
Quantization-free Lossy Image Compression Using Integer Matrix Factorization

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 Image compression is essential for efficient transmission and storage. The most
2 widely used transform coders, such as the discrete cosine transform (DCT) and
3 singular value decomposition (SVD), represent image data in continuous domains,
4 necessitating carefully designed quantizers. Notably, the performance of SVD-
5 based methods is more sensitive to quantization errors than that of DCT-based
6 methods like JPEG. Therefore, we introduce a variant of integer matrix factoriza-
7 tion (IMF) to develop a novel quantization-free image compression method. IMF
8 provides a low-rank representation of the image data as a product of two smaller
9 factor matrices with bounded integer elements, thereby eliminating the need for
10 quantization. We propose an efficient, provably convergent iterative algorithm for
11 IMF using a block coordinate descent (BCD) scheme, with subproblems having
12 closed-form solutions. The experiments demonstrate that our IMF-based compres-
13 sion method outperforms JPEG, achieving improvements of over 3 dB in PSNR
14 at rates of under 0.17 bits per pixel (bpp) on the Kodak dataset. We also assessed
15 our method’s capability to preserve visual semantic information by evaluating an
16 ImageNet pre-trained classifier on compressed images. Remarkably, our method
17 resulted in an improvement of over 10% in top-1 accuracy compared to JPEG at
18 rates under 0.23 bpp.

19

1 Introduction

20 Lossy image compression involves reducing the storage size of digital images by discarding some
21 image data that are redundant or less perceptible to the human eye. This is crucial for efficiently
22 storing and transmitting images, particularly in applications where bandwidth or storage resources are
23 limited, such as web browsing, streaming, and mobile platforms. Lossy image compression methods
24 enable adjusting the degree of compression, providing a selectable tradeoff between storage size and
25 image quality. Widely used methods such as JPEG [33] and JPEG 2000 [31] follow the *transform*
26 *coding* paradigm [17]. They use orthogonal linear transformations, such as discrete cosine transform
27 (DCT) [1] and discrete wavelet transform (DWT) [3], to decorrelate small image blocks. Since these
28 transforms map image data into a continuous domain, quantization is necessary before coding into
29 bytes. Unfortunately, as quantization errors can significantly degrade compression performance, the
30 quantizers must be carefully crafted to minimize this impact, which further complicates codec design.
31 Another promising paradigm relies on low-rank approximation methods. Particularly singular value
32 decomposition (SVD) is a representative method known to be the deterministically optimal transform
33 for energy compaction [2]. In practice, current SVD-based methods [2, 28, 20] can represent image
34 data only with components containing floating-point elements, which necessitates a quantization
35 layer prior to any byte-level processing. Nevertheless, their compression performance is suboptimal

36 due to higher sensitivity to quantization errors compared to transform-based methods like JPEG,
37 especially at low bit rates.

38 Motivated by this, we introduce a new variant of integer matrix factorization (IMF), and based on
39 that, develop an effective *quantization-free* lossy image compression method. Our IMF formulation
40 provides a low-rank representation of the image data as a product of two smaller factor matrices with
41 *bounded integer* elements. Since we can directly store and losslessly process these integer matrices at
42 the byte level, quantization is no longer needed, making IMF arguably better suited than SVD for
43 image compression. Another advantage of IMF is that the reshaped factor matrices can be treated as
44 8-bit grayscale images, allowing any lossless image compression standard to be seamlessly integrated
45 into the proposed framework. We propose an efficient iterative algorithm for IMF using a block
46 coordinate descent (BCD) scheme, where each column of a factor matrix is taken as a block and
47 updated one at a time using a closed-form solution.

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

55 2 Related Work

56 **Transform coding.** Transform coding is a widely used approach in lossy image compression,
57 leveraging mathematical transforms to decorrelate pixel values and represent image data more
58 compactly. One of the earliest and most influential methods is the discrete cosine transform (DCT)
59 [1], used in JPEG [33], which converts image data into the frequency domain, prioritizing lower
60 frequencies to retain perceptually significant information. The discrete wavelet transform (DWT)
61 [3], used in JPEG 2000 [31], offers improved performance by capturing both frequency and location
62 information, leading to better handling of edges and textures [30]. More recently, the WebP [12] and
63 HEIF [22, 18] formats combine DCT and intra-frame prediction to achieve superior compression and
64 quality compared to JPEG.

65 **Low-rank techniques.** Low-rank approximation can provide a compact representation by decom-
66 posing image data into smaller components. Notably, truncated singular value decomposition (tSVD)
67 is a classical technique that decomposes images into singular values and vectors, retaining only
68 the most significant components to achieve compression [2, 28]. Hou et al. [20] proposed sparse
69 low-rank matrix approximation (SLRMA) for data compression, which is able to explore both the
70 intra- and inter-coherence of data samples simultaneously from the perspective of optimization and
71 transformation. More recently, Yuan and Haimi-Cohen [34] introduced a graph-based low-rank
72 regularization to reduce compression artifacts near block boundaries at low bit rates.

73 **Integer Matrix Factorization.** There are applications where meaningful representation of data
74 as discrete factor matrices is crucial. While typical low-rank techniques like SVD and nonnegative
75 matrix factorization (NMF) are inappropriate for such applications, integer matrix factorization
76 (IMF) ensures the integrality of factors to achieve this goal. Lin et al. [24] investigates IMF to
77 effectively handle discrete data matrices for cluster analysis and pattern discovery. Dong et al. [14]
78 introduce an alternative least squares method for IMF, verifying its effectiveness with some data
79 mining applications. A closely related topic is boolean matrix factorization (BMF), where factors
80 are constrained to binary matrices. BMF has been applied to some data mining [25] and machine
81 learning [29] problems. However, to the best of our knowledge, IMF has not yet been explored in
82 image compression. Therefore, this work investigates IMF for image compression and argues that it
83 can serve as a powerful tool for this purpose.

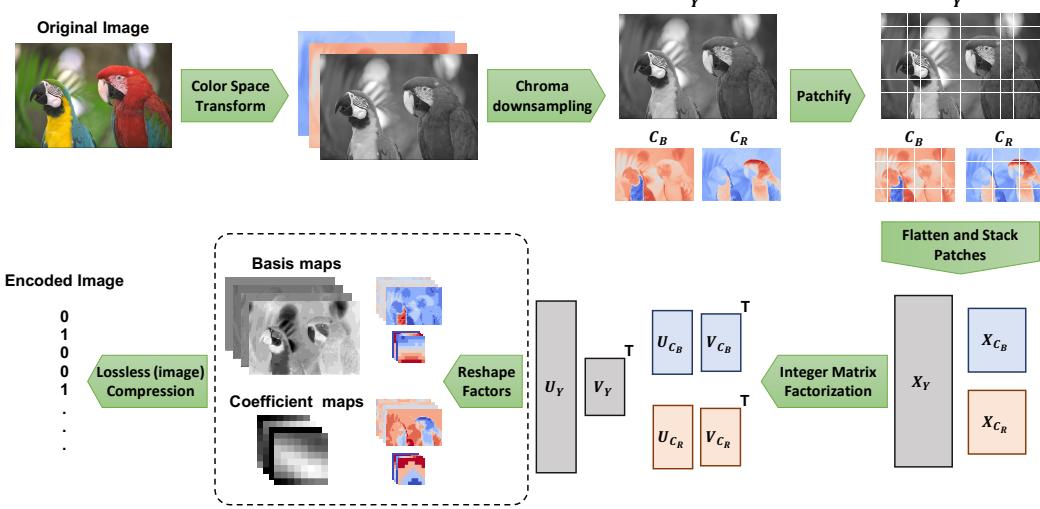


Figure 1 An illustration of the encoder for our image compression method.

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 $\mathcal{X} \in \{0, \dots, 255\}^{3 \times H \times W}$. Each step of encoding is described in the following.

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

Chroma Downsampling. After conversion to the YC_BC_R color space, the *chroma* components \mathbf{C}_B and \mathbf{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 $\mathbf{Y} \in [0, 255]^{H \times W}$ and the *chroma* components $\mathbf{C}_B, \mathbf{C}_R \in [0, 255]^{\frac{H}{2} \times \frac{W}{2}}$. 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 $\mathbf{X}_Y \in [0, 255]^{\frac{HW}{64} \times 64}$, $\mathbf{X}_{C_B} \in [0, 255]^{\frac{HW}{256} \times 64}$, and $\mathbf{X}_{C_R} \in [0, 255]^{\frac{HW}{256} \times 64}$. Later, 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 \mathbf{X}_Y , \mathbf{X}_{C_B} , and \mathbf{X}_{C_R} , which is the core of our compression method that provides a lossy compressed representation

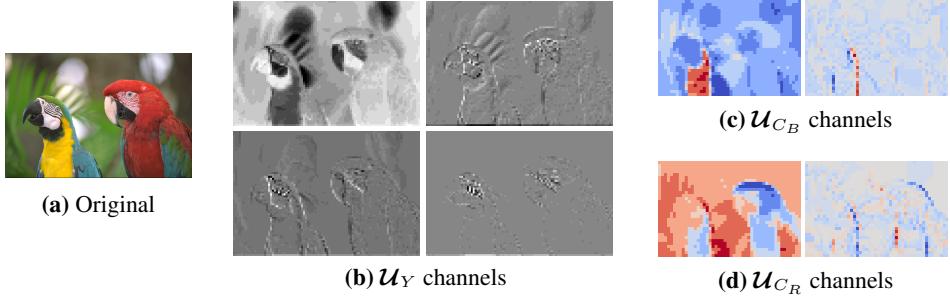


Figure 2 The channels of IMF basis maps for the kodim23 image from Kodak. (a) shows the original image. The IMF basis maps corresponding to luma (b), blue-difference (c), and red-difference chroma (d) are shown. The channels of basis map with higher energy maintain the overall texture of the original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

113 of these matrices. The low-rank approximation [15] aims to approximate a given matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$
114 by

$$\mathbf{X} \approx \mathbf{U}\mathbf{V}^\top = \sum_{r=1}^R \mathbf{U}_{:,r} \mathbf{V}_{:,r}^\top, \quad (1)$$

115 where $\mathbf{U} \in \mathbb{R}^{M \times R}$ and $\mathbf{V} \in \mathbb{R}^{N \times R}$ are *factor matrices* (or simply *factors*), $R \leq \min(M, N)$
116 represents the *rank*, $\mathbf{U}_{:,r}$ and $\mathbf{V}_{:,r}$ represent the r -th columns of \mathbf{U} and \mathbf{V} , respectively. We refer to
117 \mathbf{U} as the *basis matrix* and \mathbf{V} as the *coefficient matrix*. By selecting a sufficiently small value for
118 R , the factor matrices \mathbf{U} and \mathbf{V} , with a combined total of $(M + N)R$ elements, offer a compact
119 representation of the original matrix \mathbf{X} , which has MN elements, capturing the most significant
120 patterns in the image. Depending on the loss function used to measure the reconstruction error
121 between \mathbf{X} and the product $\mathbf{U}\mathbf{V}^\top$, as well as the constraints on the factor matrices \mathbf{U} and \mathbf{V} , various
122 formulations and variants have been proposed for different purposes [23, 13, 24]. In Section 3.3, we
123 introduce and elaborate on our variant, termed integer matrix factorization (IMF), and argue why it is
124 well-suited and effective for image compression.

125 **Lossless compression.** IMF yields factor matrices $\mathbf{U}_Y \in \{0, \dots, 255\}^{\frac{HW}{64} \times R}$ and $\mathbf{V}_Y \in$
126 $\{0, \dots, 255\}^{64 \times R}$; $\mathbf{U}_{C_B} \in \{0, \dots, 255\}^{\frac{HW}{256} \times R}$ and $\mathbf{V}_{C_B} \in \{0, \dots, 255\}^{64 \times R}$; and $\mathbf{U}_{C_R} \in$
127 $\{0, \dots, 255\}^{\frac{HW}{256} \times R}$ and $\mathbf{V}_{C_R} \in \{0, \dots, 255\}^{64 \times R}$ that correspond to \mathbf{X}_Y , \mathbf{X}_{C_B} , and \mathbf{X}_{C_R} . Since
128 these matrices have integer elements, they can be directly encoded by any standard lossless data
129 compression like zlib [11] without the need for a quantization step, which is commonly present in
130 other lossy image compression methods and adds extra complications.

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

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

133 As each channel of a *factor map* can be treated as an 8-bit grayscale image, we can encode it by
134 any standard lossless image compression method such as PNG. For images with a resolution of
135 $H, W \gg 64$, which are most common nowadays, the *basis maps* (\mathcal{U}) are significantly larger than
136 the *coefficient maps* (\mathcal{V}), accounting for the majority of the storage space. Interestingly, in practice,
137 the IMF *basis maps* turn out to be meaningful images, each capturing some visual semantic of the
138 image (see Figure 2 for an example). Therefore, our IMF approach can effectively leverage the power
139 of existing lossless image compression algorithms, offering a significant advantage over current
140 methods. However, in this work, we take the first approach and use the zlib library [11] to encode
141 factor matrices, creating a stand-alone codec that is independent from other image compression
142 methods.

143 **3.2 Decoding**

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

152 **3.3 Integer Matrix Factorization (IMF)**

153 The main building block of our method is integer matrix factorization (IMF), which is responsible
 154 for the lossy compression of matrices obtained through patchification. IMF can be framed as an
 155 optimization problem, aiming to minimize the reconstruction error between the original matrix
 156 $\mathbf{X} \in \mathbb{R}^{M \times N}$ and the product $\mathbf{U}\mathbf{V}^\top$, while ensuring that the elements of the factor matrices \mathbf{U} and
 157 \mathbf{V} are integers within a specified interval $[\alpha, \beta]$ with integer endpoints, i.e., $\alpha, \beta \in \mathbb{Z}$. Formally, the
 158 IMF problem can be expressed as:

$$\begin{aligned} & \underset{\mathbf{U}, \mathbf{V}}{\text{minimize}} \quad \|\mathbf{X} - \mathbf{U}\mathbf{V}^\top\|_F^2 \\ & \text{s.t. } \mathbf{U} \in \mathbb{Z}_{[\alpha, \beta]}^{M \times R}, \mathbf{V} \in \mathbb{Z}_{[\alpha, \beta]}^{N \times R}, \end{aligned} \quad (3)$$

159 where $\|\cdot\|_F$ denotes the Frobenius norm; $R \leq \min(M, N)$ represents the *rank*; and $\mathbb{Z}_{[\alpha, \beta]} \triangleq$
 160 $[\alpha, \beta] \cap \mathbb{Z}$ denotes the set of integers within $[\alpha, \beta]$. Without constraints on the factors, the problem
 161 would have an analytic solution through singular value decomposition (SVD), as addressed by
 162 the Eckart–Young–Mirsky theorem [15]. If only a nonnegativity constraint were applied (without
 163 integrality), variations of nonnegative matrix factorization (NMF) would emerge [23, 16]. The IMF
 164 problem (3) poses a challenging integer program, with finding its global minima known to be NP-hard
 165 [14, 32]. Only a few iterative algorithms [14, 24] have been proposed to find a “good solution” for
 166 some IMF variants in contexts other than image compression. In Section 3.4, we propose an efficient
 167 iterative algorithm for the IMF problem (3).

168 The application of SVD and NMF in image compression is problematic mainly because the resulting
 169 factors contain continuous values that must be represented as arrays of floating-point numbers. This
 170 necessitates a quantization step that not only adds extra complications but also significantly degrades
 171 compression performance due to quantization errors (as demonstrated in Section 4). In contrast, our
 172 IMF formulation produces integer factor matrices that can be directly stored and losslessly processed
 173 without incurring roundoff errors. The reason for limiting the feasible region to $[\alpha, \beta]$ in our IMF
 174 formulation is to enable more compact storage of the factors using standard integral data types, such
 175 as `int8` and `int16`, supported by programming languages. Given that the elements of the input
 176 matrix \mathbf{X} are in $[0, 255]$, we found the signed `int8` type, which represents integers from -128 to
 177 127, suitable for image compression applications. As a result, our IMF formulation is well-suited
 178 for image compression, effectively integrating the factorization and quantization steps into a single,
 179 efficient compression process.

180 **3.4 Block Coordinate Descent Scheme for IMF**

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

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

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

- 1 Initialize \mathbf{U}^{init} , \mathbf{V}^{init} using the truncated SVD method, provided by (8) and (9), and set $k = 0$
- 2 **while** stopping criterion not satisfied **do**
- 3 $k \leftarrow k + 1$
- 4 $\mathbf{A} \leftarrow \mathbf{X}\mathbf{V}^k$, $\mathbf{B} \leftarrow \mathbf{V}^{k^\top}\mathbf{V}^k$
- 5 **for** $r = 1, \dots, R$ **do**
- 6 $\mathbf{U}_{:,r}^{k+1} = \text{clamp}_{[\alpha, \beta]} \left(\text{round} \left(\frac{\mathbf{A}_{:,r} - \sum_{s=1}^{r-1} \mathbf{B}_{sr} \mathbf{U}_{:,s}^{k+1} - \sum_{s=r+1}^M \mathbf{B}_{sr} \mathbf{U}_{:,s}^k}{\|\mathbf{V}_{:,r}^k\|^2} \right) \right)$
- 7 **end**
- 8 $\mathbf{A} \leftarrow \mathbf{X}^\top \mathbf{U}^{k+1}$; $\mathbf{B} \leftarrow \mathbf{U}^{k+1^\top} \mathbf{U}^{k+1}$
- 9 **for** $r = 1, \dots, R$ **do**
- 10 $\mathbf{V}_{:,r}^{k+1} = \text{clamp}_{[\alpha, \beta]} \left(\text{round} \left(\frac{\mathbf{A}_{:,r} - \sum_{s=1}^{r-1} \mathbf{B}_{sr} \mathbf{V}_{:,s}^{k+1} - \sum_{s=r+1}^N \mathbf{B}_{sr} \mathbf{V}_{:,s}^k}{\|\mathbf{U}_{:,r}^{k+1}\|^2} \right) \right)$
- 11 **end**
- 12 **end**
- 13 **return** $(\mathbf{U}^k, \mathbf{V}^k)$

187 reached. Formally, this involves solving one of the following subproblems at a time:

$$\mathbf{u}_r \leftarrow \arg \min_{\mathbf{u}_r \in \mathbb{Z}_{[\alpha, \beta]}^{M \times 1}} \|\mathbf{E}_r - \mathbf{u}_r \mathbf{v}_r^\top\|_F^2, \quad (4)$$

$$\mathbf{v}_r \leftarrow \arg \min_{\mathbf{v}_r \in \mathbb{Z}_{[\alpha, \beta]}^{N \times 1}} \|\mathbf{E}_r - \mathbf{u}_r \mathbf{v}_r^\top\|_F^2, \quad (5)$$

188 where $\mathbf{u}_r \triangleq \mathbf{U}_{:,r}$ and $\mathbf{v}_r \triangleq \mathbf{V}_{:,r}$ represent the r -th columns of \mathbf{U} and \mathbf{V} , respectively. $\mathbf{E}_r \triangleq$
189 $\mathbf{X} - \sum_{s \neq r}^R \mathbf{u}_s \mathbf{v}_s^\top$ is the residual matrix. We define one iteration of BCD as a complete cycle
190 of updates across all the columns of both factors. In fact, the proposed algorithm is a $2R$ -block
191 coordinate descent procedure, where at each iteration, first the columns of \mathbf{U} and then the columns
192 of \mathbf{V} are updated (see Algorithm 1). Note that subproblem (5) can be transformed into the same
193 form as (4) by simply transposing its error term inside the Frobenius norm. Therefore, we only need
194 to find the best rank-1 approximation with integer elements constrained within a specific interval.
195 Fortunately, this problem has a closed-form solution, as addressed by Theorem 1 below.

196 **Theorem 1** (Monotonicity). *The global optima of subproblems (4) and (5) can be represented by
197 closed-form solutions as follows:*

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

$$\mathbf{v}_r \leftarrow \text{clamp}_{[\alpha, \beta]} \left(\text{round} \left(\frac{\mathbf{E}_r^\top \mathbf{u}_r}{\|\mathbf{u}_r\|^2} \right) \right), \quad (7)$$

198 where $\text{round}(\mathbf{Z})$ denotes an element-wise operator that rounds each element of \mathbf{Z} to the nearest
199 integer, and $\text{clamp}_{[\alpha, \beta]}(\mathbf{Z}) \triangleq \max(\alpha, \min(\mathbf{Z}, \beta))$ denotes an element-wise operator that clamps
200 each element of \mathbf{Z} to the interval $[\alpha, \beta]$.

201 *Proof.* See Appendix A.1 for the proof. □

202 It is noteworthy that the combination of $\text{round}(\cdot)$ and $\text{clamp}_{[\alpha, \beta]}(\cdot)$ in (6) and (7) can be interpreted
203 as the element-wise projector to $\mathbb{Z}_{[\alpha, \beta]}$. In Theorem 2, the convergence of the proposed algorithm
204 employing these closed-form solutions is established.

205 **Theorem 2** (Global convergence). *Let $(\mathbf{U}^k)_{k \in \mathbb{N}}$ and $(\mathbf{V}^k)_{k \in \mathbb{N}}$ be sequences generated by the pro-
206 posed Algorithm 1. Then both sequences are convergent to a locally optimal point of the optimization
207 problem (3).*

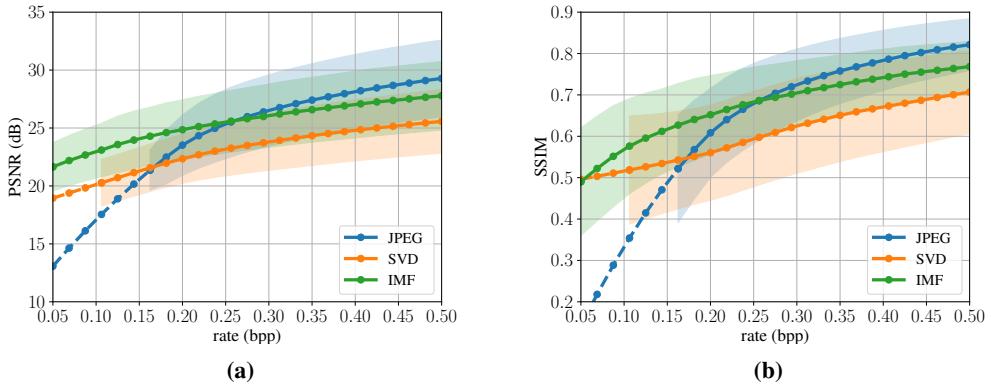


Figure 3 Rate-distortion performance on the Kodak dataset. The average PSNR and SSIM for each method is plotted as a function of bit rate (bpp). The shaded areas represent the standard deviation. The dashed line corresponds to extrapolation using quadratic splines.

208 *Proof.* See Appendix A.2 for the proof. \square

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

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

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

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

218 4 Experiments

219 In this section, the proposed IMF compression algorithm is assessed against baseline codecs following
220 the implementation details, described in Appendix B. Moreover, ablation studies are presented to
221 investigate the effect of different parameters in IMF.

222 4.1 Rate-Distortion Performance

223 Figure 3 shows the rate-distortion curves for IMF, SVD, and JPEG on the Kodak dataset. We also
224 performed experiments on the CLIC dataset, with the results provided in Appendix C. For each
225 method, PSNR and SSIM values are averaged over all images at different bit rates up to 0.5 bits per
226 pixel (bpp).

227 As seen in Figure 3, our IMF method consistently outperforms JPEG at low bit rates under 0.25 bpp
228 and remains competitive with JPEG at higher bit rates in terms of both PSNR and SSIM. It is also
229 evident that IMF significantly outperforms the SVD-based method at all the bit rates. This can be
230 attributed to the quantization errors that SVD is sensitive to during encoding and decoding, which
231 deteriorate its performance. In contrast, the quantization-free property of IMF effectively guarantees
232 higher performance at different bpp values.

233 4.2 Qualitative Performance

234 The qualitative results on the Kodak dataset are presented in Figure 4. We visualize two example
235 images compressed by each of the baseline methods at nearly the same bit rate. As visible in both

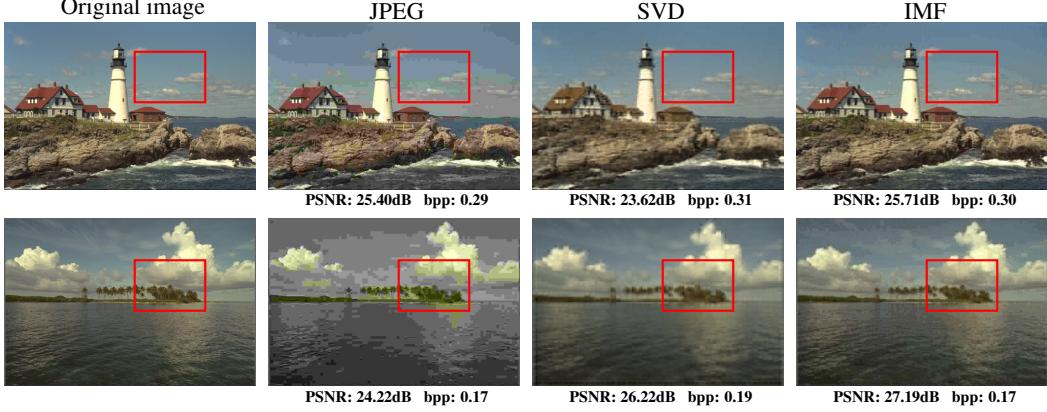


Figure 4 Qualitative performance comparison on an image from the Kodak dataset. The red bounding box highlights the artifacts produced by JPEG and the blurriness present in SVD-compressed images.

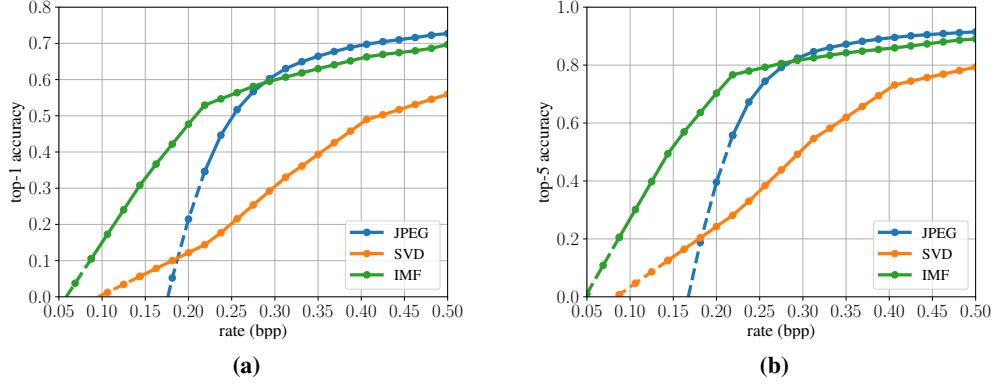


Figure 5 Impact of different compression methods on ImageNet classification accuracy. A ResNet-50 classifier pre-trained on the original ImageNet images is evaluated using validation images compressed by different methods. Panels (a) and (b) show top-1 and top-5 accuracy plotted against bpp, respectively.

examples, IMF is capable of maintaining higher-quality compression, while SVD yields more blurred images, and JPEG produces color artifacts, in the regions marked by red bounding boxes.

4.3 ImageNet Classification Performance

As another criterion, we investigate the performance of an image classifier on the images compressed by different compression algorithms. This criterion focuses on the capability of different compression methods to preserve the visual semantic information in each image. Furthermore, it highlights the importance of image compression where various vision tasks such as classification—rather than maintaining the perceived image quality—are the main objective, while keeping the requirement of resources such as memory, communication bandwidth, computation power, latency budget, etc. as limited as possible. ImageNet [10] validation set, containing 50000 images with a resolution of 224×224 in 1000 classes, is considered in this classification task done by a ResNet-50 classifier [19], pre-trained on the original ImageNet dataset. The classification performance comparison is made in Figure 5. The results suggest that IMF leads to more than a 10% improvement in top-1 accuracy over JPEG at low bit rates under 0.23 bpp and reaches a top-5 accuracy of nearly 80% at a bit rate of 0.25 bpp.

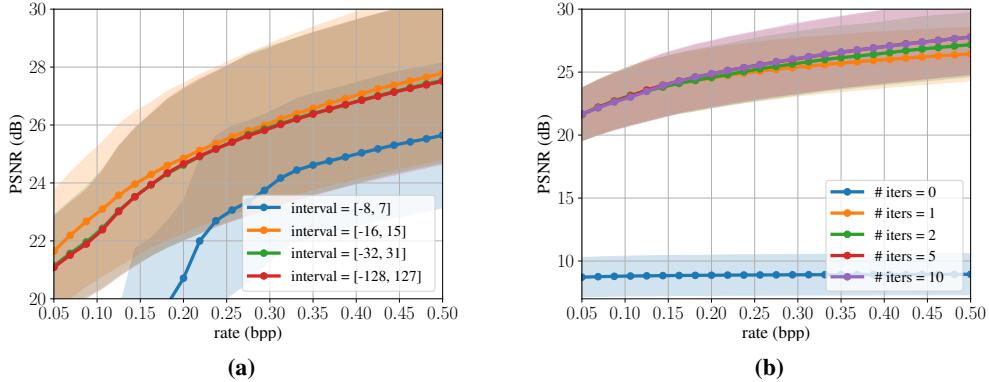


Figure 6 Ablation studies for IMF. Average PSNR on the Kodak dataset is plotted versus bit rate (bpp). (b) shows the PSNR-bpp curve for different intervals $[\alpha, \beta]$ for the elements of factor matrices. (c) shows the PSNR-bpp curve for different numbers of BCD iterations.

251 4.4 Ablation Studies

252 In this section, ablation studies are performed, focusing on factor bounds and BCD iteration numbers
 253 in the proposed IMF compression algorithm and their effect on its performance. The other ablation
 254 studies are postponed to the appendix.

255 **Factor bounds.** Figure 6a studies the compression performance of IMF with different factor bounds
 256 α and β in Algorithm 1. According to the results, the bounds $\alpha = -16$ and $\beta = 15$ lead to the best
 257 performance. Limiting factor elements within these bounds leads to dedicating fewer bits to represent
 258 the resulting narrower range, compared to the other bounds, and consequently higher compression
 259 rates without sacrificing performance.

260 **BCD iteration.** The next parameter to study is the maximum iteration number specified for the
 261 BCD updates of IMF in Algorithm (1). According to the numerical results on various datasets,
 262 the IMF performance jumps significantly after 2 iterations, while more iteration numbers lead to
 263 marginal improvement. This observation is shown for Kodak in Figure 6b. This feature makes
 264 IMF computationally efficient since with a limited number of BCD iterations a high compression
 265 performance can be achieved.

266 5 Conclusion

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

278 A limitation of IMF lies in its inability to control the entropy of the elements in the factor matrices,
 279 which could enhance the performance of the following entropy-based lossless coder. We plan to
 280 address this in the future by incorporating some entropy-aware regularization into the current IMF
 281 objective function.

282 **References**

- 283 [1] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on*
284 *Computers*, 100(1):90–93, 1974.
- 285 [2] H Andrews and CLIII Patterson. Singular value decomposition (SVD) image coding. *IEEE transactions*
286 *on Communications*, 24(4):425–432, 1976.
- 287 [3] Marc Antonini, Michel Barlaud, Pierre Mathieu, and Ingrid Daubechies. Image coding using wavelet
288 transform. *IEEE Trans. Image Processing*, 1:20–5, 1992.
- 289 [4] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic
290 and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel
291 methods. *Mathematical Programming*, 137(1):91–129, 2013.
- 292 [5] Heinz H Bauschke, Patrick L Combettes, Heinz H Bauschke, and Patrick L Combettes. *Correction to:*
293 *Convex analysis and monotone operator theory in hilbert spaces*. Springer, 2017.
- 294 [6] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM*
295 *journal on Optimization*, 23(4):2037–2060, 2013.
- 296 [7] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for
297 nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1):459–494, 2014.
- 298 [8] Alex Clark. Pillow (PIL Fork) Documentation, 2015. URL <https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf>.
- 300 [9] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing.
301 *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212, 2011.
- 302 [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical
303 image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255.
304 Ieee, 2009.
- 305 [11] Peter Deutsch and Jean-Loup Gailly. Zlib compressed data format specification version 3.3. Technical
306 report, 1996.
- 307 [12] Google Developers. WebP Compression Techniques. <https://developers.google.com/speed/webp>,
308 2011. Accessed: 2024-05-17.
- 309 [13] Chris HQ Ding, Tao Li, and Michael I Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE*
310 *transactions on pattern analysis and machine intelligence*, 32(1):45–55, 2008.
- 311 [14] Bo Dong, Matthew M Lin, and Haesun Park. Integer matrix approximation and data mining. *Journal of*
312 *scientific computing*, 75:198–224, 2018.
- 313 [15] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1
314 (3):211–218, 1936.
- 315 [16] Nicholas Gillis. *Nonnegative Matrix Factorization*. SIAM, 2020.
- 316 [17] Vivek K Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5):
317 9–21, 2001.
- 318 [18] Miska M Hannuksela, Jani Lainema, and Vinod K Malamal Vadakital. The high efficiency image file
319 format standard [standards in a nutshell]. *IEEE Signal Processing Magazine*, 32(4):150–156, 2015.
- 320 [19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition.
321 In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- 322 [20] Junhui Hou, Lap-Pui Chau, Nadia Magnenat-Thalmann, and Ying He. Sparse low-rank matrix approxi-
323 mation for data compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(5):
324 1043–1054, 2015.
- 325 [21] Eastman Kodak. Kodak lossless true color image suite, 1993. URL <https://r0k.us/graphics/kodak/>.
- 327 [22] Jani Lainema, Miska M Hannuksela, Vinod K Malamal Vadakital, and Emre B Aksu. HEVC still image
328 coding and high efficiency image file format. In *2016 IEEE International Conference on Image Processing*
329 (*ICIP*), pages 71–75. IEEE, 2016.

- 330 [23] Daniel Lee and H. Sebastian Seung. Algorithms for Non-negative Matrix Factorization. In T. Leen,
331 T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13.
332 MIT Press, 2000. URL https://proceedings.neurips.cc/paper_files/paper/2000/file/f9d1152547c0bde01830b7e8bd60024c-Paper.pdf.
- 334 [24] Matthew M Lin, Bo Dong, and Moody T Chu. Integer matrix factorization and its application.
- 335 [25] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete
336 basis problem. *IEEE transactions on knowledge and data engineering*, 20(10):1348–1362, 2008.
- 337 [26] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM
338 Journal on Optimization*, 22(2):341–362, 2012.
- 339 [27] Organizers. Challenge on Learned Image Compression, 2024. URL <https://www.compression.cc/>.
- 340 [28] HS Prasanth, HL Shashidhara, and KN Balasubramanya Murthy. Image compression using SVD. In
341 *International conference on computational intelligence and multimedia applications (ICCI-MA 2007)*,
342 volume 3, pages 143–145. IEEE, 2007.
- 343 [29] Siamak Ravanbakhsh, Barnabás Póczos, and Russell Greiner. Boolean matrix factorization and noisy
344 completion via message passing. In *International Conference on Machine Learning*, pages 945–954.
345 PMLR, 2016.
- 346 [30] Jerome M Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on
347 signal processing*, 41(12):3445–3462, 1993.
- 348 [31] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The JPEG 2000 still image com-
349 pression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.
- 350 [32] Peter van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a
351 lattice. *Technical Report, Department of Mathematics, University of Amsterdam*, 1981.
- 352 [33] Gregory K Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):
353 30–44, 1991.
- 354 [34] Xin Yuan and Raziel Haimi-Cohen. Image compression based on compressive sensing: End-to-end
355 comparison with JPEG. *IEEE Transactions on Multimedia*, 22(11):2889–2904, 2020.

356 **A Omitted Proofs**

357 **A.1 Proof of Theorem 1**

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

$$\arg \min_{\mathbf{u}_r \in \mathbb{Z}_{[\alpha, \beta]}^M} \|\mathbf{E}_r - \mathbf{u}_r \mathbf{v}_r^\top\|_F = \arg \min_{\mathbf{u}_r \in \mathbb{Z}_{[\alpha, \beta]}^M} \sum_{i=1}^M \sum_{j=1}^N (e_{ij}^r - u_i^r v_j^r)^2, \quad (10)$$

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

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

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

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

367 where ∇_x is the partial derivative with respect to x . Since q_i has a constant curvature (second
 368 derivative) and $q_i(\bar{u}_i^r + d)$ is nondecreasing with increasing $|d|$, the value in the set $\mathbb{Z}_{[\alpha, \beta]}$ which
 369 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
 370 the set $\mathbb{Z}_{[\alpha, \beta]}$, namely $u_i^{r^*} = \text{clamp}_{[\alpha, \beta]}(\text{round}(\sum_{j=1}^N e_{ij}^r v_j^r / \sum_{j=1}^N v_j^{r^2}))$, which is presented for all
 371 $i \in \{1, \dots, M\}$ in a compact form in (6).

372 **A.2 Proof of Theorem 2**

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

$$\begin{aligned} & \underset{U_{:r} \in \mathbb{R}^M, V_{:r} \in \mathbb{R}^N, \forall r \in \{1, \dots, R\}}{\text{minimize}} \quad \Psi(\mathbf{U}, \mathbf{V}) := f_0(\mathbf{U}, \mathbf{V}) + \sum_{r=1}^R f_r(U_{:r}) + \sum_{r=1}^R g_r(V_{:r}), \\ & \text{where} \quad f_0(\mathbf{U}, \mathbf{V}) := \|\mathbf{X} - \mathbf{U}\mathbf{V}^\top\|_F^2, \\ & \quad f_r(U_{:r}) := \delta_{[a, b]}(U_{:r}) + \delta_{\mathbb{Z}}(U_{:r}), \\ & \quad g_r(V_{:r}) := \delta_{[a, b]}(V_{:r}) + \delta_{\mathbb{Z}}(V_{:r}), \end{aligned} \quad (13)$$

375 with $\delta_{\mathcal{B}}(\cdot)$ as the indicator function of the nonempty set \mathcal{B} where $\delta_{\mathcal{B}}(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathcal{B}$ and $\delta_{\mathcal{B}}(\mathbf{x}) =$
 376 $+\infty$, otherwise. By the definition of functions above, it is easy to confirm that the problems (3) and
 377 (13) are equivalent.

378 The unconstrained optimization problem (13) consists of the sum of a differentiable (smooth), convex
 379 function f_0 with nonsmooth, nonconvex functions f_r and g_r . This problem has been extensively
 380 studied in the literature under the class of nonconvex nonsmooth minimization problems. One of the
 381 common algorithms applied to such a problem class is the well-known forward-backward-splitting
 382 (FBS) algorithm [9, 5]. In Algorithm 1, the blocks $U_{:r}$ and $V_{:r}$ are updated sequentially following
 383 block coordinate (BC) descent minimization algorithms, also often called Gauss-Seidel updates or
 384 alternating minimization [26, 4]. Hence, in this convergence study, we are interested in algorithms that
 385 allow BC-type updates for the nonconvex nonsmooth problem of (13) [6, 7]. Specifically, we focus
 386 on the proximal alternating linearized minimization (PALM) algorithm [7], to relate its convergence
 387 behavior to that of Algorithm 1. To that end, we show that the updates of Algorithm 1 are equivalent
 388 to the updates of PALM on the recast problem of (13), and all the assumptions necessary for the
 389 convergence of PALM are satisfied by our problem setting.

390 The PALM algorithm can be summarized as follows:

- 391 1. Initialize $\mathbf{U}^{\text{init}} \in \mathbb{R}^{M \times R}$, $\mathbf{V}^{\text{init}} \in \mathbb{R}^{N \times R}$
 392 2. For each iteration $k = 0, 1, \dots$
- $$(a) \quad \mathbf{U}^{k+1} \in \text{prox}_{c_k}^f \left(\mathbf{U}^k - \frac{1}{c_k} \nabla_{\mathbf{U}} H(\mathbf{U}^k, \mathbf{V}^k) \right), \text{ with } c_k > L_1(\mathbf{V}^k) \quad (14)$$
- $$(b) \quad \mathbf{V}^{k+1} \in \text{prox}_{d_k}^g \left(\mathbf{V}^k - \frac{1}{d_k} \nabla_{\mathbf{V}} H(\mathbf{U}^{k+1}, \mathbf{V}^k) \right), \text{ with } d_k > L_2(\mathbf{U}^{k+1})$$

393 where the proximal map for an extended proper lower semicontinuous (nonsmooth) function $\varphi : \mathbb{R}^n \rightarrow (-\infty, +\infty]$ and $\gamma > 0$ is defined as $\text{prox}_{\gamma}^{\varphi}(\mathbf{x}) := \arg \min_{\mathbf{w} \in \mathbb{R}^n} \{ \varphi(\mathbf{w}) + \frac{\gamma}{2} \|\mathbf{w} - \mathbf{x}\|_2^2 \}$, and
 394 $L_1 > 0$, $L_2 > 0$ are local Lipschitz moduli, defined in the following proposition. It is also remarked
 395 that the iterates in (14) can be extended to more than two blocks, which is our case in Algorithm 1
 396 with a block representing a column of \mathbf{U} or \mathbf{V} , without violation of the convergence. However, for
 397 the sake of presentation, we study these BC-type iterates with two blocks of the form (14).
 398

399 The following proposition investigates the necessary assumptions (cf. [7, Asm. 1 and Asm. 2]) for
 400 convergence iterates in (14).

401 **Proposition 1** (Meeting required assumptions). *The assumptions necessary for the convergence of
 402 iterates in (14) are satisfied by the functions involved in the problem (13), specifically:*

- 403 1. *The indicator functions $\delta_{[a,b]}$ and $\delta_{\mathbb{Z}}$ are proper and lower semicontinuous functions, so do
 404 the functions f and g ;*
 405 2. *For any fixed \mathbf{V} , the partial gradient $\nabla_{\mathbf{U}} H(\mathbf{U}, \mathbf{V})$ is globally Lipschitz continuous with
 406 modulus $L_1(\mathbf{V}) = \|\mathbf{V}^T \mathbf{V}\|_{\text{F}}$ defined by*

$$\|\nabla_{\mathbf{U}} H(\mathbf{U}_1, \mathbf{V}) - \nabla_{\mathbf{U}} H(\mathbf{U}_2, \mathbf{V})\| \leq L_1(\mathbf{V}) \|\mathbf{U}_1 - \mathbf{U}_2\|, \quad \forall \mathbf{U}_1, \mathbf{U}_2 \in \mathbb{R}^{M \times R},$$

407 where $\|\cdot\|$ in this section denotes the ℓ_2 -norm of the vectorized input with the proper
 408 dimension (here, with the input in $\mathbb{R}^{MR \times 1}$). The similar Lipschitz continuity is evident for
 409 $\nabla_{\mathbf{V}} H(\mathbf{U}, \mathbf{V})$ as well with modulus $L_2(\mathbf{U}) = \|\mathbf{U} \mathbf{U}^T\|_{\text{F}}$.

- 410 3. *The sequences \mathbf{U}^k and \mathbf{V}^k are bounded due to the indicator functions $\delta_{[a,b]}$ with bounded a
 411 and b . Hence the moduli $L_1(\mathbf{V}^k)$ and $L_2(\mathbf{U}^k)$ are bounded from below and from above for
 412 all $k \in \mathbb{N}$.*
 413 4. *The function H is twice differentiable, hence, its full gradient $\nabla H(\mathbf{U}, \mathbf{V})$ is Lipschitz
 414 continuous on the bounded set $\mathbf{U} \in [a, b]^{M \times R}$, $\mathbf{V} \in [a, b]^{N \times R}$. Namely, with $M > 0$:*

$$\begin{aligned} \|(\nabla_{\mathbf{U}} H(\mathbf{U}_1, \mathbf{V}_1) - \nabla_{\mathbf{U}} H(\mathbf{U}_2, \mathbf{V}_2), \nabla_{\mathbf{V}} H(\mathbf{U}_1, \mathbf{V}_1) - \nabla_{\mathbf{V}} H(\mathbf{U}_2, \mathbf{V}_2))\| \\ \leq M \|(\mathbf{U}_1 - \mathbf{U}_2, \mathbf{V}_1 - \mathbf{V}_2)\|, \end{aligned} \quad (15)$$

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

- 416 5. *The sets $[a, b]$ and integer numbers are semi-algebraic; so are their indicator functions. The
 417 function H is also polynomial, hence it is semi-algebraic. The sum of these functions results
 418 in a semi-algebraic function Ψ in (13), hence Ψ is a Kurdyka-Łojasiewicz (KL) function.*

419 By Proposition 1, the optimization problem (13) can be solved by the BC iterates in (14), due to the
 420 following proposition:

421 **Proposition 2** (Global convergence [7]). *With the assumptions in proposition 1 being met by the
 422 problem (13), let $((\mathbf{U}^k, \mathbf{V}^k))_{k \in \mathbb{N}}$ be a sequence generated by the BC iterates in (14). Then the
 423 sequence converges to a critical point $(\mathbf{U}^*, \mathbf{V}^*)$ of the problem (13), where $0 \in \partial \Psi(\mathbf{U}^*, \mathbf{V}^*)$, with
 424 ∂ as the subdifferential of Ψ .*

425 In the following, we highlight that the iterates in (14) can be implemented more simply and more
 426 efficiently by Algorithm 1 for the problem of image compression. It is noted that the so-called *forward*
 427 steps $\mathbf{U}^k - \frac{1}{c_k} \nabla_{\mathbf{U}} H(\mathbf{U}^k, \mathbf{V}^k)$ and $\mathbf{V}^k - \frac{1}{d_k} \nabla_{\mathbf{V}} H(\mathbf{U}^{k+1}, \mathbf{V}^k)$ in the prox operators can be replaced
 428 by the simple closed-form solutions $E_r v_r / \|v_r\|^2$ and $E_r^T u_r / \|u_r\|^2$ presented in (6) and (7), respectively
 429 (in the case where the iterates (14) are extended to multi-block updates, each block representing

430 one column). This is thanks to the special form of the functions $H(\cdot, \mathbf{V}^k)$ and $H(\mathbf{U}^{k+1}, \cdot)$ being
431 quadratic functions, each having a global optimal point which ensures a descent in each forward
432 step. Furthermore, the proximal operators $\text{prox}_{c_k}^f$ and $\text{prox}_{d_k}^g$ can efficiently be implemented by
433 the operators round and clamp $_{[\alpha,\beta]}$ in (6) and (7). The equivalence of these steps is proven in the
434 following lemma.

435 **Lemma 1** (prox implementation). *Consider the operators round and clamp $_{[\alpha,\beta]}$ defined in (6) and
436 (7). Then $\text{prox}_{c_k}^f(\mathbf{W}) = \text{round}(\text{clamp}_{[\alpha,\beta]}(\mathbf{W}))$ and $\text{prox}_{d_k}^g(\mathbf{Z}) = \text{round}(\text{clamp}_{[\alpha,\beta]}(\mathbf{Z}))$ for any
437 $\mathbf{W} \in \mathbb{R}^{M \times R}$, $\mathbf{Z} \in \mathbb{R}^{N \times R}$, and round(clamp $_{[\alpha,\beta]}(\cdot)$) being an elementwise operator on the input
438 matrices.*

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

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

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

$$\text{round}(\mathbf{W}) = \arg \min_{\mathbf{U} \in \mathbb{Z}^{M \times R}} \{\|\mathbf{U} - \mathbf{W}\|_F^2\}. \quad (16)$$

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

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

442 The first equality is due to the definition of prox which is equivalent to the second equality. In
443 the third equality the matrices $\mathbf{A} \in \mathbb{R}^{M \times R}$ and $\mathbf{B} \in \mathbb{R}^{M \times R}$ have elements all equal to a and b ,
444 respectively. The third equality is due to the fact that replacing $\|\mathbf{U} - \mathbf{W}\|_a^2 + \|\mathbf{U} - \mathbf{W}\|_b^2$ with
445 $\|\mathbf{U} - \mathbf{A}\|_a^2 + \|\mathbf{U} - \mathbf{B}\|_b^2$ has no effect on the solution of the minimization. The fourth equality is
446 also trivial due to the involved norms in the third equality. The fifth equality can be easily confirmed
447 by the definition of clamp $_{[\alpha,\beta]}$. Finally, in the last equality (16) is invoked. It is noted that in the
448 implementation, $\text{round}(\text{clamp}_{[\alpha,\beta]}(\cdot)) = \text{clamp}_{[\alpha,\beta]}(\text{round}(\cdot))$ due to the integrality of the bounds
449 $\alpha, \beta \in \mathbb{Z}$. A similar proof can be trivially followed for $\text{prox}_{d_k}^g(\mathbf{Z}) = \text{round}(\text{clamp}_{[\alpha,\beta]}(\mathbf{Z}))$ as
450 well. \square

451 Now that the equivalence of iterates (14) with the simple and closed-form steps in Algorithm 1 is
452 fully established, and the assumptions required for the convergence are verified in proposition 1 to be
453 met by problems (13) and (3), proposition 2 can be trivially invoked to establish the convergence of
454 Algorithm 1 to a locally optimal point of problem (3).

455 B Implementation Details

456 **Parameter Setup.** We used a patch size of 8×8 for patchification. The number of BCD iterations
457 for IMF is by default set to 10 (although our ablation studies in Section 4.4 suggests that even 2
458 iterations may be sufficient in practice). For lossless compression of factor maps, we employed the
459 WebP codec implemented in the Pillow library [8]. The default factor bounds are $[-16, 15]$.

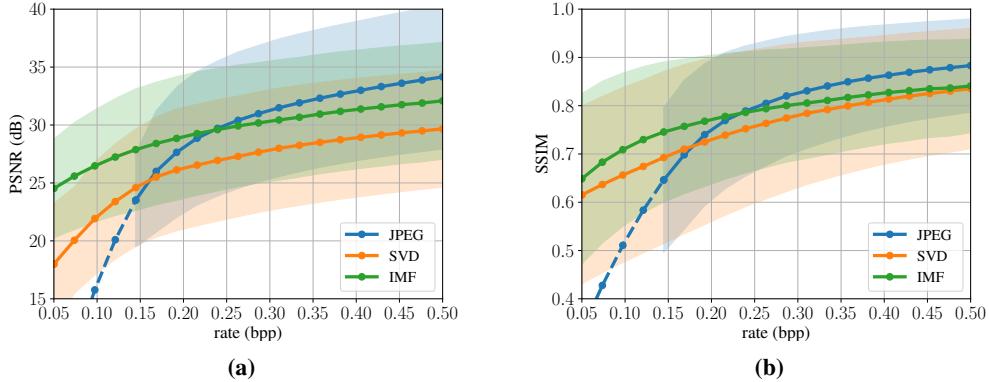


Figure 7 Rate-distortion performance on the CLIC dataset. The average PSNR and SSIM for each method is plotted as a function of bits per pixel (bpp). The shaded areas represent the standard deviation. The dashed line corresponds to extrapolation using quadratic splines.

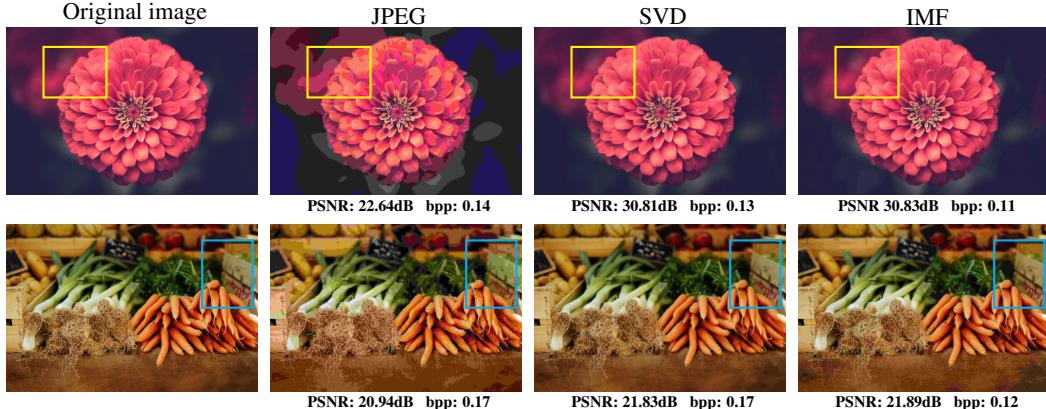


Figure 8 Qualitative performance comparison on an image from the CLIC dataset. The bounding box highlights the artifacts produced by JPEG.

460 **Evaluation.** For comparison, we experimented with the widely-used Kodak dataset [21], with 24
 461 lossless images of 768×512 resolution. To assess the robustness of our proposed method, we also
 462 tested it using the CLIC 2024 validation dataset [27], consisting of 30 high-resolution, high-quality
 463 images. To evaluate the rate-distortion performance, we measured the bit rate in bits per pixel (bpp)
 464 and assessed the quality of reconstructed images using the peak signal-to-noise ratio (PSNR) and
 465 structural similarity index measure (SSIM) metrics. We plotted rate-distortion (RD) curves, such as
 466 the PSNR-bpp curve, for each method to demonstrate the compression performance.

467 **Baseline Codecs.** For JPEG compression, we employed the Pillow library [8]. Our SVD-based
 468 compression baseline follows the same framework as the proposed IMF compression algorithm.
 469 However, it employs truncated SVD instead of IMF, followed by uniform quantization of the factors
 470 and lossless compression using the zlib library [11]. This differs from the IMF algorithm where
 471 factors are first reshaped into factor maps and then compressed losslessly as images using WebP.

472 C Experiments on CLIC

473 Figure 7 shows the rate-distortion curves for IMF, SVD, and JPEG on the CLIC dataset. For each
 474 method, PSNR and SSIM values are averaged over all images at different bit rates, up to 0.5 bits
 475 per pixel (bpp). Similar to the results provided for the Kodak dataset in Figure 3, our IMF method
 476 consistently outperforms JPEG at low bit rates and significantly outperforms the SVD-based method
 477 at all bit rates, in terms of both PSNR and SSIM.

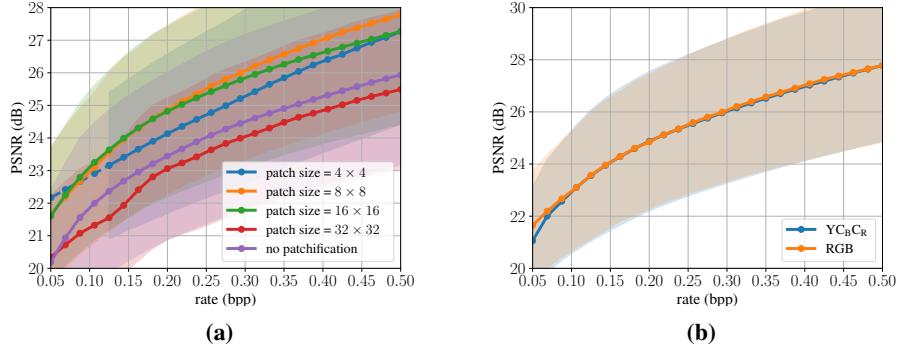


Figure 9 Ablation studies for IMF. The average PSNR on the Kodak dataset is plotted against the rate (bpp). (a) shows the PSNR-bpp curve for different patch sizes. (b) shows the PSNR-bpp curve for the use of YCbCr and RGB color spaces.

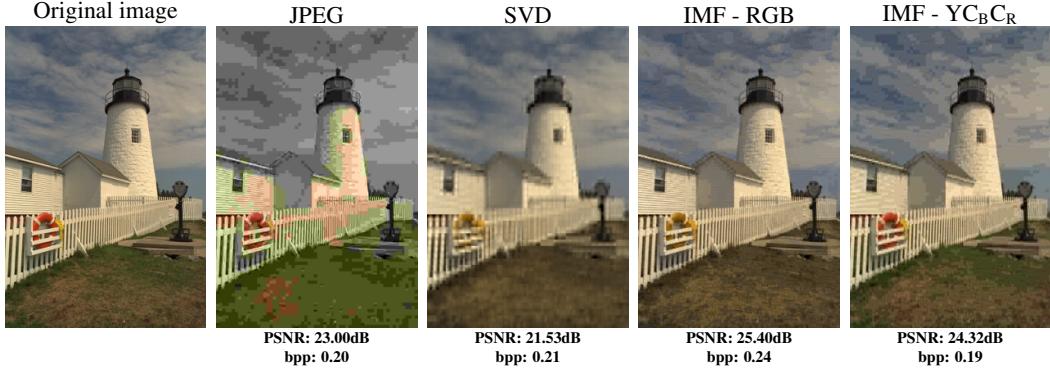


Figure 10 Qualitative comparison on an image from Kodak.

478 D More Ablations Studies

479 **Patchification.** In Figure 9a, patchification effect with different patch sizes is investigated. First, it
 480 can be concluded that the considered patchification technique has a positive effect on performance
 481 since it captures the locality and spatial dependencies of neighboring pixels. Hence, IMF has more
 482 representation power to reconstruct the original pattern in each patch. The performance on various
 483 datasets has shown that patches of size 8 × 8 lead to the best performance. The same conclusion is
 484 evident for the Kodak dataset example presented in Figure 9a.

485 **Color space.** The compression performance of IMF is studied with two color spaces, namely RGB
 486 and YC_BC_R, in Figure 9b. Although the compression performance remains unchanged in terms of
 487 PSNR, qualitative results reported in Figure 10 indicate that YC_BC_R color space can maintain natural
 488 colors more effectively.

489 E Run Time

490 The encoding and decoding times for each method, at a bit rate of 0.2 bpp, are detailed in Table 1. All
 491 experiments were conducted on a Xeon Gold 6140 CPU@2.3 GHz with 192 GiB RAM. In terms of
 492 decoding speed, IMF outperforms JPEG by more than four times and SVD by two times. However,
 493 when it comes to encoding, IMF is four times slower than JPEG and two times slower than SVD.

Table 1 The average encoding and decoding CPU times over Kodak for different compression methods at a bit rate of 0.2 bpp.

	JPEG	SVD	IMF
encoding time	16.3 ms	20.8 ms	41.3 ms
decoding time	4.1 ms	2.5 ms	1 ms