Quantization-free Lossy Image Compression Using Integer Matrix Factorization

Pooya Ashtari[1*†]
pooya.ashtari@esat.kuleuven.be

Pourya Behmandpoor[1†]
pourya.behmandpoor@kuleuven.be

Fateme Nateghi Haredasht[2]
fnateghi@stanford.edu

Jonathan H. Chen[2]
jonc101@stanford.edu

Panagiotis Patrinos[1]
panos.patrinos@esat.kuleuven.be

Sabine Van Huffel[1]
sabine.vanhuffel@kuleuven.be

[1]Department of Electrical Engineering (ESAT), STADIUS Center, KU Leuven, Belgium
[2]Department of Medicine, Stanford University, Stanford, CA, USA

Abstract

Lossy image compression is essential for efficient transmission and storage. Most widely used methods, such as JPEG, use transforms like the discrete cosine transform (DCT) that map image data into a continuous domain, which necessitates carefully designed quantization techniques. Another promising approach relies on low-rank approximation methods, such as singular value decomposition (SVD). While current methods within this approach also require a quantization layer, their compression performance is unfortunately suboptimal due to higher sensitivity to quantization errors than JPEG. Therefore, we introduce a variant of integer matrix factorization (IMF) that forms the basis for our quantization-free lossy image compression technique. 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. Moreover, the reshaped factor matrices can be treated as grayscale images, allowing any lossless image compression standard to be seamlessly integrated into the proposed IMF framework. We propose an efficient, provably convergent iterative algorithm for IMF using a block coordinate descent (BCD) scheme, where each column of a factor matrix is updated one at a time using a closed-form solution. The experiments show that our IMF method outperforms JPEG, achieving improvements of over 4 dB and 7 dB in PSNR at 0.15 bits per pixel (bpp) on the Kodak and CLIC datasets, respectively. We also assessed the ability of the methods to preserve visual semantic information by evaluating an ImageNet pre-trained classifier on compressed images. Notably, the IMF method yielded over a 20% relative improvement in top-1 accuracy over JPEG at bit rates under 0.25 bpp.

1 Introduction

Lossy image compression involves reducing the storage size of digital images by permanently discarding some image data details 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

*Corresponding author. Emails:pooya.ashtari@esat.kuleuven.be, pooya.ash@gmail.com
†Equal contribution

Preprint. Under review.

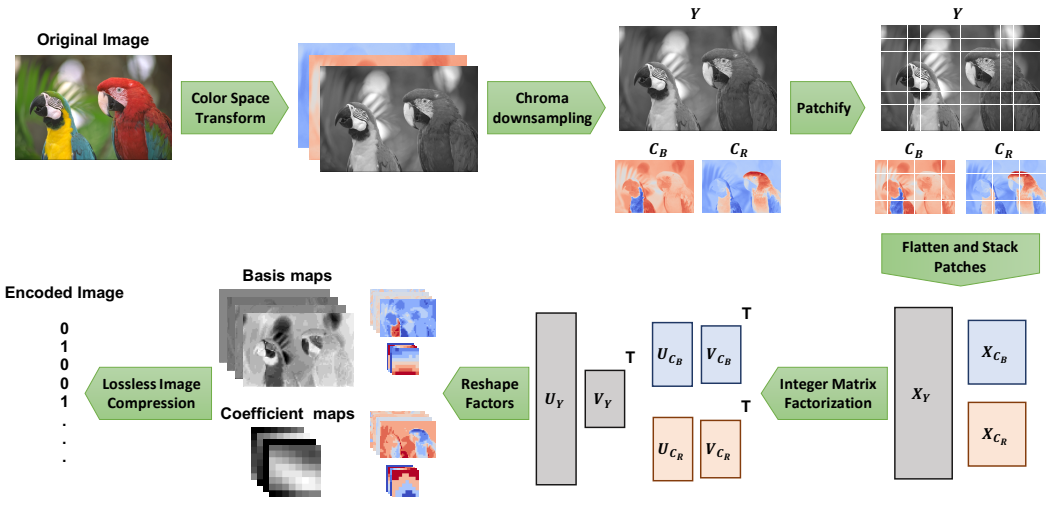


Figure 1 An illustration of the encoder for our image compression method, based on integer matrix factorization.

compression methods enable adjusting the degree of compression, providing a selectable tradeoff between storage size and image quality.

Widely used methods such as JPEG [17] and JPEG 2000 [15] follow the transform coding approach [11]. They use orthogonal linear transformations, such as the discrete cosine transform (DCT) [1] and the discrete wavelet transform (DWT) [2], to decorrelate small image blocks. Since these transforms map image data into a continuous domain, quantization is necessary before coding into bytes. However, as quantization errors can significantly degrade compression performance, the quantizers must be carefully crafted to minimize this impact, which further complicates the codec design.

Another promising paradigm riles on low-rank approximation methods [?] such as singular value decomposition (SVD). Current low-rank methods can represent the image data only with components containing floating point elements, which again necessitates a quantization layer prior to any byte-level processing. However, their compression is unfortunately suboptimal due to the higher sensitivity to quantization error compared to transform-based methods like JPEG, particularly at low bit rates.

2 Related Work

Transform-based compression.

Low-rank approximation for compression.

Integer Matrix Factorization.

3 Method

3.1 Overall Encoding Framework

Figure 1 illustrates an overview of the encoding pipeline for our proposed image compression method using 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 integer matrix factorization (IMF). Note that this patchification technique differs from the block splitting in JPEG, where each block is subject to discrete cosine transform (DCT) and processed independently. This patchification technique not only captures the locality and spatial dependencies of neighboring pixels but also performs better with the matrix decomposition approach to image compression. ← rephrase the sentence.

Low-rank approximation. We now apply a low-rank approximation to the matrices $\boldsymbol{X}_Y$, $\boldsymbol{X}_{C_B}$, and $\boldsymbol{X}_{C_R}$, which is the core of our compression method that provides a lossy compressed representation of these matrices. The low-rank approximation [9] aims to approximate a given matrix $\boldsymbol{X} \in \mathbb{R}^{M \times N}$ by

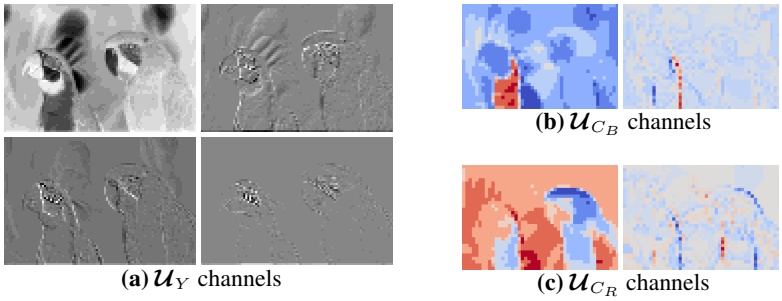
$$\boldsymbol{X} \approx \boldsymbol{U}\boldsymbol{V}^{\mathsf{T}} = \sum_{r=1}^{R} U_{:r} {V_{:r}}^{\mathsf{T}}, \tag{1}$$

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 ← make sure you cite some works here and/or in the intro. 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.

Reshape factors. IMF yields integers factor matrices $\boldsymbol{U}_Y \in \{0, \ldots, 255\}^{\frac{HW}{64} \times R}$ and $\boldsymbol{V}_Y \in \{0, \ldots, 255\}^{64 \times R}$; $\boldsymbol{U}_{C_B} \in \{0, \ldots, 255\}^{\frac{HW}{256} \times R}$ and $\boldsymbol{V}_{C_B} \in \{0, \ldots, 255\}^{64 \times R}$; and $\boldsymbol{U}_{C_R} \in \{0, \ldots, 255\}^{\frac{HW}{256} \times R}$ and $\boldsymbol{V}_{C_R} \in \{0, \ldots, 255\}^{64 \times R}$ that correspond to $\boldsymbol{X}_Y$, $\boldsymbol{X}_{C_B}$, and $\boldsymbol{X}_{C_R}$. We reshape these matrices by unfolding their first dimension to obtain R-channel 2D spatial maps, referred to as *factor maps* and represented by the following tensors:

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

Lossless image compression. Since *factor maps* are already integer tensors, we do not need a quantization step, which is commonly present in other lossy image compression methods and adds extra complications. This is the main advantage of our approach. Moreover, each channel of a *factor map* can be treated as an 8-bit grayscale image and therefore can be directly encoded by any standard lossless image compression method, such as PNG or WebP. 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



(a) $\mathcal{U}_Y$ channels (b) $\mathcal{U}_{C_B}$ channels (c) $\mathcal{U}_{C_R}$ channels

Figure 2 The first channels of IMF factor maps for the `kodim23` image from Kodak, corresponding to luma (Y), blue-difference (C_B), and red-difference (C_R) chroma.

the coefficient maps ($\mathcal{V}$), 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.

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 maps defined in (2) are produced by losslessly decompressing the encoded image. These maps are then transformed into factor matrices by flattening their spatial dimensions. The matrices $\boldsymbol{X}_Y$, $\boldsymbol{X}_{C_B}$, and $\boldsymbol{X}_{C_R}$ are calculated through the product of the corresponding factor matrices due 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 bilinear 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 values α and β, 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 the singular value decomposition (SVD), as addressed by the Eckart–Young–Mirsky theorem [9]. If only a nonnegativity constraint were applied (without integrality), variations of nonnegative matrix factorization (NMF) would emerge [13, 10]. The IMF problem (3) poses a challenging integer program, with finding its global minima known to be NP-hard [8, 16]. Only a few iterative algorithms [8, 14] 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). Conversely, our

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**
    $U_{:r}^{k+1} = \text{round}\left(\text{clamp}_{[\alpha,\beta]}\left(\frac{A_{:r} - \sum_{s=1}^{r-1} B_{sr}U_{:s}^{k+1} - \sum_{s=r+1}^{M} B_{sr}U_{:s}^{k}}{\|V_{:r}^k\|^2}\right)\right)$
  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**
    $V_{:r}^{k+1} = \text{round}\left(\text{clamp}_{[\alpha,\beta]}\left(\frac{A_{:r} - \sum_{s=1}^{r-1} B_{sr}V_{:s}^{k+1} - \sum_{s=r+1}^{N} B_{sr}V_{:s}^{k}}{\|U_{:r}^{k+1}\|^2}\right)\right)$
  end
end
return $(\boldsymbol{U}^k, \boldsymbol{V}^k)$

IMF formulation produces integer factor matrices that can be directly stored and losslessly processed without incurring roundoff errors. The reason for limiting the feasible region to $[\alpha, \beta]$ in our IMF formulation is to enable more compact storage of the factors using standard integral data types, such as `int8` and `int16`, supported by programming languages. Given that the elements of the input matrix $\boldsymbol{X}$ are in $[0, 255]$, we found the signed `int8` type, which represents integers from -128 to 127, suitable for image compression applications. As a result, our IMF formulation is well-suited for image compression, effectively integrating the factorization and quantization steps into a single, efficient compression process.

3.4 Block Coordinate Descent Scheme for IMF

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

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

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

where $\boldsymbol{u}_r \coloneqq U_{:r}$ and $\boldsymbol{v}_r \coloneqq V_{:r}$ represent the r-th columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. $\boldsymbol{E}_r \coloneqq \boldsymbol{X} - \sum_{s \neq r}^{R} \boldsymbol{u}_s\boldsymbol{v}_s^{\mathsf{T}}$ 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 (Integer rank-1 approximation)**.** *The global optima of subproblems* (4) *and* (5) *can be represented by closed-form solutions. Specifically, these subproblems can be replaced by the*

following:

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

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

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 will 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)*.*

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. $\tilde{\boldsymbol{U}} \in \mathbb{R}^{M\times R}$ contains the corresponding left-singular vectors in its columns, and $\tilde{\boldsymbol{V}} \in \mathbb{R}^{N\times R}$ contains the corresponding right-singular vectors in its columns. The initial factors are then calculated as follows:

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

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

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. Moreover, ablation studies are presented to investigate the effect of different parameters in IMF.

4.1 Qualitative Performance

The qualitative comparison is made in Figure 3 where the compressed images by different algorithms are shown in different bpp values. As can be seen, the proposed IMF compression algorithm is capable of maintaining quality compressed images in bpp values as low as 0.15, outperforming JPEG and SVD-based methods which suffer from maintaining balanced colors in low bpp values. The artifacts in color produced by JPEG are visible in both example images.

4.2 Rate-Distortion Performance

In this section, we plot the PSNR-bpp curves for each algorithm to compare their compression performance. The corresponding SSIM-bpp curves are postponed to the appendix. In the figures, for each compressed image, bpp and PSNR values are calculated. Then, for each compression method, PSNR values are interpolated linearly in some fixed bpp values in the range $(0.05, 0.5)$. In

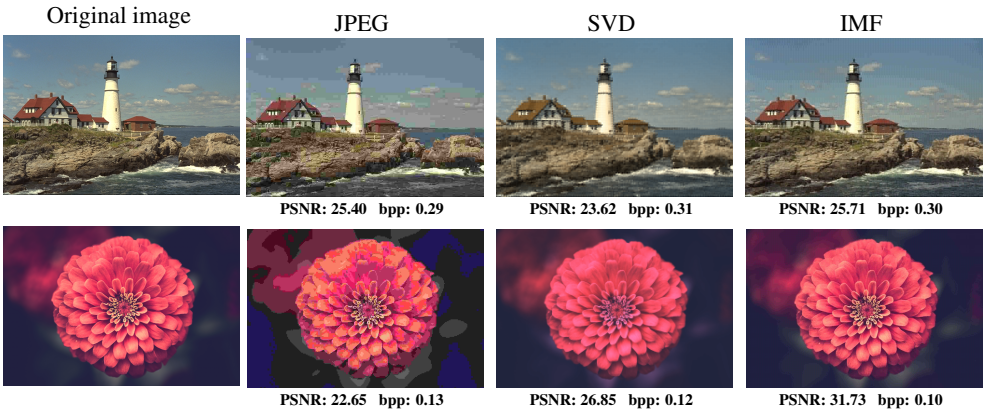
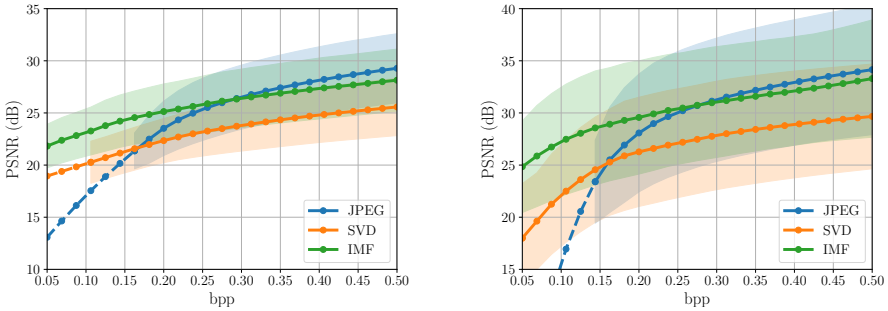


Figure 3 Qualitative performance comparison on an image from Kodak (top row) and CLIC (bottom row).



(a) performance on Kodak **(b)** performance on CLIC

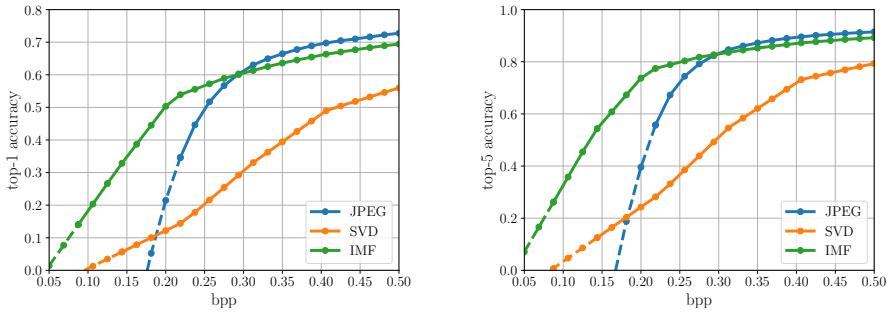
Figure 4 Performance comparison of JPEG, SVD, and IMF compression algorithms on Kodak and CLIC.

the following plots, the average performance over all images is reported, along with the standard deviation which is presented as shadows. For the missing bpp values, the average is extrapolated quadratically and is shown by dashed lines.

In Figure 4, the compression performance on the Kodak and CLIC datasets is reported. In this figure, the proposed IMF compression algorithm outperforms the SVD-based compression, which can be attributed to the quantization errors that SVD is prone to during encoding and decoding, deteriorating its performance. In this view, the quantization-free property of IMF effectively guarantees higher performance in different bpp values. It is also evident that the IMF compression algorithm outperforms JPEG in low bpp values.

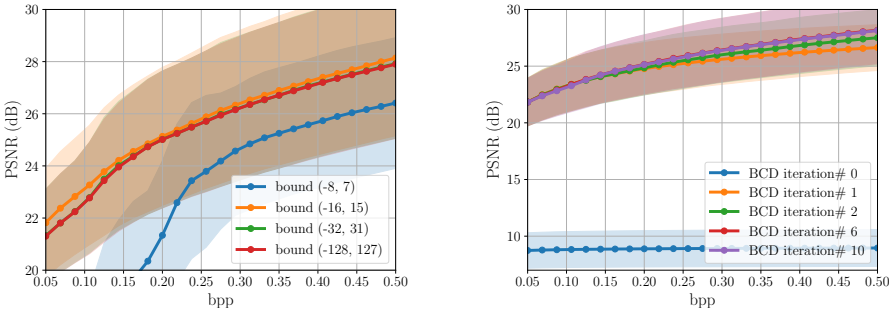
4.3 ImageNet Classification Performance

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 [**?**] validation set, consisting of 50000 224×224 RGB images in 1000 classes, is considered in this classification task done by a ResNet-50 classifier [**?**], 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 25% relative improvement in top-1 accuracy over JPEG at the low bitrate of 0.23 bpp and reaches top-5 accuracy of more than 70% at the bpp value of 0.2.



(a) top-1 accuracy **(b)** top-5 accuracy

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.



(a) ablation on factor bounds **(b)** ablation on BCD iteration number

Figure 6 Ablation studies for IMF. Average PSNR on the Kodak dataset is reported versus bpp.

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 and Future Work

Acknowledgments and Disclosure of Funding

References

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

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

[3] 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.

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

[5] Alex Clark. Pillow (pil fork) documentation, 2015. URL https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

[6] 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.

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

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

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

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

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

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

[13] 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.

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

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

[16] 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.

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

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:

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

The objective functions 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 $\mathbb{Z}_{[\alpha,\beta]}$. Specifically, following simple calculus, for u_i^r the solution is $u_i^{r^\star} = \mathrm{round}(\mathrm{clamp}_{[\alpha,\beta]}(\sum_{j=1}^N e_{ij}^r v_j^r / \sum_{j=1}^N v_j^{r\,2}))$, which is presented in a compact form in (6).

A.2 Proof of Theorem 2

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

$$\begin{aligned} \underset{\boldsymbol{U}\in\mathbb{R}^{M\times R},\boldsymbol{V}\in\mathbb{R}^{N\times R}}{\text{minimize}} \quad & \Psi(\boldsymbol{U},\boldsymbol{V}) \coloneqq H(\boldsymbol{U},\boldsymbol{V}) + f(\boldsymbol{U}) + g(\boldsymbol{V}), \\ \text{where} \quad & H(\boldsymbol{U},\boldsymbol{V}) \coloneqq \|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}\|_{\mathrm{F}}^2, \\ & f(\boldsymbol{U}) \coloneqq \delta_{[a,b]}(\boldsymbol{U}) + \delta_{\mathbb{Z}}(\boldsymbol{U}), \\ & g(\boldsymbol{V}) \coloneqq \delta_{[a,b]}(\boldsymbol{V}) + \delta_{\mathbb{Z}}(\boldsymbol{V}), \end{aligned} \tag{12}$$

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 (12) are equivalent.

The unconstrained optimization problem (12) consists of the sum of a differentiable (smooth), convex function H with nonsmooth, nonconvex functions f and g. 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 [6, 3]. In Algorithm 1, the variables $\boldsymbol{U}$ and $\boldsymbol{V}$ are updated sequentially following block coordinate (BC) descent minimization algorithms, also often called Gauss-Seidel updates or alternating minimization [**?** **?**]. Hence, in this convergence study, we are interested in algorithms that allow BC-type updates for the nonconvex nonsmooth problem of (12) [**?** 4]. Specifically, we focus on the proximal alternating linearized minimization (PALM) algorithm [4], 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 (12), 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 $\boldsymbol{U}^{\mathrm{init}} \in \mathbb{R}^{M\times R}$, $\boldsymbol{V}^{\mathrm{init}} \in \mathbb{R}^{N\times R}$
2. For each iteration $k = 0, 1, \dots$

$$\begin{aligned} &(a) \quad \boldsymbol{U}^{k+1} \in \mathrm{prox}_{c_k}^{f}\left(\boldsymbol{U}^k - \frac{1}{c_k}\nabla_{\boldsymbol{U}} H(\boldsymbol{U}^k,\boldsymbol{V}^k)\right), \text{ with } c_k > L_1(\boldsymbol{V}^k) \\ &(b) \quad \boldsymbol{V}^{k+1} \in \mathrm{prox}_{d_k}^{g}\left(\boldsymbol{V}^k - \frac{1}{d_k}\nabla_{\boldsymbol{V}} H(\boldsymbol{U}^{k+1},\boldsymbol{V}^k)\right), \text{ with } d_k > L_2(\boldsymbol{U}^{k+1}) \end{aligned} \tag{13}$$

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 $\operatorname{prox}_\gamma^\varphi(\boldsymbol{x}) := \arg\min_{\boldsymbol{w}\in\mathbb{R}^n} \left\{\varphi(\boldsymbol{w}) + \frac{\gamma}{2}\|\boldsymbol{w}-\boldsymbol{x}\|_2^2\right\}$, ∇_x is the partial derivative with respect to x, and $L_1 > 0$, $L_2 > 0$ are local Lipschitz moduli, defined in the following proposition. It is also remarked that the iterates in (13) 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 (13).

The following proposition investigates the necessary assumptions (cf. [4, Asm. 1 and Asm. 2]) for convergence iterates in (13).

Proposition 1 (Meeting required assumptions)**.** *The assumptions necessary for the convergence of iterates in* (13) *are satisfied by the functions involved in the problem* (12)*, 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 $M > 0$:*

$$\begin{aligned}\|(\nabla_{\boldsymbol{U}} H(\boldsymbol{U}_1,\boldsymbol{V}_1) - \nabla_{\boldsymbol{U}} H(\boldsymbol{U}_2,\boldsymbol{V}_2), \nabla_{\boldsymbol{V}} H(\boldsymbol{U}_1,\boldsymbol{V}_1) - \nabla_{\boldsymbol{V}} H(\boldsymbol{U}_2,\boldsymbol{V}_2))\| \\ \le M\|(\boldsymbol{U}_1 - \boldsymbol{U}_2, \boldsymbol{V}_1 - \boldsymbol{V}_2)\|,\end{aligned} \tag{14}$$

   where $(\cdot,\cdot)$ denotes the concatination 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* (12)*, hence Ψ is a Kurdyka-Łojasiewicz (KL) function.*

By Proposition 1, the optimization problem (12) can be solved by the BC iterates in (13), due to the following proposition:

Proposition 2 (Global convergence [4])**.** *With the assumptions in proposition 1 being met by the problem* (12)*, let $((\boldsymbol{U}^k,\boldsymbol{V}^k))_{k\in\mathbb{N}}$ be a sequence generated by the BC iterates in* (13)*. Then the sequence converges to a critical point $(\boldsymbol{U}^\star,\boldsymbol{V}^\star)$ of the problem* (12)*, 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 (13) 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 $\boldsymbol{E}_r^{\mathrm{T}}\boldsymbol{u}_r/\|\boldsymbol{u}_r\|^2$ presented in (6) and (7), respectively (in the case where the iterates (13) 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 $\operatorname{prox}_{c_k}^f$ and $\operatorname{prox}_{d_k}^g$ can efficiently be implemented by the operators round and $\operatorname{clamp}_{[\alpha,\beta]}$ 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* $\text{prox}^f_{c_k}(\boldsymbol{W}) = \text{round}(\text{clamp}_{[\alpha,\beta]}(\boldsymbol{W}))$ *and* $\text{prox}^f_{d_k}(\boldsymbol{Z}) = \text{round}(\text{clamp}_{[\alpha,\beta]}(\boldsymbol{Z}))$ *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 $\boldsymbol{W} \in \mathbb{R}^{M\times R}$:

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

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

$$\text{round}(\boldsymbol{W}) = \underset{\boldsymbol{U}\in\mathbb{Z}^{M\times R}}{\arg\min}\{\|\boldsymbol{U}-\boldsymbol{W}\|_F^2\}. \tag{15}$$

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

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

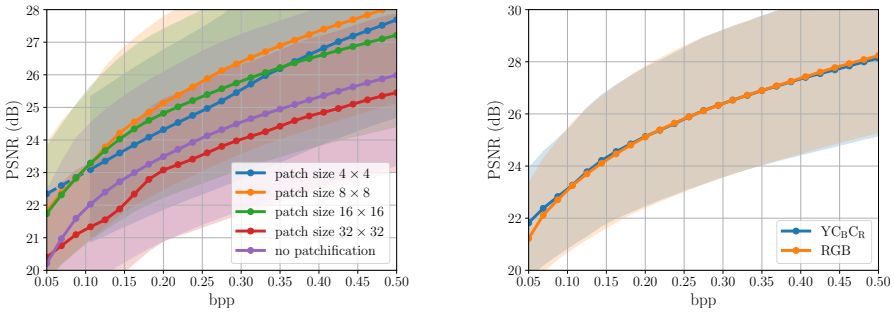
The first equality is due to the definition of prox which is equivalent to the second equality. In the third equality the matrices $\boldsymbol{A} \in \mathbb{R}^{M\times R}$ and $\boldsymbol{B} \in \mathbb{R}^{M\times R}$ have elements all equal to a and b, respectively. The third equality is due to 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 (15) is invoked. A similar proof can be trivially followed for $\text{prox}^f_{d_k}(\boldsymbol{Z}) = \text{round}(\text{clamp}_{[\alpha,\beta]}(\boldsymbol{Z}))$ as well. □

Now that the equivalence of iterates (13) 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 (12) and (3), proposition 2 can be travially 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 [5]. ← factor bounds are missing.

Evaluation. For comparison, we experimented with the widely-used Kodak dataset [12], 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 [**?**], 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.



(a) ablation on patch size **(b)** ablation on color space

Figure 7 Ablation studies for IMF. Average PSNR on the Kodak dataset is reported versus bpp.

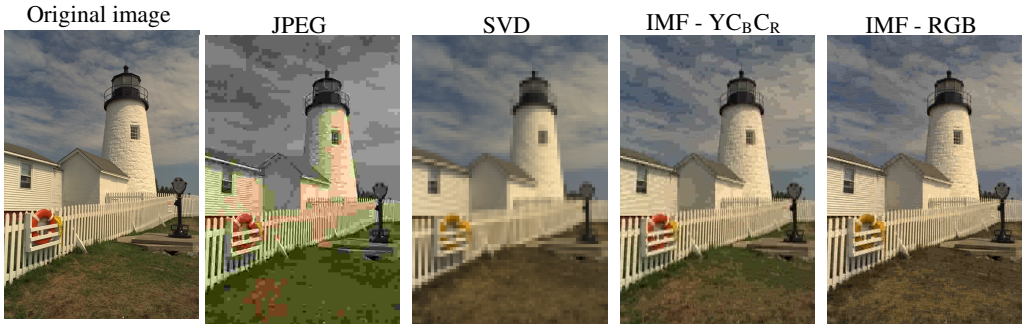


Figure 8 Qualitative performance comparison on an image from Kodak.

Baseline Codecs. For JPEG compression, we employed the Pillow library [5]. 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 [7]. This differs from the IMF algorithm where factors are first reshaped into factor maps and then compressed losslessly as images using WebP.

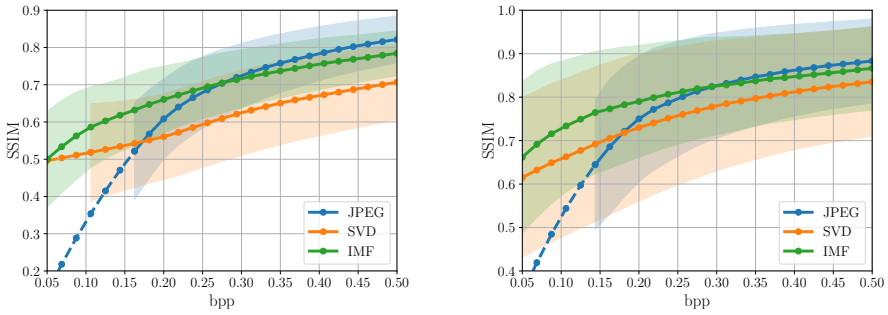
C More Ablations Studies

Patchification. In Figure 7a, 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 7a.

Color space. The compression performance of IMF is studied with two color spaces, namely RGB and YC_BC_R, in Figure 7b. Although the compression performance remains unchanged in terms of PSNR, qualitative results reported in Figure 8 indicate that YC_BC_R color space can maintain natural colors more effectively.

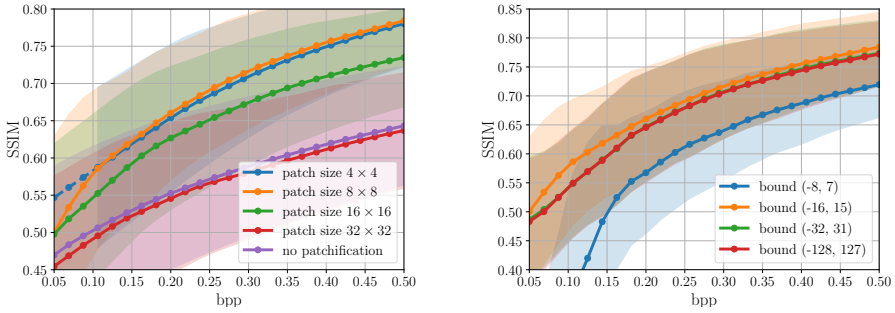
D Structural Similarity Index Measure (SSIM) Metrics

In Figures 9 and 10, average SSIM values for the images of the Kodak and CLIC datasets are presented.

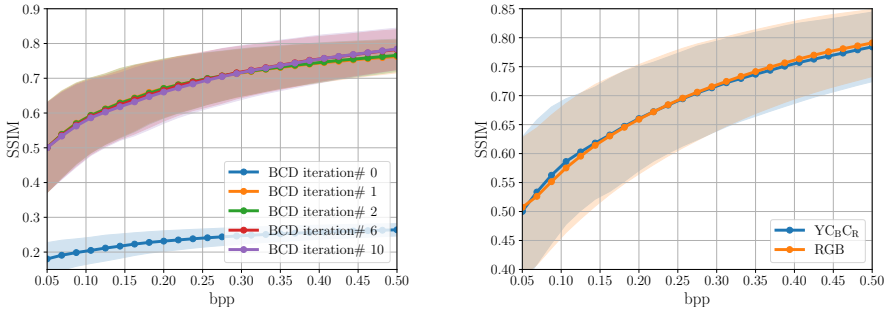


(a) performance on Kodak **(b)** performance on CLIC

Figure 9 Performance comparison of JPEG, SVD, and IMF compression algorithms on Kodak and CLIC.



(a) ablation on patch size **(b)** ablation on factor bounds



(c) ablation on BCD iteration number **(d)** ablation on color space

Figure 10 Ablation studies for IMF. Average SSIM on the Kodak dataset is reported versus bpp.