Quantization-free Lossy Image Compression Using Integer Matrix Factorization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Image compression is essential for efficient transmission and storage.The most widely used transform coders, such as the discrete cosine transform (DCT) and singular value decomposition (SVD), represent image data in continuous domains, necessitating carefully designed quantizers. Notably, the performance of SVD-based methods is more sensitive to quantization errors than that of DCT-based methods like JPEG. Therefore, we introduce a variant of integer matrix factorization (IMF) to develop a novel quantization-free 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. The experiments demonstrate that our IMF-based compression method outperforms JPEG, achieving improvements of over 3 dB in PSNR at rates of under 0.17 bits per pixel (bpp) on the Kodak dataset. We also assessed our method's capability to preserve visual semantic information by evaluating an ImageNet pre-trained classifier on compressed images. Remarkably, our method resulted in an improvement of over 10% in top-1 accuracy compared to JPEG at rates under 0.23 bpp.

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 [33] and JPEG 2000 [31] follow the *transform coding* paradigm [17]. They use orthogonal linear transformations, such as discrete cosine transform (DCT) [1] and discrete wavelet transform (DWT) [3], to decorrelate small image blocks. Since these transforms map image data into a continuous domain, 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 methods. Particularly singular value decomposition (SVD) is a representative method known to be the deterministically optimal transform for energy compaction [2]. In practice, current SVD-based methods [2, 28, 20] can represent image data only with components containing floating-point elements, which necessitates a quantization layer prior to any byte-level processing. Nevertheless, their compression performance is suboptimal

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

due to higher sensitivity 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) [1], used in JPEG [33], which converts image data into the frequency domain, prioritizing lower frequencies to retain perceptually significant information. The discrete wavelet transform (DWT) [3], used in JPEG 2000 [31], offers improved performance by capturing both frequency and location information, leading to better handling of edges and textures [30]. More recently, the WebP [12] and HEIF [22, 18] formats combine DCT and intra-frame prediction to achieve superior compression and quality compared to JPEG.

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 [2, 28]. Hou et al. [20] 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 [34] 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. [24] investigates IMF to effectively handle discrete data matrices for cluster analysis and pattern discovery. Dong et al. [14] 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 [25] and machine learning [29] 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.

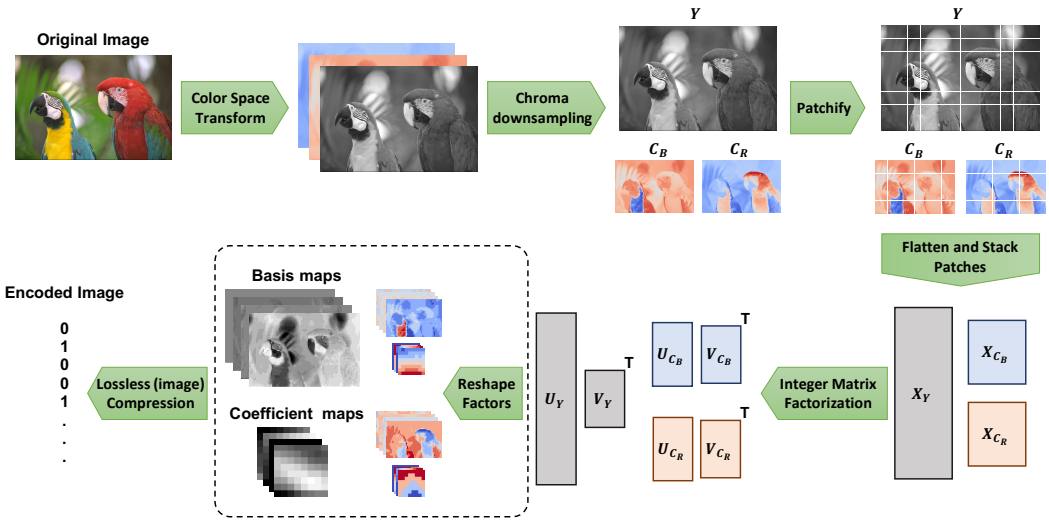


Figure 1 An illustration of the encoder for our image compression method.

3 Method

3.1 Overall Encoding Framework

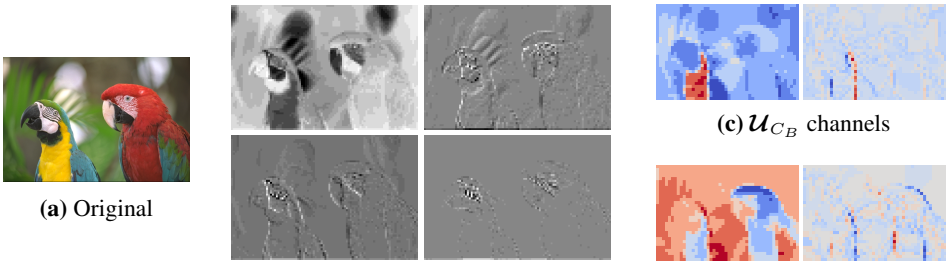
The proposed compression method follows a *transform coding* paradigm, but it does not involve quantization. Figure 1 illustrates an overview of our encoding pipeline based on integer matrix factorization (IMF). The encoder accepts an RGB image with dimensions $H \times W$ and a color depth of 8 bits,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



(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.

of these matrices. The low-rank approximation [15] 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, 13, 24]. 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 $\boldsymbol{U}_Y \in \{0,\dots,255\}^{\frac{HW}{64}\times R}$ and $\boldsymbol{V}_Y \in \{0,\dots,255\}^{64\times R}$; $\boldsymbol{U}_{C_B} \in \{0,\dots,255\}^{\frac{HW}{256}\times R}$ and $\boldsymbol{V}_{C_B} \in \{0,\dots,255\}^{64\times R}$; and $\boldsymbol{U}_{C_R} \in \{0,\dots,255\}^{\frac{HW}{256}\times R}$ and $\boldsymbol{V}_{C_R} \in \{0,\dots,255\}^{64\times R}$ 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 [11] without the need for a quantization step, which is commonly present in other lossy image compression methods and adds extra complications.

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 [11] 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 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 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:

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

where $\|\cdot\|_{\mathrm{F}}$ 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 [15]. If only a nonnegativity constraint were applied (without integrality), variations of nonnegative matrix factorization (NMF) would emerge [23, 16]. The IMF problem (3) poses a challenging integer program, with finding its global minima known to be NP-hard [14, 32]. Only a few iterative algorithms [14, 24] 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

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

Input: $\boldsymbol{X} \in \mathbb{R}^{M \times N}$, factorization rank R
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 *stopping criterion not satisfied* **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**
    $U_{:r}^{k+1} = \text{clamp}_{[\alpha,\beta]}\left(\text{round}\left(\frac{A_{:r} - \sum_{s=1}^{r-1} B_{sr}U_{:s}^{k+1} - \sum_{s=r+1}^{M} B_{sr}U_{:s}^{k}}{\|V_{:r}^k\|^2}\right)\right)$
  end
  $\boldsymbol{A} \leftarrow \boldsymbol{X}^{\mathsf{T}}\boldsymbol{U}^{k+1}$; $\boldsymbol{B} \leftarrow {\boldsymbol{U}^{k+1}}^{\mathsf{T}}\boldsymbol{U}^{k+1}$
  for $r = 1, \ldots, R$ **do**
    $V_{:r}^{k+1} = \text{clamp}_{[\alpha,\beta]}\left(\text{round}\left(\frac{A_{:r} - \sum_{s=1}^{r-1} B_{sr}V_{:s}^{k+1} - \sum_{s=r+1}^{N} B_{sr}V_{:s}^{k}}{\|U_{:r}^{k+1}\|^2}\right)\right)$
  end
end
return $(\boldsymbol{U}^k, \boldsymbol{V}^k)$

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

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

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

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 1 below.

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

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

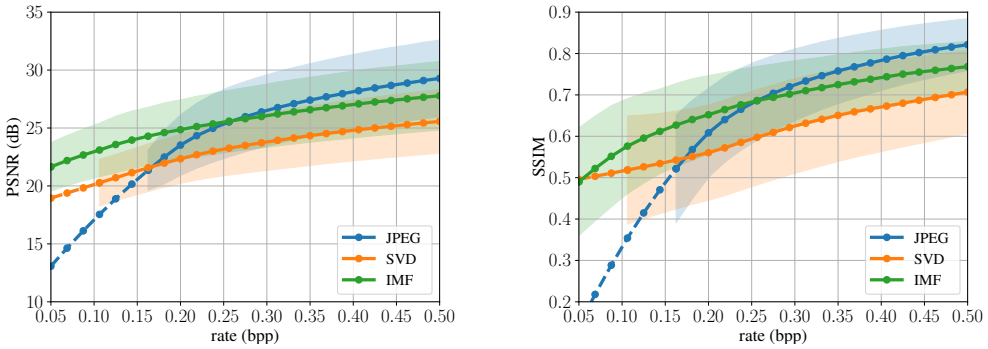
$$\boldsymbol{v}_r \leftarrow \text{clamp}_{[\alpha,\beta]}\left(\text{round}\left(\frac{\boldsymbol{E}_r^{\mathsf{T}}\boldsymbol{u}_r}{\|\boldsymbol{u}_r\|^2}\right)\right), \tag{7}$$

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

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

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

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



(a) **(b)**

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.

Proof. See Appendix A.2 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

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

4.1 Rate-Distortion Performance

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

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

4.2 Qualitative Performance

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

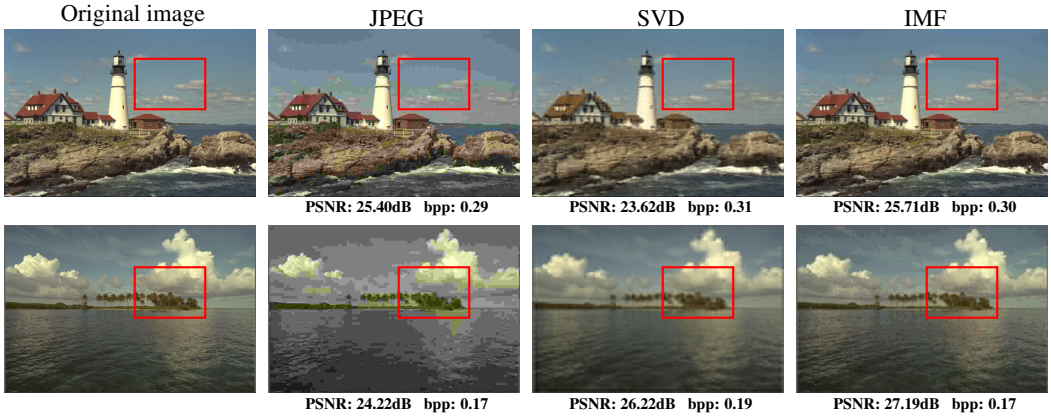


Figure 4 Qualitative performance comparison on an image from the Kodak dataset. The red bounding box highlights the artifacts produced by JPEG and the blurriness present in SVD-compressed images.

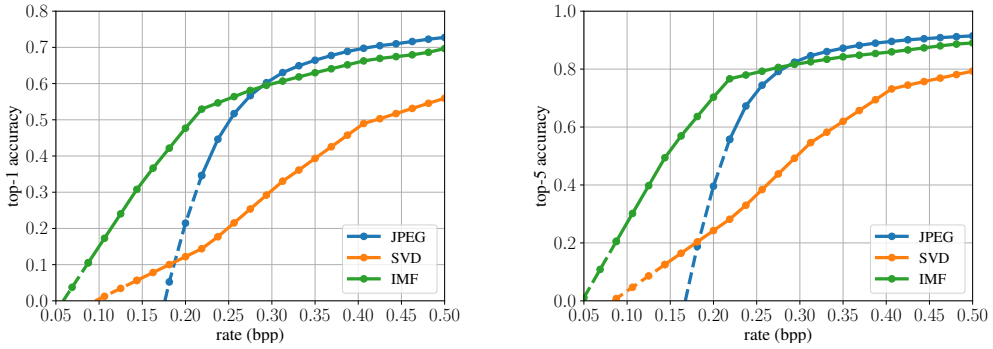
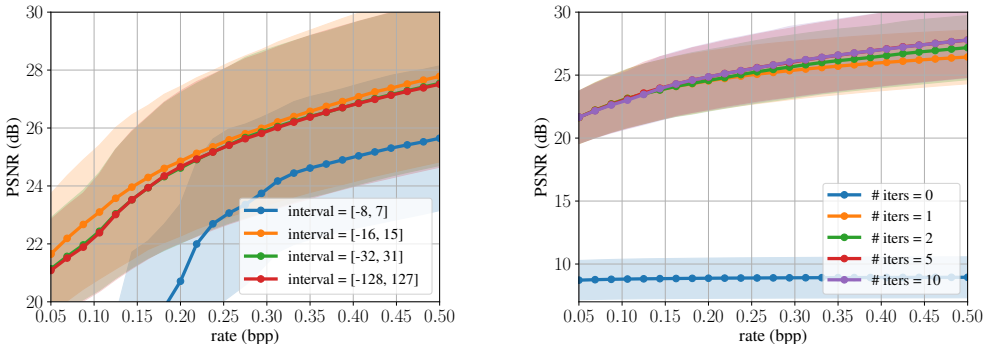


Figure 5 Impact of different compression methods on ImageNet classification accuracy. A ResNet-50 classifier pre-trained on the original ImageNet images is evaluated using validation images compressed by different methods. Panels (a) and (b) show top-1 and top-5 accuracy plotted against bpp, respectively.

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.



(a) **(b)**

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.

4.4 Ablation Studies

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

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

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

5 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 efficiency and flexibility of the system. 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, particularly at low bit rates, and maintains better visual semantic information. This advancement underscores the potential of IMF to set a new standard in lossy image compression, bridging the gap between factorization and quantization.

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

References

[1] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[2] H Andrews and CLIII Patterson. Singular value decomposition (SVD) image coding. *IEEE transactions on Communications*, 24(4):425–432, 1976.

[3] Marc Antonini, Michel Barlaud, Pierre Mathieu, and Ingrid Daubechies. Image coding using wavelet transform. *IEEE Trans. Image Processing*, 1:20–5, 1992.

[4] 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. *Mathematical Programming*, 137(1):91–129, 2013.

[5] Heinz H Bauschke, Patrick L Combettes, Heinz H Bauschke, and Patrick L Combettes. *Correction to: Convex analysis and monotone operator theory in hilbert spaces*. Springer, 2017.

[6] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization*, 23(4):2037–2060, 2013.

[7] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1):459–494, 2014.

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

[9] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212, 2011.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] Peter Deutsch and Jean-Loup Gailly. Zlib compressed data format specification version 3.3. Technical report, 1996.

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

[13] Chris HQ Ding, Tao Li, and Michael I Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):45–55, 2008.

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

[15] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1 (3):211–218, 1936.

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

[17] Vivek K Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5): 9–21, 2001.

[18] Miska M Hannuksela, Jani Lainema, and Vinod K Malamal Vadakital. The high efficiency image file format standard [standards in a nutshell]. *IEEE Signal Processing Magazine*, 32(4):150–156, 2015.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Junhui Hou, Lap-Pui Chau, Nadia Magnenat-Thalmann, and Ying He. Sparse low-rank matrix approximation for data compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(5): 1043–1054, 2015.

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

[22] Jani Lainema, Miska M Hannuksela, Vinod K Malamal Vadakital, and Emre B Aksu. HEVC still image coding and high efficiency image file format. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 71–75. IEEE, 2016.

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

[24] Matthew M Lin, Bo Dong, and Moody T Chu. Integer matrix factorization and its application.

[25] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE transactions on knowledge and data engineering*, 20(10):1348–1362, 2008.

[26] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

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

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

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

[30] Jerome M Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on signal processing*, 41(12):3445–3462, 1993.

[31] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

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

[33] Gregory K Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4): 30–44, 1991.

[34] Xin Yuan and Raziel Haimi-Cohen. Image compression based on compressive sensing: End-to-end comparison with JPEG. *IEEE Transactions on Multimedia*, 22(11):2889–2904, 2020.

A Omitted Proofs

A.1 Proof of Theorem 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 $u_i^{r^\star} = \mathrm{clamp}_{[\alpha,\beta]}(\mathrm{round}(\sum_{j=1}^{N} e_{ij}^r v_j^r / \sum_{j=1}^{N} {v_j^r}^2))$, which is presented for all $i \in \{1, ..., M\}$ in a compact form in (6).

A.2 Proof of Theorem 2

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

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

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 (13) are equivalent.

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

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 $k = 0, 1, ...$

$$\begin{aligned} &(a) \quad \boldsymbol{U}^{k+1} \in \operatorname{prox}^{f}_{c_k}\left(\boldsymbol{U}^k - \frac{1}{c_k}\nabla_{\boldsymbol{U}} H(\boldsymbol{U}^k, \boldsymbol{V}^k)\right), \text{ with } c_k > L_1(\boldsymbol{V}^k) \\ &(b) \quad \boldsymbol{V}^{k+1} \in \operatorname{prox}^{g}_{d_k}\left(\boldsymbol{V}^k - \frac{1}{d_k}\nabla_{\boldsymbol{V}} H(\boldsymbol{U}^{k+1}, \boldsymbol{V}^k)\right), \text{ with } d_k > L_2(\boldsymbol{U}^{k+1}) \end{aligned} \tag{14}$$

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 $\operatorname{prox}^{\varphi}_{\gamma}(\boldsymbol{x}) := \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} \left\{\varphi(\boldsymbol{w}) + \frac{\gamma}{2}\|\boldsymbol{w} - \boldsymbol{x}\|_2^2\right\}$, and $L_1 > 0$, $L_2 > 0$ are local Lipschitz moduli, defined in the following proposition. It is also remarked that the iterates in (14) can be extended to more than two blocks, which is our case in Algorithm 1 with a block representing a column of $\boldsymbol{U}$ or $\boldsymbol{V}$, without violation of the convergence. However, for the sake of presentation, we study these BC-type iterates with two blocks of the form (14).

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

Proposition 1 (Meeting required assumptions)**.** *The assumptions necessary for the convergence of iterates in* (14) *are satisfied by the functions involved in the problem* (13)*, 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 $\nabla_{\boldsymbol{U}} H(\boldsymbol{U}, \boldsymbol{V})$ is globally Lipschitz continuous with modulus $L_1(\boldsymbol{V}) = \|\boldsymbol{V}^T\boldsymbol{V}\|_{\mathrm{F}}$ defined by*

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

   where $\|\cdot\|$ in this section 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 $\nabla_{\boldsymbol{V}} H(\boldsymbol{U}, \boldsymbol{V})$ as well with modulus $L_2(\boldsymbol{U}) = \|\boldsymbol{U}\boldsymbol{U}^T\|_{\mathrm{F}}$.
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 H is twice differentiable, hence, its full gradient $\nabla H(\boldsymbol{U}, \boldsymbol{V})$ 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 $M > 0$:*

$$\begin{aligned} \|(\nabla_{\boldsymbol{U}} H(\boldsymbol{U}_1, \boldsymbol{V}_1) - \nabla_{\boldsymbol{U}} H(\boldsymbol{U}_2, \boldsymbol{V}_2), \nabla_{\boldsymbol{V}} H(\boldsymbol{U}_1, \boldsymbol{V}_1) - \nabla_{\boldsymbol{V}} H(\boldsymbol{U}_2, \boldsymbol{V}_2))\| \\ \le M\|(\boldsymbol{U}_1 - \boldsymbol{U}_2, \boldsymbol{V}_1 - \boldsymbol{V}_2)\|, \end{aligned} \tag{15}$$

   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 H is also polynomial, hence it is semi-algebraic. The sum of these functions results in a semi-algebraic function Ψ in* (13)*, hence Ψ is a Kurdyka-Łojasiewicz (KL) function.*

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

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

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

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

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

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

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

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

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

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

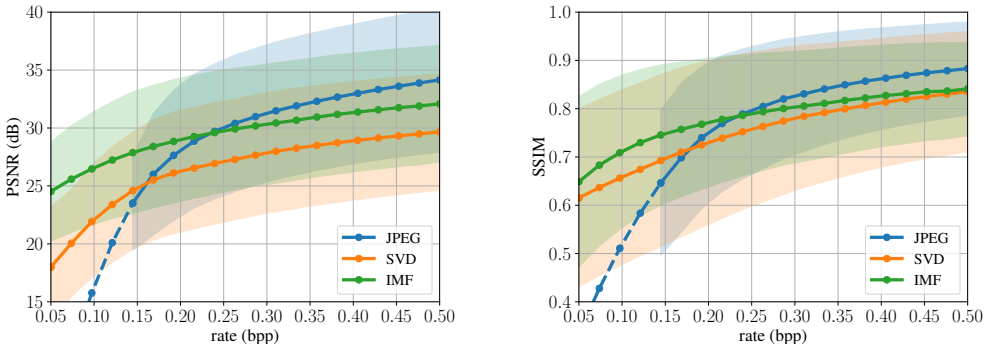
$$\begin{aligned}
\text{prox}_{c_k}^f(\boldsymbol{W}) &= \underset{\boldsymbol{U}\in\mathbb{R}^{M\times R}}{\arg\min}\{\delta_{[a,b]}(\boldsymbol{U}) + \delta_{\mathbb{Z}}(\boldsymbol{U}) + \frac{c_k}{2}\|\boldsymbol{U}-\boldsymbol{W}\|_F^2\} \\
&= \underset{\boldsymbol{U}\in\mathbb{Z}_{[a,b]}^{M\times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

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 $\text{prox}_{d_k}^f(\boldsymbol{Z}) = \text{round}(\text{clamp}_{[\alpha,\beta]}(\boldsymbol{Z}))$ as well. □

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

B Implementation Details

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



(a) **(b)**

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.

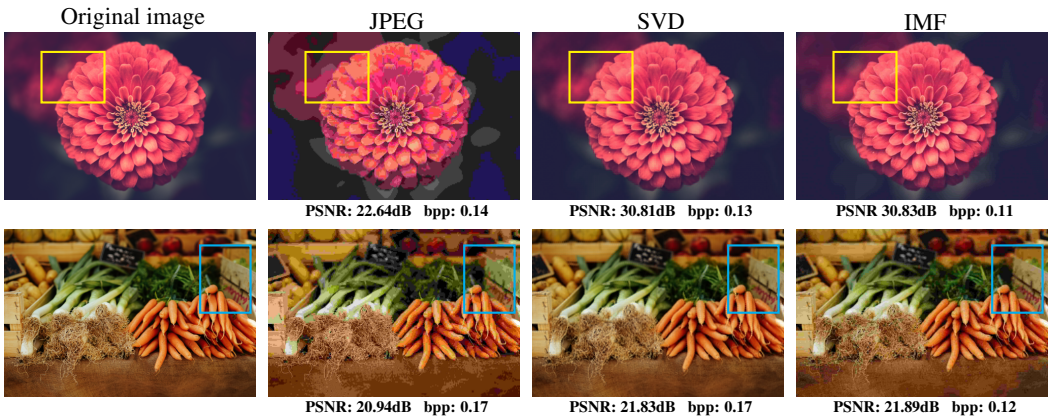


Figure 8 Qualitative performance comparison on an image from the CLIC dataset. The bounding box highlights the artifacts produced by JPEG.

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

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

C Experiments on CLIC

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

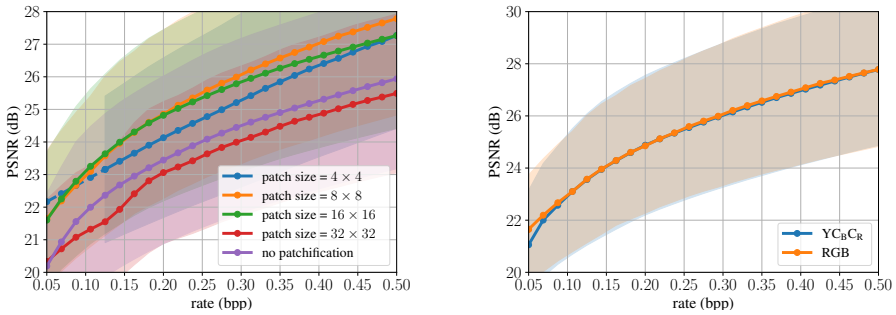


Figure 9 Ablation studies for IMF. The average PSNR on the Kodak dataset is plotted against the rate (bpp). (a) shows the PSNR-bpp curve for different patch sizes. (b) shows the PSNR-bpp curve for the use of YCbCr and RGB color spaces.

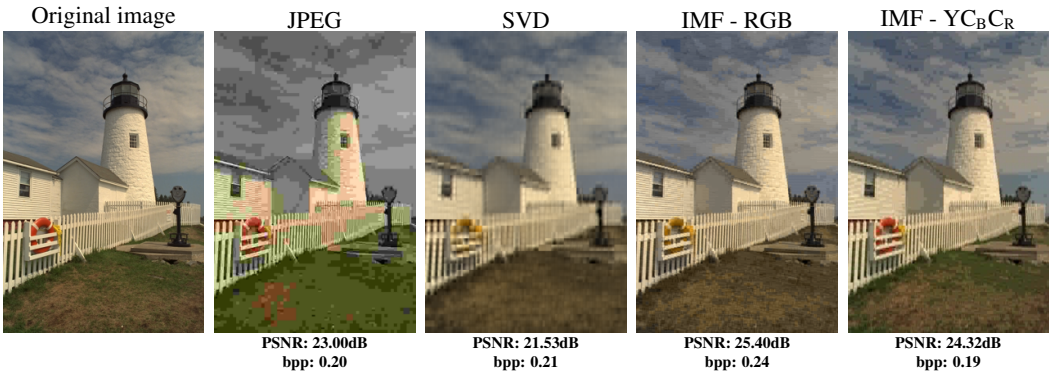


Figure 10 Qualitative comparison on an image from Kodak.

D More Ablations Studies

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

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

E Run Time

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