Quantization-free Lossy Image Compression Using Integer Matrix Factorization*

Pooya Ashtari[†1], Pourya Behmandpoor[‡1], Fateme Nateghi Haredasht[2], Jonathan H. Chen[2], Panagiotis Patrinos[1] and Sabine Van Huffel[1]

[1]*Department of Electrical Engineering (ESAT), STADIUS Center, KU Leuven, Belgium*
[2]*Stanford Center for Biomedical Informatics Research, Stanford University, CA, USA*

Abstract

Lossy image compression is essential for efficient transmission and storage. Traditional compression methods mainly rely on discrete cosine transform (DCT) or singular value decomposition (SVD), both of which represent image data in continuous domains and therefore necessitate carefully designed quantizers. Notably, SVD-based methods are more sensitive to quantization errors than DCT-based methods like JPEG. To address this issue, we introduce a variant of integer matrix factorization (IMF) to develop a novel quantization-free lossy image compression method. IMF provides a low-rank representation of the image data as a product of two smaller factor matrices with bounded integer elements, thereby eliminating the need for quantization. We propose an efficient, provably convergent iterative algorithm for IMF using a block coordinate descent (BCD) scheme, with subproblems having closed-form solutions. Our experiments on the Kodak and CLIC 2024 datasets demonstrate that our IMF compression method consistently outperforms JPEG at low bit rates below 0.25 bits per pixel (bpp) and remains comparable at higher bit rates. We also assessed our method's capability to preserve visual semantics by evaluating an ImageNet pre-trained classifier on compressed images. Remarkably, our method improved top-1 accuracy by over 5 percentage points compared to JPEG at bit rates under 0.25 bpp. The project is available at https://github.com/pashtari/lrf.

Keywords: Image Compression, Integer Matrix Factorization, Quantization-free Compression, Low-rank Approximation.

1. Introduction

Lossy image compression involves reducing the storage size of digital images by discarding some image data that are redundant or less perceptible to the human eye. This is crucial for efficiently storing and transmitting images, particularly in applications where bandwidth or storage resources are limited, such as web browsing, streaming, and mobile platforms. Lossy image compression methods enable adjusting the degree of compression, providing a selectable tradeoff between storage size and image quality. Widely used methods such as JPEG [1] and JPEG 2000 [2] follow the *transform coding* paradigm [3]. They use orthogonal linear transformations, such as discrete cosine transform (DCT) [4] and discrete wavelet transform (DWT) [5], to decorrelate small image blocks. Since these transforms map image data into a continuous do-

*This research received funding from the Flemish Government (AI Research Program). Sabine Van Huffel and Pooya Ashtari are affiliated with Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium.

[†]Corresponding author: Pooya Ashtari. *Email:* pooya.ashtari@esat.kuleuven.be

[‡]Pooya Ashtari and Pourya Behmandpoor contributed equally to this work.

main, quantization is necessary before coding into bytes. Unfortunately, as quantization errors can significantly degrade compression performance, the quantizers must be carefully crafted to minimize this impact, which further complicates codec design.

Another promising paradigm relies on low-rank approximation techniques, with singular value decomposition (SVD) being a notable example. SVD is recognized as the deterministically optimal transform for energy compaction [6]. In practice, current SVD-based methods [6, 7, 8] can represent image data only with factors that contain floating-point elements, necessitating a quantization layer prior to any byte-level processing. However, this quantization often results in suboptimal compression performance, as SVD is more sensitive to quantization errors compared to transform-based methods like JPEG, especially at low bit rates.

Motivated by this, we introduce a new variant of integer matrix factorization (IMF), and based on that, develop an effective *quantization-free* lossy image compression method. Our IMF formulation provides a low-rank representation of the image data as a product of two smaller factor matrices with *bounded integer* elements. Since we can directly store and losslessly process these integer matrices at the byte level, quantization is no longer needed, making IMF arguably better suited than SVD for image compression. Another advantage of IMF is that the reshaped factor matrices can be treated as 8-bit grayscale images, allowing any lossless image compression standard to be seamlessly integrated into the proposed framework. We propose an efficient iterative algorithm for IMF using a block coordinate descent (BCD) scheme, where each column of a factor matrix is taken as a block and updated one at a time using a closed-form solution.

Our contributions are summarized as follows. We propose a new IMF formulation that enables *quantization-free* image compression. Moreover, we introduce an efficient algorithm for the IMF problem and prove its convergence. Finally, to the best of our knowledge, this work is the first effort to explore IMF for image compression, presenting the first algorithm based on a low-rank approach that significantly outperforms SVD and competes favorably with JPEG, particularly at low bit rates. Our method narrows the gap between factorization and quantization by integrating them into a single layer and optimizing the compression system.

2. Related Work

Transform Coding. Transform coding is a widely used approach in lossy image compression, leveraging mathematical transforms to decorrelate pixel values and represent image data more compactly. One of the earliest and most influential methods is the discrete cosine transform (DCT) [4], used in JPEG [1], which converts image data into the frequency domain, prioritizing lower frequencies to retain perceptually significant information. The discrete wavelet transform (DWT) [5], used in JPEG 2000 [2], offers improved performance by capturing both frequency and location information, leading to better handling of edges and textures [9]. More recently, the WebP [10] and HEIF [11, 12] formats combine DCT and intra-frame prediction to achieve superior compression and quality compared to JPEG.

Learned Image Compression (LIC). Recently, learned image compression (LIC) has gained attention for potentially outperforming traditional methods by leveraging deep neural networks. Ballé et al. [13] pioneered this area with an end-to-end trainable convolutional neural network based on variational autoencoders. Cheng et al. [14] incorporated a simplified attention module and discretized Gaussian mixture likelihoods for achieving a more accurate and flexible entropy model. Liu et al. [15] combined transformers and CNNs to exploit the local modeling ability of convolutions and the global modeling ability of the attention mechanism. Yang and Mandt [16] introduced diffusion models into LIC, using a denoising decoder to iteratively reconstruct a compressed image. Despite these advancements, the high computational complexity of LIC methods remains a significant limitation, particularly for real-time applications and resource-constrained

environments.

Low-rank Techniques. Low-rank approximation can provide a compact representation by decomposing image data into smaller components. Notably, truncated singular value decomposition (tSVD) is a classical technique that decomposes images into singular values and vectors, retaining only the most significant components to achieve compression [6, 7]. Hou et al. [8] proposed sparse low-rank matrix approximation (SLRMA) for data compression, which is able to explore both the intra- and inter-coherence of data samples simultaneously from the perspective of optimization and transformation. More recently, Yuan and Haimi-Cohen [17] introduced a graph-based low-rank regularization to reduce compression artifacts near block boundaries at low bit rates.

Integer Matrix Factorization. There are applications where meaningful representation of data as discrete factor matrices is crucial. While typical low-rank techniques like SVD and nonnegative matrix factorization (NMF) are inappropriate for such applications, integer matrix factorization (IMF) ensures the integrality of factors to achieve this goal. Lin et al. [18] investigates IMF to effectively handle discrete data matrices for cluster analysis and pattern discovery. Dong et al. [19] introduce an alternative least squares method for IMF, verifying its effectiveness with some data mining applications. A closely related topic is boolean matrix factorization (BMF), where factors are constrained to binary matrices. BMF has been applied to some data mining [20] and machine learning [21] problems. However, to the best of our knowledge, IMF has not yet been explored in image compression. Therefore, this work investigates IMF for image compression and argues that it can serve as a powerful tool for this purpose.

3. Method

3.1. Overall Encoding Framework

The proposed compression method follows a *transform coding* paradigm, but it does not involve quantization. Figure 1 illustrates an overview of our encoding pipeline based on integer matrix factorization (IMF). The encoder accepts an RGB image with dimensions $H \times W$ and a color depth of 8 bits, represented by the tensor $\boldsymbol{\mathcal{X}} \in \{0, \dots, 255\}^{3 \times H \times W}$. Each step of encoding is described in the following.

Color Space Transformation. Analogous to the JPEG standard, the image is initially transformed into the $\mathrm{YC_BC_R}$ color space. Let $\boldsymbol{Y} \in [0, 255]^{H \times W}$ represent the *luma* component, and $\boldsymbol{C}_B, \boldsymbol{C}_R \in [0, 255]^{\frac{H}{2} \times \frac{W}{2}}$ represent the blue-difference and red-difference *chroma* components, respectively. Note that as a result of this transformation, the elements of the *luma* ($\boldsymbol{Y}$) and *chroma* ($\boldsymbol{C}_B$, $\boldsymbol{C}_R$) matrices are not limited to integers and can take any value within the interval $[0, 255]$.

Chroma Downsampling. After conversion to the $\mathrm{YC_BC_R}$ color space, the *chroma* components $\boldsymbol{C}_B$ and $\boldsymbol{C}_R$ are downsampled using average-pooling with a kernel size of $(2, 2)$ and a stride of $(2, 2)$, similar to the process used in JPEG. This downsampling exploits the fact that the human visual system perceives far more detail in brightness information (*luma*) than in color saturation (*chroma*).

Patchification. After *chroma* downsampling, we have three components: the *luma* component $\boldsymbol{Y} \in [0, 255]^{H \times W}$ and the *chroma* components $\boldsymbol{C}_B, \boldsymbol{C}_R \in [0, 255]^{\frac{H}{2} \times \frac{W}{2}}$. Each of the matrices is split into non-overlapping 8×8 patches. If a dimension of a matrix is not divisible by 8, the matrix is first padded to the nearest size divisible by 8 using reflection of the boundary values. These patches are then flattened into row vectors and stacked vertically to form matrices $\boldsymbol{X}_Y \in [0, 255]^{\frac{HW}{64} \times 64}$, $\boldsymbol{X}_{C_B} \in [0, 255]^{\frac{HW}{256} \times 64}$, and $\boldsymbol{X}_{C_R} \in [0, 255]^{\frac{HW}{256} \times 64}$. Later,

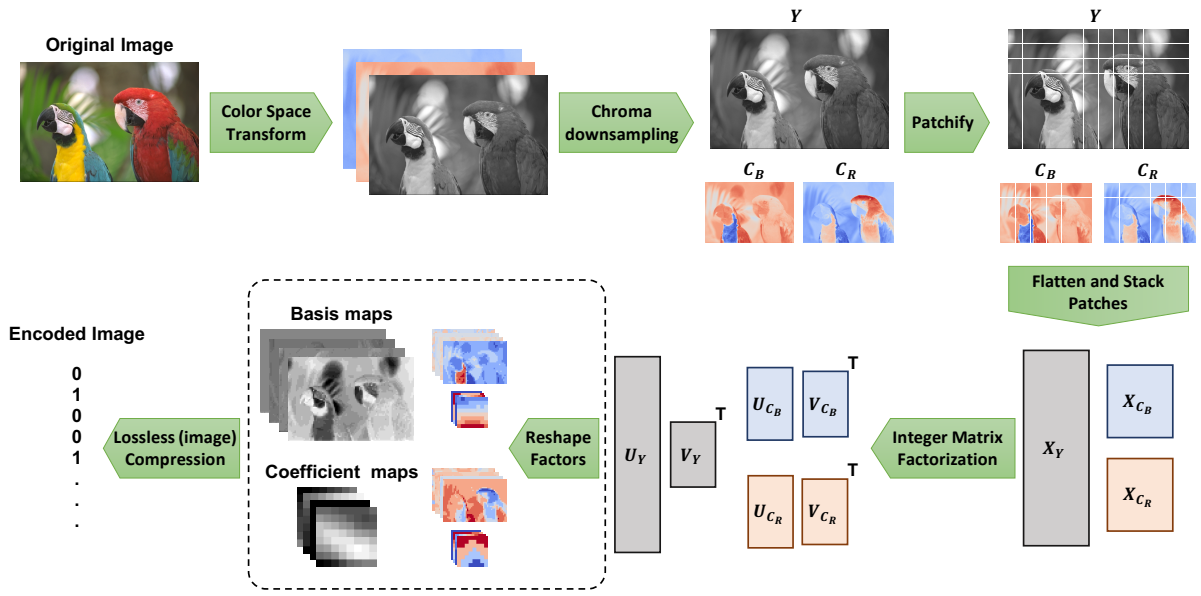


Figure 1: An illustration of the encoder for our image compression method.

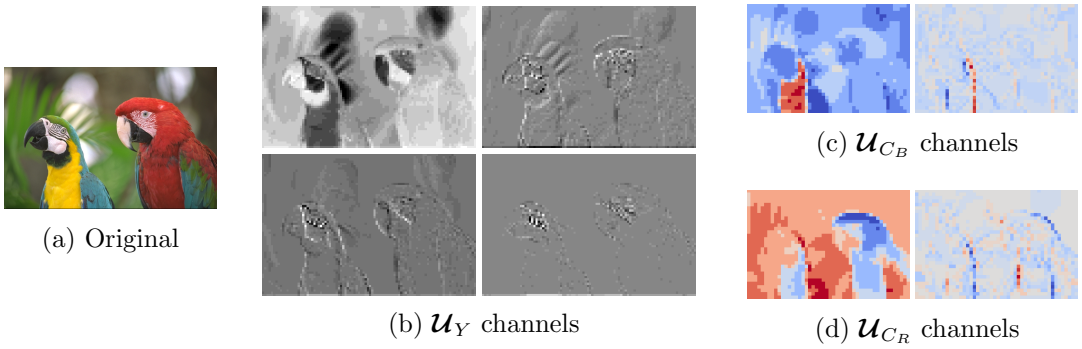
these matrices will be low-rank approximated using IMF. Note that this patchification technique differs from the block splitting in JPEG, where each block is subject to DCT individually and processed independently. This patchification technique not only captures the locality and spatial dependencies of neighboring pixels but also performs better when combined with the matrix decomposition approach for image compression.

Low-rank Approximation. We now apply a low-rank approximation to the matrices $\boldsymbol{X}_Y$, $\boldsymbol{X}_{C_B}$, and $\boldsymbol{X}_{C_R}$, which is the core of our compression method that provides a lossy compressed representation of these matrices. The low-rank approximation [22] aims to approximate a given matrix $\boldsymbol{X} \in \mathbb{R}^{M\times N}$ by

$$\boldsymbol{X} \approx \boldsymbol{U}\boldsymbol{V}^{\mathsf{T}} = \sum_{r=1}^{R} U_{:r} {V_{:r}}^{\mathsf{T}}, \tag{1}$$

where $\boldsymbol{U} \in \mathbb{R}^{M\times R}$ and $\boldsymbol{V} \in \mathbb{R}^{N\times R}$ are *factor matrices* (or simply *factors*), $R \leq \min(M, N)$ represents the *rank*, $U_{:r}$ and $V_{:r}$ represent the r-th columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. We refer to $\boldsymbol{U}$ as the *basis matrix* and $\boldsymbol{V}$ as the *coefficient matrix*. By selecting a sufficiently small value for R, the factor matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, with a combined total of $(M + N)R$ elements, offer a compact representation of the original matrix $\boldsymbol{X}$, which has MN elements, capturing the most significant patterns in the image. Depending on the loss function used to measure the reconstruction error between $\boldsymbol{X}$ and the product $\boldsymbol{U}\boldsymbol{V}^{\mathsf{T}}$, as well as the constraints on the factor matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, various formulations and variants have been proposed for different purposes [23, 24, 18]. In Section 3.3, we introduce and elaborate on our variant, termed integer matrix factorization (IMF), and argue why it is well-suited and effective for image compression.

Lossless Compression. IMF yields factor matrices $\boldsymbol{U}_Y \in \{0, \ldots, 255\}^{\frac{HW}{64}\times R}$ and $\boldsymbol{V}_Y \in \{0, \ldots, 255\}^{64\times R}$; $\boldsymbol{U}_{C_B} \in \{0, \ldots, 255\}^{\frac{HW}{256}\times R}$ and $\boldsymbol{V}_{C_B} \in \{0, \ldots, 255\}^{64\times R}$; and $\boldsymbol{U}_{C_R} \in \{0, \ldots, 255\}^{\frac{HW}{256}\times R}$ and $\boldsymbol{V}_{C_R} \in \{0, \ldots, 255\}^{64\times R}$ that correspond to $\boldsymbol{X}_Y$, $\boldsymbol{X}_{C_B}$, and $\boldsymbol{X}_{C_R}$. Since these matrices have integer elements, they can be directly encoded by any standard lossless data compression like zlib [25] without the need for a quantization step, which is commonly present in other lossy image compression methods and adds extra complications.



(a) Original

(b) $\boldsymbol{\mathcal{U}}_Y$ channels

(c) $\boldsymbol{\mathcal{U}}_{C_B}$ channels

(d) $\boldsymbol{\mathcal{U}}_{C_R}$ channels

Figure 2: The channels of IMF basis maps for the `kodim23` image from Kodak. (a) shows the original image. The IMF basis maps corresponding to luma (b), blue-difference (c), and red-difference chroma (d) are shown. The channels of basis map with higher energy maintain the overall texture of the original image, where channels with lower energy focus more on subtle changes.

Alternatively, we can first reshape the factor matrices by unfolding their first dimension to obtain R-channel 2D spatial maps, referred to as *factor maps* and represented by the following tensors:

$$\begin{aligned} \boldsymbol{\mathcal{U}}_Y &\in \{0,\dots,255\}^{R\times\frac{H}{8}\times\frac{W}{8}}, \\ \boldsymbol{\mathcal{U}}_{C_B},\boldsymbol{\mathcal{U}}_{C_R} &\in \{0,\dots,255\}^{R\times\frac{H}{16}\times\frac{W}{16}}, \\ \boldsymbol{\mathcal{V}}_Y,\boldsymbol{\mathcal{V}}_{C_B},\boldsymbol{\mathcal{V}}_{C_R} &\in \{0,\dots,255\}^{R\times 8\times 8}. \end{aligned} \tag{2}$$

As each channel of a *factor map* can be treated as an 8-bit grayscale image, we can encode it by any standard lossless image compression method such as PNG. For images with a resolution of $H, W \gg 64$, which are most common nowadays, the *basis maps* ($\boldsymbol{\mathcal{U}}$) are significantly larger than the *coefficient maps* ($\boldsymbol{\mathcal{V}}$), accounting for the majority of the storage space. Interestingly, in practice, the IMF *basis maps* turn out to be meaningful images, each capturing some visual semantic of the image (see Figure 2 for an example). Therefore, our IMF approach can effectively leverage the power of existing lossless image compression algorithms, offering a significant advantage over current methods. However, in this work, we take the first approach and use the zlib library [25] to encode factor matrices, creating a stand-alone codec that is independent from other image compression methods.

3.2. Decoding

The decoder receives an encoded image and reconstructs the RGB image by applying the inverse of the operations used by the encoder, starting from the last layer and moving to the first. Initially, the factor matrices are produced by losslessly decompressing the encoded image. The matrices $\boldsymbol{X}_Y$, $\boldsymbol{X}_{C_B}$, and $\boldsymbol{X}_{C_R}$ are calculated through the product of the corresponding factor matrices, according to (1). The *luma* and downsampled *chroma* components are then obtained by reshaping $\boldsymbol{X}_Y$, $\boldsymbol{X}_{C_B}$, and $\boldsymbol{X}_{C_R}$ back into their spatial forms, following the inverse of the patchification step. Subsequently, the downsampled *chroma* components are upsampled to their original size using nearest-neighbor interpolation. Finally, the $\mathrm{YC_BC_R}$ image is converted back into an RGB image.

3.3. Integer Matrix Factorization (IMF)

The main building block of our method is integer matrix factorization (IMF), which is responsible for the lossy compression of matrices obtained through patchification. IMF can be framed

Algorithm 1 The proposed block coordinate descent (BCD) algorithm for IMF.

Input: $\boldsymbol{X} \in \mathbb{R}^{M\times N}$, factorization rank R, factor bounds $[\alpha, \beta]$, # iterations K
Output: Factor matrices $\boldsymbol{U} \in \mathbb{Z}_{[\alpha,\beta]}^{M\times R}$ and $\boldsymbol{V} \in \mathbb{Z}_{[\alpha,\beta]}^{N\times R}$

Initialize $\boldsymbol{U}^{\text{init}}$, $\boldsymbol{V}^{\text{init}}$ using the truncated SVD method, provided by (8) and (9), and set $k = 0$
while $k < K$ **do**
  $k \leftarrow k + 1$
  $\boldsymbol{A} \leftarrow \boldsymbol{X}\boldsymbol{V}^k$
  $\boldsymbol{B} \leftarrow {\boldsymbol{V}^k}^{\mathsf{T}}\boldsymbol{V}^k$
  for $r = 1, \ldots, R$ **do**
    $U_{:r}^{k+1/2} \leftarrow \dfrac{A_{:r} - \sum_{s=1}^{r-1} B_{sr}U_{:s}^{k+1} - \sum_{s=r+1}^{M} B_{sr}U_{:s}^{k}}{\|V_{:r}^k\|^2}$
    $U_{:r}^{k+1} \leftarrow \text{clamp}_{[\alpha,\beta]}(\text{round}(U_{:r}^{k+1/2}))$
  end for
  $\boldsymbol{A} \leftarrow \boldsymbol{X}^{\mathsf{T}}\boldsymbol{U}^{k+1}$
  $\boldsymbol{B} \leftarrow {\boldsymbol{U}^{k+1}}^{\mathsf{T}}\boldsymbol{U}^{k+1}$
  for $r = 1, \ldots, R$ **do**
    $V_{:r}^{k+1/2} \leftarrow \dfrac{A_{:r} - \sum_{s=1}^{r-1} B_{sr}V_{:s}^{k+1} - \sum_{s=r+1}^{N} B_{sr}V_{:s}^{k}}{\|U_{:r}^{k+1}\|^2}$
    $V_{:r}^{k+1} \leftarrow \text{clamp}_{[\alpha,\beta]}(\text{round}(V_{:r}^{k+1/2}))$
  end for
end while
return $(\boldsymbol{U}^K, \boldsymbol{V}^K)$

as an optimization problem, aiming to minimize the reconstruction error between the original matrix $\boldsymbol{X} \in \mathbb{R}^{M\times N}$ and the product $\boldsymbol{U}\boldsymbol{V}^{\mathsf{T}}$, while ensuring that the elements of the factor matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are integers within a specified interval $[\alpha, \beta]$ with integer endpoints, i.e., $\alpha, \beta \in \mathbb{Z}$. Formally, the IMF problem can be expressed as:

$$\begin{aligned} \underset{\boldsymbol{U},\boldsymbol{V}}{\text{minimize}} \quad & \|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{V}^{\mathsf{T}}\|_{\text{F}}^2 \\ \text{subject to} \quad & \boldsymbol{U} \in \mathbb{Z}_{[\alpha,\beta]}^{M\times R}, \boldsymbol{V} \in \mathbb{Z}_{[\alpha,\beta]}^{N\times R}, \end{aligned} \tag{3}$$

where $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm; $R \leq \min(M, N)$ represents the *rank*; and $\mathbb{Z}_{[\alpha,\beta]} \triangleq [\alpha, \beta] \cap \mathbb{Z}$ denotes the set of integers within $[\alpha, \beta]$. Without constraints on the factors, the problem would have an analytic solution through singular value decomposition (SVD), as addressed by the Eckart–Young–Mirsky theorem [22]. If only a nonnegativity constraint were applied (without integrality), variations of nonnegative matrix factorization (NMF) would emerge [23, 26]. The IMF problem (3) poses a challenging integer program, with finding its global minima known to be NP-hard [19, 27]. Only a few iterative algorithms [19, 18] have been proposed to find a "good solution" for some IMF variants in contexts other than image compression. In Section 3.4, we propose an efficient iterative algorithm for the IMF problem (3).

The application of SVD and NMF in image compression is problematic mainly because the resulting factors contain continuous values that must be represented as arrays of floating-point numbers. This necessitates a quantization step that not only adds extra complications but also significantly degrades compression performance due to quantization errors (as demonstrated in Section 4). In contrast, our IMF formulation produces integer factor matrices that can be directly stored and losslessly processed without incurring roundoff errors. The reason for limiting the feasible region to $[\alpha, \beta]$ in our IMF formulation is to enable more compact storage of the

factors using standard integral data types, such as `int8` and `int16`, supported by programming languages. Given that the elements of the input matrix $\boldsymbol{X}$ are in $[0, 255]$, we found the signed `int8` type, which represents integers from -128 to 127, suitable for image compression applications. As a result, our IMF formulation is well-suited for image compression, effectively integrating the factorization and quantization steps into a single, efficient compression process.

3.4. Block Coordinate Descent Scheme for IMF

We propose an efficient algorithm for IMF using the block coordinate descent (BCD) scheme (aka alternating optimization). The pseudocode is provided in Algorithm 1. Starting with some initial parameter values, this approach involves sequentially minimizing the cost function with respect to a single column of a factor at a time, while keeping the other columns of that factor and the entire other factor fixed. This process is repeated until a stopping criterion is met, such as when the change in the cost function value falls below a predefined threshold or the maximum number of iterations is reached. Formally, this involves solving one of the following subproblems at a time:

$$\boldsymbol{u}_r \leftarrow \underset{\boldsymbol{u}_r \in \mathbb{Z}_{[\alpha,\beta]}^{M\times 1}}{\arg\min} \ \|\boldsymbol{E}_r - \boldsymbol{u}_r \boldsymbol{v}_r^{\mathsf{T}}\|_{\mathrm{F}}^2, \tag{4}$$

$$\boldsymbol{v}_r \leftarrow \underset{\boldsymbol{v}_r \in \mathbb{Z}_{[\alpha,\beta]}^{N\times 1}}{\arg\min} \ \|\boldsymbol{E}_r - \boldsymbol{u}_r \boldsymbol{v}_r^{\mathsf{T}}\|_{\mathrm{F}}^2, \tag{5}$$

where $\boldsymbol{u}_r \triangleq U_{:r}$ and $\boldsymbol{v}_r \triangleq V_{:r}$ represent the r-th columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. $\boldsymbol{E}_r \triangleq \boldsymbol{X} - \sum_{s\neq r}^{R} \boldsymbol{u}_s \boldsymbol{v}_s^{\mathsf{T}}$ is the residual matrix. We define one iteration of BCD as a complete cycle of updates across all the columns of both factors. In fact, the proposed algorithm is a $2R$-block coordinate descent procedure, where at each iteration, first the columns of $\boldsymbol{U}$ and then the columns of $\boldsymbol{V}$ are updated (see Algorithm 1). Note that subproblem (5) can be transformed into the same form as (4) by simply transposing its error term inside the Frobenius norm. Therefore, we only need to find the best rank-1 approximation with integer elements constrained within a specific interval. Fortunately, this problem has a closed-form solution, as addressed by Theorem 3.1 below.

Theorem 3.1 (Monotonicity)**.** *The global optima of subproblems* (4) *and* (5) *can be represented by closed-form solutions as follows:*

$$\boldsymbol{u}_r \leftarrow \mathrm{clamp}_{[\alpha,\beta]}\left(\mathrm{round}\left(\frac{\boldsymbol{E}_r \boldsymbol{v}_r}{\|\boldsymbol{v}_r\|^2}\right)\right), \tag{6}$$

$$\boldsymbol{v}_r \leftarrow \mathrm{clamp}_{[\alpha,\beta]}\left(\mathrm{round}\left(\frac{\boldsymbol{E}_r^{\mathsf{T}} \boldsymbol{u}_r}{\|\boldsymbol{u}_r\|^2}\right)\right), \tag{7}$$

where $\mathrm{round}(\boldsymbol{Z})$ *denotes an element-wise operator that rounds each element of* $\boldsymbol{Z}$ *to the nearest integer, and* $\mathrm{clamp}_{[\alpha,\beta]}(\boldsymbol{Z}) \triangleq \max(\alpha, \min(\boldsymbol{Z}, \beta))$ *denotes an element-wise operator that clamps each element of* $\boldsymbol{Z}$ *to the interval* $[\alpha, \beta]$*. Moreover, the cost function in* (3) *is monotonically nonincreasing over BCD iterations of Algorithm 1, which involve sequential updates of* (6) *and* (7) *over columns of* $\boldsymbol{U}$ *and* $\boldsymbol{V}$*.*

Proof. See Appendix A for the proof. □

It is noteworthy that the combination of $\mathrm{round}(\cdot)$ and $\mathrm{clamp}_{[\alpha,\beta]}(\cdot)$ in (6) and (7) can be interpreted as the element-wise projector to $\mathbb{Z}_{[\alpha,\beta]}$. In addition, updates (6) and (7) are presented in Algorithm 1 at steps 7 and 8, and steps 13 and 14, respectively. In Theorem 3.2, the convergence of Algorithm 1 employing these closed-form solutions is established.

Theorem 3.2 (Convergence)**.** *Let* $(U_{:r}^k)_{k\in\mathbb{N}}$ *and* $(V_{:r}^k)_{k\in\mathbb{N}}$ *for* $r \in \{1, \ldots, R\}$ *be sequences generated by the proposed Algorithm 1. Then all sequences are bounded and convergent to a locally optimal point of optimization problem* (3)*.*

Proof. See Appendix B for the proof. □

Initialization. The initial values of factors can significantly impact the convergence performance of the BCD algorithm. We found that the convergence with naive random initialization can be too slow. To address this issue, we propose an initialization method using SVD. The procedure is straightforward. First, the truncated SVD of the input matrix $\boldsymbol{X} \in \mathbb{R}^{M \times N}$ is computed as $\tilde{\boldsymbol{U}}\boldsymbol{\Sigma}\tilde{\boldsymbol{V}}^{\mathsf{T}}$, where $\boldsymbol{\Sigma} \in \mathbb{R}^{R \times R}$ is a diagonal matrix corresponding to the R largest singular values. $\tilde{\boldsymbol{U}} \in \mathbb{R}^{M \times R}$ and $\tilde{\boldsymbol{V}} \in \mathbb{R}^{N \times R}$ contain the corresponding left-singular vectors and right-singular vectors in their columns, respectively. The initial factors are then calculated as follows:

$$\boldsymbol{U}^{\text{init}} = \text{clamp}_{[\alpha,\beta]}(\text{round}(\tilde{\boldsymbol{U}}\boldsymbol{\Sigma}^{\frac{1}{2}})), \tag{8}$$

$$\boldsymbol{V}^{\text{init}} = \text{clamp}_{[\alpha,\beta]}(\text{round}(\boldsymbol{\Sigma}^{\frac{1}{2}}\tilde{\boldsymbol{V}})). \tag{9}$$

Essentially, this means we first low-rank approximate $\boldsymbol{X}$ and then project the elements of the resulting factor matrices into $\mathbb{Z}_{[\alpha,\beta]}$.

4. Experiments

4.1. Setup

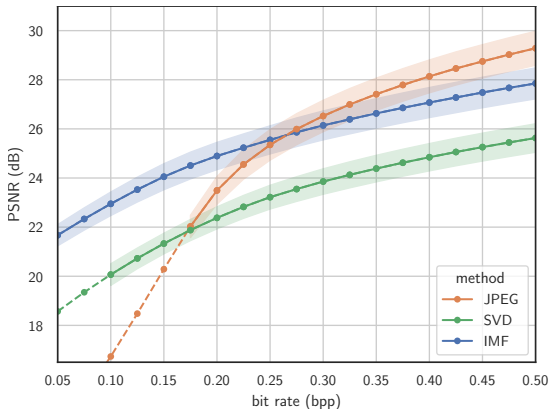
IMF Configuration. In our IMF implementation, we used a default patch size of 8×8. The default factor bounds were set to $[-16, 15]$. Unless otherwise specified, the number of BCD iterations was set to 10 although our ablation studies in Section 4.6 suggest that even 2 iterations may suffice in practice (see Figure 6b). For lossless compression of factors, we encoded and decoded each column of a factor separately using the zlib library [25].

Baseline Codecs. We compared our IMF method against JPEG and SVD baselines. For JPEG compression, we used the Pillow library [28]. Our SVD-based baseline follows the same framework as the proposed method (described in 3.1) but substitutes truncated SVD for IMF. This is followed by uniform quantization of the SVD factor matrices before lossless compression using zlib [25]. This differs from IMF compression, which benefits from the integrality of factors by directly encoding them with zlib, eliminating the need for any intermediate quantization step.

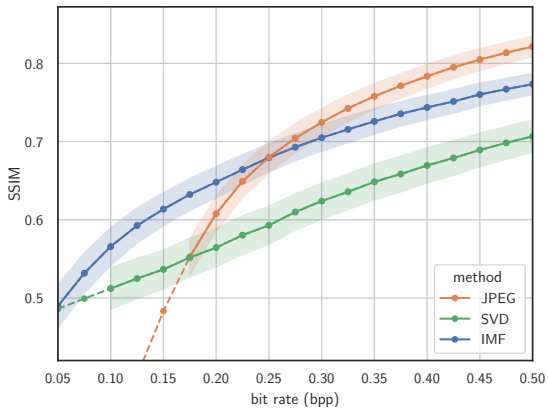
Datasets. To validate the effectiveness of our method, we conducted experiments using the widely-used **Kodak** dataset [29], consisting of 24 lossless images with a resolution of 768×512. To evaluate the robustness of our method in a higher-resolution setting, we also experimented with the **CLIC 2024** validation dataset [30], which contains 30 high-resolution, high-quality images. Additionally, we assessed the compression methods by their ability to retain visual semantics. This was achieved by evaluating a pre-trained ImageNet classifier on compressed images from the **ImageNet** validation set [31], consisting of 50,000 images with a resolution of 224×224 across 1,000 classes.

Metrics. To evaluate the rate-distortion performance of methods on the Kodak and CLIC 2024 datasets, we measured the bit rate in bits per pixel (bpp) and assessed the quality of the reconstructed images using peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM). Then, these metrics were plotted as functions of bit rate for each method to illustrate their rate-distortion performance.

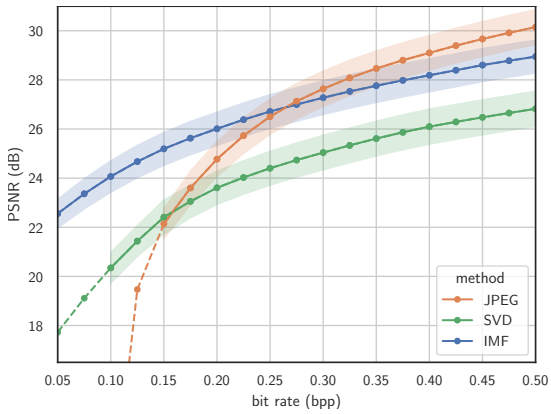
More precisely, to construct a rate-distortion curve for each method on each dataset, we first measured the PSNR/SSIM values at feasible bit rates for each image. Then, PSNR/SSIM was interpolated at evenly spaced bit rates ranging from 0.05 bpp to 0.5 bpp using LOESS (locally



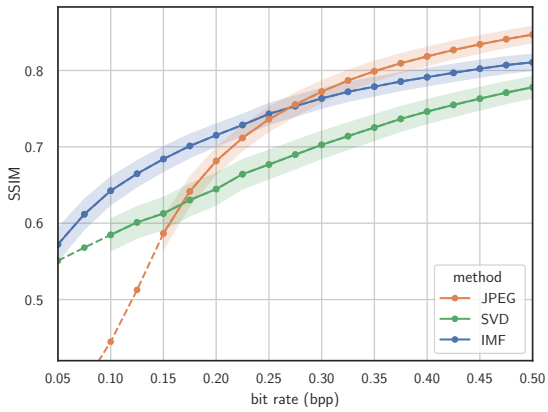
(a) Kodak - PSNR



(b) Kodak - SSIM



(c) CLIC 2024 - PSNR



(d) CLIC 2024 - SSIM

Figure 3: Rate-distortion performance on the Kodak (top panels) and CLIC 2024 (bottom panels) datasets. The average PSNR (left panels) and average SSIM (right panels) for each method are plotted as functions of bit rate. Shaded areas represent standard errors. Dashed lines indicate extrapolated values predicted using LOESS [32] for extremely low bit rates that are otherwise unattainable.

estimated scatterplot smoothing) [32]. Finally, the interpolated values were averaged over all images at each of these bit rates.

4.2. Rate-Distortion Performance

Figure 3 illustrates the rate-distortion curves comparing the performance of IMF, SVD, and JPEG compression methods.

Kodak. On the Kodak dataset, as shown in Figures 3a and 3b, our IMF method consistently outperforms JPEG at low bit rates below 0.25 bpp and remains comparable at higher bit rates in terms of both PSNR and SSIM. Furthermore, IMF significantly surpasses the SVD-based baseline across all bit rates.

CLIC 2024. A similar trend is observed with the CLIC 2024 dataset, as shown in Figures 3c and 3d. Here, the PSNR (Figure 3c) and SSIM (Figure 3d) results further confirm the competitive performance of IMF across all bit rates, with a particularly notable margin at bit rates lower than 0.25 bpp. Specifically, at a bit rate of 0.15 bpp, IMF achieves an PSNR of over 25 dB, compared to approximately 22 dB for both JPEG and SVD. This supports the robustness of IMF in preserving visual quality across different datasets.

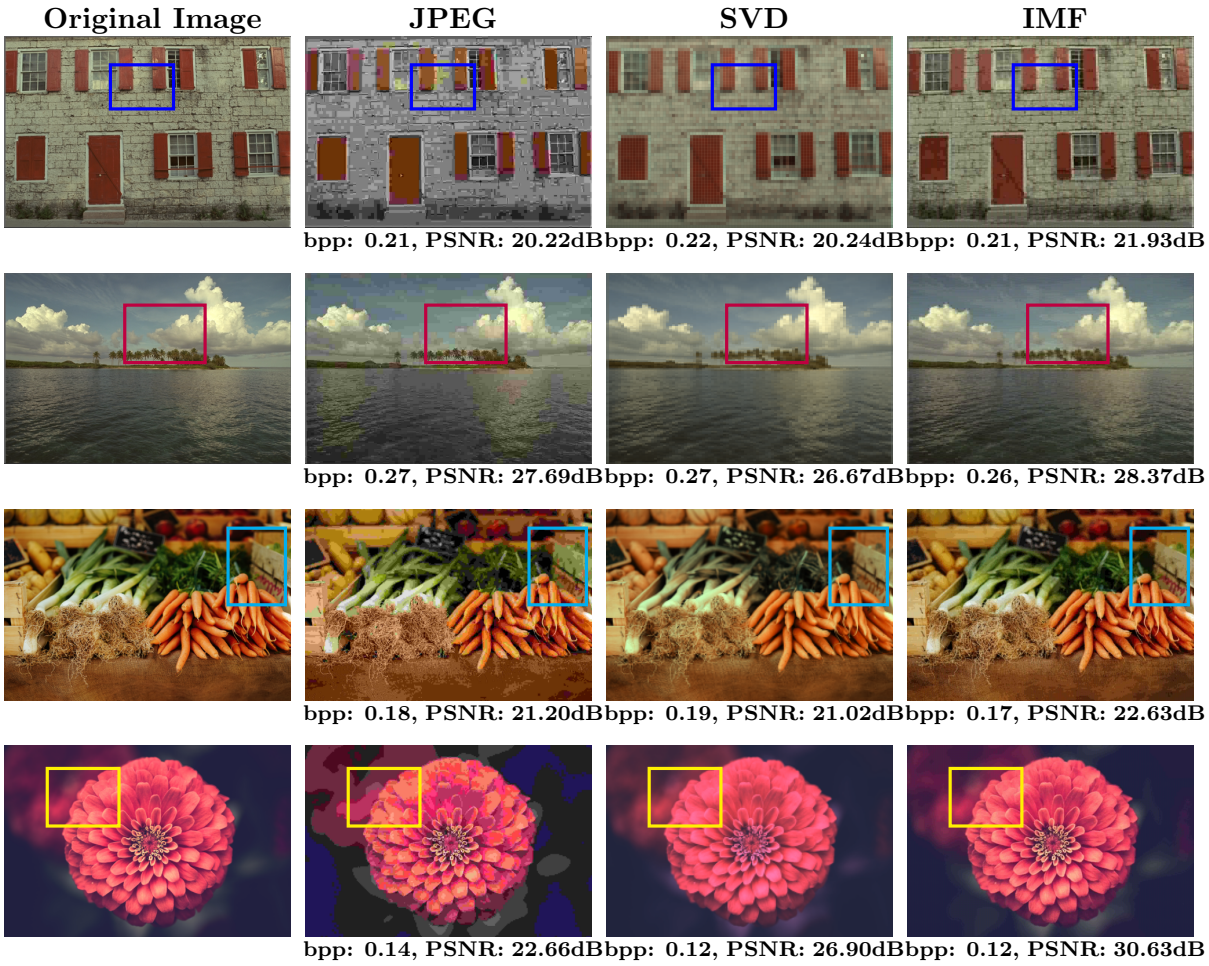


Figure 4: Qualitative performance comparison on example images from the Kodak (top two rows) and the CLIC 2024 (bottom two rows) datasets. Each column shows the original image, JPEG, SVD, and IMF compression results respectively. The bit rate and PSNR values for each compressed image is reported. The colored bounding boxes highlight artifacts produced by JPEG and SVD compression.

4.3. Qualitative Performance

Figure 4 compares various compression methods using images from the Kodak (top two rows) and CLIC 2024 (bottom two rows) datasets, compressed at similar bit rates.

In the building image (first row), JPEG compression, with a PSNR of 20.22 dB at a bit rate of 0.21 bpp, introduces *blocking artifacts* and changes the facade color, as visible in the blue boxes. SVD compression reduces these artifacts but causes blurriness. Our IMF compression, with a similar bit rate but a higher PSNR (21.93 dB), maintains both texture and sharpness with minimal artifacts.

In the seascape image (second row), JPEG causes blocking and significant *color bleeding artifacts*, such as the redness in the cloud area marked by the red boxes and also on the water surface (outside the red box). SVD reduces color distortion but still has blockiness and blurriness. IMF preserves the color and texture of clouds and water more effectively, resulting in a more visually pleasing image.

In the vegetables image (third row), JPEG yields visible color distortion (marked by the cyan boxes), while SVD introduces significant blurriness. IMF, however, effectively preserves the

Table 1: Mean encoding and decoding CPU times for different compression methods at bit rates of 0.15 bpp and 0.25 bpp, measured on the Kodak and CLIC 2024 datasets.

Method	Kodak				CLIC 2024			
	Bitrate = 0.15 bpp		Bitrate = 0.25 bpp		Bitrate = 0.15 bpp		Bitrate = 0.25 bpp	
	Encoding	Decoding	Encoding	Decoding	Encoding	Decoding	Encoding	Decoding
JPEG	9.40 ms	4.54 ms	9.76 ms	4.23 ms	60.01 ms	26.76 ms	60.33 ms	25.75 ms
SVD	27.99 ms	1.33 ms	25.63 ms	1.23 ms	96.83 ms	5.29 ms	98.82 ms	4.82 ms
IMF	64.04 ms	2.82 ms	82.57 ms	2.66 ms	256.15 ms	9.91 ms	374.52 ms	9.06 ms

color fidelity and detail.

In the flower image (fourth row), JPEG compression, with a PSNR of 20.22 dB at a bit rate of 0.14 bpp, exhibits severe *color banding artifacts* around the flower boundary. SVD compression offers smoother gradients but remains blurry. Our IMF compression maintains the gradient fidelity and intricate petal distinctions, achieving a significantly higher PSNR of 30.63 dB at a lower bit rate of 0.12 bpp.

4.4. Run Time

The encoding and decoding times at bit rates of 0.15 bpp and 0.25 bpp for each method on Kodak and CLIC 2024 are reported in Table 1. All experiments in this section were conducted on 2 Xeon Gold 6140 CPUs @ 2.3 GHz (Skylake), each with 18 cores, and with 192 GiB RAM.

JPEG compression consistently has a lower encoding time compared to IMF and SVD. However, IMF and SVD have a significant advantage in decoding speed over JPEG, with SVD being the fastest. Specifically, IMF decodes more than twice as fast as JPEG on the CLIC 2024 dataset across all bit rates. This is due to the heavier FFT operation in the JPEG decoder compared to the lighter matrix multiplication in the IMF decoder. Overall, IMF is preferable for applications requiring fast decoding and high image quality at low bit rates.

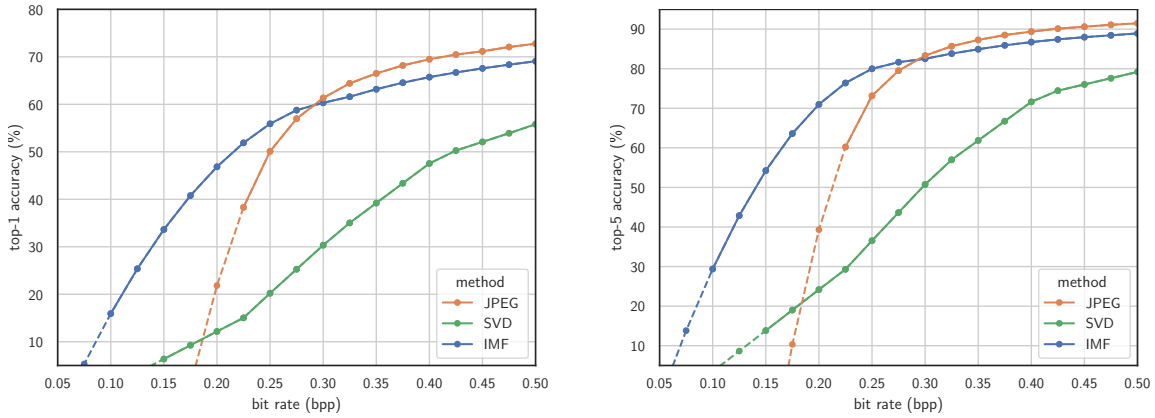
4.5. ImageNet Classification Performance

It is relevant to assess the ability of different compression methods in preserving the visual semantic information in images. To this end, we investigate the performance of an image classifier on images compressed using various compression methods. This is particularly crucial in scenarios where the ultimate goal is a vision task such as image classification, rather than maintaining perceived image quality, and we compress images before classification to minimize resource requirements, such as memory and communication bandwidth.

In this experiment, we employed a ResNet-50 classifier [33], pre-trained on the original ImageNet [34] dataset, to classify compressed images from the ImageNet validation set using different compression methods. The classification performance comparison is presented in Figure 5. Notably, the results indicate that IMF compression achieves over a 5% improvement in top-1 accuracy compared to JPEG at bit rates under 0.25 bpp and reaches a top-5 accuracy exceeding 70% at a bit rate of 0.2 bpp. IMF compression leads to higher classification accuracies than JPEG at bit rates up to approximately 0.30 bpp.

4.6. Ablation Studies

We conducted ablation studies to investigate the impact of factor bounds, the number of BCD iterations, and patch size on the compression performance of our IMF method. All experiments in this section were performed using the Kodak dataset. We followed the IMF configuration



(a) (b)

Figure 5: Impact of different compression methods on ImageNet classification accuracy. A ResNet-50 classifier pre-trained on the original ImageNet images is evaluated using validation images compressed by different methods. Panels (a) and (b) show top-1 and top-5 accuracy plotted as a function of bit rate, respectively. Dashed lines indicate extrapolated values predicted using LOESS [32] for extremely low bit rates that are otherwise unattainable.

described in Section 4.1 and varied only the parameters under ablation one at a time.

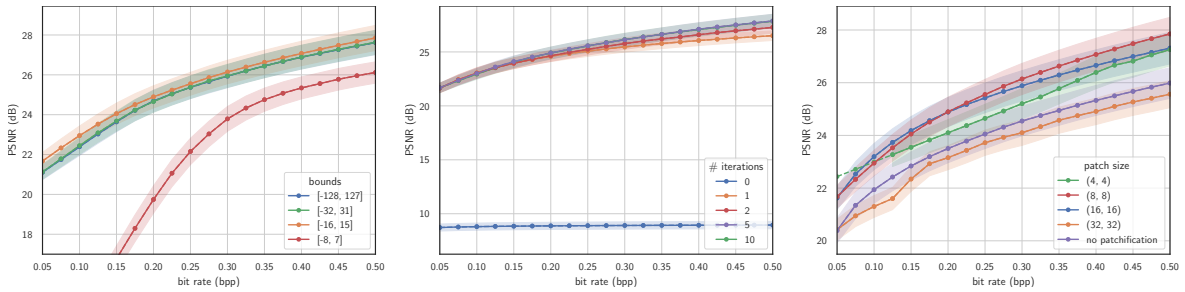
Factor Bounds. Figure 6a shows the average PSNR as a function of bit rate for IMF using various factor bounds $[\alpha, \beta]$ in Algorithm 1. The results indicate that the interval $[-16, 15]$ yields the optimal performance, showing moderate improvement over both $[-32, 31]$ and $[-128, 127]$, while significantly outperforming $[-8, 7]$. In fact, constraining the factor elements within a sufficiently narrow range can reduce the bit allocation needed, thereby leading to higher compression ratios. Note that in all these cases, the factor elements are represented as the `int8` data type.

BCD Iterations. The next parameter studied is the number of BCD iterations K in Algorithm 1, where each BCD iteration involves one complete cycle of updates across all the columns of both factors. Figure 6a shows the average PSNR plotted against the bit rate for IMF with different numbers of iterations $K \in \{0, 1, 2, 5, 10\}$. As expected, more iterations consistently resulted in higher PSNR for IMF compression. Without any BCD iterations ($K = 0$) and relying solely on the SVD-based initialization given by (8) and (9), the results became very poor. However, performance improved significantly after a few iterations, with more than $K = 5$ iterations yielding only marginal gains. We found that $K = 10$ iterations are sufficient in practice for image compression applications. This makes IMF computationally efficient, as decent compression performance can be achieved even with a limited number of BCD iterations.

Patchification. Figure 6c explores the impact of different patch sizes on IMF performance in terms of PSNR. As observed, a patch size of $(8, 8)$ yields the best performance. A patch size of $(16, 16)$ follows closely, with only a slight reduction in PSNR at higher bit rates. with only marginally lower PSNR at higher bit rates. Conversely, larger patch sizes like $(32, 32)$ or omitting the patchification step altogether significantly degrade compression performance.

5. Discussion

All our comparative results (Figure 3 and Figure 5) consistently show that our IMF method outperforms JPEG in both maintaining image quality and preserving visual semantics at low bit rates and remains comparable at higher bit rates. Moreover, IMF consistently demonstrates superior performance compared to SVD across all bit rates. This superiority can be attributed



(a) (b) (c)

Figure 6: Ablation studies for IMF. The average PSNR on the Kodak dataset is plotted as a function of bit rate under various experimental conditions: (a) varying the bounds $[\alpha, \beta]$ for the elements of the factor matrices, (b) changing the number of BCD iterations, and (c) adjusting the patch size.

to the quantization-free nature of IMF, which allows for more accurate reconstruction. In contrast, the high sensitivity of SVD to quantization errors during encoding and decoding degrades reconstruction quality.

As observed in Figure 6a, contracting the IMF factor bounds from $[-128, 127]$ to $[-16, 15]$ consistently improves the rate-distortion performance. Generally, narrowing the factor bounds $[\alpha, \beta]$ can potentially lower the entropy, thereby improving the effectiveness of lossless compression in the final stage of our framework and subsequently reducing the bit rate. However, this reduction in entropy comes at the cost of increased reconstruction error, as the feasible set in (3) becomes more constrained. This trade-off between entropy and reconstruction quality limits the compression performance of IMF. Therefore, it would be beneficial to moderately expand the factor bounds $[\alpha, \beta]$ while simultaneously controlling the entropy of the elements in the factor matrices. We plan to address this in the future by incorporating an entropy-aware regularization term into the current IMF objective function.

Patchification with an appropriate patch size (e.g., $(8, 8)$) helps capture local spatial dependencies and, as confirmed by our results in Figure 6c, positively impacts the performance of IMF and SVD. However, discontinuities at patch boundaries can introduce *blocking artifacts*, similar to JPEG compression at very low bit rates (see the building image example in Figure 4). Moreover, while JPEG suffers more from *color distortion* (e.g., *color bleeding* and *color banding*) at low bit rates, IMF and SVD are more affected by *blurriness*, as observed in the seascape image example in Figure 4. As a potential solution for future work, a deep neural network could be trained to remove these artifacts and then integrated as a post-processing module to further enhance the quality of IMF-compressed images.

6. Conclusion

This work presents a novel quantization-free lossy image compression method based on integer matrix factorization (IMF). By representing image data as a product of two smaller matrices with bounded integer elements, the proposed IMF approach effectively eliminates the need for quantization, a significant source of error in traditional compression methods like JPEG and SVD. The reshaped factor matrices are compatible with existing lossless compression standards, enhancing the overall flexibility and efficiency of our method. Our proposed iterative algorithm, utilizing a block coordinate descent scheme, has proven to be both efficient and convergent. Experimental results demonstrate that the IMF method significantly outperforms JPEG in terms of PSNR and SIMM at low bit rates and maintains better visual semantic information. This advantage underscores the potential of IMF to set a new standard in lossy image compression,

bridging the gap between factorization and quantization.

A. Proof of Theorem 3.1

We start by proving the closed-form solution (6), noting that the proof for (7) follows the same reasoning. The objective function in the subproblem (4) can be reformulated as follows:

$$\underset{\boldsymbol{u}_r \in \mathbb{Z}^M_{[\alpha,\beta]}}{\arg\min} \|\boldsymbol{E}_r - \boldsymbol{u}_r \boldsymbol{v}_r^{\mathsf{T}}\|_{\mathrm{F}} = \underset{\boldsymbol{u}_r \in \mathbb{Z}^M_{[\alpha,\beta]}}{\arg\min} \sum_{i=1}^{M}\sum_{j=1}^{N} (e_{ij}^r - u_i^r v_j^r)^2, \tag{10}$$

where e_{ij}^r denotes the element of matrix $\boldsymbol{E}_r$ in the ith row and jth column, and u_i^r and v_j^r are the ith and jth elements of vectors $\boldsymbol{u}_r$ and $\boldsymbol{v}_r$, respectively. Since the elements of $\boldsymbol{E}_r$ and $\boldsymbol{v}_r$ are fixed in problem (4), the optimization (10) can be decoupled into M optimizations as follows:

$$\begin{aligned} &\underset{u_i^r \in \mathbb{Z}_{[\alpha,\beta]}}{\arg\min}\ q_i(u_i^r), \quad \forall i \in \{1,\dots,M\}, \\ &\text{where}\ \ q_i(u_i^r) \coloneqq \sum_{j=1}^{N} (e_{ij}^r - u_i^r v_j^r)^2. \end{aligned} \tag{11}$$

The objective functions $q_i(u_i^r)$ in (11) are single-variable quadratic problems. Hence, the global optimum in each decoupled optimization problem can be achieved by finding the minimum of each quadratic problem and then projecting it onto the set $\mathbb{Z}_{[\alpha,\beta]}$. The minimum of each quadratic function in (11), denoted by $\bar{u}_i^r$, can be simply found by

$$\nabla_{u_i^r} q_i(u_i^r) = 0 \implies \bar{u}_i^r = \textstyle\sum_{j=1}^{N} e_{ij}^r v_j^r / \sum_{j=1}^{N} {v_j^r}^2, \tag{12}$$

where ∇_x is the partial derivative with respect to x. Since q_i has a constant curvature (second derivative) and $q_i(\bar{u}_i^r + d)$ is nondecreasing with increasing $|d|$, the value in the set $\mathbb{Z}_{[\alpha,\beta]}$ which is closest to $\bar{u}_i^r$ is the global minimizer of (11). This value can be reached by projecting $\bar{u}_i^r$ onto the set $\mathbb{Z}_{[\alpha,\beta]}$, namely $u_i^{r\star} = \mathrm{clamp}_{[\alpha,\beta]}(\mathrm{round}(\bar{u}_i^r))$, which is presented for all $i \in \{1, ..., M\}$ in a compact form in (6).

Since $\boldsymbol{u}_r^\star \coloneqq (u_1^{r\star}, \dots, u_M^{r\star})$ is the global optimum of optimization (10), it is evident that

$$\|\boldsymbol{E}_r - \boldsymbol{u}_r^\star \boldsymbol{v}_r^{\mathsf{T}}\|_{\mathrm{F}} \le \|\boldsymbol{E}_r - \boldsymbol{u}_r \boldsymbol{v}_r^{\mathsf{T}}\|_{\mathrm{F}}. \tag{13}$$

This inequality guarantees a nonincreasing cost function over one update of $\boldsymbol{u}_r$. Following the same reasoning for updates of $\boldsymbol{v}_r$ in (7), it can be concluded that in each update of (6) and (7), the cost function is nonincreasing. Therefore, the sequential updates over the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ in Algorithm 1 result in a monotonically nonincreasing cost function in (3).

B. Proof of Theorem 3.2

To study the convergence of the proposed Algorithm 1, we recast the optimization problem (3) into the following equivalent problem:

$$\underset{U_{:r} \in \mathbb{R}^M, V_{:r} \in \mathbb{R}^N, \forall r \in \{1,\dots,R\}}{\text{minimize}} \quad \Psi(\boldsymbol{U}, \boldsymbol{V}) \tag{14}$$

where

$$\begin{aligned} \Psi(\boldsymbol{U}, \boldsymbol{V}) &\coloneqq f_0(\boldsymbol{U}, \boldsymbol{V}) + \sum_{r=1}^{R} f(U_{:r}) + \sum_{r=1}^{R} g(V_{:r}), \\ f_0(\boldsymbol{U}, \boldsymbol{V}) &\coloneqq \|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}\|_{\mathrm{F}}^2, \\ f(U_{:r}) &\coloneqq \delta_{[a,b]}(U_{:r}) + \delta_{\mathbb{Z}}(U_{:r}), \\ g(V_{:r}) &\coloneqq \delta_{[a,b]}(V_{:r}) + \delta_{\mathbb{Z}}(V_{:r}), \end{aligned}$$

with $\delta_{\mathcal{B}}(\cdot)$ as the indicator function of the nonempty set $\mathcal{B}$ where $\delta_{\mathcal{B}}(\boldsymbol{x}) = 0$ if $\boldsymbol{x} \in \mathcal{B}$ and $\delta_{\mathcal{B}}(\boldsymbol{x}) = +\infty$, otherwise. By the definition of functions above, it is easy to confirm that the problems (3) and (14) are equivalent.

The unconstrained optimization problem (14) consists of the sum of a differentiable (smooth) function f_0 and nonsmooth functions f and g. This problem has been extensively studied in the literature under the class of nonconvex nonsmooth minimization problems. In Algorithm 1, the blocks $U_{:r}$ and $V_{:r}$ are updated sequentially following block coordinate descent (BCD) minimization algorithms, also often called Gauss-Seidel updates or alternating optimization [35, 36]. Hence, in this convergence study, we are interested in algorithms that allow BCD-like updates for the nonconvex nonsmooth problem of (14) [37, 38]. Specifically, we focus on the proximal alternating linearized minimization (PALM) algorithm [38], to relate its convergence behavior to that of Algorithm 1. To that end, we show that the updates of Algorithm 1 are related to the updates of PALM on the recast problem of (14), and all the assumptions necessary for the convergence of PALM are satisfied by our problem setting. It is noted that, for the sake of presentation and without loss of generality, in this proof, we assume each of the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ has only one column ($R = 1$); hence, we only have two blocks in the BCD updates. The iterates in PALM and the presented proof can be trivially extended for more than two blocks.

The PALM algorithm can be summarized as follows:

1. Initialize $\boldsymbol{U}^{\text{init}} \in \mathbb{R}^{M\times R}$, $\boldsymbol{V}^{\text{init}} \in \mathbb{R}^{N\times R}$
2. For each iteration $k = 0, 1, ...$

$$\begin{aligned} &(a)\ \boldsymbol{U}^{k+1} \in \operatorname{prox}^{f}_{c_k}\left(\boldsymbol{U}^k - \frac{1}{c_k}\nabla_{\boldsymbol{U}} f_0(\boldsymbol{U}^k, \boldsymbol{V}^k)\right), \\ &(b)\ \boldsymbol{V}^{k+1} \in \operatorname{prox}^{g}_{d_k}\left(\boldsymbol{V}^k - \frac{1}{d_k}\nabla_{\boldsymbol{V}} f_0(\boldsymbol{U}^{k+1}, \boldsymbol{V}^k)\right), \end{aligned} \tag{15}$$

where the proximal map for an extended proper lower semicontinuous (nonsmooth) function $\varphi : \mathbb{R}^n \to (-\infty, +\infty]$ and $\gamma > 0$ is defined as $\operatorname{prox}^{\varphi}_{\gamma}(\boldsymbol{x}) \coloneqq \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} \left\{\varphi(\boldsymbol{w}) + \frac{\gamma}{2}\|\boldsymbol{w} - \boldsymbol{x}\|_2^2\right\}$. In (15), $c_k > L_1(\boldsymbol{V}^k)$ and $d_k > L_2(\boldsymbol{U}^{k+1})$ where $L_1 > 0$, $L_2 > 0$ are local Lipschitz moduli, defined in the following proposition.

The following proposition investigates the necessary assumptions (cf. [38, Asm. 1 and Asm. 2]) for convergence of iterates in (15).

Proposition B.1 (Meeting required assumptions)**.** *The assumptions necessary for the convergence of iterates in* (15) *are satisfied by the functions involved in the problem* (14)*, specifically:*

1. *The indicator functions $\delta_{[a,b]}$ and $\delta_{\mathbb{Z}}$ are proper and lower semicontinuous functions, so do the functions f and g;*
2. *For any fixed $\boldsymbol{V}$, the partial gradient $\nabla_{\boldsymbol{U}} f_0(\boldsymbol{U}, \boldsymbol{V})$ is globally Lipschitz continuous with modulus $L_1(\boldsymbol{V}) = \|\boldsymbol{V}^T\boldsymbol{V}\|_{\text{F}}$. Therefore, for all $\boldsymbol{U}_1, \boldsymbol{U}_2 \in \mathbb{R}^{M\times R}$ the following holds*

$$\|\nabla_{\boldsymbol{U}} f_0(\boldsymbol{U}_1, \boldsymbol{V}) - \nabla_{\boldsymbol{U}} f_0(\boldsymbol{U}_2, \boldsymbol{V})\| \le L_1(\boldsymbol{V})\|\boldsymbol{U}_1 - \boldsymbol{U}_2\|,$$

   where $\|\cdot\|$ denotes the ℓ_2-norm of the vectorized input with the proper dimension (here, with the input in $\mathbb{R}^{MR\times 1}$). The similar Lipschitz continuity is evident for $\nabla_{\boldsymbol{V}} f_0(\boldsymbol{U}, \boldsymbol{V})$ as well with modulus $L_2(\boldsymbol{U}) = \|\boldsymbol{U}\boldsymbol{U}^T\|_{\text{F}}$.
3. *The sequences $\boldsymbol{U}^k$ and $\boldsymbol{V}^k$ are bounded due to the indicator functions $\delta_{[a,b]}$ with bounded a and b. Hence the moduli $L_1(\boldsymbol{V}^k)$ and $L_2(\boldsymbol{U}^k)$ are bounded from below and from above for all $k \in \mathbb{N}$.*

4. *The function f_0 is twice differentiable, hence, its full gradient $\nabla f_0(\boldsymbol{U}, \boldsymbol{V})$ is Lipschitz continuous on the bounded set $\boldsymbol{U} \in [a, b]^{M\times R}$, $\boldsymbol{V} \in [a, b]^{N\times R}$. Namely, with $M > 0$:*

$$\begin{aligned}\|\big(&\nabla_{\boldsymbol{U}} f_0(\boldsymbol{U}_1, \boldsymbol{V}_1) - \nabla_{\boldsymbol{U}} f_0(\boldsymbol{U}_2, \boldsymbol{V}_2),\\ &\nabla_{\boldsymbol{V}} f_0(\boldsymbol{U}_1, \boldsymbol{V}_1) - \nabla_{\boldsymbol{V}} f_0(\boldsymbol{U}_2, \boldsymbol{V}_2)\big)\|\\ &\leq M\|(\boldsymbol{U}_1 - \boldsymbol{U}_2, \boldsymbol{V}_1 - \boldsymbol{V}_2)\|,\end{aligned}$$

where $(\cdot, \cdot)$ denotes the concatenation of the two arguments.

5. *The sets $[a, b]$ and integer numbers are semi-algebraic; so are their indicator functions. The function f_0 is also polynomial, hence it is semi-algebraic. The sum of these functions results in a semi-algebraic function Ψ in (14), hence Ψ is a Kurdyka-Łojasiewicz (KL) function.*

By Proposition B.1, the optimization problem (14) can be solved by the iterates in (15), due to the following proposition:

Proposition B.2 (Global convergence [38])**.** *With the assumptions in proposition B.1 being met by the problem (14), let $((\boldsymbol{U}^k, \boldsymbol{V}^k))_{k\in\mathbb{N}}$ be a sequence generated by the iterates in (15). Then the sequence converges to a critical point $(\boldsymbol{U}^\star, \boldsymbol{V}^\star)$ of the problem (14), where $0 \in \partial\Psi(\boldsymbol{U}^\star, \boldsymbol{V}^\star)$, with ∂ as the subdifferential of Ψ.*

It is noted that the so-called *forward* steps $\boldsymbol{U}^k - \frac{1}{c_k}\nabla_{\boldsymbol{U}} f_0(\boldsymbol{U}^k, \boldsymbol{V}^k)$ and $\boldsymbol{V}^k - \frac{1}{d_k}\nabla_{\boldsymbol{V}} f_0(\boldsymbol{U}^{k+1}, \boldsymbol{V}^k)$ in the prox operators in (15) are replaced by the simple closed-form solutions $\boldsymbol{E}_r\boldsymbol{v}_r/\|\boldsymbol{v}_r\|^2$ and $\boldsymbol{E}_r^\mathsf{T}\boldsymbol{u}_r/\|\boldsymbol{u}_r\|^2$ in Algorithm 1 at steps 7 and 13 (cf. (6) and (7)), respectively. In the case where the iterates (15) are extended to multi-block updates, each block represents one column. This is thanks to the special form of the functions $f_0(\cdot, \boldsymbol{V}^k)$ and $f_0(\boldsymbol{U}^{k+1}, \cdot)$ being quadratic functions, each having a global optimal point, which ensures a descent in each forward step. Furthermore, the proximal operators $\mathrm{prox}^f_{c_k}$ and $\mathrm{prox}^g_{d_k}$ can efficiently be implemented by the operators round and $\mathrm{clamp}_{[\alpha,\beta]}$ in (6) and (7) (and equivalently in Algorithm 1 at steps 8 and 14). The equivalence of these steps is proven in the following lemma.

Lemma B.1 (prox implementation)**.** *Consider the operators* round *and* $\mathrm{clamp}_{[\alpha,\beta]}$ *defined in (6) and (7). Then* $\mathrm{prox}^f_{c_k}(\boldsymbol{W}) = \mathrm{round}(\mathrm{clamp}_{[\alpha,\beta]}(\boldsymbol{W}))$ *and* $\mathrm{prox}^g_{d_k}(\boldsymbol{Z}) = \mathrm{round}(\mathrm{clamp}_{[\alpha,\beta]}(\boldsymbol{Z}))$ *for any $\boldsymbol{W} \in \mathbb{R}^{M\times R}$, $\boldsymbol{Z} \in \mathbb{R}^{N\times R}$, and* $\mathrm{round}(\mathrm{clamp}_{[\alpha,\beta]}(\cdot))$ *being an elementwise operator on the input matrices.*

Proof. Define the following norms for a given matrix $\boldsymbol{W} \in \mathbb{R}^{M\times R}$:

$$\|\boldsymbol{W}\|^2_{[a,b]} \coloneqq \sum_{i,j|a\leq\boldsymbol{W}_{ij}\leq b} \boldsymbol{W}_{ij}^2, \quad \|\boldsymbol{W}\|^2_a \coloneqq \sum_{i,j|\boldsymbol{W}_{ij}<a} \boldsymbol{W}_{ij}^2,$$
$$\|\boldsymbol{W}\|^2_b \coloneqq \sum_{i,j|\boldsymbol{W}_{ij}>b} \boldsymbol{W}_{ij}^2.$$

Moreover, note that the round operator can be equivalently driven by the following proximal operator:

$$\mathrm{round}(\boldsymbol{W}) = \underset{\boldsymbol{U}\in\mathbb{Z}^{M\times R}}{\arg\min}\{\|\boldsymbol{U} - \boldsymbol{W}\|_F^2\}. \tag{16}$$

The proximal operator $\text{prox}_{c_k}^{f}(\boldsymbol{W})$ can be rewritten as

$$\begin{aligned}
\text{prox}_{c_k}^{f}(\boldsymbol{W}) &= \underset{\boldsymbol{U}\in\mathbb{R}^{M\times R}}{\arg\min}\{\delta_{[a,b]}(\boldsymbol{U}) + \delta_{\mathbb{Z}}(\boldsymbol{U}) + \frac{c_k}{2}\|\boldsymbol{U}-\boldsymbol{W}\|_F^2\} \\
&= \underset{\boldsymbol{U}\in\mathbb{Z}_{[a,b]}^{M\times R}}{\arg\min}\{\|\boldsymbol{U}-\boldsymbol{W}\|_F^2\} \\
&= \underset{\boldsymbol{U}\in\mathbb{Z}_{[a,b]}^{M\times R}}{\arg\min}\{\|\boldsymbol{U}-\boldsymbol{W}\|_{[a,b]}^2 + \|\boldsymbol{U}-\boldsymbol{A}\|_a^2 + \|\boldsymbol{U}-\boldsymbol{B}\|_b^2\} \\
&= \underset{\boldsymbol{U}\in\mathbb{Z}^{M\times R}}{\arg\min}\{\|\boldsymbol{U}-\boldsymbol{W}\|_{[a,b]}^2 + \|\boldsymbol{U}-\boldsymbol{A}\|_a^2 + \|\boldsymbol{U}-\boldsymbol{B}\|_b^2\} \\
&= \underset{\boldsymbol{U}\in\mathbb{Z}^{M\times R}}{\arg\min}\{\|\boldsymbol{U}-\text{clamp}_{[\alpha,\beta]}(\boldsymbol{W})\|_F^2\} \\
&= \text{round}(\text{clamp}_{[\alpha,\beta]}(\boldsymbol{W})).
\end{aligned}$$

The first equality is due to the definition of prox which is equivalent to the second equality. In the third equality the matrices $\boldsymbol{A}\in\mathbb{R}^{M\times R}$ and $\boldsymbol{B}\in\mathbb{R}^{M\times R}$ have elements all equal to a and b, respectively. The third equality is due to the fact that replacing $\|\boldsymbol{U}-\boldsymbol{W}\|_a^2 + \|\boldsymbol{U}-\boldsymbol{W}\|_b^2$ with $\|\boldsymbol{U}-\boldsymbol{A}\|_a^2 + \|\boldsymbol{U}-\boldsymbol{B}\|_b^2$ has no effect on the solution of the minimization. The fourth equality is also trivial due to the involved norms in the third equality. The fifth equality can be easily confirmed by the definition of $\text{clamp}_{[\alpha,\beta]}$. Finally, in the last equality, (16) is invoked. It is noted that in the implementation, $\text{round}(\text{clamp}_{[\alpha,\beta]}(\cdot)) = \text{clamp}_{[\alpha,\beta]}(\text{round}(\cdot))$ due to the integrality of the bounds $\alpha,\beta\in\mathbb{Z}$. A similar proof can be trivially followed for $\text{prox}_{d_k}^{g}(\boldsymbol{Z}) = \text{round}(\text{clamp}_{[\alpha,\beta]}(\boldsymbol{Z}))$ as well. □

Now that the equivalence of iterates (15) with the simple and closed-form steps in Algorithm 1 is fully established, and the assumptions required for the convergence are verified in proposition B.1 to be met by problems (14) and (3), proposition B.2 can be trivially invoked to establish the convergence of Algorithm 1 to a locally optimal point of problem (3).

References

[1] Gregory K Wallace. "The JPEG still picture compression standard". In: *Communications of the ACM* 34.4 (1991), pp. 30–44.

[2] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. "The JPEG 2000 still image compression standard". In: *IEEE Signal processing magazine* 18.5 (2001), pp. 36–58.

[3] Vivek K Goyal. "Theoretical foundations of transform coding". In: *IEEE Signal Processing Magazine* 18.5 (2001), pp. 9–21.

[4] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. "Discrete cosine transform". In: *IEEE transactions on Computers* 100.1 (1974), pp. 90–93.

[5] Marc Antonini et al. "Image coding using wavelet transform." In: *IEEE Trans. Image Processing* 1 (1992), pp. 20–5.

[6] H Andrews and CLIII Patterson. "Singular value decomposition (SVD) image coding". In: *IEEE transactions on Communications* 24.4 (1976), pp. 425–432.

[7] HS Prasantha, HL Shashidhara, and KN Balasubramanya Murthy. "Image compression using SVD". In: *International conference on computational intelligence and multimedia applications (ICCIMA 2007)*. Vol. 3. IEEE. 2007, pp. 143–145.

[8] Junhui Hou et al. "Sparse low-rank matrix approximation for data compression". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.5 (2015), pp. 1043–1054.

[9] Jerome M Shapiro. "Embedded image coding using zerotrees of wavelet coefficients". In: *IEEE Transactions on signal processing* 41.12 (1993), pp. 3445–3462.

[10] Google Developers. *WebP Compression Techniques*. https://developers.google.com/speed/webp. Accessed: 2024-05-17. 2011.

[11] Jani Lainema et al. "HEVC still image coding and high efficiency image file format". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 71–75.

[12] Miska M Hannuksela, Jani Lainema, and Vinod K Malamal Vadakital. "The high efficiency image file format standard [standards in a nutshell]". In: *IEEE Signal Processing Magazine* 32.4 (2015), pp. 150–156.

[13] Johannes Ballé et al. "Variational image compression with a scale hyperprior". In: *arXiv preprint arXiv:1802.01436* (2018).

[14] Zhengxue Cheng et al. "Learned image compression with discretized gaussian mixture likelihoods and attention modules". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 7939–7948.

[15] Jinming Liu, Heming Sun, and Jiro Katto. "Learned image compression with mixed transformer-CNN architectures". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 14388–14397.

[16] Ruihan Yang and Stephan Mandt. "Lossy image compression with conditional diffusion models". In: *Advances in Neural Information Processing Systems* 36 (2024).

[17] Xin Yuan and Raziel Haimi-Cohen. "Image compression based on compressive sensing: End-to-end comparison with JPEG". In: *IEEE Transactions on Multimedia* 22.11 (2020), pp. 2889–2904.

[18] Matthew M Lin, Bo Dong, and Moody T Chu. "Integer matrix factorization and its application". In: ().

[19] Bo Dong, Matthew M Lin, and Haesun Park. "Integer matrix approximation and data mining". In: *Journal of scientific computing* 75 (2018), pp. 198–224.

[20] Pauli Miettinen et al. "The discrete basis problem". In: *IEEE transactions on knowledge and data engineering* 20.10 (2008), pp. 1348–1362.

[21] Siamak Ravanbakhsh, Barnabás Póczos, and Russell Greiner. "Boolean matrix factorization and noisy completion via message passing". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 945–954.

[22] Carl Eckart and Gale Young. "The approximation of one matrix by another of lower rank". In: *Psychometrika* 1.3 (1936), pp. 211–218.

[23] Daniel Lee and H. Sebastian Seung. "Algorithms for Non-negative Matrix Factorization". In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000. URL: https://proceedings.neurips.cc/paper_files/paper/2000/file/f9d1152547c0bde01830b7e8bd60024c-Paper.pdf.

[24] Chris HQ Ding, Tao Li, and Michael I Jordan. "Convex and semi-nonnegative matrix factorizations". In: *IEEE transactions on pattern analysis and machine intelligence* 32.1 (2008), pp. 45–55.

[25] Peter Deutsch and Jean-Loup Gailly. *Zlib compressed data format specification version 3.3*. Tech. rep. 1996.

[26] Nicholas Gillis. *Nonnegative Matrix Factorization*. SIAM, 2020.

[27] Peter van Emde Boas. "Another NP-complete problem and the complexity of computing short vectors in a lattice". In: *Tecnical Report, Department of Mathmatics, University of Amsterdam* (1981).

[28] Alex Clark. *Pillow (PIL Fork) Documentation*. 2015. URL: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

[29] Eastman Kodak. *Kodak lossless true color image suite*. 1993. URL: https://r0k.us/graphics/kodak/.

[30] Organizers. *Challenge on Learned Image Compression*. 2024. URL: https://www.compression.cc/.

[31] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115 (2015), pp. 211–252.

[32] William S Cleveland and Susan J Devlin. "Locally weighted regression: an approach to regression analysis by local fitting". In: *Journal of the American statistical association* 83.403 (1988), pp. 596–610.

[33] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[34] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[35] Yu Nesterov. "Efficiency of coordinate descent methods on huge-scale optimization problems". In: *SIAM Journal on Optimization* 22.2 (2012), pp. 341–362.

[36] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods". In: *Mathematical Programming* 137.1 (2013), pp. 91–129.

[37] Amir Beck and Luba Tetruashvili. "On the convergence of block coordinate descent type methods". In: *SIAM journal on Optimization* 23.4 (2013), pp. 2037–2060.

[38] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. "Proximal alternating linearized minimization for nonconvex and nonsmooth problems". In: *Mathematical Programming* 146.1 (2014), pp. 459–494.