a linear dimensionality reduction technique (algorithm), converts a collection of correlated variables (p) into a smaller k (kp) number of uncorrelated variables called principal components. Image compression, a method to reduce the size of an image in bytes while maintaining as much of the image's quality as feasible, is one of the applications for PCA.

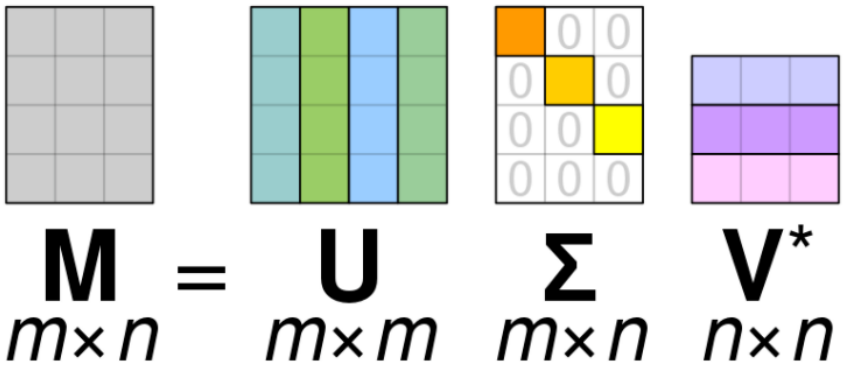
Algorithm 1 Principal Component Analysis

function PCA(X)
  Compute the sample mean and the sample covariance matrix by:
  $\mu_x = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$
  $\Sigma_x = \frac{1}{n-1}\sum_{i=1}^{n} (\mathbf{x}_i - \mu_x)(\mathbf{x}_i - \mu_x)^T$
  Compute the eigenvalues and eigenvectors of Σ_x
  Define the transformation matrix $T = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d]$ with the d eigenvectors associated to the d largest eigenvalues.
  Project the data X into the PCA subspace:
  $\mathbf{y}_i = T\mathbf{x}_i \quad \text{for} \quad i = 1, 2, \ldots, n$
  return Y
end function

SINGULAR VALUE DECOMPOSITION (SVD)

According to (SVD), every (m n) matrix A can be written as a product. A=U V T
Where U and V are orthogonal matrices and the Σ matrix is made up of descending non-negative values on its diagonal and zeros everywhere else. The entries σ1 ≥ σ2 ≥ σ3 ≥ … ≥ 0 on the diagonal of are known as the singular values (SVs) of A. Σ Maps the j^{th} unit coordinate vector of n-dimensional space to the j^{th} coordinate vector of m-dimensional space, scaled by factor σj. U and V are orthogonal because they correspond to rotations (possibly followed by reflections) of m-dimensional and n-dimensional space, respectively. As a result, Σ only changes the length of vectors.

Images are represented in a rectangular array, with each element representing a pixel's grayscale value. For colored images, we have a 3-D array of size nm3, where n and m represent the number of pixels vertically and horizontally, and we store the intensity for red, green, and blue for each pixel. We generate a low-rank approximation of a matrix that represents an image separately for each color. The resulting 3-dimensional array will be a close match to the original image.



NON-NEGATIVE MATRIX FACTORIZATION (NMF)

NMF is a method for factorizing a non-negative matrix, V, into the product of two lower rank matrices, W and H, such that WH approximates an optimal solution of V. This is an unsupervised learning algorithm for reducing data dimensionality to lower-dimensional spaces. This method is commonly employed in recommendation systems, text mining, and image analysis applications. NMF is a cutting-edge feature extraction algorithm that is useful when there are many ambiguous features with low predictability. It is capable of producing meaningful patterns, topics, and themes.

General framework of NMF algorithms

Input: A nonnegative matrix $\boldsymbol{V} \in \mathbb{R}_{+}^{n \times m}$ and the factorization rank r

- Generate some initial matrices $\boldsymbol{W}^{(0)} \geq 0, \boldsymbol{H}^{(0)} \geq 0$
- **for** $t = 1, 2, \ldots,$ stopping time
  $\boldsymbol{W}^{(t)} = \text{update } (\boldsymbol{V}, \boldsymbol{H}^{(t-1)}, \boldsymbol{W}^{(t-1)})$
  $\boldsymbol{H}^{(t)^T} = \text{update } (\boldsymbol{V}^T, \boldsymbol{W}^{(t)^T}, \boldsymbol{H}^{(t-1)^T})$
  end for

Output: A rank-r NMF of $\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H}$, for $(\boldsymbol{W}, \boldsymbol{H}) \geq 0$

The goal here is to use NMF factorization as a tool to obtain an optimally compressed version of an original image while retaining its main characteristics.

The columns of the target matrix V will be made up of the image's pixel intensities. The columns of W will be the basis vectors and the image's main features. The columns of H are known as encodings, and they are the coefficients that represent the image with a linear combination of basis vectors.

EXPERIMENTAL SETUP

The algorithms discussed above were implemented in Python 3.8.8 using libraries such as sklearn, NumPy, pandas, matplotlib, PIL, ipywidgets, skimage, and others.

We used the inbuilt Kmeans method to compress the image for K means and did not change any default parameters except the number of clusters k. Based on the number of clusters k, each pixel in the image is assigned to one of the clusters k based on its color in this method. By plotting the Squared Loss Vs k graph, we can determine the best value of k. We saw the results for the following values of k = [2,5,10,50,100,1000] to compare and understand how the algorithms work.

PCA was also implemented in sklearn using the built-in method, with the number of principal components simply specified. We accomplished this by first converting the color image to greyscale and then using pca on it. For our image compression, we plotted the explained variance vs number of components plot and chose the value that explains 95% of the variance. But to see the difference in compression for different number of components we selected the following values [25, 50, 100, 130, 250,300].

To calculate rank-k approximations in SVD, we used NumPy linalg functions. The image was divided into three distinct channels, each of which was subjected to SVD before being merged to produce the compressed color image. We created an interactive slider to select the rank and then compress the image. The implementation of NMF was very similar to that of SVD because their formulas are very similar. The only difference here is that we applied our compression to a greyscale image. We tried NMF for different values of rank [5, 50, 100,200,250,300] and for 300 iterations each.

Results

In Kmeans we saw that the best way to select the value of k (number of clusters) was by plotting a graph for k vs squared loss and choosing the value of k where the elbow is formed to retain most of the quality and minimize the loss. From the plot below we can see that an optimal value will be around k=100. To compare the compressed image with original and a bad compression, below we have shown the original image, k=2 image and k=100 image.

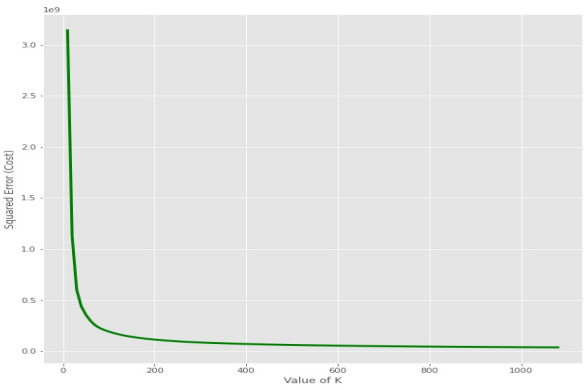
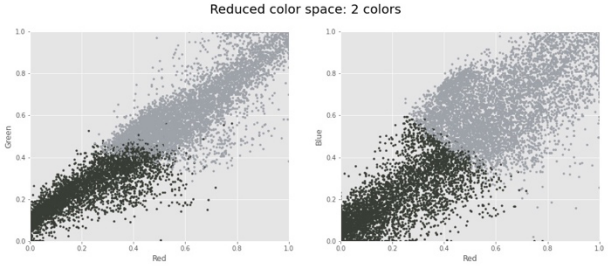
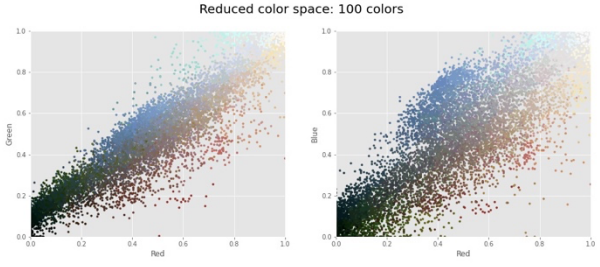


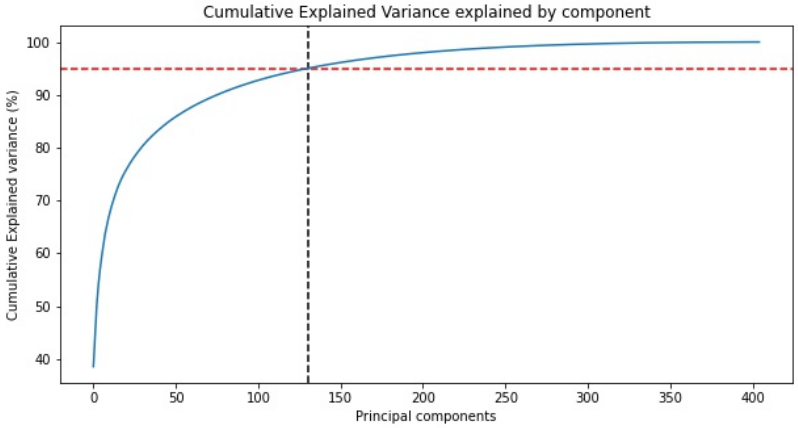
Figure:(left to right) Original image, Compressed image k = 2, compressed image k = 100

The way k means work is that we group the pixels into k clusters based on their color, hence when k = 2 we can see only 2 colors in the image. But when k = 100 we see 100 colors. There are 16million possible colors in an 8-bit color image, but human eyes can't perceive all that. We take advantage of this fact to compress the image, reduce the size and keep the quality. The reduced color spaces can be seen below

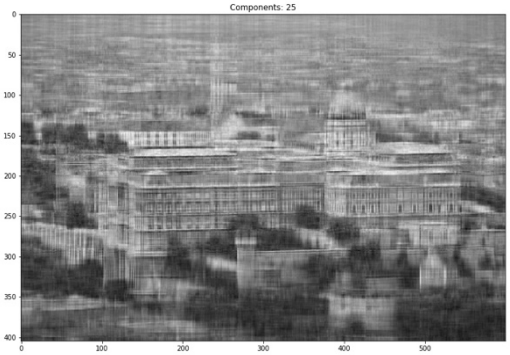




In PCA the selection criteria for the number of components were by plotting the number of components vs % explained variance, from this graph we get a clear picture of for what number of components 95% variance of the data is explained. Which is shown below.



From this plot we know that selecting 130 components would give us the best compression for the image. Below are the results for 30 components, 130 components and the original image. We can compress the image in color; we just must apply PCA for each of the R G B channels.



Next, we look at how SVD performs for image compression. To select the best approximation, we plot a graph of accuracy vs rank and select the rank where accuracy is more than 90%. We have implemented this for all three channels R G B and made it interactive so you can select the rank for approximation and run to see the output, below are the results from the same.

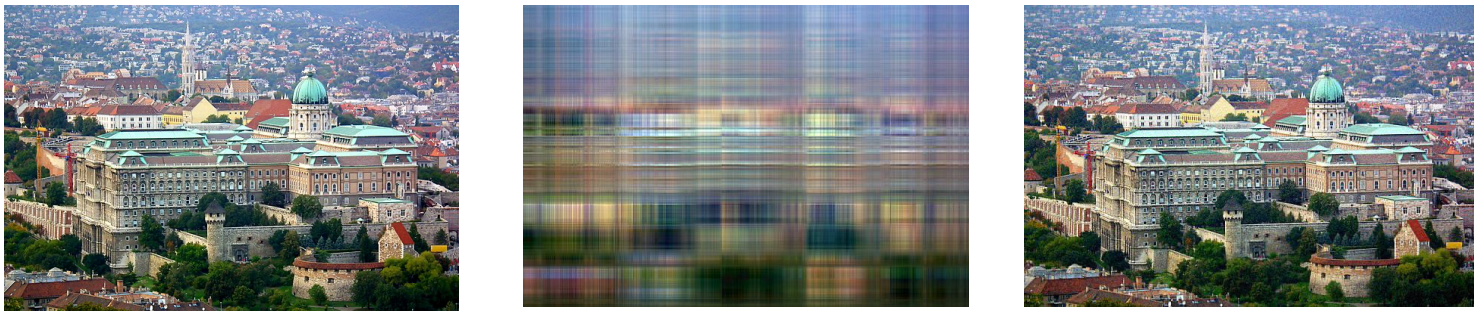
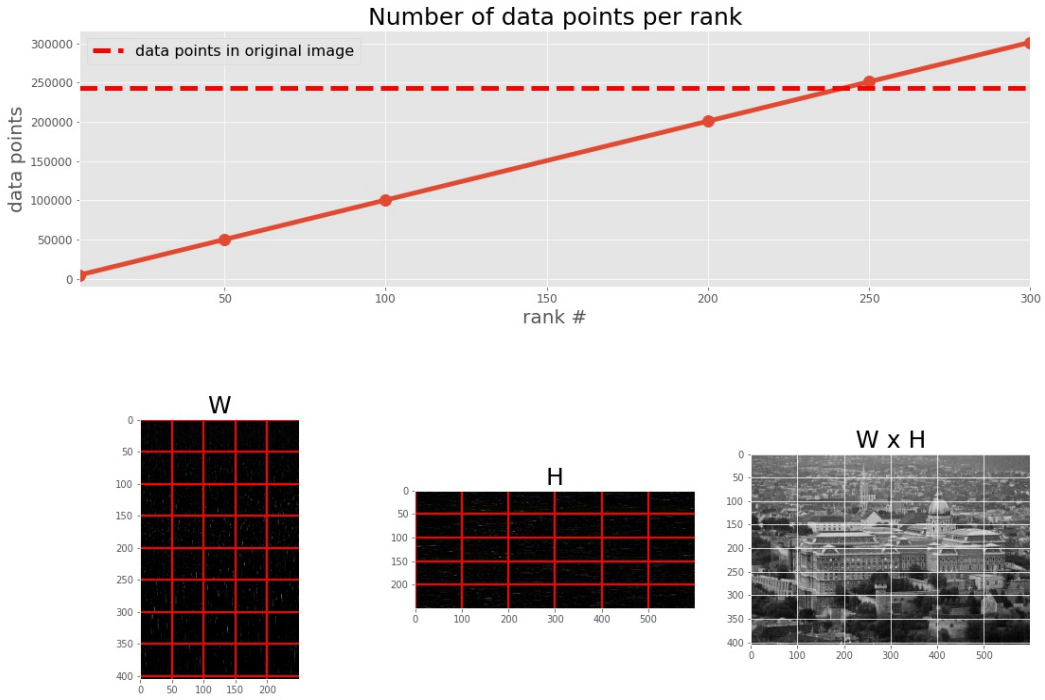
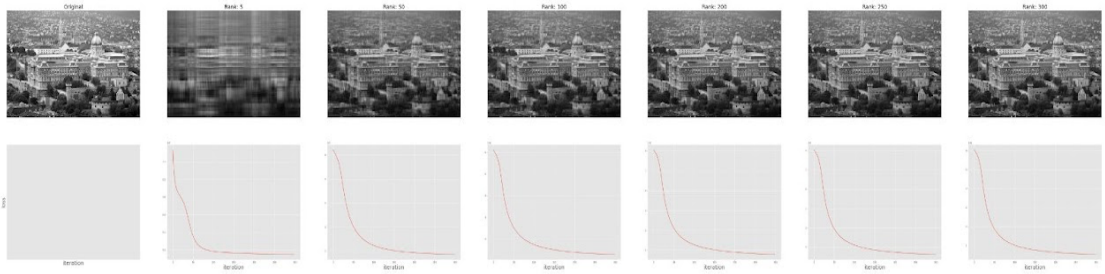


Figure:(left to right) Original image, rank 3 compressed image, compressed rank 150 image

NMF works in a very similar way to SVD, but it is more useful to extract the features from a face in image processing. In our experiments we used a grayscale image to compress the images. It can be done for a color image as well; we just must apply NMF for RGB channels. Below graphs show how the size of the image's changes over different ranks. The below image shows how NMF works. And we also have shown the comparison of compressed and original images.





Conclusions

Key findings

1) K-means is a good color image compression method in which we can specify the number of colors as k clusters which gives us control over the colors.

2) For PCA, by plotting the explained variance graph, we can easily see the number of components that explain 90% of the variance and choose that to compress our images. It works better when we have a group of similar images.

3) We can use SVD to compare the rank K approximations to the original image and choose the best k value for compression.

4) In NMF, think about the columns in matrix W as features of V which means the coefficients in H are the weights of each of these features. Due to this reason, it is best for feature selection in face recognition systems.

Limitations

1) These linear image compression methods work quick for images with small file size, but they computationally take a lot more time if the size of the image is large.

2) Processing time required for color image processing is a lot for the methods discussed here. We should look for other better methods for this process.

3) Although NMF works well for image compression, the main use of NMF in image processing is for feature extraction from a face

Future work

1) Implement color image processing for all the methods discussed here.

2) Optimize the Algorithms for faster processing of images with large file sizes.

3) Create a web application where the end user can upload an image file and select the method they want to use for compression and select the parameters. Once the image is compressed, we will give an option to save the compressed image in the format they like.

CONTRIBUITION

Considering our team was only two people, we divided the work equally and were each accountable for two techniques. Saurav Subash Prasad did the PCA and K-means techniques, while Vishwas Shivakumar performed the SVD and NMF approaches.

REFERENCES

1) Kumar, S. (2021b, December 14). Image Compression using K-Means Clustering - Towards Data Science. Medium. https://towardsdatascience.com/image-compression-using-k-means-clustering-aa0c91bb0eeb

2) Beezer. (2014). Linear Methods for Image Compression - Math 420, Prof. Beezer. http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-meacham-i+mage-compression-present.pdf

3) Lazorchak, B. (2013, May 22). Hey Content Creator: Make Mine Lossless! | The Signal. https://blogs.loc.gov/thesignal/2013/05/hey-content-creator-make-mine-lossless/

4) Kulchandani, S., Pal, H., & Dangarwala, J. (2014, November 11). Image Compression: Review and Comparative Analysis. IJERT. https://www.ijert.org/research/image-compression-review-and-comparative-analysis-IJERTV3IS110464

5) Parmar. (2012). A Review of Image Compression. Research Gate. https://www.researchgate.net/publication/257868354_A_Review_of_Image_Compression

6) Al Bovik, The Essential Guide to Image Processing, Academic Press, 2009, ISBN 9780123744579, https://doi.org/10.1016/B978-0-12-374457-9.00031-7 (https://www.sciencedirect.com/science/article/pii/B9780123744579000317)

7) Xing Wan 2019 IOP Conf. Ser.: Mater. Sci. Eng. 563 052042 (https://iopscience.iop.org/article/10.1088/1757-899X/563/5/052042/pdf)

8) S, Prasantha. (2007). Image compression using SVD.(https://www.researchgate.net/profile/Prasantha-S/publication/338621459_Image_compression_using_SVD/links/5ef89a7792851c52d6041ba4/Image-compression-using-SVD.pdf)

9) Lee, D., Seung, H. Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999). https://doi.org/10.1038/44565