Insert here your thesis' task.

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.

April 11, 2019

Acknowledgements

THANKS (remove entirely in case you do not with to thank anyone)

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 particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on April 11, 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

V několika větách shrňte obsah a přínos této práce v českém jazyce.

Klíčová slova Replace with comma-separated list of keywords in Czech.

Abstract

Summarize the contents and contribution of your work in a few sentences in English language.

Keywords Replace with comma-separated list of keywords in English.

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 kinds 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 very 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 resolution (1920x1080 pixels, the second most common resolution used on desktop devices [5]. When storing an image with these dimensions 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 platofrms on the internet are based on image sharing and image hosting. One modern social media platform centered around image sharing had over 67 million new posts each day [34]. Being able to obtain these images fast and in good quality is therefore highly important for both users, just as it is important for the owners of these services to be able to store this data.

In the recent years with growht related to machine learning and similar areas, such as artifical intelligence, new applications of mathematical concepts were 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 [30].

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 are essentially also 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 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 [23]. 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 the state of the art image compression algorithms and possible points related to further analysis or will be explored.

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.

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 [28], but the works which have given this problem far more popularity are the works of Lee and Seung [21], 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 $n \times p$ non-negative matrix, (i.e. with $x_{ij} \geq 0$, denoted $X \geq 0$), and $r > 0$ an integer. Non-negative matrix factorization consists in finding an approximation

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

where W, H are $n \times r$ and $r \times p$ 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 than 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 [9].

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 decomposition, whereas *NMF* is in reality an approximation. Thus, the problem is called *non-negative matrix approximation* in certain other works, such as [37].

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 [21], 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 [13]. 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) [28].

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 [17]. However, they will not be explored within this thesis.

1.2.1 Multiplicative updates

The algorithm described here is described more thoroughly in [22], 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 how 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 [22] 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 [22] 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 [22] for solving this problem. The multiplicative updates can be described as an algorithm below:

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:

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

and

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

Algorithm 1: Multiplicative update algorithm for NMF

It is shown in [22] 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. [28] 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 evaluation 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.

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

- Initialize W as 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. [29]

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 concrete examples 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 [21] was the human activity of recognizing objects from basic parts - especially when using human vision which is shown to be designed 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 like the following:

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

[13]

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 [21] 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 [21], 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 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 [13].

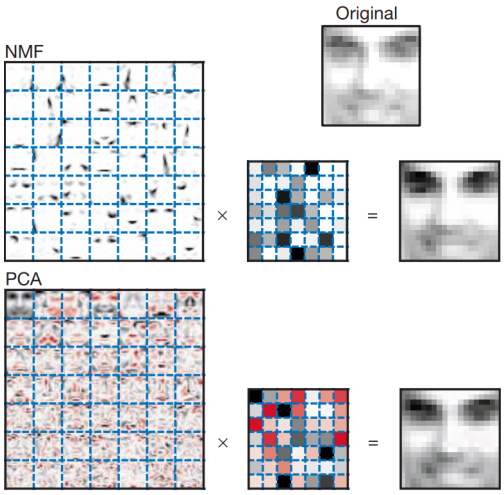


Figure 1.1: Comparison between dimensionality reduction algorithms - *NMF* and *PCA*, as shown in [21].

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.

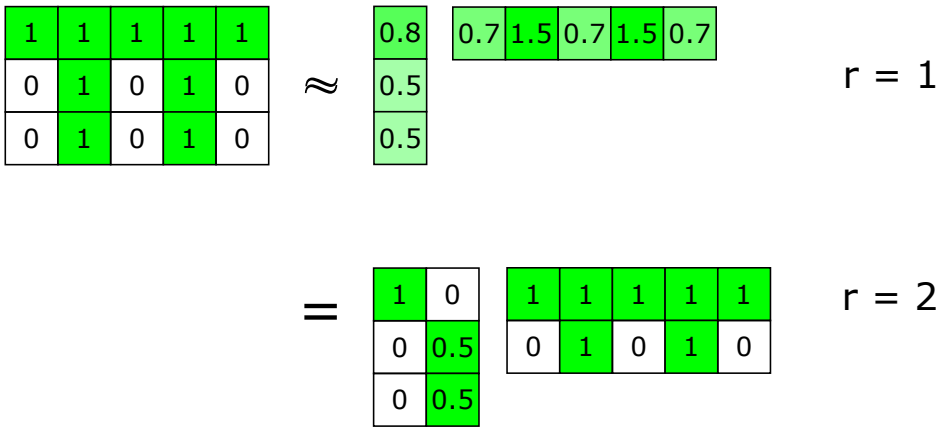
The first property which will be shown is that the solution to non-negative matrix factorization is **not unique**. A matrix and its inverse can transform the two factorization matrices, for example as:

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

If the matrices $\bar{W} = WB$ and $\bar{H} = B^{-1}H$ are non-negative then they form another solution to the *NMF* problem.[41]

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. [26]



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.[35] With certain constraints, it is however possible to solve the *NMF* problem in polynomial time - for example it is shown in [18] 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).[10]

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.

[10]

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*.

[19]

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.

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 looked upon as a lossy compression tool (more on lossy compression in chapter 3), as an original matrix of size $n \times p$ 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 $n \times p$, the dimensions of matrix W $n \times r$ and the dimensions of matrix H $r \times p$, then this requirement can be formalized as:

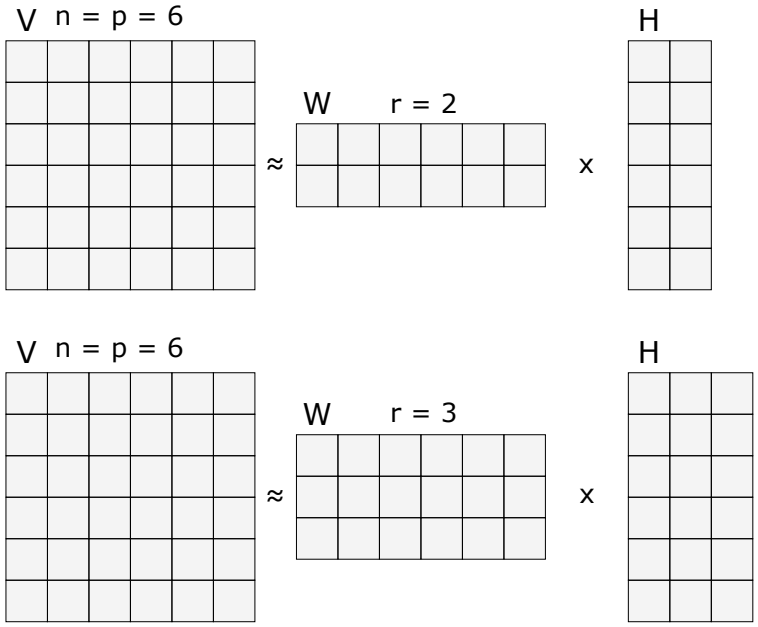
$$\begin{aligned} n \times r + r \times p &< n \times p \\ r(n+p) &< n \times p \\ r &< \frac{n \times p}{n+p} \end{aligned}$$

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

$$\begin{aligned} r &< \frac{n \times n}{n+n} \\ r &< \frac{n^2}{2n} \\ r &< \frac{n}{2} \end{aligned}$$

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 and their representations color models. 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 spaces 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*. More about sampling can be found for example in [42].

The value assigned to a pixel $I[m, n]$ determines its color. The following sections explore the common color encoding options.

2.2 Color models

A color model is a mathematical model used for describing the way of representing colors, usually as tuples of numbers (although this is not necessary, as in i.e. the *grayscale* model). When associated with a description of how the tuple components are interpreted, colors can be interpreted. The color models described are the *RGB* color model, the $Y'C_BC_R$ color space (the set of colors which can be used) which is a transformation using the *RGB* model and the *grayscale* model.

2.2.1 RGB

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.[14] In order to represent a color, components of each color (red, green and blue) are added together. As these components are added together, the *RGB* model is considered to be an additive one.

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 $width * height * 3$ 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 on the figure 2.1

2.2.2 $Y'C_BC_R$

Y'CbCr, also written as $Y'C_BC_R$ 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.

2.2.2.1 Luma

The *luma* value is represented in the $Y'C_BC_R$ model by the symbol Y' and representes 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. [25] 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]:

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

2.2.2.2 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:

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

[12]

2.2.2.3 Use of $Y'C_BC_R$

Common usage of $Y'C_BCR$ stems from the human eye being more sensitive to the differences in luminance than color differences. By transforming colors into the $Y'C_BCR$ color space, it is possible to separate the color data from luminance and reduce the amount of color difference signals - thus less resolution can be allocated for chrominance while keeping high resolution for the luma information. This process is called *chroma subsampling* and is widely used in video encoding schemes, as well as in the JPEG image format. [20]

An image decomposed into the Y', C_B and C_R components can be seen on the figure 2.2

2.2.3 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 $width * height$, not counting the header and other data used by the specific file format.

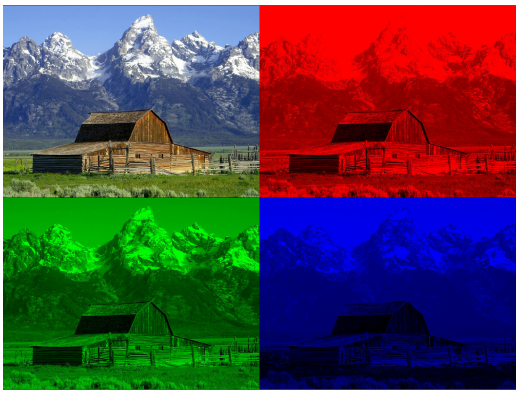


Figure 2.1: An image decomposed into the R, G and B channels as used in the RGB color model.

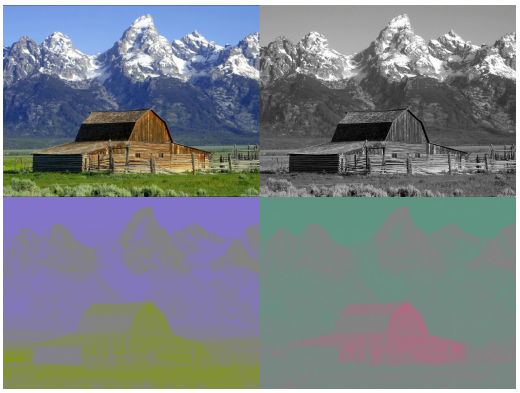


Figure 2.2: An image decomposed into the Y', C_B and C_R components, as used in the $Y'C_BC_R$ color space. The Y' component also represents the grayscale image.

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 algorithms* or *lossy algorithms*. Lossy algorithms generally provide much higher compression than lossless algorithms do, but at the cost of losing information.[33]

3.1.1 Lossless algorithms

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

3.1.2 Lossy algorithms

However, not all usage scenarios 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 with 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.

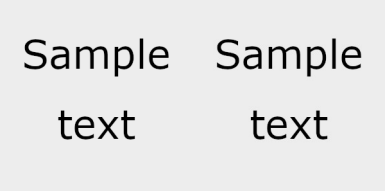


Figure 3.1: A sample image showing how accurate a lossy algorithm can be. The left image has been compressed using a lossless algorithm, the right image has been compressed using a lossy algorithm (JPEG). In this case, essentially no artifact can be seen unless zoomed in (as seen in the figure 3.2).

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.

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 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).[33]

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.[36] Thus, a sequence such as

$$10000, 10002, 10001, 99999, 10000$$

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

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

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

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.

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.[33]

While compressing image data using standard compression techniques, 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.[31]

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 signal source. The redundancies commonly present in images are these ones:

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

[38]

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, an information about how many pixels of the same color are stored in a row. 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.

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 algorithms, it potentially stops being the most important metric when evaluating lossy algorithms. When evaluating lossy algorithms, the important information is not only quantification of how much was the data compressed but also evaluating how much information was actually lost in the compression process. As an example, if a lossy 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 when measuring the performance of a lossy compression algorithm are the *mean squared error* (MSE) and *peak signal-to-noise ratio* (PSNR).[32]

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.

[24][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.[15] 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 $N \times N$ 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.

[39]

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 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.

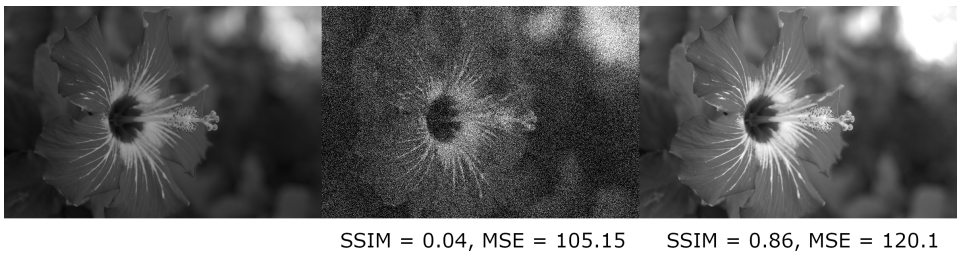


Figure 3.4: Figure visualizing the problems of MSE and PSPNR when compared to SSIM as an image quality metric. Original photography source: [4]

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.

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.[11]

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 RGB to $Y'C_BC_R$.
- The resolution of chroma is reduced - this is done as the human eye is less sensitive to color details than brightness.
- The image is split into 8×8 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 8×8 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 8×8 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.

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 algorithm of choice (more on the specifics of implementation in the following chapter).

This scheme is visualized in the figure 4.1.

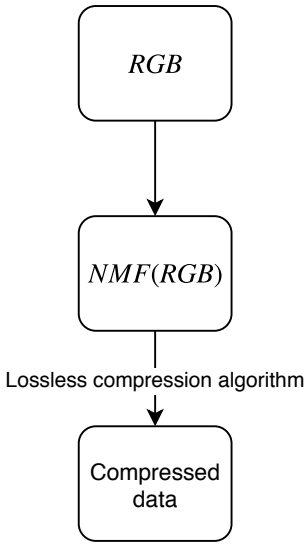


Figure 4.1: Visualization of the naive compression scheme. Note that NMF(RGB) actually represents two matrices which are stored next to each other in one data structure.

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. More on the specifics in the following chapter which is related to implementation.

This scheme is visualized in the figure 4.2.

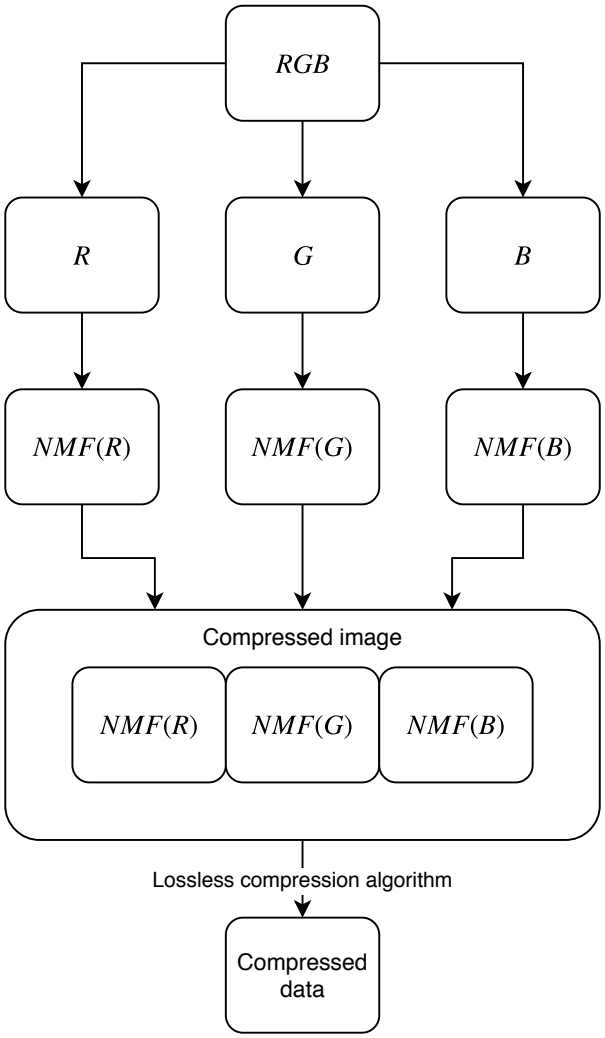


Figure 4.2: Visualization of the compression scheme which performs NMF on each of the RGB components.

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.

4.1.4 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:

- 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.
- When using a lossless 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.

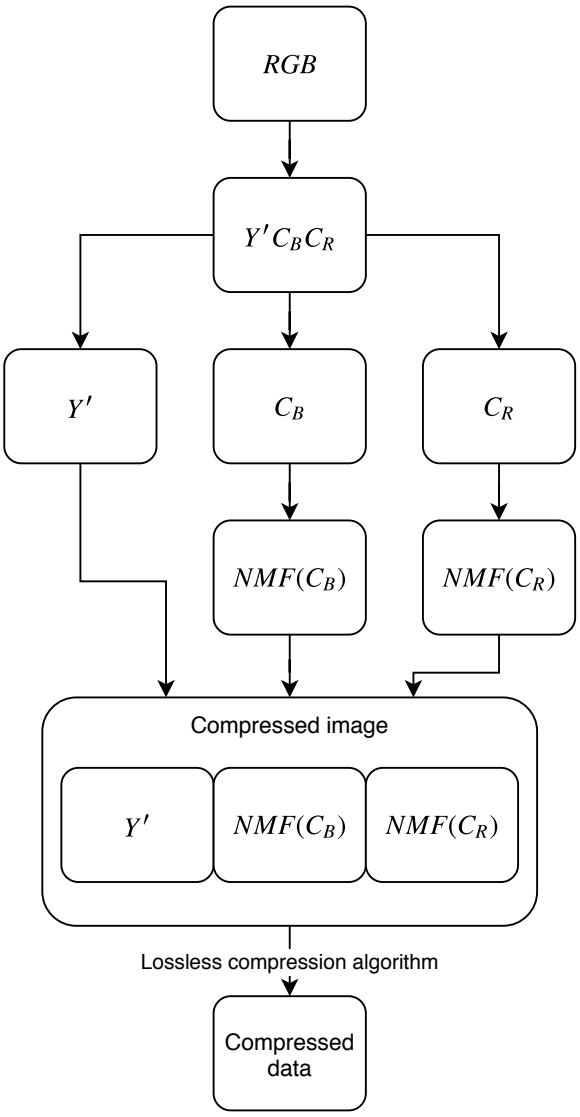


Figure 4.3: Visualization of the compression scheme based on using $Y'C_BC_R$.

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 in the *Python programming language*, which has been chosen for its high readability as well as its good support for 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.[27]

- *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.[43]

- *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 YC_BC_R).[40]

- *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 it is a library commonly used for compression. The other reason for choosing zlib was also that it is a part of the Python standard library.

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 evaluating 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.

5.3 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 YC_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 $height \times width$ (luma matrix in the $Y'C_BC_R$ scheme), $width \times rank$ or $rank \times height$ (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 zlib with the highest possible compression setting.

5.4 Possible improvements

While the compression schemes were implemented, there are still some areas which would still allow for a more efficient implementation. Some of the areas noted are the following:

- Performing independent non-negative matrix factorization in parallel (such as in the case of the YC_BC_R scheme, where factorizing the C_B and the C_R matrices could run in parallel as these two operations are fully independent).
- 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 zlib 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 [16] used constrained non-negative matrix factorization for compressing grayscale images and reports significantly faster compression time and 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.

Chapter 6

Experiments and results

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

- Peak signal-to-noise ratio - calculated both as an average of PSNR for each RGB component and averaged as well as evaluating PSNR solely using the luma component.
- Mean squared error - calculated both as an average of MSE for each RGB component and averaged as well as evaluating MSE solely using the luma component.
- Compression ratio.
- Compression time.

Together with that, the compressed images will be analyzed subjectively.

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 compression scheme (NNDSVD, as noted in the previous chapter). On the other hand, the effect of rank and the maximum number of iterations on the quality of compression will be explored - thus these values will not be constant.

6.2 Choice of images

http://imagecompression.info/test_images/

Conclusion

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

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

[11] 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

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

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

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

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

[16] 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.

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

[18] 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

[19] 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.

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

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

[22] 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

[23] 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.

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

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

[26] 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

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

[28] 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.

[29] 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.

[30] 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

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

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

[33] 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.

[34] 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.

[35] 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

[36] 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.

[37] 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

[38] 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.

[39] 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.

[40] 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

[41] 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

[42] Young, Ian T., Gerbrands, Jan J., van Vliet, Lucas J., data Koninklijke Bibliotheek, Cip, Haag, Den, Theodore, Young Ian, Jacob, Gerbrands Jan, Vliet, Van, and Jozef, Lucas. *Fundamentals Of Image Processing*, 1995.

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

Appendix A

Acronyms

NMF Non-negative matrix factorization

PNG Portable Network Graphics

JPEG

PCA Principal Component Analysis

SVD Singular Value Decomposition

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format