FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER'S THESIS

Title:	Evaluating performance of an image compression scheme based on non-negative matrix factorization
Student:	Bc. Marek Pikna
Supervisor:	doc. Ing. Ivan Šimeček, Ph.D.
Study Programme:	Informatics
Study Branch:	System Programming
Department:	Department of Theoretical Computer Science
Validity:	Until the end of summer semester 2019/20

Instructions

The aim of this work is to design and implement an image compression scheme based on non-negative matrix factorization (NMF) and to evaluate the performance of this compression scheme. In order to achieve this, the goals are following:

- Study the theory related to image compression [1].
- Study NMF and its current applications [2] [3].
- Analyze existing formats used for representing uncompressed images and decide how can they be used for a compression scheme using NMF.
- Design and implement a compression scheme using NMF [4].
- Evaluate using both objective and subjectiv metrics the performance of this compression scheme with respect to various parameters of NMF (amount of iterations, rank, variation of NMF algorithm).
- Decide whether a certain representation of an uncompressed image is better suited for this compression scheme than another one.
- Compare the results with currently used image compression methods.

References

[1] RABBANI, Majid a Paul W. JONES. Digital image compression techniques. Bellingham, Wash., USA: Spie Optical Engineering Press, 1991. ISBN 978-081-9406-484.
[2] Lee, Daniel & Sebastian Seung, H. (1999). Learning the Parts of Objects by Non-Negative Matrix Factorization. Nature. 401. 788-91.
[3] Gillis, Nicolas. (2014). The Why and How of Nonnegative Matrix Factorization. Regularization, Optimization, Kernels, and Support Vector Machines. 12.
[4] SAHU, Khushboo Kumar a K. J. SATAO. Image Compression Methods using Dimension Reduction and Classification through PCA and LDA: A Review [online]. , 4 [cit. 2019-01-11].
Dostupné z:https://www.ijsr.net/archive/v5i5/NOV163957.pdf

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 16, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Evaluating performance of an image compression scheme based on non-negative matrix factorization

Bc. Marek Pikna

Department of Theoretical Computer Science

Supervisor: doc. Ing. Ivan Šimeček, Ph.D.

May 9, 2019

Acknowledgements

I would like to thank my supervisor doc. Ing. Ivan Šimeček, Ph.D. for all the valuable feedback provided during the course of writing this thesis. I would also like to thank my family for all they have done for me and my friends for being there for me whenever I was in need. Last but not least, I would like to thank Brandon McCartney for his unlasting support and positivity.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on May 9, 2019 .....

Czech Technical University in Prague

Faculty of Information Technology

Citation of this thesis

Pikna, Marek. *Evaluating performance of an image compression scheme based on non-negative matrix factorization.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Tato magisterská práce analyzuje potenciál komprese obrazu pomocí faktorizace nezáporných matic, matematické metody, která faktorizuje matici obsahující nezáporné elementy do dvou faktorů. Jelikož tato technika pouze aproximuje matici a není přesnou faktorizací, dochází ke ztrátě informací. Z toho důvodu lze chápat tuto metodu také jako ztrátovou kompresní metodu. V práci jsou navržena tři kompresní schémata, založena na různých barevných modelech a zároveň jsou tato schémata otestována. Nejlepší kompresní schéma je také porovnáno s JPEG kompresní metodou. Výsledky ukazují, že ačkoliv tyto metody nedosahují lepších kompresních poměrů, ztráta informace dovede být nižší a kvalita obrazu tedy vyšší, než v případě JPEGu. Práce zároveň navrhuje další možná vylepšení, která by mohla vylepšit kompresní poměry a kompresní časy.

Klíčová slova Faktorizace nezáporných matic, redukce dimenzionality, barevné modely, barevné prostory, ztrátová komprese, komprese obrazu

Abstract

This master's thesis analyzes the potential of image compression utilizing non-negative matrix factorization, a mathematical technique which factorizes a matrix containing non-negative elements into two factors. As this technique is only an approximation and not an exact factorization, information loss occurs, rendering the technique a lossy compression technique. Three compression schemes are proposed, based on different color models, and tested, with the best performing one compared to JPEG. The results show that while non-negative matrix factorization does not achieve better compression ratio, the information loss can be lower and the image quality therefore higher. Further improvements of the algorithm which could improve the compression ratio are proposed.

Keywords Non-negative matrix factorization, dimensionality reduction, color models, color spaces, lossy compression, image compression

Contents

List of Figures

List of Tables

Introduction

Data encoding and consequently data compression are both problems which lie at the heart of many modern technologies - digital television, videogames, mobile communications, security cameras and all other forms of multimedia. As the amount of data only grows in the current world, the quality of compression ends up becoming a very serious problem, since good compression can significantly reduce the costs of data storage as well as the costs and speed of data transfer.

To put the problem into perspective, in order to store images in the resolutions currently considered as high resolutions (1920x1080 pixels, the second most common resolution used on desktop devices [5]) using an uncompressed standard encoding, the filesize would be almost 6 megabytes. Such a high size impacts many areas, such as speed of transfer or storage costs. Fortunately, modern image compression formats such as *PNG* or *JPEG* are able to reduce this size remarkably.

Currently, images contribute to the amount of data on the internet significantly - not only are images a common form of media for professional purposes but some of the currently largest platforms on the internet are based on image sharing and image hosting. One modern social media platform centered around image sharing needs to be able to store and serve over 67 million new posts each day [43]. Being able to obtain these images fast and in good quality is therefore highly important for both users, as well as for the owners of these services, in order to store data efficiently.

In the recent years with growth related to machine learning and related areas, such as artifical intelligence, new applications of mathematical concepts have been discovered. Some of these concepts which are enjoying high success are algorithms related to *dimensionality reduction*. These algorithms aim to

reduce the number of random variables under consideration [39].

While these algorithms have been enjoying success mostly in areas such as data mining or machine learning, their nature of reducing the amount of random variables under inspection means that the algorithms can also be understood as compression algorithms. One of the algorithms used for dimensionality reduction and the one which this thesis focuses on is *non-negative matrix factorization* (abbreviated as *NMF*). The *NMF* algorithm is currently used in areas such as facial recognition or astronomy. Heart of the algorithm is the factorization of a matrix consisting of non-negative values into two factor matrices.

The research in image compression methods using dimensionality reduction algorithms is currently being performed - another dimensionality reduction algorithm and its potential usage for image compression which has been well researched is the *singular value decomposition* algorithm, for example in [30]. This algorithm, just like the *NMF*, factorizes a matrix - however, without restricting the values to be non-negative. While certain similar research to see whether *NMF* can be used for image compression exists, the works are related to very specific use-case scenarios.

Thus, this thesis aims to analyze the potential of the *NMF* algorithm as a tool for image compression. In order to achieve this, the following points will be explored in the thesis:

- *NMF* and its current applications will be studied (Chapter 1).
- The theory and practice related to digital image encoding and image compression will be explored and described together with modern image compression. (Chapters 2 and 3)

By analyzing these concepts, a proof of concept image compression algorithm using *NMF* will be designed and implemented. By doing so, these issues will be addressed:

- Whether a certain way of representing an uncompressed image is better suited for non-negative matrix factorization.
- Utilizing both subjective as well as objective metrics commonly used for evaluating quality of image compression, the performance of the proof of concept compression scheme will be evaluated.
- How well suited *NMF* is for usage as an algorithm for image compression.

At the end of the thesis, the proof of concept algorithm will be compared to a state of the art image compression algorithm and possible points related to further analysis or further improvements of the algorithm or its implementation will be discussed.

The first three chapters are related to the theoretical part of the problem, studying *NMF*, image encoding and image compression. The following chapters are related to the practical part of this thesis - design and implementation of the image compression algorithm and its evaluation.

Related work

Non-negative matrix factorization is a relatively new technique and majority of the existing works related to non-negative matrix factorization are related to its applications in machine learning and others. Some of the research existing in the field of audio/video processing is related, but not limited, to:

- Separating voices in speech mixtures or voice from background sounds. [49]
- Separating singing voice or musical instrument in polyphonic mixtures. [10]
- Video action recognition. [25]
- Compression.

While the potential for the usage of non-negative matrix is high, its current usage is quite limited. The existing works which are specifically related to image compression using non-negative matrix factorization found are the following:

- Image compression using Constrained Non-Negative Matrix Factorization [20] is a work focusing mainly on non-negative matrix factorization and the effect of using a slightly different method on image compression. The work reports better time results when using constrained non-negative matrix factorization when compared to using the general approach, but with very low improvements in the image quality.
- An Image Compression Scheme in Wireless Multimedia Sensor Networks Based on NMF [24] utilizes a compression scheme based on non-negative matrix factorization but focuses more on the possibility of using this compression scheme in a sensor network, rather than on the image compression itself and its potential quality.

Compressing images using non-negative matrix factorization has also been found in other works, but usually as an illustrative example rather than the target of research.

This work aims to analyze the potential usability of an image compression scheme based on NMF strictly as a compression tool - the effect of NMF parameters on the image quality and explore a number of possible schemes related to the way the images would be represented.

Chapter 1

Non-negative matrix factorization

This chapter discusses the *non-negative matrix factorization* - defines the problem, describes some of the existing solutions to the problem and offers some observations. By doing so, the basics for the rest of the thesis are provided.

Non-negative matrix factorization as a problem was first formulated by Paatero and Tapper in [36], but the works which have given this problem far more popularity are the works of Lee and Seung [28], where *NMF* was applied to areas of machine learning and artificial intelligence - more specifically to facial recognition and discovering semantic features in encyclopedic articles.

1.1 Problem definition

Let V be a $n \cdot p$ non-negative matrix, (i.e. with $x_{ij} \geq 0$, denoted $X \geq 0$), and $r > 0$ an integer. Non-negative matrix factorization consists of finding an approximation

$$V \approx WH \tag{1.1}$$

where W, H are $n \cdot r$ and $r \cdot p$ non-negative matrices, respectively - meaning all the elements of the matrices are non-negative. In practice, the rank r is often chosen such that $r \ll min(n, p)$. This is due to the reason that in many common applications of non-negative matrix factorization, the information contained in the matrix V is summarized and split into r factors as the columns of W [11].

It should be noted here that the name of the problem might be misleading, as the term *factorization* is usually understood more as an exact decompo-

sition, whereas *NMF* is in reality an approximation. Thus, the problem is called *non-negative matrix approximation* in certain other works, such as [46].

1.2 Problem solution

In this section, two of the commonly used algorithms for solving the non-negative matrix factorization problem will be described. The first of these two algorithms will be the algorithm based on *multiplicative updates* as used in [28], where the attention to part-based analysis, simplicity of the *multiplicative updates* and interpretability of the results helped to spread the influence into many other research fields, such as image processing or text processing [15]. Due to these reasons, this will be one of the algorithms described. The second algorithm which will be described is the *alternating least squares* algorithm, which is the earliest algorithm proposed for solving the non-negative matrix factorization problem (positive matrix factorization in the original work) [36].

The reason for choosing these two algorithms is that both of them are very commonly used in practice. Other algorithms exist and are often researched, example being the *projected gradient* method [21]. However, they will not be explored within this thesis.

1.2.1 Multiplicative updates

The algorithm described here is described more thoroughly in [29], a work by Lee and Seung where the algorithms are described in detail together with proving correctness of the algorithms.

In order to find an approximate factorization $V \approx WH$, a way to quantify the quality of the approximation needs to be defined. Such a metric (or a cost function) can be constructed by measuring the distance between two non-negative matrices A and B. One of the measures provided in [29] is the square of the Euclidean distance between A and B.

$$||A - B||^2 = \sum_{ij}(A_{ij} - B_{ij})^2 \tag{1.2}$$

This distance is lower bounded by zero and vanishes if and only if $A = B$.

By using this cost function, the non-negative matrix factorization can be formulated as an optimization problem:

Problem 1 *Minimize* $||V - WH||^2$ *with respect to* W *and* H*, subject to the constraints* $W, H \geq 0$.

It is shown in [29] that an algorithm cannot realistically solve this problem by finding a global minimum. However, it is possible using various techniques from numerical optimization which make it possible to find local minimum.

Thus, the multiplicative update rules are a compromise between speed and ease of implementation offered in [29] for solving this problem. The multiplicative updates can be described as an algorithm in the following way:

Input: Non-negative matrix V
Output: Non-negative factors W and H

- Initialize W and H as non-negative matrices
- Until $||V - WH||^2$ is minimized, update W and H by computing the following, with n as an index of the iteration:

$$H_{[i,j]}^{n+1} = H_{[i,j]}^{n} \frac{((W^n)^T V)_{[i,j]}}{((W^n)^T W^n H^n)_{[i,j]}} \tag{1.3}$$

and

$$W_{[i,j]}^{n+1} = W_{[i,j]}^{n} \frac{(V(H^{n+1})_{[i,j]}^T}{W^n H^{n+1} (H^{n+1})_{[i,j]}^T} \tag{1.4}$$

Algorithm 1: Multiplicative update algorithm for NMF

It is shown in [29] that the Euclidean distance $||V - WH||$ (and consequently the cost function $||V - WH||^2$) is nonincreasing under these rules. The work by Lee and Seung also considers another possible cost function and proves the convergence of these rules.

1.2.2 Alternating least squares method

Alternating least squares method is the first algorithm proposed for solving the non-negative matrix factorization problem, which was proposed in the work by Paatero. [36] Fixing either of the factors W or H, the problem essentially becomes a least squares problem, which is commonly used in regression analysis.

The alternating least squares problem then solves NMF using the algorithm shown in 2.

As the least squares algorithm does not enforce the constraint of non-negativity, all the negative elements in matrices are set to 0 after each evalu-

Input: Non-negative matrix V
Output: Non-negative factors W and H

- Initialize W as a random dense matrix
- Until a stopping condition:
  (LS) Solve $min_{H \geq 0} ||V - WH||^2$
  (NONNEG) Set all the negative elements of H to 0.
  (LS) Solve $min_{W \geq 0} ||V^T - H^T W^T||^2$
  (NONNEG) Set all the negative elements of W to 0.

Algorithm 2: Basic Alternating least squares algorithm for NMF. [37]

ation of the least squares problem.

The name of the algorithm reflects its nature where it keeps alternating between solving the least squares problem for one fixed matrix and then the other.

1.3 Common NMF applications

In this section, a short example of a common application of non-negative matrix factorizations will be provided, for the purpose of showing a specific example so that the reader may become more adjusted to the problem as well as showing certain observations about the properties of non-negative matrix factorization.

1.3.1 Part-based analysis

What has inspired the work of Lee and Seung [28] was the human activity of recognizing objects from basic parts - especially when using human vision which is shown to detect the presence or absence of features (parts) of physical objects in order to recognize them [7].

Thus, assuming that features of an object would be independent and it would be possible to compile all the features together, an object could be described formally as:

$$Object_i = Part_1(b_{i1}) \text{ with } Part_2(b_{i2}) \text{ with...}, \tag{1.5}$$

where b_{ij} either has the value *present* if part i is present in object j or the value *absent* if part i is absent in object j.

If the possible states *present* and *absent* in the model are replaced by non-negative values, it is possible to signify not only the presence or absence of a

feature but also its quantity or significance ($b_{ij} \geq 0$). Thus, mathematically, a description of an object could look the following:

$$Object_i = b_{i1} \cdot Part_1 + b_{i2} \cdot Part_2 + ... \tag{1.6}$$

[15]

When utilizing non-negative matrix factorization, the matrix W can be considered the set of features present in the data and matrix H the set of hidden variables. Non-negative matrix factorization can also be implemented as:

$$v_i \approx W h_i \tag{1.7}$$

Represented this way, the concept of non-negative matrix factorization can be understood intuitively - each column in the original matrix V is a data point. Each column in the matrix W is a basis element and columns of the matrix H give the coordinates of a data point in the basis W. The product matrix WH (the approximation of V) is a linear combination of the column vectors - the features extracted from the data points and its significance in the data point.

This way, features and parts can be extracted from the original data and understood, as shown in the next subsection on the image learning example.

1.3.2 Image learning

Digital image processing is a field which is currently enjoying high popularity when it comes to research. Image processing is the field thanks to which it is possible to extract features from images or recognize various patterns. Principal component analysis *(PCA)* is a dimensionality reduction technique similar to *NMF* commonly used for recognizing faces in images [6] - however, without the non-negativity constraint. It has been shown in [28] that *NMF* can be used in a similar fashion while potentially classifying the data in a way which is easier to be understood.

The Figure 1.1 has been taken from [28], where a database of 2,429 facial images has been taken and used to create a matrix V. By applying both *NMF* as well as *PCA* to the matrix, feature and coeffiecient matrices were created and particular instance of a face was approximately represented by a linear superposition of basis images. The coefficients used in the linear superposition are shown in the montages, where black pixels indicate positive values and red

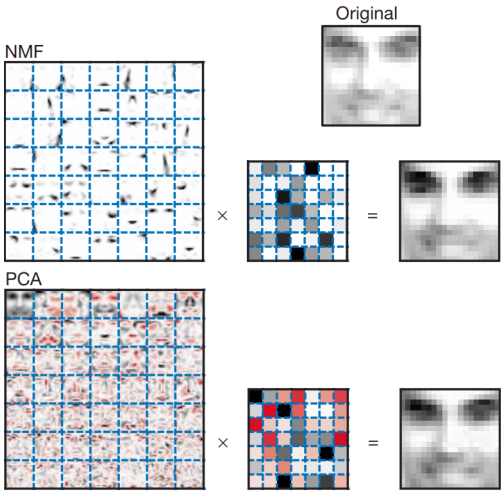


Figure 1.1: Comparison between dimensionality reduction algorithms - *NMF* and *PCA*, as shown in [28].

pixels indicate negative values. It can be seen on these montages that while *NMF* can be used for the same purposes as *PCA*, the main strength of the algorithm is that certain parts can be recognized meaningfully - for example parts of a nose and such, whereas in the case of *PCA*, discovering meaning in the matrices is difficult.

It is for these reasons that non-negativity is a powerful and meaningful constraint, as parts are never subtracted from the particular instance (as it can happen in the case of *PCA*), but are only added together [15].

1.3.3 NMF properties

Previous sections in this chapter have described non-negative matrix factorization, more specifically the definition of the problem, common solutions and an example usage of *NMF*. In this section, certain properties of non-negative matrix factorization will be pointed out. When the proof of concept image compression scheme is designed, the usefulness of these properties for compression will be discussed.

The first property which will be shown is that the solution to non-negative matrix factorization is **not unique**. A matrix and its inverse can transform the two factorization matrices, for example as:

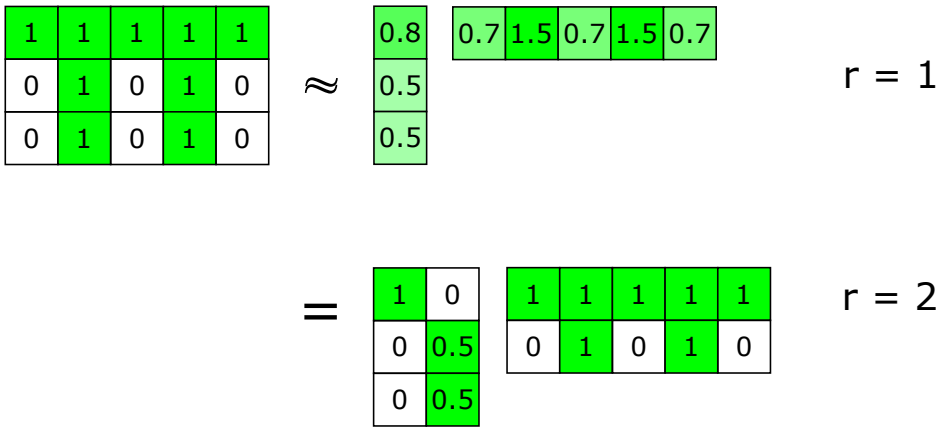
$$V \approx WH = WBB^{-1}H \tag{1.8}$$

If the matrices $\bar{W} = WB$ and $\bar{H} = B^{-1}H$ are non-negative then they form

another solution to the *NMF* problem.[52]

Another important property is that the non-negative matrix factorization is not hierarchical, meaning that the factor matrices using rank r can be completely different to those of rank $r + 1$, as shown in 1.2, where choice of rank $r = 1$ provides only approximation of the target matrix V yet rank $r = 2$ makes it possible to calculate $V = WH$ instead of the approximation. Due to these reasons, the results provided by using *NMF* highly depend on choice of the rank parameter.

Figure 1.2: Visual display of non-hierarchical properties of NMF. [34]



However, one of the most important properties to note is that non-negative matrix factorization is very difficult to solve and the existing general algorithms solve *NMF* as an optimization problem. Not only that but it has even been proven that non-negative matrix factorization is an *NP-hard* problem.[44] With certain constraints, it is however possible to solve the *NMF* problem in polynomial time - for example it is shown in [22] that in case the matrix V is symmetric and contains a diagonal principal submatrix of rank r, it is possible to solve the problem in polynomial time $O(rm^2)$.

The last property of non-negative matrix factorization to be noted has been shown in the figures above. The factors extracted are often *sparse* - it is precisely for this reason that interpreting results of non-negative matrix factorization is easy in fields such as image processing or text mining (but many others as well).[12]

1.4 NMF parameters

When utilizing non-negative matrix factorization, a number of variables can be chosen - most importantly the rank r affecting the dimensions of the factor matrices and the initialization technique for initializing the factor matrices.

1.4.1 Choice of rank r

The choice of rank r is one of the most important decisions when utilizing non-negative matrix factorization - as when used for the examples above, choice of rank r changes the amount of features to be extracted in order to approximate the target matrix V.

There are several common methods used for choosing the rank r:

- Trial and error - try different values of r and choose the one performing the best for application at hand.
- Estimate the rank r using various statistical approaches (such as by using *SVD*).
- The use of expert insights.

[12]

1.4.2 Initialization techniques

As it has been previously shown, non-negative matrix factorization is an optimization problem. The results of NMF highly depend on the initial values of NMF variables due to the existence of local minima. Some of the ways of initializing the initial factor matrices are:

- Using random non-negative values.
- Searching good values via genetic algorithms.
- Initialize matrices based on feature extraction using *PCA*.
- Initialize matrices based on feature extraction using *SVD*.

[23]

It should be noted here that some of the initialization techniques involve randomness (such as purely using random non-negative values) and some do not - and are rather based on extracting data from the input matrix. The following initialization techniques are utilized and tested for image compression usage in this thesis:

- Nonnegative double singular value decomposition (NNDSVD)
- Random vcol
- Random seeding

Nonnegative double singular value decomposition is a method designed to enhance the initialization stage of the non-negative matrix factorization. The basic algorithm contains no randomization and is based on two SVD processes, one approximating the data matrix and the other approximating positive sections of the resulting partial SVD factors. [8]

Random vcol is an initialization method which forms an initialization of each column of the basis matrix (W) by averaging p random columns of target matrix (V). Random vcol also forms an initialization of each row of the mixture matrix (H) by averaging p random rows of target matrix. [27]

The *random* seeding method is the simplest non-negative matrix factorization seeding method possible and generates matrices randomly.

1.5 NMF and compression

Most common use-case scenarios of non-negative matrix factorization are related to machine learning, as shown in the previous sections. However, *NMF* can also be considered a lossy compression tool (more on lossy compression in chapter 3), as an original matrix of size $n \cdot p$ is approximated by the product of two smaller matrices. Assuming the target matrix V is represented in the same way as the factor matrices W and H, if the amount of elements contained in matrices W and H is lower than the amount of elements in matrix V, then *NMF* was used to perform compression.

Whether the amount of elements in factor matrices W and H is lower than the amount of elements in target matrix V depends on the choice of rank r. More specifically, if the dimensions of matrix V are $n \cdot p$, the dimensions of matrix W $n \cdot r$ and the dimensions of matrix H $r \cdot p$, then this requirement can be formalized as:

$$\begin{aligned} n \cdot r + r \cdot p &< n \cdot p \\ r(n+p) &< n \cdot p \\ r &< \frac{n \cdot p}{n+p} \end{aligned}$$

Considering a square matrix ($n = p$), the following relationship can also be deduced:

$$r < \frac{n \cdot n}{n + n}$$
$$r < \frac{n^2}{2n}$$
$$r < \frac{n}{2}$$

This relationship is also demonstrated in the figure 1.3.

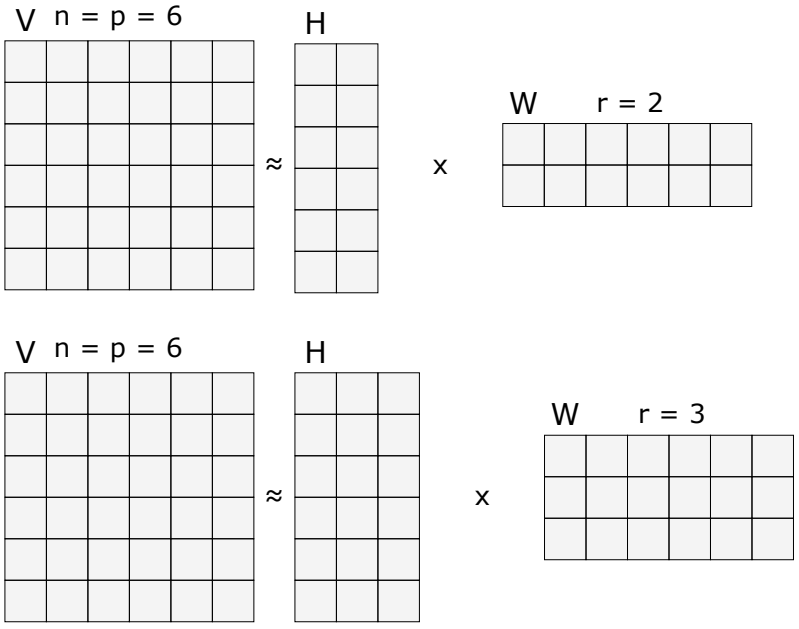


Figure 1.3: Visualization of NMF as a compression tool. When using a rank r lower than $n/2$, the total amount of elements in matrices W and H is less than the amount of elements in matrix V. When the rank r reaches the value $n/2$, the amount of elements is the same.

Chapter 2

Digital image encoding

The second chapter of this thesis is related to digital images, color models and color spaces which are utilized to define digital images. The importance of this chapter is related to creating the image compression scheme and its empirical testing on various different digital image representations. The possible ways of specifying colors in digital images which will be explored are *RGB*, *grayscale* and $Y'C_BC_R$. As this thesis is related to image compression and non-negative matrix factorization and not image processing, only the most important elements of image encoding and color models will be explored.

2.1 Digital images

A digital image I is stored as a matrix of *pixels* (abbreviation for a *picture element*). These matrices described as 2D discrete space are derived from analog images in 2D continuous space through the process called *sampling*. The other important operation which is done when images are converted into a 2D discrete space is *quantization*, which corresponds to a discretizaiton of the intensity values.

The value assigned to a pixel $I[m, n]$ determines its color. Various ways of representing color information (color models and color spaces) exist - as well as transformations between them. The following sections explore the common color encoding options, which will be utilized in the image compression schemes.

2.1.1 Color models and color spaces

A color model is a mathematical model used for describing the way colors can be represented, usually as tuples of numbers (although this is not necessary,

as in i.e. the *grayscale* model, where only one number value is necessary for describing light intensity).

On the other hand, a color space is an organization of colors, which allows for reproducible representations of colors. Color spaces can be defined in an arbitrary way, with particular colors for example associated with numbers (such as in the *Pantone* collection) or can also be defined mathematically - such as is the case of, for example, *sRGB* (standard Red Green Blue) color space.

The difference between a color model and a color space is that a color model is an abstract mathematical model describing the way colors can be represented using numbers (or tuples of numbers). A color space provides the actual mapping which defines the colors represented, using the model. A visualization of different color spaces (Adobe RGB and sRGB) can be seen in the figure 2.1, which shows two color spaces which use the same model for describing colors - but are able to display a different set of colors (also called gamut).[19]

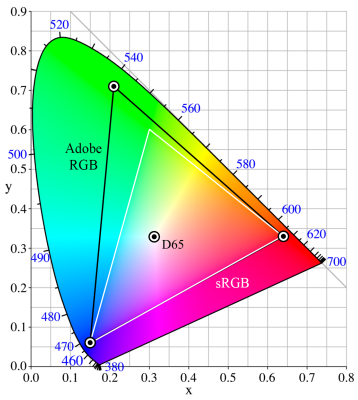


Figure 2.1: Comparison between two different color spaces which both use an RGB color model. The RGB color model contains the visible color spectrum, while both color spaces are able to represent only a subset of these colors. Image source: [32]

The color representations which are going to be explored are the *RGB* color model, the $Y'C_BC_R$ color space which is a transformation using a color space based on the *RGB* model and the *grayscale* model.

2.1.2 RGB color model

The *RGB* color model is closely related to the way the human eye perceives colors with the r (red), g (green) and b (blue) receptors in our retinas.[17] In

order to represent a color, components of each color (red, green and blue) are added together. Since these components are added together, the *RGB* model is also called an additive model.

In order to store image data using the *RGB* color model, the color components need to be quantified. Common way of storing the values of components in a pixel is storing the color intensity value using 8 bits (range [0, 255], where the value 0 indicates no inclusion of the color component and 255 indicates maximum possible inclusion of the component). If all the values of components are equal to 0, the resulting color is black, if all the values of components are equal to the defined maximum value (255 in this case), the resulting color is white. An uncompressed image format which represents images this way is, for example, the Windows BMP.[1]

Thus, encoding an uncompressed image using the *RGB* color scheme with the common 8-bit per component component representation results in each pixel being represented by 24 bits. The size of an image in bytes would then be $width \cdot height \cdot 3$ bytes (not counting the header and other data used by the specific file format).

A colored image together with its decomposition into the R, G and B channels can be seen in the figure 2.2

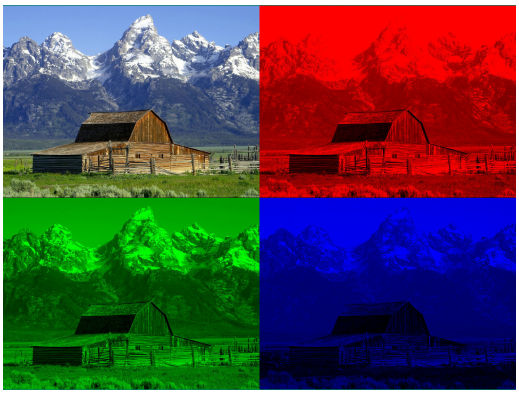


Figure 2.2: An image decomposed into the R, G and B channels as used in the *RGB* color model.

2.1.3 $Y'C_BC_R$ color space

$Y'C_BC_R$, also written as *Y'CbCr*, is an encoding system of colors commonly used in digital image systems, which is defined by a mathematical coordinate transformation from an associated RGB color space. The Y' represents the *luma* value (brightness of an image). The C_B and C_R values are considered the *chroma components*, and represent the color information.

2.1.3.1 Luma

The *luma* value is represented in the $Y'C_BC_R$ model by the symbol Y' and represents the brightness of an image. Y itself is considered to be the *relative luminance*. Relative luminance is a metric of light intensity as it appears to the human eye. The prime symbol (Y') denotes that *gamma correction* has been utilized. Gamma correction is an operation related to nonlinearity of light perception - when twice the number of photons hit a camera sensor, twice the signal is received, denoting a linear relationship. However, the human eye does not perceive change of light in a linear way. Gamma correction thus aims to translate the human eye's light sensitivity and that of a camera. [33] As gamma correction is not an important topic for the rest of this thesis, it will not be explored further.

Luma is calculated as the weighted sum of gamma-compressed $R'G'B'$ components. The prime again represents gamma correction. Luma can be calculated in the following way, as described in [2]:

$$Y' = 0.2126R' + 0.7512G' + 0.0722B' \tag{2.1}$$

2.1.3.2 Chrominance

Chrominance is the signal conveying the color information of a picture, separately from the accompanying luma. the C_B and C_R values represent the blue-difference (and red-difference, respectively) when compared to the luma. Multiple ways of calculating C_B and C_R exist, such as the one for HDTVs in [2]. In digital images, other transformations exist, such as the one used in the JPEG image format:

$$\begin{aligned} C_B &= 128 - (0.168736R') - (0.331264G') + (0.5B') \\ C_R &= 128 + (0.5R') - (0.418688G') + (0.081312B') \end{aligned} \tag{2.2}$$

[14]

An image decomposed into the Y', C_B and C_R components can be seen in the figure 2.3. The way colors are represented in the $Y'C_BC_R$ color model can be seen in the figure 2.4, which shows the C_BC_R plane at constant luma ($Y' = 0.5$).

2.1.3.3 Use of $Y'C_BC_R$

Common use of $Y'C_BCR$ stems from the human eye being more sensitive to the differences in luma than color differences. Therefore, reduced bandwidth can be given for chrominance components, allowing compression artifacts (or

Figure 2.3: An image decomposed into the Y', C_B and C_R components, as used in the $Y'C_BC_R$ color space. The Y' component also represents the grayscale image.

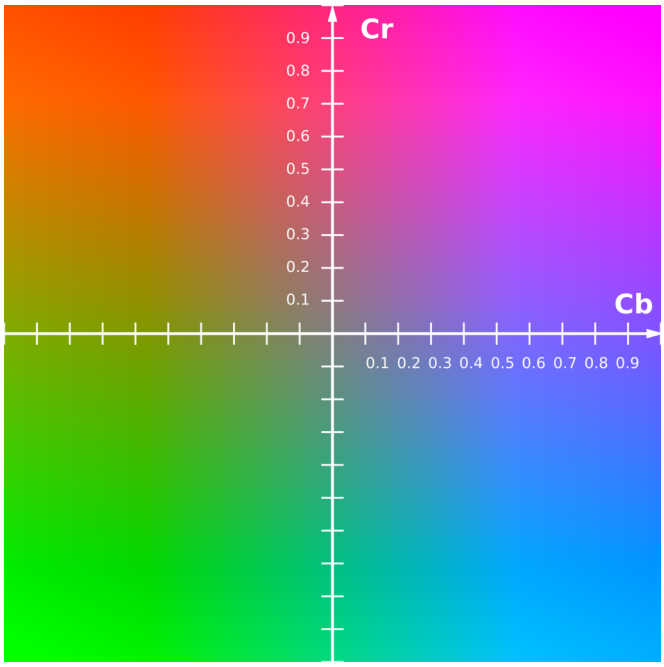


Figure 2.4: The C_BC_R plane at constant luma $Y' = 0.5$.

even transmission errors, i.e. when encoding image data for television signals) to be more efficiently masked, as the human eye is not as sensitive to these errors.

Another usage of $Y'C_BC_R$ is called *chroma subsampling*, Chroma subsampling refers to allocating less resolution for chroma information than the luma

information. This process is commonly used in video encoding schemes as well as in the JPEG image compression technique.[26]

2.1.4 Grayscale

Grayscale images are images composed exclusively of shades of gray. When using the grayscale model, each pixel therefore carries only the intensity information. Commonly, grayscale images are stored using 8 bits per pixel. The range of colors represented by these 8 bits goes from black (the value 0) through possible shades of gray to white (255 or another maximum possible value).

It is possible to transform colors from an *RGB* color space into the grayscale model by calculating the luma using the formula 2.1, as luma is a representation of an image's brightness.

Encoding an uncompressed grayscale image which uses the common 8 bit per pixel representation would therefore create an image of a size of $width \cdot height$, not counting the header and other data used by the specific file format.

Chapter 3

Image compression

As it has been shown in the introduction of this thesis, storing uncompressed images requires a lot of space. The art of reducing the amount of space required for storing data by encoding information with less bits than would otherwise be necessary is called data compression.

This chapter will explore the basics of data compression in order to build a framework for the rest of this thesis. Image compression basics with examples of current algorithms used for image compression will be described. The end of the chapter will explore compression metrics which can be used for determining quality of data compression (as well as image compression).

3.1 Data compression

When the terms *compression algorithm* or *compression technique* are used, they refer to two algorithms. One algorithm which takes an input X and generates a representation Y, which requires fewer bits to store. The second algorithm is the reconstruction algorithm, which operates on the representation Y and generates the reconstruction Z. Depending on whether the reconstruction Z is completely identical to the original data source X, the compression algorithms can be divided into two broad classes, being either *lossless compression algorithms* or *lossy compression algorithms*. Lossy compression algorithms generally provide much higher compression ratios than lossless compression algorithms do, but at the cost of losing information.[42]

3.1.1 Lossless compression algorithms

Lossless compression techniques involve no loss of information - by using a lossless compression algorithm, the original data can be recovered perfectly from the compressed data. The applications for lossless compression algorithms

commonly include text data, where small differences could easily result in wrong statements (errors in e.g. bank records are, for obvious reasons, very undesirable).

Many different lossless compression techniques - and different approaches altogether - exist and this thesis does not aim to describe them all. Therefore, the only techniques which are going to be described are the ones which are very commonly used - both as general data compression techniques as well as specifically used in image compression schemes.

3.1.1.1 Huffman coding

Huffman codes are particular types of optimal *prefix codes* (a type of code system where no whole code word in the system is a prefix of any other code word in the system) which are commonly used for lossless data compression. The technique was developed by David Huffman as part of a class assignment in the first ever taught class in the area of information theory.[42]

Huffman coding produces a table from data input, which encodes a source symbol using variable-length code. The algorithm used for deriving this table does so from estimated probabilites (or frequencies) of occurences for each possible value of source symbols. Symbols which occur more frequently are then represented using fewer bits than less common symbols.[16]

It is possible to prove that Huffman coding produces optimal codes, except for certain use-cases where Huffman coding is unable to do so - such as in the case where the probability mass function would be unknown, for example due to data changing over time or receiving new information during the coding process.

3.1.1.2 LZ77

Huffman coding is a technique which produces optimal codes, but also assumes a source which generates a sequence of independent symbols. However, very often, most sources are correlated. Thus it is common that the coding step is preceded by a decorrelation step. This step is often done using techniques which incoroporate the structure in the data to increase the amount of compression. These techniques are commonly called *dictionary techniques*, as they build a list of commonly occuring patterns and encode these patterns by transmitting their index in the list. These methods tend to be most useful with sources which generate a small number of patterns quite frequently - such as text sources or image sources.[42]

These dictionary methods are often distinguished by the way the dictio-

naries are built, commonly into two groups - *static* dictionaries and *adaptive* dictionaries. The static dictionary techniques are more appropriate when considerable prior knowledge about the source is available - such as when records are compressed and ahead of time, certain words are going to appear in almost all records (for example the word Name).

Adaptive techniques build dictionaries adaptively, during the course of compression. One popular technique, which has given rise to a number of variations, is the technique named *LZ77*. This approach creates a dictionary as a portion of the previously encoded sequence. The encoder examines the input sequence through a sliding window, which consists of two parts - a search buffer, which contains a portion of the recently encoded sequence, and a look-ahead buffer, wihch contains the next portion of the sequence to be encoded.

In order to encode a sequence in the look-ahead buffer, the encoder moves a search pointer through the search buffer until it encounters a match to the first symbol in the look-ahead buffer. The distance of the pointer from the look-ahead buffer is called the offset. The encoder then examines the symbols following the symbol at the pointer location to see if they match consecutive symbols in the look-ahead buffer. Once the longest match has been found, the encoder encodes it with a triple (o, l, c), where o is the offset, l is the length of the match and c is the codeword corresponding to the symbol in the look-ahead buffer that follows the match.[54]

3.1.1.3 Use in image compression

Both Huffman coding as well as LZ77 are both utilized in one of the most used lossless image compression formats available - PNG. The PNG (Portable Network Graphics) standard was developed due to patent issues, where companies would charge royalties to authors of software that included support for GIF. Due to these reasons, the internet community decided to implement a patent-free replacement for GIF, which would become known as PNG.[38]

The compression algorithm used in PNG is based on the *DEFLATE* implementation of LZ77, which combines both LZ77 and Huffman coding. While the encoder encodes the source using LZ77, it also represents the index values using a Huffman code. The algorithm producing DEFLATE files is widely considered to be implementable in a manner not covered by patents, which has led to a widespread use not only in image compression, but also in the ZIP file format, for which it was originally designed.[9]

3.1.2 Lossy compression algorithms

However, not all use cases require compression to be lossless. In these cases, the requirement of retrieving absolutely identical data can be relaxed. By relaxing this requirement, very often higher compression ratios (more on compression ratio in section 3.2.2) can be obtained, at the cost of having distortions in the reconstruction.

Very common scenarios where lossy compression algorithms become often more desirable than lossless ones are when storing audio content. Modern audio compression algorithms are almost all lossy - the often used *MP3* format used for storing audio stores audio using a lossy compression algorithm. As long as audio is stored without audible artifacts (distortions which are not present in the original data), the quality of sound does not have to be perfect - especially when compressing speech. When compressing other forms of audio, such as music, the compression needs to be more accurate, but still does not have to be absolutely perfect - as long as the listener does not notice the difference.

Other typical use case for a lossy compression algorithm is compressing images or video content. Small distortions are acceptable in images and video as long as they are not easily noticable by the human eye. However, this can also depend on the use case - for example photos commonly taken can be compressed in a lossy way, but medical images often have to be compressed in a lossless way, as artifacts can be very undesirable for medical usage.

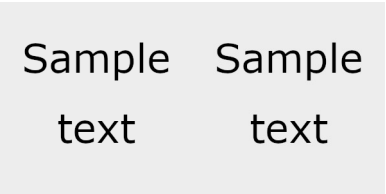


Figure 3.1: A sample image showing how accurate a lossy compression algorithm can be. The left image has been compressed using a lossless technique, the right image has been compressed using a lossy compression technique (JPEG). In this case, essentially no artifact can be seen unless zoomed in (as seen in the figure 3.2).

3.1.3 Modeling and coding

One of the important factors when designing a compression scheme which needs to be accounted for are characteristics of data which is to be compressed. An approach which works the best will depend to a large extent on

Figure 3.2: A zoomed in image used in the figure 3.1. While almost impossible to notice any artifact in the former example, the artifacts can be seen on the borders between the text and the gray background easily once zoomed in. Color levels were adjusted in order to emphasize the presence of artifacts.

the redundancies (more on redundancies in section 3.2.1) inherent to the data compressed.

In the phase of modeling, information about any redundancy is extracted in a form of a model, which can then be utilized in the compression algorithm. The second phase important in a compression scheme is *coding*. A description of model and how data differs from the model can be encoded (usually using a binary alphabet).[42]

A very simple example of modeling and coding and its importance can be seen on *delta encoding* (sometimes also called *delta compression*). Delta encoding encodes a sequence of messages not by their values but by calculating the difference between two elements.[45] Thus, a sequence such as

$$10000, 10002, 10001, 9999, 10000$$

can be modelled as a sequence where the numbers have very small differences. When shown as a sequence of differences, with the original value, the sequence then becomes:

$$10000, +2, -1, -2, +1$$

After a representation of this data sequence has been modeled, the coding phase creates a way how to encode this sequence using this model. One possible coding is shown in the table 3.1. Utilizing this coding, the differences require only 2 bits in order to be stored.

However, delta encoding would not be efficient in a scenario when the data has large differences - especially if the differences would be larger than storing the data itself. Understanding the type of data when creating a model is therefore highly important.

+1	00
+2	01
-1	11
-2	11

Table 3.1: Example coding for a sample data sequence using delta encoding

3.2 Image compression

Commonly, data compression algorithms are discussed as universal techniques which are able to compress essentially any data. While these algorithms might still have preferred applications, literature commonly separates techniques from the applications. However, image compression is a specific field where separating techniques from the application is essentially impossible as the techniques are meant specifically for compressing images.[42]

While compressing image data using standard compression techniques is possible, none of them are satisfactory for color or grayscale images (which were discussed in chapter 2). For example, statistical methods can be very good for compressing data where each value has different probability (such as in standard text, where certain letters appear far less than others). However, in images, certain colors or shades of gray often have the same probabilities.[40]

3.2.1 Redundancies in images

In order to understand how image compression techniques work, certain aspects of images need to be characterized, as compression techniques try to use these aspects for their advantage. A fundamental component of image compression is reducing the amount of redundancies present in the original image. The redundancies commonly present in images are the following:

- Coding redundancy
- Interpixel redundancy (also sometimes called spatial redundancy)
- Psychovisual redundancy

[47]

Coding redundancy occurs when length of the code words is larger than required. A simple example of coding redundancy can be shown on a grayscale image where only certain shades of gray are used. Instead of requiring 8 bits per shade of gray, such an image might require far less bits to encode, as can be seen in the figure 3.3.

Interpixel redundancy refers to the fact that usually, neighbouring pixels tend to have similar colors. Therefore, it can often be possible to predict the color of neighbouring pixels. As a simple example, black-and-white images might be encoded using *run-length encoding*, where instead of encoding the specific colors per pixel, information about how many pixels in a row have the same color is stored. This can be particularly efficient when considering bi-level (black-and-white) images, as can be seen in the figure 3.3.

The last kind of redundancy which is commonly utilized in image compression algorithms is called *psychovisual redundancy*. The human eye is not able to perceive certain visual information in an image - such information could be discarded without any noticeable artifacts present in the compressed image. A tool commonly used for reducing psychovisual redundancies in an image is *discrete cosine transform*, which is also used in the JPEG image format.

Figure 3.3: A sample image showing some of the redundancies which can be present in an image. As there are only 4 levels of gray color, storing them using 8 bits per possible gray color would result in high coding redundancy. Also, neighbouring pixels almost always have the same colors.

3.2.2 Compression metrics

When evaluating the quality of a compression technique, utilizing certain metrics is necessary - especially when attempting to compare compression algorithms. The following sections discuss various possible ways of evaluating compression techniques and consecutively compare their performance.

Both general compression metrics which can be utilized in any technique will be introduced, as well as techniques which are specifically related to evaluating the performance of image compression schemes.

3.2.2.1 Compression ratio

One of the most important metrics is the *compression ratio*. Compression ratio is a very simple metric which quantifies the reduction in data representation needed. It can be easily defined in the following way:

$$Compression\ Ratio = \frac{Uncompressed\ size}{Compressed\ size} \tag{3.1}$$

It is easy to see that compression ratio is an essential metric as it makes it very easy to compare compression algorithms or even quantify their compression power in general. Certain metrics extend compression ratio and use it for scoring purposes, such as the *Weissman score* which can be used for lossless compression algorithms.

However, while compression ratio is a very valuable metric and highly valuable for lossless compression algorithms, it potentially stops being the most important metric when evaluating performance of lossy compression algorithms. When evaluating lossy compressing algorithms, the important information is not only quantification of how well was the data compressed but also evaluating how much information was actually lost in the compression process. As an example, if a lossy compression algorithm has better compression ratio than another one, it might still not be the algorithm of choice, in the case where it would introduce too many artifacts into the data.

3.2.2.2 Mean squared error

Therefore, when analyzing the quality of lossy compression, it is necessary to introduce other compression metrics, which can be utilized for evaluating the performance of lossy compression algorithms, related to information loss - or in other words, the errors present in data after reconstruction. The two error metrics which are commonly used for measuring the performance of a lossy compression algorithm are the *mean squared error* (MSE) and *peak signal-to-noise ratio* (PSNR).[41]

Mean squared error is a statistical metric measuring the average of the squared of errors (the difference between original values and the values after decompressing data and obtaining the reconstruction). When used for image compression, the metric can be formalized in the following way:

$$MSE = \frac{1}{MN} \sum_{y=1}^{M} \sum_{x=1}^{N} (I(x,y) - I'(x,y))^2 \tag{3.2}$$

where

- M and N are the dimensions of an image
- $I(x, y)$ is the value of a pixel on coordinates x, y in the original image.
- $I'(x, y)$ is the value of a pixel on coordinates x, y in the reconstructed image.

A lower value of MSE is better, as it directly displays less errors present in the decompressed image.

3.2.2.3 Peak signal-to-noise ratio

Peak signal-to-noise ratio is a metric derived from MSE describing the ratio between the maximum possible power of a signal and the power of corrupting noise which affects the representation. Due to signals possibly having a very wide dynamic range, PSNR is usually expressed in the logarithmic scale.

With MSE defined as above, PSNR can be formalized as:

$$PSNR = 20 \cdot log10 \left(\frac{MAX_I^2}{MSE} \right) \tag{3.3}$$

where MAX_I is the maximum possible value of a pixel in an image. In the case of grayscale images stored with 8 bits per pixel, this value would be 255. Unlike MSE, higher PSNR values are better, as they are a sign of less errors in data - signal being the original colors and noise being errors.

When evaluating MSE (and consecutively PSNR) for color images, the calculation becomes slightly more difficult, as a pixel value of I[x, y] holds multiple values for multiple components, which are all perceived differently by the human eye. Some of possible approaches can therefore be the following:

- Evaluate MSE and PSNR for all the color components together using the RGB matrix.
- Due to brightness being the most important element for human eye, rather than color components, evaluate MSE and PSNR values only of luma (or simply the grayscale image). The definition of luma as a weighted sum has been explored in chapter 2.

[31][3]

However, it needs to be noted that PSNR is not a perfect metric in the sense that it would provide absolute definite conclusions. When utilizing PSNR, it is still important to also consider data subjectively, as it is only conclusively valid in scenarios where the compared results come from the same

codec (or codec type) and the same content. Not only that, but it is a metric which is not performing strongly as a quality metric when it comes to human perception of image data.[18] Still, it is a metric often utilized to evaluate lossy compression of images.

3.2.2.4 Structural similiarity index

Because of these reasons, another metric has recently been used for measuring the similarity between two images - the structural similarity index (abbreviated as *SSIM*). SSIM aims to improve on MSE and PSNR, which estimate *absolute errors* which occur during compression. SSIM on the other hand is a perception-based model, which considers image degradation as perceived change in structural information. Structural information refers to the concept that pixels have strong inter-dependencies when they are spatially close. These dependencies are important as they carry important information about the structure of objects in the visual scene.

The SSIM index is calculated on various windows (sub-samples), rather than on an entire image, like MSE is. SSIM between two windows x and y of common size $N \cdot N$ is calculated using the following formula:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.4}$$

where:

- μ_x is the average of x,
- μ_y is the average of y,
- σ_x^2 is the variance of x,
- σ_y^2 is the variance of y.
- σ_{xy} is the covariance of x and y,
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ are two variables to stabilize the division with weak denominator,
- L is the dynamic range of the pixel-values (in case of 8 bits per color the value is 255),
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

[50]

Usually, this formula is applied only on luma, as the most important aspect of the image in relation to eye perception. However, SSIM may also be applied

on color or chromatic values. The possible values of SSIM are in the range $(-1; 1]$, where SSIM of 1 can only be reached in the case of two images being identical.

The figure 3.4 shows the possible strength of SSIM when compared to MSE or PSNR. While the second image has a lot of noise in it, its MSE compared to the original image is actually *lower* than the MSE of the last image, even though the only difference when compared to the original is added contrast. By utilizing only MSE (or PSNR), it would appear the second image is more similar to the original, even though that is not the case for the human eye.

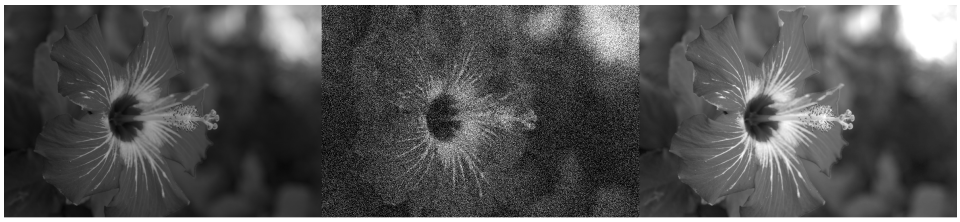


Figure 3.4: Figure visualizing the problems of MSE and PSNR when compared to SSIM as an image quality metric. Original photography source: [4]

3.2.2.5 Compression time

The last metric to note is not related to quality of the resulting data specifically, but is nonetheless very important - compression (or decompression) time. Techniques with lower compression time are generally more preferrable.

3.2.3 JPEG image compression

As noted in chapter 1, non-negative matrix factorization can be understood as a lossy compression algorithm - approximating the data in an original matrix by two smaller matrices which have less elements in total than the original matrix. As this process necessarily involves data loss (as non-negative matrix factorization solves the problem of approximating the original matrix, not finding an exact representation), a compression algorithm based on non-negative matrix factorization has to be a lossy one. Thus, the image compression algorithms which are going to be discussed in this thesis are lossy ones. Also, as this thesis is related to creating a proof of concept image compression scheme based on non-negative matrix factorization and not describing all the existing lossy image compression schemes, only the JPEG compression scheme will be explored, as the results of the *NMF* compression scheme will be compared to JPEG.

The JPEG compression method is commonly used for compressing digital photographies and makes it possible to adjust the degree of compression - making it possible to decide the tradeoff between image quality and compression. Commonly, JPEG is able to achieve 10:1 compression ratio without significant perceptible loss in image quality.[13]

The JPEG compression algorithm can use various modes of operation, but the most popular one works in the following way:

- An image is converted from an RGB color space to an $Y'C_BC_R$ color space.
- The resolution of chroma is reduced - this is done as the human eye is less sensitive to color details than brightness (chroma subsampling).
- The image is split into $8 \cdot 8$ pixel blocks. For each component of $Y'C_BC_R$ the 8x8 block is transformed using the discrete cosine transform. By doing so, the image data is represented in the frequency domain.
- The amplitudes of the frequencies are quantized, meaning that components with high frequencies are stored with less accuracy than the ones with lower frequencies.
- The resulting data of $8 \cdot 8$ blocks is further compressed using lossless encoding.

These steps are all able to reduce all three redundancies presented in the previous sections - psychovisual redundancies are removed by storing higher frequencies with lower accuracy and interpixel redundancy by working on $8 \cdot 8$ pixel blocks. Coding redundancy is reduced using a lossless compression algorithm at the end.

One of the possible parameters of JPEG compression is the quality setting. The quality setting can be chosen in the range [1; 100] and affects quantization, with the value 100 meaning no quantization. The effect which the quality setting can have on an image is shown in the figure 3.5, where the original image is compressed multiple times with different quality settings.

Figure 3.5: Effect of JPEG compression on an image with varying quality settings. First image is the original and the following images improve on the quality settings of a previous image. Original photography source: [4]

Chapter 4

NMF compression scheme

The previous three chapters all explored the essentials required to construct an image compression scheme based on non-negative matrix factorization. Utilizing these concepts, proof of concept compression schemes will be designed in this chapter. The implementation of these schemes is described in the following chapter 5.

4.1 Compression scheme design

As it has been shown in chapter 1, non-negative matrix factorization can be understood as a lossy compression scheme for matrices, as the original matrix can be approximated by factor matrices which possibly require less space to store. However, as there are multiple ways of representing and encoding an image with different tradeoffs, multiple compression schemes will be constructed and compared with each other - ones based on factorizing RGB images and a scheme based on compressing chroma values.

It should be emphasized that these compression schemes are proof of concept only and their main goal is to empirically evaluate the effect non-negative matrix factorization has on images. Therefore, storage size reduction is in this case only the secondary goal.

All the compression schemes compress data using non-negative matrix factorization. As it is necessary to reconstruct the matrices from one-dimensional data, a header containing the data related to the width and height of the original image and the NMF rank used is put at the start of the compressed data.

4.1.1 Naive compression scheme (RGB)

As an image with colors encoded using the RGB values is already a 2D matrix, a naive approach would be to simply use non-negative matrix factorization with the image as the input matrix. Afterwards, the two matrices will be compressed using a lossless data compression tool in order to save storage space.

As non-negative matrix factorization results in creating two factor matrices, these factor matrices are flattened and afterwards concatenated into a byte string, resulting in one long string of bytes which is afterwards compressed using a lossless compression algorithm of choice (more on the specifics of implementation in the following chapter).

This scheme is visualized in the figure 4.1.

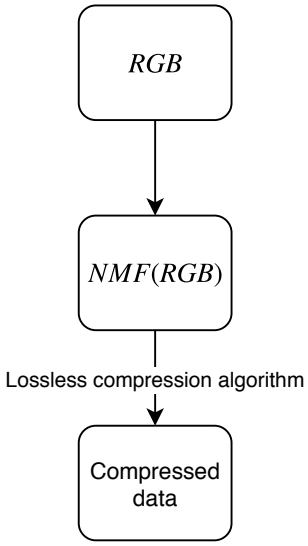


Figure 4.1: Visualization of the naive compression scheme. Note that NMF(RGB) actually represents two matrices which are stored next to each other in one data structure.

4.1.2 Separate RGB compression

Another scheme which will be explored will be one where the RGB components are extracted into their own matrices and non-negative matrix factorization is performed on each of these matrices individually. One aspect of this approach compared to the previous one is that the matrices which are going to be factorized are going to be smaller (contain less columns, as two color components will be missing in each matrix).

Like in the previous scheme, the factor matrices will be flattened and concatenated, followed by using a lossless compression technique to store the data.

This scheme is visualized in the figure 4.2.

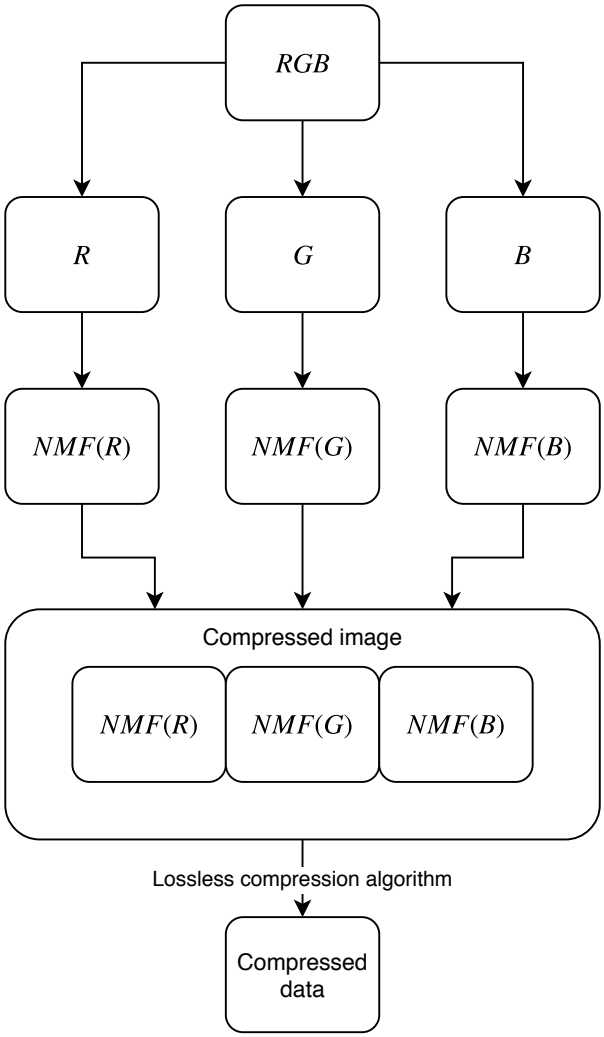


Figure 4.2: Visualization of the compression scheme which performs NMF on each of the RGB components.

4.1.3 $Y'C_BC_R$ compression scheme

The compression scheme based on $Y'C_BC_R$ aims to utilize the concept of human eye not noticing loss of information as strongly when it is related to colors, rather than brightness. This is achieved by working on each component separately:

- The luma (Y') component is not compressed by a lossy compression algorithm but a lossless one.
- The chroma components are separately compressed both using non-negative matrix factorization.

By not factorizing the luma component or using any lossy compression, the original luma information is not lost - only the chroma information. By doing so, less artifacts should be introduced into the images, however at the possible cost of worse compression ratios.

After factorizing the matrices containing the chroma components, all 5 matrices are flattened and concatenated into one long byte string, which is compressed using a lossless compression technique of choice.

This scheme is visualized in the figure 4.3.

4.2 Decompression

Decompressing images using these compression schemes is a reversed operation. As it is essentially the same for each scheme, only decompression of the $Y'C_BC_R$ scheme will be described.

The data compressed using a lossless data compression algorithm of choice is decompressed with the associated decompression algorithm. Afterwards, the header and the data related to the Y' component and the factor matrices are extracted from the compressed data. The header includes the width and height of the original matrices and the NMF rank used. By using this information, all the matrices are reconstructed from the flattened data. The factor matrices are multiplied together to reconstruct the C_B and C_R components. All three components are put together to construct an image in the $Y'C_BC_R$ color space, followed by converting the image into an RGB color space.

4.3 Non-deterministic properties of NMF

One of the properties which has been noted in chapter 1 was that the solution to non-negative matrix factorization is not unique. Not only that, but due to the problem being an optimization problem, there is no guarantee that various runs of any of these compression schemes will create the same images, unless the initial matrices have been initialized in the same way.

As a result, unless the initial matrices have been initialized in the same way, two aspects of these compression schemes should be noted:

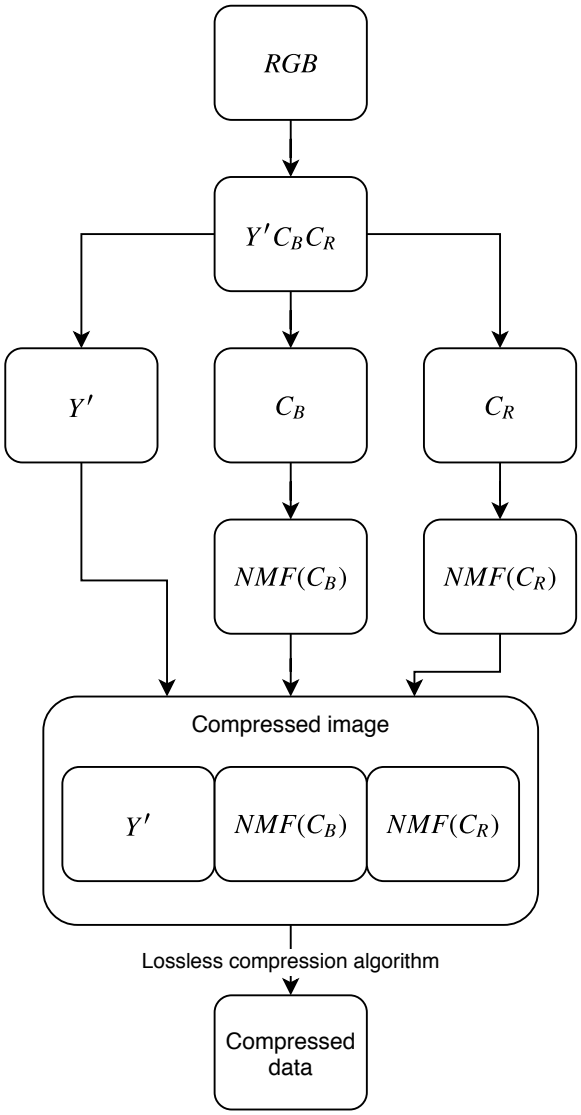


Figure 4.3: Visualization of the compression scheme based on compressing only the chroma values.

- As the approximation will be different for each run, the information loss will also be different, meaning that the visual quality of compression can differ - depending on the seeding technique used.
- When using a lossless compression algorithm to store the data, the storage size required to store the matrices will be different, as the source data compressed will also be different.

Chapter 5

Implementation

This chapter is related to the specific implementation of the schemes presented in the previous chapter. An outline of technologies used will be provided as well as the actual specifics of the compressed data.

5.1 Technologies used

The proof of concept compression schemes, as well as the analysis scripts, were implemented using the *Python programming language*, which has been chosen for its high readability as well as its good support for general working with data. The libraries which were used for implementing the compression schemes were the following:

- *NumPy* - a Python library useful for matrix calculations and other scientific work. NumPy also provides support for vectorizing array operations, thus often significantly speeding up matrix calculations. The library also works as an efficient container of generic data with specifiable datatypes. NumPy was used for the majority of matrix operations, together with flattening compressed matrices and reshaping them. This library was chosen for its high efficiency when it comes to working with matrices, ease of use and good readability.[35]
- *Nimfa* - a library implementing non-negative matrix factorization with implementations of multiple factorization methods, initialization approaches and other capabilities. This library uses the capabilities of NumPy for matrix calculations. It was chosen for its established use in academic environment and having rich capabilities, such as different seeding methods.[53]
- *Pillow* - a fork of the Python Imaging Library (PIL). Pillow implements many functionalities related to image processing. The functionalities

used are related to parsing existing image data and converting between different pixel representations (RGB, grayscale (luma) and $Y'C_BC_R$). Pillow was chosen as the library for working with images as it is currently considered the de facto standard Python library for these purposes and is internally used by many other libraries which work with images.[51]

- *zlib* - a compression library which is a higher-level abstraction for the DEFLATE lossless compression algorithm. Zlib was chosen as the lossless compression technique after performing non-negative matrix factorization as DEFLATE is a technique commonly used for compression - as well as image compression (as noted in chapter 3.1. The other reason for choosing zlib was also that it is a part of the Python standard library, meaning that using zlib requires little dependencies.
- *scikit-image* - an image processing library for Python. Used for calculating PSNR and SSIM.[48]

The source codes together with the experiments used can be found either in the enclosed CD or online in a GitHub repository using the url `https://github.com/prki/mastersthesis`.

5.2 Performing non-negative matrix factorization

Non-negative matrix factorization was implemented using the Nimfa library, which allows choosing various parameters, most notably the rank, number of iterations and the seeding technique for initializing factor matrices.

The method chosen for solving NMF was the standard NMF approach of multiplicative updates with the Euclidean distance update equations. The method for seeding the factor matrices which was chosen was the non-negative double singular value decomposition (NNDSVD)[8], which is a method containing no randomization and is based on SVD processes. This method has been chosen for two reasons:

- It is a method which does not use randomization. Due to that, the compression schemes always create the same factor matrices, meaning that the results can be repeated.
- As the SVD processes initialize the factor matrices based on the input data, the algorithm often performs better than random initialization does and rapidly reduces the approximation error.[8]

The rank and number of iterations of the NMF algorithm are parameters of the compression scheme and their effect on quality of image compression is explored in the following chapter.

5.2.1 Data types and NMF compression

Compression using NMF has been briefly discussed in chapter 1. One of the aspects noted was the possible upper bound of rank to attain a compression ratio greater than 1. However, it has also been noted that it is necessary for the elements in factor matrices to require the same amount of storage that the elements of the original matrix do.

However, that is not the case here, as the original data consists of 8 bit values, therefore natural numbers only. Factor matrices however consist of real number values. Thus, the data in factor matrices is represented using floating point numbers. In order to save space, the single-precision floating-point format was chosen - doing so however potentially results in more information loss than using double-precision would.

5.3 Compressed image data structure

Each of the compression schemes utilizes non-negative matrix factorization to compress color data - be it either factorizing RGB matrix, matrices containing each RGB component or factorizing the chroma components of $Y'C_BC_R$ values. All of the compression schemes compress this data using a lossless compression algorithm in order to save space.

This data structure which contains the data to be compressed is the same for all of the compression schemes - with the only difference being how much data does it contain, as the separate RGB compression scheme creates 6 factor matrices which need to be stored, whereas the naive RGB compression scheme creates only 2 factor matrices. Since the data structure differs only in this aspect, the general explanation of the structure will be provided.

Data area	Data content	Data size	Data type
Header	height	4B	uint32
	width	4B	uint32
	rank	4B	uint32
Compressed image data	flat matrix	matrix size · 4B	float32
	flat matrix	matrix size · 4B	float32
	other flat matrices	matrix size · 4B	float32

Table 5.1: Table showing how the compressed image data is stored. Flat matrices represent the matrices after they have been flattened into a 1-D array. Matrix size depends on what kind of matrix is stored, as explained in the section 5.3

A visualization of the data structure which is compressed can be seen in the

table 5.1. The first 12 bytes of this data structure are used for the header. This header stores height, width and the rank parameter of non-negative matrix factorization - this information is required for restoring the image data, as the dimensions of matrices which were stored were either $height \cdot width$ (luma matrix in the $Y'C_BC_R$ scheme), $width \cdot rank$ or $rank \cdot height$ (in the case of factor matrices). As the data size thus depends on the specific matrix stored, the table only describes the data size as matrix size · 4B.

This data structure is compressed using the zlib module with the highest possible compression setting.

5.4 Decompressing data

An important note about decompressing the image data is related to matrix multiplication. The factor matrices consist of real numbers, but the image data consists of natural numbers. Due to that, the elements of the multiplied matrix have to be rounded to the nearest number. This results in further possible information loss.

5.5 Possible improvements

Certain aspects of image compression using non-negative matrix factorization were noticed during the time this thesis was being written, however, as they were out of scope, they were not implemented or analyzed. These aspects which were noted would allow for a more efficient implementation or would possibly improve the image quality. Some of the aspects noted are the following:

- Performing independent non-negative matrix factorization in parallel (such as in the case of the $Y'C_BC_R$ scheme, where factorizing the C_B and the C_R matrices could run in parallel as these two operations are completely independent). This would result in faster compression times.
- Storing factor matrices as sparse matrices - as non-negative matrix factorization creates sparse factor matrices (and the seeding technique which was used should also create sparse factor matrices), representing and storing the factor matrices utilizing a technique for storing sparse matrices could save more space.
- Using a different lossless compression technique - while DEFLATE was chosen for its proven efficiency, it was not chosen for any specific property of the data - another lossless compression technique could prove to save more space.

- Utilizing a different method for solving the non-negative matrix factorization problem or changing the variation of the non-negative matrix factorization problem. For example, the research paper [20] used constrained non-negative matrix factorization for compressing grayscale images and reports significantly faster compression time together with slightly better compression ratio with no significant image quality differences - however, many specifics were not presented in the experiments and results (such as image dimensions, number of iterations or rank). The standard non-negative matrix factorization method was chosen for its general purpose approach.

- Representing the factor matrices using a different datatype, possibly using fixed point representation. It is unclear whether this would be an improvement or if doing so would have a significant impact on the image quality, as deciding that requires further analysis.

- Splitting up the image into multiple blocks and performing the compression scheme on these blocks. This approach was not performed because early experiments showed that doing so would be very time consuming - however, the results might also significantly improve the compression quality.

Chapter 6

Experiments and results

This chapter describes experiments performed using the compression schemes presented in the previous two chapters and analyzes their performance. Performance of compression schemes will be measured using the following metrics:

- Peak signal-to-noise ratio, which will be calculated on the luma component (on the grayscale image).
- Structural similarity index
- Compression ratio.
- Compression time.

The effect of compression on image quality will also be analyzed subjectively. The compression scheme performing the best will also be compared with JPEG image compression.

The results can also be found and used for further analysis in a comma-separated value format online in a GitHub repository using the url `https://github.com/prki/mastersthesis` or in the enclosed CD.

6.1 Compression scheme variables

As noted in the previous chapters, non-negative matrix factorization can perform very differently depending on the parameters of NMF, such as the matrix rank, number of iterations of NMF, the chosen algorithm used for solving NMF as well as matrix initialization.

In order for the experiments to be replicable, the algorithm used and the method used for matrix initialization is the same for each run of each com-

pression scheme (NNDSVD, as noted in the previous chapter). On the other hand, the effect of rank on the quality of compression will be explored - thus, rank will not be constant. In the case of the compression scheme performing the best, the effect of maximum number of iterations on PSNR and SSIM will also be explored. The best performing compression scheme will also be compared with JPEG.

The quality setting of JPEG compression is set to be constant for all the runs, with the value of 75. In the runs where rank was the variable, maximum number of iterations was set to 300. In the runs where maximum number of iterations was the variable, the rank was set to 150.

6.2 Choice of images

A set of 6 images has been chosen from the Image Compression Benchmark [4]. The benchmark contains a set of multiple high-resolution high-precision images from which 6 pictures were chosen. The pictures chosen include a computer-generated image as well as photographies in order to experiment using images from various sources. All the images were scaled down to lower resolution than the original ones (while retaining the original aspect ratio) due to the high time complexity of solving non-negative matrix factorization. While these images can be obtained in higher precision than 8 bits per component, the experiments were performed using the 8 bit precision, as it is currently still very commonly used.

The images chosen from the set are shown, together with their names, on the figure 6.1. The information about their resolution changes can be found in the table 6.1.

Image name	**Original resolution**	**New resolution**
artificial	3072x2048	950x633
bridge	2749x4049	679x1000
deer	4043x2641	980x640
hdr	3072x2048	1000x667
nightshot_iso_1600	3136x2352	960x720
spider_web	4256x2848	960x642

Table 6.1: Resolutions of benchmark images after scaling them down. Scaling was done due to high time requirements of non-negative matrix factorization.

Figure 6.1: Images used in the experiments together with their names as found in the image benchmark dataset.

6.3 Results using the naive RGB scheme

The naive RGB scheme, which performs non-negative matrix factorization on the entire RGB matrix, has proven to be very lossy. Not only that, but as will be shown in the following subsections, the naive RGB compression scheme is able to introduce artifacts into images which make certain structures impossible to notice well.

6.3.1 Image quality

While the image compression can be very lossy, the structural similarity of images can still be fairly high - achieving values over 0.75 in majority of images already with rank 5. The choice of rank has strong effect on the image quality - in some test cases, having a noticable effect on the structural similarity, but more noticably, the effect of rank can be seen on PSNR, which measures entire error and where rank changes the PSNR value significantly (i.e. from PSNR of ca. 22dB to over 27.5dB). The measured SSIM and PSNR values can be seen in the figure 6.2.

However, the peak signal-to-noise ratio values are still quite low and are usually not considered to result in images which would have good quality.

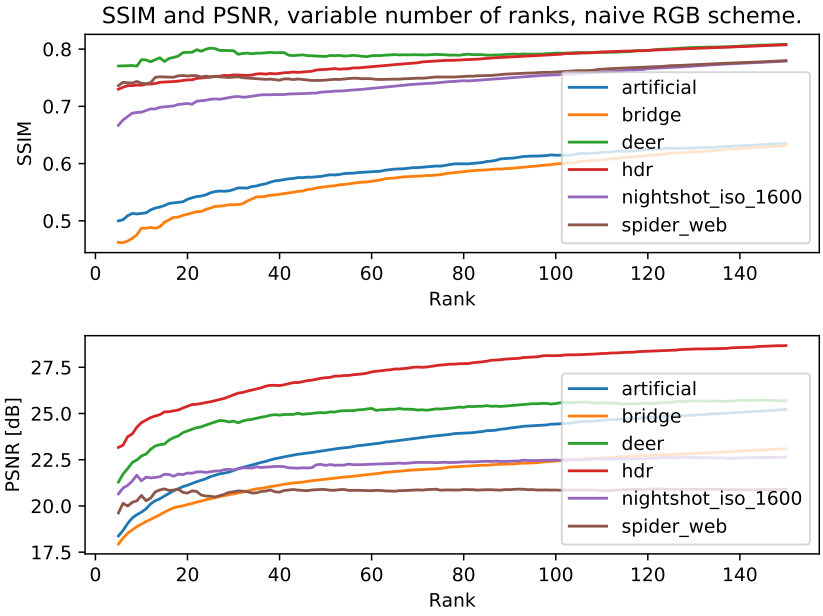


Figure 6.2: Results of measuring SSIM and PSNR on benchmark images using a variable rank and the naive RGB compression scheme. In all cases, higher rank results in better image quality - with noticable improvements of both metrics.

However, since structure of images can be kept relatively well, the images could still be used for computer image processing, rather than for the human eye. The effect of image compression using the naive RGB scheme and the artifacts which the human eyes see are explored in the subsection 6.3.3.

6.3.2 Compression time/ratio

The naive RGB compression scheme achieved the best compression ratios out of all three schemes. However, time-wise, it falls short when compared to the $Y'C_BC_R$ scheme - even though the $Y'C_BC_R$ compression scheme performs NMF on two different matrices (although containing slightly less data). While the compression ratio can be fairly high at lower ranks, unfortunately the image information loss is very easy to notice at low ranks. A graph showing the results can be seen in the figure 6.3.

6.3.3 Subjective analysis

The effect of rank is very easy to see when compressing images using the naive RGB scheme. At lower ranks, essentially only the general structure of an image can be seen. At higher ranks, the image becomes clearer and the

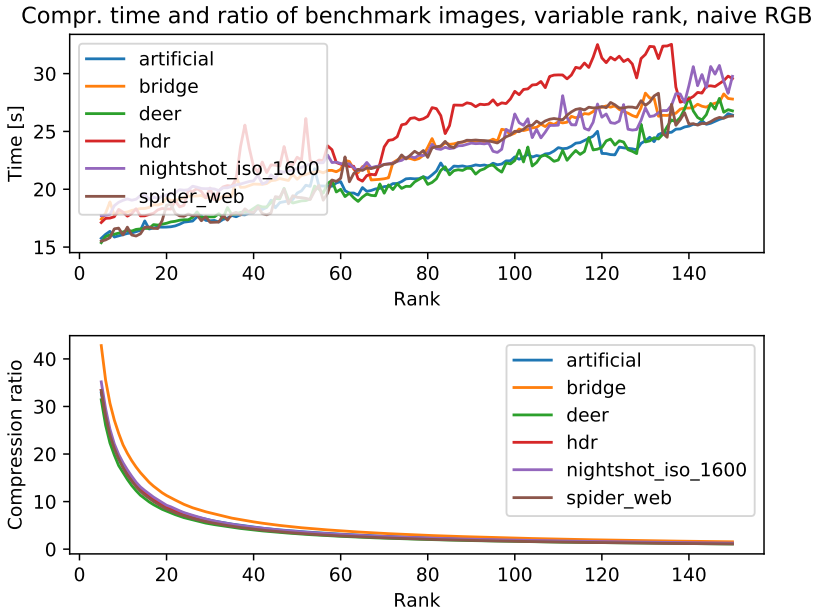


Figure 6.3: Figure showing the effect of rank on compression time and compression ratio when using the naive RGB scheme. Compression ratio can be very good for lower ranks, but as shown in the experiments, the image quality is very low when using these ranks.

details of an image become easy to see. However, the images still contain a lot of noise in colors - regions with the same color seem to contain a lot of grain. This can be seen in the figure 6.4, which shows the same image compressed using different rank side-by-side.

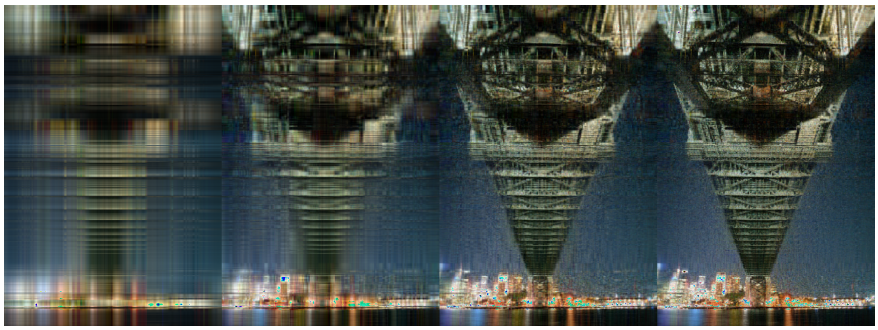


Figure 6.4: Compressing the *bridge* image using the naive RGB scheme. Ranks used (from left to right): 5, 20, 75, 150. Even though higher ranks result in significant improvement of image quality, the images are very lossy and contain a lot of noise.

One of the problems of the naive RGB scheme which however make it difficult to use is that the scheme introduces artifacts which are very easy to notice and which corrupt the original image severely. Likely, this is not only a problem of the naive RGB scheme but possibly a problem of the multiplicative updates method or of the NNDSVD seeding method, as it seems these artifacts are a result of stopping in a local optimum. These artifacts do not disappear with higher ranks, but only change their shape very slightly. These artifacts can be seen in the image 6.5.

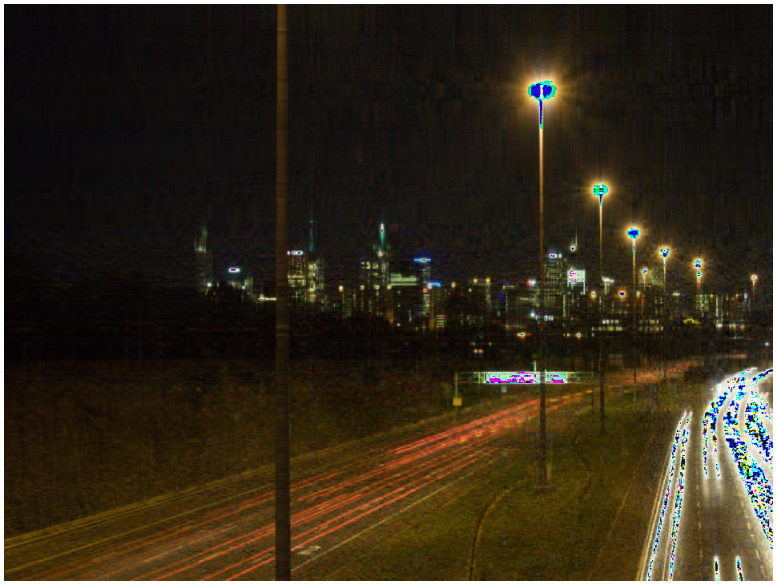


Figure 6.5: Image compressed using the naive RGB compression scheme where some of the artifacts are very easy to notice - especially on the right side, where almost all original data was lost. These artifacts hardly disappear with higher rank.

6.4 Results using the separate RGB scheme

The separate RGB scheme is an improvement over the naive RGB scheme. By performing non-negative matrix factorization on separate RGB channels (and thus factorizing smaller matrices), the approximations are slightly more accurate. However, even though this is the case, the improvements are very small, as seen in the following subsections. Not only that but the exact same artifacts which occur in naive RGB compression scheme occur when the separate RGB scheme is used..

6.4.1 Image quality

The image quality of images compressed using the separate RGB scheme is very similar to the one of images compressed using the naive RGB scheme - and changing rank has very similar effect as well. Even then, the metrics still score this compression scheme noticably better, with better SSIM/PSNR gain with higher ranks and higher achieved SSIM/PSNR values altogether. The measured results can be seen visualized in the figure 6.6.

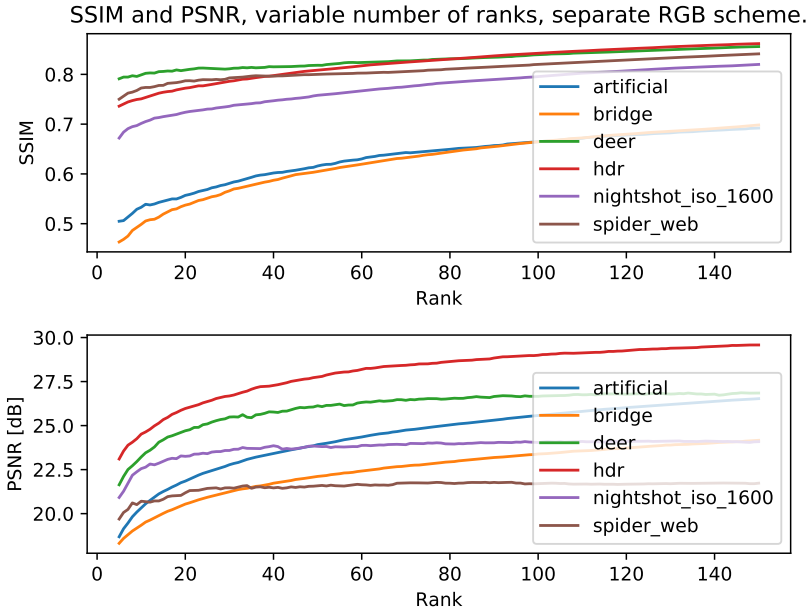


Figure 6.6: Results of measuring SSIM and PSNR on benchmark images using variable rank and the separate RGB compression scheme. In all cases, higher rank, just like in the naive RGB scheme, results in better image quality. Compared to the naive RGB scheme, the graphs have very similar shapes, but score slightly better.

Even though the SSIM and PSNR values are an improvement over the naive RGB compression scheme, the improvement is still very low - while the image structure is generally kept, many errors are introduced into the images, resulting in fairly low PSNR values.

6.4.2 Compression time/ratio

The separate RGB compression scheme achieves essentially the same compression times as its naive counterpart.(about 15 seconds for lower ranks and 30 seconds when used with higher ranks). The main difference between separate and naive RGB schemes in this case can be seen in the compression ratios.

While the relationship between rank and compression ratio is essentially the same, the naive RGB scheme outperforms the separate RGB scheme slightly, achieving higher compression ratios.

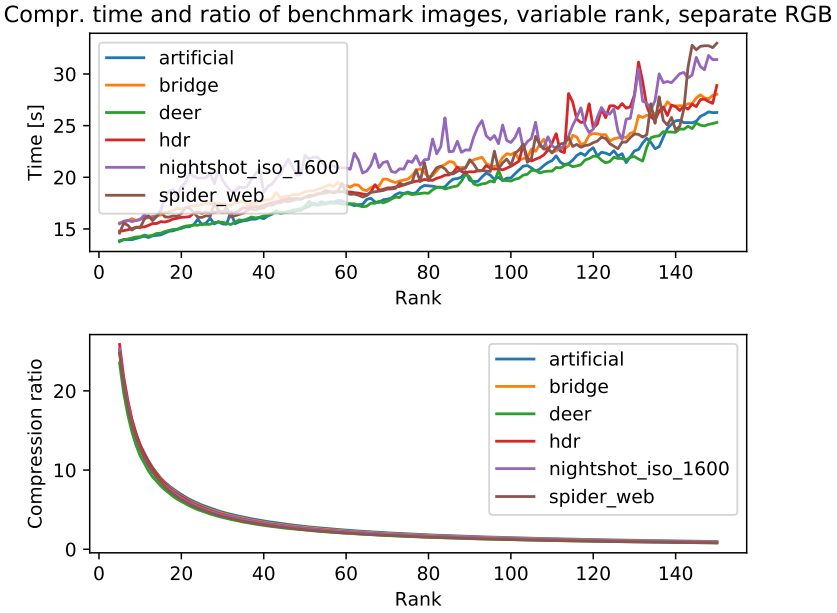


Figure 6.7: Figure showing the effect of rank on compression time and compression ratio when using the separate RGB scheme. As in the previous case with image quality, the graphs look very similar to the ones of the naive RGB scheme. While compression time is essentially the same as in the case of the naive RGB scheme, compression ratios do not achieve values as high as in the previous case for lower ranks.

However, even though this is the case, the lower ranks still achieve very low PSNR values in both cases, meaning the better compression ratio for the naive RGB scheme when lower ranks are used is not as much of an advantage, as such high information loss is not desirable.

6.4.3 Subjective analysis

The effect on rank and the image compression quality looks very similar as in the case of the naive RGB scheme. The effect can be seen in the figure 6.8. Therefore, even though the loss of information is slightly lower than in the previous case, the compression scheme still has many problems which make it hard for use. As noted before, it is not clear whether these artifacts occur specifically because of non-negative matrix factorization or if a different seeding method would possibly lead to better results.

Figure 6.8: Compressing the *hdr* image using the separate RGB scheme. Ranks used (from left to right): 5, 25 and 75.. Even though higher ranks result in significant improvement of image quality, the images are very lossy and contain a lot of noise, making it almost impossible to read certain texts in the images. Also, some artifacts can be noticed, which do not disappear at higher ranks, such as the green artifacts on the left side.

6.5 Results using the $Y'C_BC_R$ scheme

Altogether, the performance of the scheme based on compressing only the chroma information using non-negative matrix factorization showed itself to be much better than in the previous two cases, mostly achieving very high SSIM values and high PSNR values. At the same time, thanks to converging slightly faster, the compression times were also not as high.

6.5.1 Image quality

The effect of rank on the compression quality seems to be tied together closely - higher the rank, less information would be lost, generally. Only one image showed itself to be problematic and required higher rank to achieve good quality (the image called *artificial*). In all the other cases, it would have been possible to set the rank to a very low value and still achieve very high compression quality. The results have shown themselves to be very similar both when using PSNR and SSIM as the metrics. The visualized results can be seen in the figure 6.9.

The maximum obtained SSIM/PSNR values were essentially the same for all images, reaching SSIM over 0.99 in most cases and reaching PSNR values of ca. 48 dB. In most experiments with other rank settings, these values were essentially not improved on using this configuration.

6.5.2 Compression time/ratio

As in all the cases before, rank affects both the compression time as well as the compression ratio. Compared to both previous schemes, the compression times improve significantly - starting at 10 seconds for lower ranks and taking

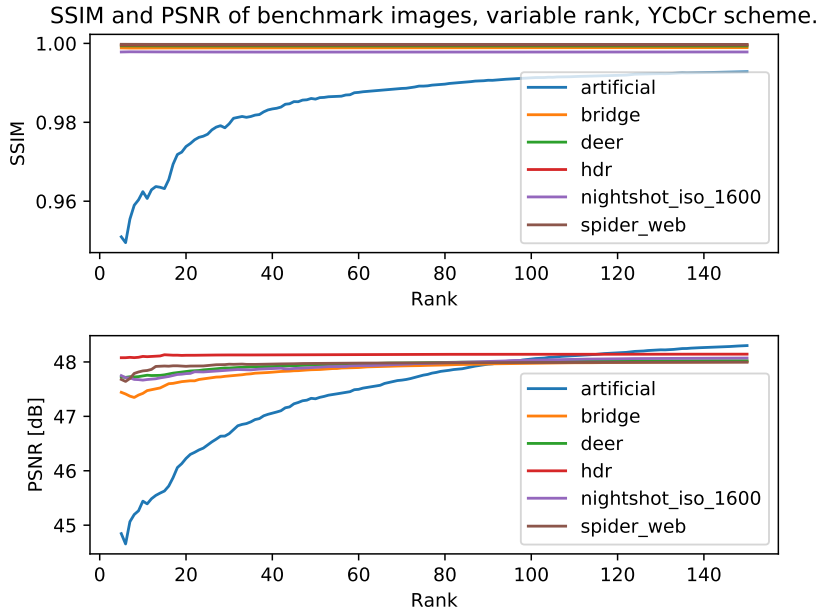


Figure 6.9: Results of measuring SSIM and PSNR on benchmark images using a variable rank and the $Y'C_BC_R$ compression scheme. Note that except for one image, the SSIM changes only extremely slightly with higher rank, thus making it possible to use very low rank and still achieve good compression quality.

about 20 seconds in most cases at rank 150 (unlike previous compression schemes, which took 15 to 30 seconds).

While the compression time is a significant improvement in this case, the compression ratio is not - mostly due to the fact that the luma values are not compressed using a lossy compression technique at all, thus requiring more space. However, unlike the previous cases, lower ranks can be used while still achieving very good image compression quality, as only one image has proven to be problematic (and the only one, which wasn't a photography) for image compression - in the rest of the cases, rank did not affect SSIM/PSNR significantly. As a result, for practical use, the $Y'C_BC_R$ compression scheme could achieve better compression ratios as lower ranks would still lead to desirable image qualities.

6.5.3 Comparison with JPEG

The scheme based on compressing chroma information has shown itself to possibly perform even better than JPEG with optimal rank choice for each

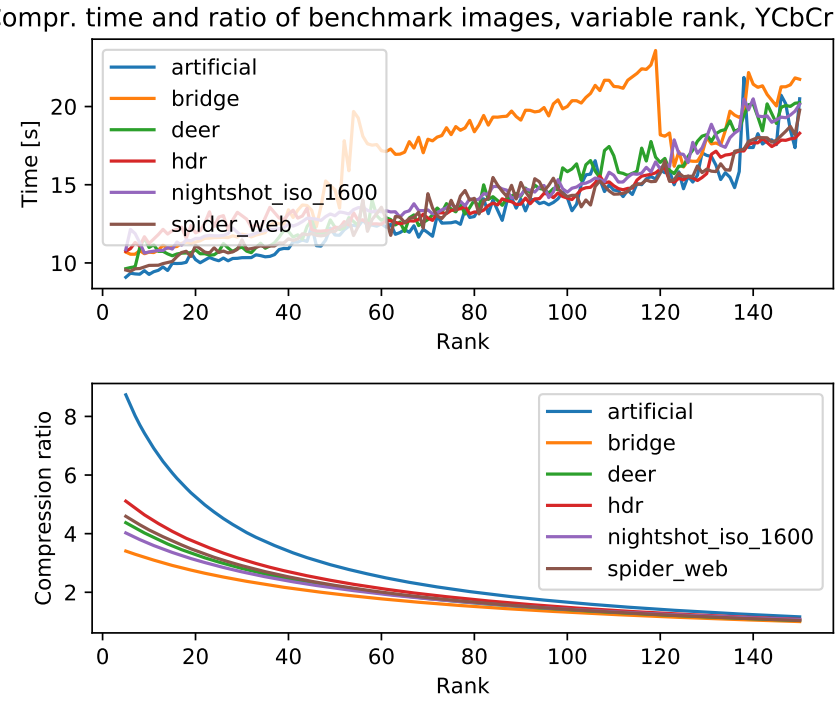


Figure 6.10: Visualization of the effect of rank on compression time and ratio on the $Y'C_BC_R$ compression scheme.

image. In the case of all six images, the values of SSIM and PSNR achieved higher values in their maximum than the SSIM and PSNR values of the same images compressed using JPEG. The comparison can be seen in the table 6.2. Visually, image compression using NMF can produce better images and certain artifacts which are common in JPEG images are not present in images compressed in this way (as shown in the next section).

However, even though the compression scheme is able to lose less information content, it is significantly beaten by JPEG when it comes to size of the images. This can be seen visually in the graph 6.11, which shows images compressed using rank 20 and their respective filesizes, compared to the filesizes of images compressed with JPEG.

6.5.4 Image quality improvement with more NMF iterations

As the $Y'C_BC_R$ compression scheme has proven to be the one performing the best out of the three, additional experiments were performed related to the effect of number of maximum possible iterations on the image quality. In

Image name	SSIM	PSNR	SSIM (JPEG)	PSNR (JPEG)
artificial	0.993	48.30	0.974	38.25
bridge	0.999	48.00	0.950	35.40
deer	0.999	48.02	0.957	40.29
hdr	0.999	48.14	0.975	41.88
nightshot_iso_1600	0.998	48.07	0.935	38.86
spider_web	0.999	47.99	0.980	42.61

Table 6.2: A table showing the maximum achieved values of chosen metrics when using the $Y'C_BC_R$ compression scheme and comparing them with the results using JPEG.

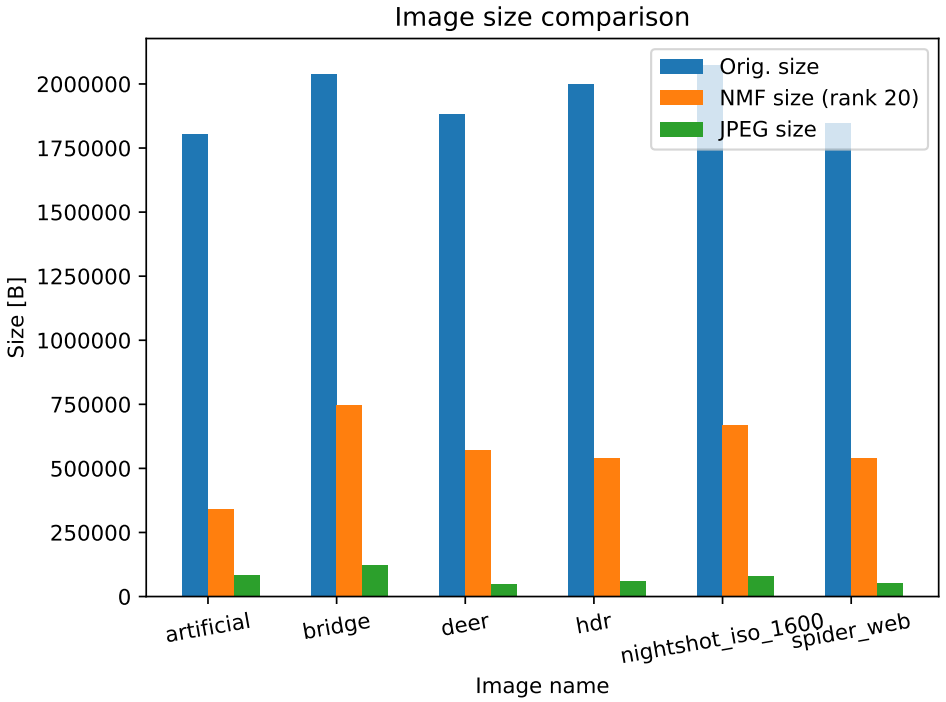


Figure 6.11: Comparison between original image sizes and image sizes of images compressed using NMF ($Y'C_BC_R$ scheme) at a specific rank and JPEG image sizes. Even though NMF image compression can produce high quality images with far lower sizes, JPEG still significantly outperforms the NMF compression scheme.

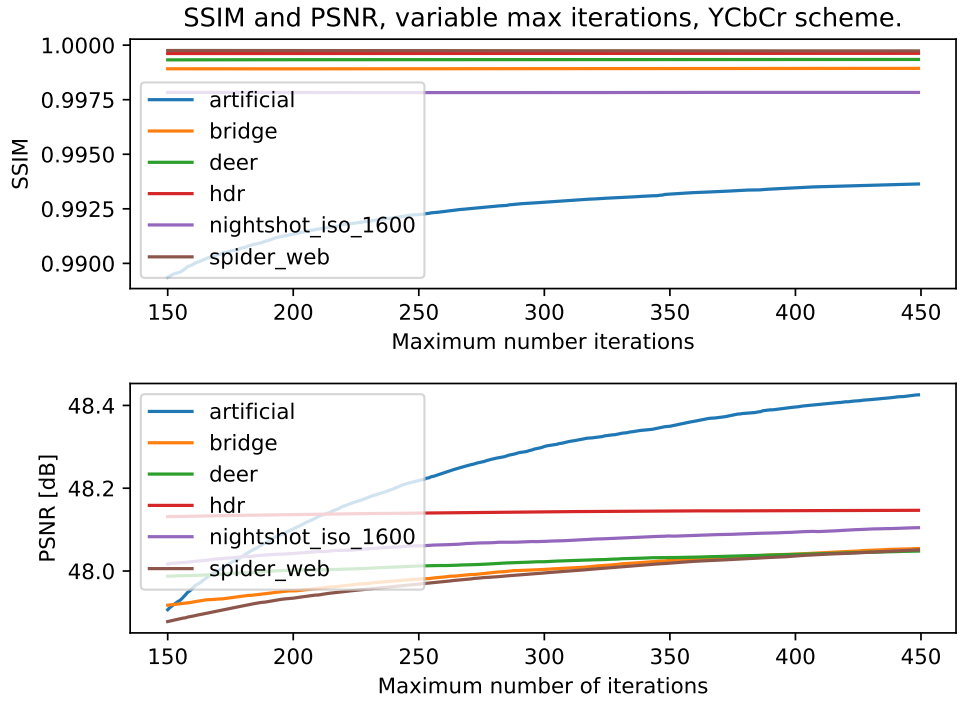


Figure 6.12: A figure displaying the effect of maximum number of multiplicative updates iterations on the image quality ($Y'C_BC_R$ scheme). While both SSIM as well as PSNR can be improved upon, the difference is very small, in most cases not resulting in very significant PSNR gain.

these experiments, the value of rank was set to 150 and the maximum number of iterations was in the range $[150; 450]$. Visualization of these experiments' results can be seen in the figure 6.12.

While maximum number of iterations results in images closer to the original (as the approximation improves), the improvement has been shown to be relatively small. The effect on SSIM is essentially unnoticable in all images but one, where the SSIM gain is still quite low. However, in the case of PSNR, the improvements are easier to see - however, these improvements are still very low.

By lowering the maximum number of iterations, the compression time can be significantly lowered as well, while not having a high impact on the compression quality, as shown in the figure 6.13. While the effect of number of iterations on image quality has not shown to be very significant, it does

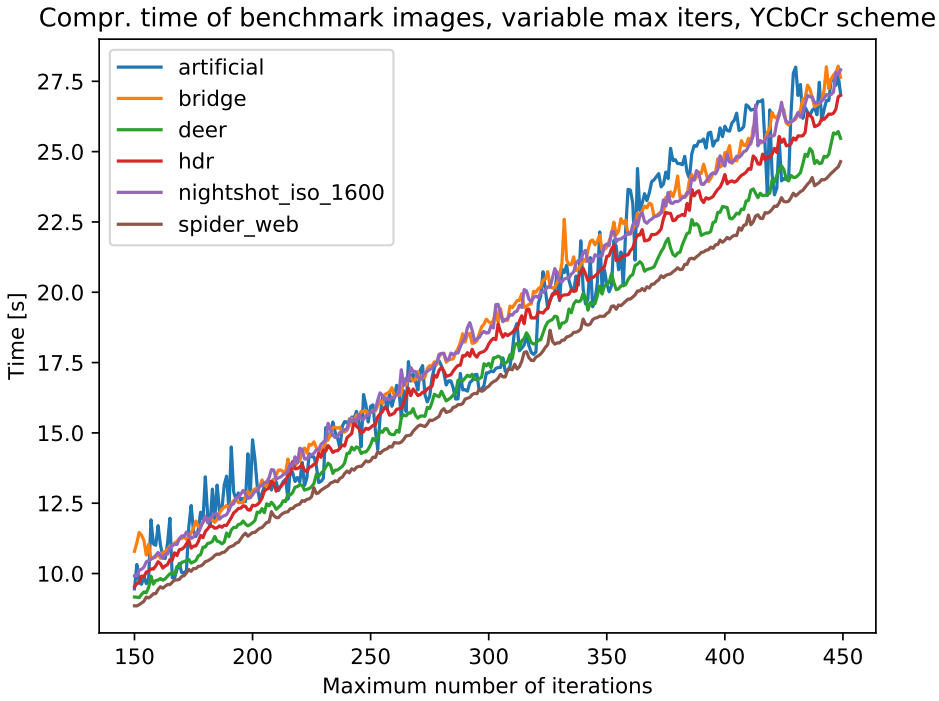


Figure 6.13: A figure displaying compression time depending on the maximum number of iterations ($Y'C_BC_R$ scheme).

have a linear effect on compression time requirements.

6.5.5 Subjective analysis of results

While the results show that images compressed using this method can be very accurate, there is still information lost in the process related to chroma values. This can be easily seen on the image named *artificial* where compressing certain regions of an image creates many artifacts, especially if these regions mainly consist of certain colors (red, purple and dark blue). It hasn't been found out if certain colors are more prone to these artifacts by default or if these artifacts are rather related to certain image properties. A comparison between a region of an image showing the difference between using JPEG and the $Y'C_BC_R$ compression scheme can be seen in the figures 6.14 and 6.15.

SImilar problems however occur when compressing the same image using JPEG. The exact same image region has problems with the same colors and also produces artifacts in the neighbouring areas to the cubes, as JPEG compresses small $8 \cdot 8$ pixel regions. This is one of the main advantages of the

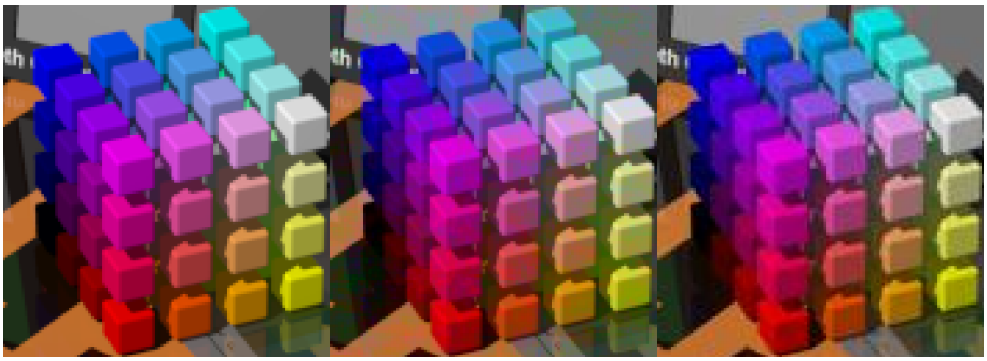


Figure 6.14: Region of the image named *artificial* showcasing the difference between our compression scheme using NMF and between JPEG. First image shows the uncompressed vesion, second image using the $Y'C_BC_R$ compression scheme and the last image is compressed using JPEG. While small differences are noticable, second image loses less information on certain colors (mostly on the right cubes).

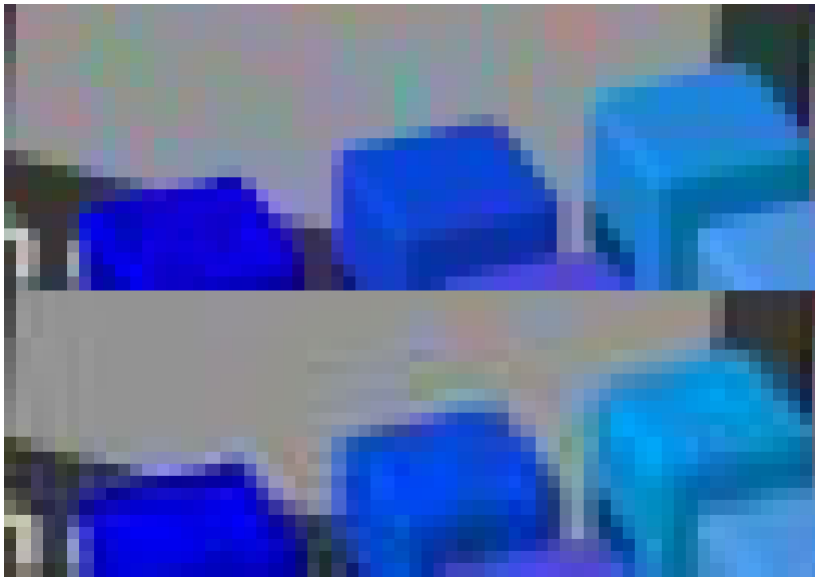


Figure 6.15: Zoomed in region from the figure 6.14. First image was compressed using the $Y'C_BC_R$ scheme, second one has been compressed using JPEG. The JPEG artifacts surrounding the cubes are easier to notice and the NMF compression scheme is able to keep the structure of an image better. However, the NMF compression scheme also produces artifacts in the area behind the cubes in the form of slightly blue tones in the background.

$Y'C_BC_R$ compression scheme when compared to JPEG - structure of an image is, for the most part, very unaffected, as only the color values change.

In most cases where the compression loses information, the human eye does not perceive the information loss as artifacts per se but rather simply loss of color information - resulting in colors being slightly inaccurate (less bright or a bit darker). This can be seen on the image named *hdr*. While the structure of an image is kept, some of the colors are slightly inaccurate - such as the small region of an image showed in the figure **??**, where small regions with slightly brighter colors can be seen. These changes are, however, difficult to notice without focusing on them.

6.6 Randomized seeding methods

In the course of implementing the compression schemes and their testing, certain interesting aspects of the compression schemes were found. During the course of development and testing, multiple seeding methods were tested - in the previous sections, the NNDSVD seeding method was used in all cases. Early on, randomized seeding methods were also used - until it was decided the NNDSVD method would be better suited, not only due to the fact that the results would be absolutely replicable, but also due to the fact that the randomized seeding methods did not perform well in certain cases. This section shows some of these aspects in order to show how much can different configuration change the results of image compression schemes based on non-negative matrix factorization.

6.6.1 Rank affecting image quality

The *nightshot_iso_1600* image from the benchmark set was compressed using randomized seeding methods using the naive RGB scheme. The results were essentially the same in case of both randomized seeding methods used (fully random and random_vcol). However, when working on a photography, the SSIM and PSNR values were clearly better when compared to the results with NNDSVD used. In the case of PSNR, the gain was not that high - but in SSIM the difference was as much as entire 0.1 points. This improvement is shown in the figure 6.16.

However, even though the randomized methods improve on the naive RGB scheme when used on this image, the artifacts still occur and do not disappear with higher ranks, which would make it seem that these artifacts occur not because of the seeding method but rather because of the way non-negative matrix factorization is solved itself.

These results would make it seem that randomized methods are an im-

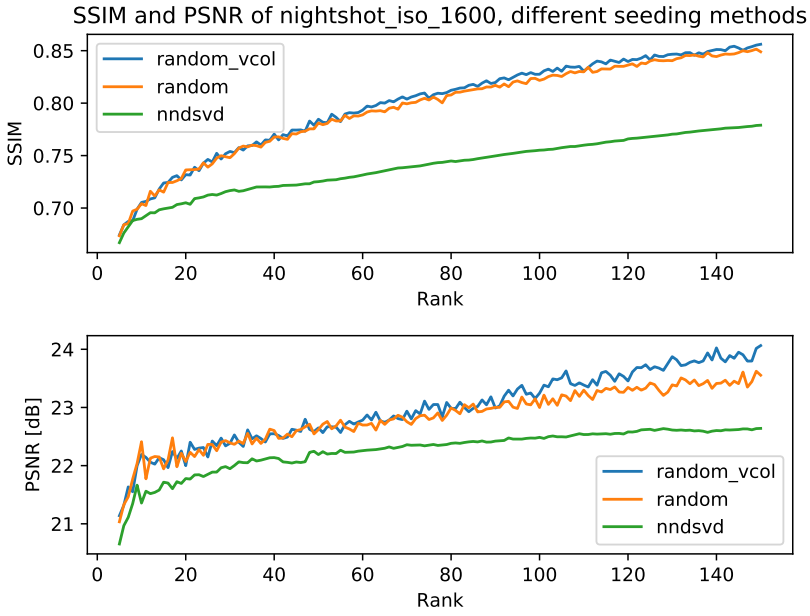


Figure 6.16: Randomized seeding methods improve the image quality of the naive RGB compression scheme on the *nightshot_iso_1600* image. While the PSNR gain is not that significant, the SSIM values are noticably higher.

provement over the NNDSVD method and perform far better. During the course of development of the compression schemes, other images were artificially created for testing purposes. Some of the results on these images were very surprising and would appear to have contradictory results compared to the rest of the findings - more specifically, using higher rank could significantly corrupt the image and lead to very noticable information loss.

This can be seen in particular in the figure 6.17, where an artifically created image was compressed using the RGB schemes. When random seeding was used, higher ranks corrupted the image extremely. In the case of NNDSVD, this was not the case and higher rank had absolutely no effect on the image quality. These errors also did not occur when the random_vcol seeding method was used.

As randomized seeding method could corrupt an image in such significant way, other seeding methods were preferred - in this case, NNDSVD, which did not have this problem in any of the test cases and would also lead to replicable test results due to lack of randomization.

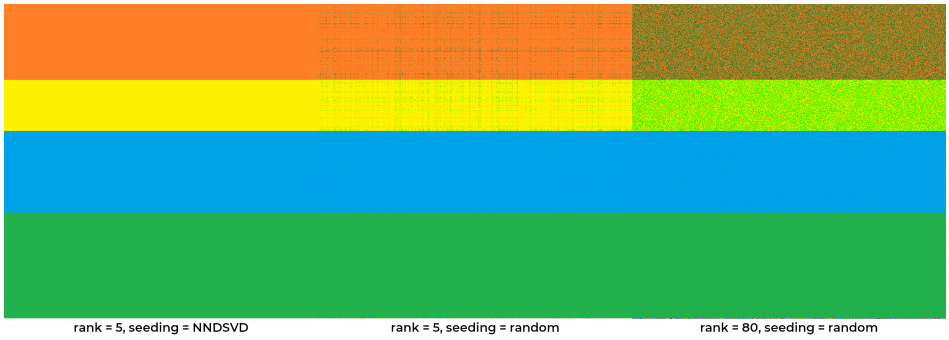


Figure 6.17: A sample image where random seeding method produces inconsistent results when increasing rank.

Conclusions and future work

In this thesis, non-negative matrix factorization was explored and its potential use for image compression was analyzed. Together with that, the essential information related to encoding digital images was also analyzed and explored, as well as image compression basics. By analyzing these topics, it was possible to create three compression schemes - a naive one factorizing an image as an entire RGB matrix, one which factorizes separate RGB channels and one which factorizes matrices containing the values related to colors by transforming the image into the $Y'C_BC_R$ color space. These compression schemes were implemented using the *Python* programming language and tested on various benchmark images coming from different sources (both artificially created images as well as digital photographies).

These compression schemes were evaluated using the peak signal-to-noise ratio and structural similarity index metrics and the effect of choosing different rank values on the image quality was explored. The other metrics which were explored were the compression times and compression ratios.

Out of these compression schemes, the ones based on compressing RGB color information proved to be quite inaccurate - even introducing very noticable artifacts. On the other hand, the compression scheme factorizing C_B and C_R matrices has proven to be very accurate - even more accurate than current state of the art lossy compression technique (JPEG), however at the cost of image size and compression times.

Future work

As noted in the first chapter, the area of image compression using non-negative matrix factorization is currently very underdeveloped. While this thesis offers framework for exploring the potential of using non-negative matrix factoriza-

tion for image compression, there are still many areas which can be explored.

The $Y'C_BC_R$ compression scheme has proven to be the fastest one (out of the explored compression schemes utilizing non-negative matrix factorization) by a large margin. The accuracy of the $Y'C_BC_R$ when it comes to retaining the image quality is even higher than in the case of JPEG, being one of the currently most used lossy compression techniques for image compression. However, the compression scheme still falls short of JPEG when it comes to compression time and compression ratio.

Potential improvements which could lead to better compression ratious have been proposed in chapter 5, such as representing the matrices in a way which would take less space (for example by using a sparse matrix representation, if possible), performing non-negative matrix factorization on small image regions, performing further quantization in order to compress a range of values into a single quantum value or utilizing chroma subsampling. Compression times could be significantly improved by evaluating non-negative matrix factorization of the C_B and C_R information in parallel.

Bibliography

[1] *Microsoft Windows Bitmap File Format Summary.*
URL http://www.fileformat.info/format/bmp/egff.htm

[2] Recommendation ITU-R BT.709-6, 06/2015 ed.

[3] Netpbm documentation, 2014.
URL http://netpbm.sourceforge.net/doc/

[4] *Image Compression Benchmark*, 2015.
URL http://imagecompression.info/

[5] *Desktop Screen Resolution Stats Worldwide.* http://gs.statcounter.com/screen-resolution-stats/desktop/worldwide, 2019. Accessed: 2019-02-20.

[6] Agarwal, M., Agrawal, H., Jain, N., and Kumar, M. *Face Recognition Using Principle Component Analysis, Eigenface and Neural Network.* In 2010 International Conference on Signal Acquisition and Processing, pp. 310–314. 2010. doi:10.1109/ICSAP.2010.51.

[7] Biederman, Irving. *Recognition-by-components: A theory of human image understanding.* Psychological Review, 94:115–147, 1987.

[8] Boutsidis, C. and Gallopoulos, E. *SVD Based Initialization: A Head Start for Nonnegative Matrix Factorization.* Pattern Recogn., 41(4):1350–1362, 2008. ISSN 0031-3203. doi:10.1016/j.patcog.2007.09.010.
URL http://dx.doi.org/10.1016/j.patcog.2007.09.010

[9] Deutsch, P. *DEFLATE Compressed Data Format Specification Version 1.3*, 1996.

[10] Févotte, Cédric, Bertin, Nancy, and Durrieu, Jean-Louis. *Nonnegative Matrix Factorization with the Itakura-Saito Divergence: With Application to Music Analysis.* Neural Computation, 21(3):793–830, 2009.
URL http://dblp.uni-trier.de/db/journals/neco/neco21.html#FevotteBD09

[11] Gaujoux, Renaud. *An introduction to NMF package.* https://cran.r-project.org/web/packages/NMF/vignettes/NMF-vignette.pdf, 2018. Accessed: 2019-02-21.

[12] Gillis, Nicolas. *The Why and How of Nonnegative Matrix Factorization.* arXiv e-prints, arXiv:1401.5226, 2014.

[13] Haines, Richard F. and Chuang, Sherry L. *The effects of video compression on acceptability of images for monitoring life sciences experiments.* Tech. rep., NASA, 1992.
URL https://ntrs.nasa.gov/search.jsp?R=19920024689

[14] Hamilton, Eric. *JPEG File Interchange Format.* Tech. rep., C-Cube Microsystems, Milpitas, CA, USA, 1992.
URL http://www.w3.org/Graphics/JPEG/jfif3.pdf

[15] Ho, Ngoc-Diep. Nonnegative matrix factorization algorithms and applications. Ph.D. thesis, 2008.

[16] Huffman, David A. *A Method for the Construction of Minimum-Redundancy Codes.* Proceedings of the Institute of Radio Engineers, 40(9):1098–1101, 1952.

[17] Hunt, R. W. G. The reproduction of colour. Hoboken, NJ: John Wiley, c2004, 6th ed ed. ISBN 04-700-2425-9.

[18] Huynh-Thu, Q. and Ghanbari, M. *Scope of validity of PSNR in image/video quality assessment.* Electronics letters, 44(13):800–801, 2008.

[19] Ibraheem, Noor, Hasan, Mokhtar, Khan, Rafiqul Zaman, and K Mishra, Pramod. *Understanding Color Models: A Review.* ARPN Journal of Science and Technology, 2, 2012.

[20] Inuganti, Srilakshmi and Gampala, Veerraju. *Image compression using Constrained Non-Negative Matrix Factorization.* In International Journal of Advanced Research in Computer Science and Software Engineering, pp. 498–503. 2013. ISSN 2277 128X.

[21] jen Lin, Chih. *Projected gradient methods for non-negative matrix factorization.* Tech. rep., Neural Computation, 2007.

[22] Kalofolias, V. and Gallopoulos, E. *Computing symmetric nonnegative rank factorizations.* Linear Algebra and its Applications, 436(2):421 – 435, 2012. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2011.03.016. Special Issue devoted to the Applied Linear Algebra Conference (Novi Sad 2010).
URL http://www.sciencedirect.com/science/article/pii/S0024379511002199

[23] Kitamura, D. and Ono, N. *Efficient initialization for nonnegative matrix factorization based on nonnegative independent component analysis.* In 2016 IEEE International Workshop on Acoustic Signal Enhancement (IWAENC), pp. 1–5. 2016. doi:10.1109/IWAENC.2016.7602947.

[24] Kong, Shikang, Sun, Lijuan, Han, Chong, and Guo, Jian. *An Image Compression Scheme in Wireless Multimedia Sensor Networks Based on NMF.* Information, 8:26, 2017. doi:10.3390/info8010026.

[25] Krausz, Barbara and Bauckhage, Christian. *Action Recognition in Videos Using Nonnegative Tensor Factorization.* pp. 1763–1766. 2010. doi:10.1109/ICPR.2010.435.

[26] Lambrecht, Christian J. Van den Branden. Vision Models and Applications to Image and Video Processing. Norwell, MA, USA: Kluwer Academic Publishers, 2001. ISBN 0792374223.

[27] Langville, Amy Nicole, Meyer, Carl Dean, Albright, Russell, Cox, James, and Duling, David. *Algorithms, Initializations, and Convergence for the Nonnegative Matrix Factorization.* CoRR, abs/1407.7299, 2014.
URL http://arxiv.org/abs/1407.7299

[28] Lee, Daniel D. and Seung, H. Sebastian. *Learning the parts of objects by nonnegative matrix factorization.* Nature, 401:788–791, 1999.

[29] Lee, Daniel D. and Seung, H. Sebastian. *Algorithms for Non-negative Matrix Factorization.* In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, Advances in Neural Information Processing Systems 13, pp. 556–562. MIT Press, 2001.
URL http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

[30] Mathews, Brady. *Image Compression using Singular Value Decomposition (SVD).* http://www.math.utah.edu/~goller/F15_M2270/BradyMathews_SVDImage.pdf, 2014. Accessed: 2019-02-21.

[31] MATLAB. version R2018b. Natick, Massachusetts: The MathWorks Inc., 2018.

[32] Mbearnstein37. *CIExy1931 AdobeRGB vs sRGB*, 2013. [Online; accessed May 1, 2019].
URL https://en.wikipedia.org/wiki/File:Atomic_force_microscope_block_diagram.svg

[33] McHugh, Sean. *Gamma correction.*
URL https://www.cambridgeincolour.com/tutorials/gamma-correction.htm

[34] Miettinen, Pauli. *Lecture notes in Data Mining and Matrices*, 2017.
URL https://www.mpi-inf.mpg.de/fileadmin/inf/d5/teaching/ss17_dmm/lectures/2017-05-29-intro-to-nmf.pdf

[35] Oliphant, Travis E. *A guide to NumPy.* USA: Trelgol Publishing, 2006– . [Online; accessed ¡today¿].
URL http://www.numpy.org/

[36] Paatero, Pentti and Tapper, Unto. *Positive Matrix Factorization: A Non-Negative Factor Model with Optimal Utilization of Error Estimates of Data Values.* Environmetrics, 5:111–126, 1994. doi:10.1002/env.3170050203.

[37] Rezghi, M and Yousefi, Masoud. *A projected alternating least square approach for computation of nonnegative matrix factorization.* Journal of Sciences, Islamic Republic of Iran, 26:273–279, 2015.

[38] Roelofs, Greg. *Portable Network Graphics.*
URL http://www.libpng.org/pub/png/

[39] Roweis, S. T. and Saul, L. K. *Nonlinear Dimensionality Reduction by Locally Linear Embedding.* In Science, vol. 290, pp. 2323–2326. ISSN 00368075. doi:10.1126/science.290.5500.2323.
URL http://www.sciencemag.org/cgi/doi/10.1126/science.290.5500.2323

[40] Salomon, David. Data Compression: The Complete Reference. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 1846286026.

[41] Sampath, Satish. *An Introduction to Image Compression.*
URL http://www.debugmode.com/imagecmp/index.htm

[42] Sayood, Khalid. Introduction to Data Compression, Third Edition (Morgan Kaufmann Series in Multimedia Information and Systems). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN 012620862X.

[43] Schultz, Jeff. *How Much Data is Created on the Internet Each Day?* https://blog.microfocus.com/how-much-data-is-created-on-the-internet-each-day/, 2017. Accessed: 2019-02-18.

[44] Shitov, Yaroslav. *A short proof that NMF is NP-hard*. 2016. URL https://www.researchgate.net/publication/303217568_A_short_proof_that_NMF_is_NP-hard

[45] Smith, Steven W. The Scientist and Engineer's Guide to Digital Signal Processing. San Diego, CA, USA: California Technical Publishing, 1997. ISBN 0-9660176-3-3.

[46] Sra, Suvrit and Dhillon, Inderjit S. *Generalized Nonnegative Matrix Approximations with Bregman Divergences*. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, Advances in Neural Information Processing Systems 18, pp. 283–290. MIT Press, 2006. URL http://papers.nips.cc/paper/2757-generalized-nonnegative-matrix-approximations-with-bregman-divergences.pdf

[47] T, Prabhakar, Naveen, Dr Jagan, Annam, Lakshmi Prasanthi, and Santhi, G.Vijaya. *Image Compression Using DCT and Wavelet Transformations*. International Journal of Signal Processing, Image Processing and Pattern Recognition (IJSIP) Korea, Vol.4:pages.61–74,, 2011.

[48] van der Walt, Stéfan, Schönberger, Johannes L., Nunez-Iglesias, Juan, Boulogne, François, Warner, Joshua D., Yager, Neil, Gouillart, Emmanuelle, Yu, Tony, and the scikit-image contributors. *scikit-image: image processing in Python*. PeerJ, 2:e453, 2014. ISSN 2167-8359. doi:10.7717/peerj.453. URL https://doi.org/10.7717/peerj.453

[49] Virtanen, T. *Monaural Sound Source Separation by Nonnegative Matrix Factorization With Temporal Continuity and Sparseness Criteria*. IEEE Transactions on Audio, Speech, and Language Processing, 15(3):1066–1074, 2007. ISSN 1558-7916. doi:10.1109/TASL.2006.885253.

[50] Wang, Zhou and Bovik, Alan. *Bovik, A.C.: Mean squared error: love it or leave it? - A new look at signal fidelity measures. IEEE Sig. Process. Mag. 26, 98-117*. Signal Processing Magazine, IEEE, 26:98 – 117, 2009. doi:10.1109/MSP.2008.930649.

[51] Wiredfool, Clark, Alex, , Hugo, Murray, Andrew, Karpinsky, Alexander, Gohlke, Christoph, Crowell, Brian, Schmidt, David, Houghton, Alastair, Johnson, Steve, Mani, Sandro, Ware, Josh, Caro, David, Kossouho, Steve, Brown, Eric W., Lee, Antony, Korobov, Mikhail, Michał Górny, Santana, Esteban Santana,

Pieuchot, Nicolas, Tonnhofer, Oliver, Brown, Michael, Benoit Pierre, Abela, Joaquín Cuenca, Solberg, Lars Jørgen, Reyes, Felipe, Buzanov, Alexey, Yifu Yu, Eliempje, and Tolf, Fredrik. *Pillow: 3.1.0*, 2016. doi:10.5281/zenodo.44297.
URL `https://zenodo.org/record/44297`

[52] Xu, Wei, Liu, Xin, and Gong, Yihong. *Document Clustering Based on Non-negative Matrix Factorization.* In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, SIGIR '03, pp. 267–273. New York, NY, USA: ACM, 2003. ISBN 1-58113-646-3. doi:10.1145/860435.860485.
URL `http://doi.acm.org/10.1145/860435.860485`

[53] Zitnik, Marinka and Zupan, Blaz. *Nimfa: A Python Library for Nonnegative Matrix Factorization.* Journal of Machine Learning Research, 13:849–853, 2012.

[54] Ziv, J. and Lempel, A. *A Universal Algorithm for Sequential Data Compression.* IEEE Trans. Inf. Theor., 23(3):337–343, 2006. ISSN 0018-9448. doi:10.1109/TIT.1977.1055714.
URL `http://dx.doi.org/10.1109/TIT.1977.1055714`

Appendix A

Acronyms

JPEG Joint Photographic Experts Group

MSE Mean squared error

NMF Non-negative matrix factorization

NNDSVD Nonnegative double singular value decomposition

PCA Principal component analysis

PNG Portable Network Graphics

PSNR Peak signal-to-noise ratio

SSIM Structural similarity index

SVD Singular value decomposition

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory with source codes
  - `nmf_compressor` ..... the directory containing Python source files
  - `thesis` ..... the directory containing LaTeX source files and images
- `text` ..... the directory containing the thesis in PDF file