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.

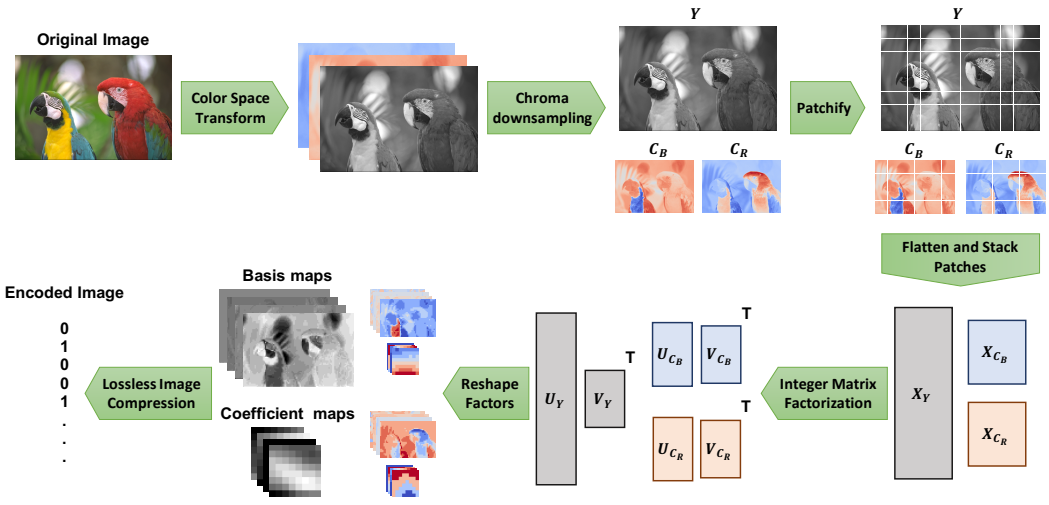


Figure 1 An illustration of the encoder for our image compression method, based on integer matrix factorization.

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

Widely used methods such as JPEG [15] and JPEG 2000 [13] follow the transform coding approach [9]. 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.

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 [7] 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*), and $R \leq \min(M, N)$ represents the *rank*. 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. 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 a 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 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.

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 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. 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. 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 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 $[\alpha, \beta]$. 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 [7]. If only a nonnegativity constraint were applied (without integrality), variations of nonnegative matrix factorization (NMF) would emerge [11, 8]. The IMF problem (3) poses a challenging integer program, with finding its global minima known to be NP-hard [6, 14]. Only a few iterative algorithms [6, 12] 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 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). 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 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:

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

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

where $\boldsymbol{u}_r = U_{:r}$ and $\boldsymbol{v}_r = V_{:r}$ represent the r-th columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. $\boldsymbol{E}_r = \boldsymbol{X} - \sum_{s \neq r}^R \boldsymbol{u}_s \boldsymbol{v}_s^\mathsf{T}$ 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)**.** *Let $\boldsymbol{E} \in \mathbb{R}^{M \times N}$, $\boldsymbol{u} \in \mathbb{Z}^{M \times 1}_{[\alpha,\beta]}$, and $\boldsymbol{v} \in \mathbb{R}^{N \times 1}$. A global solution to the problem*

$$\boldsymbol{u}^* \in \underset{\boldsymbol{u} \in \mathbb{Z}^{M \times 1}_{[\alpha,\beta]}}{\arg\min} \|\boldsymbol{E} - \boldsymbol{u}\boldsymbol{v}^\mathsf{T}\|_F^2 \tag{6}$$

is given by

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

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

Proof. See Appendix A for the proof. □

Using Theorem 1, we obtain the following update formulas for our BCD algorithm: where $\boldsymbol{u}_r = U_{:r}$ and $\boldsymbol{v}_r = V_{:r}$ represent the r-th columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. $\boldsymbol{E}_r = \boldsymbol{X} - \sum_{s \neq r}^R \boldsymbol{u}_s \boldsymbol{v}_s^\mathsf{T}$ 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. To do so, we replace subproblems (4) and (5) by the following closed-form solutions:

$$\boldsymbol{u}_r \leftarrow \mathrm{round}\left(\mathrm{clamp}_{[\alpha,\beta]}\left(\frac{\boldsymbol{E}_r \boldsymbol{v}_r}{\|\boldsymbol{v}_r\|^2}\right)\right), \tag{8}$$

$$\boldsymbol{v}_r \leftarrow \mathrm{round}\left(\mathrm{clamp}_{[\alpha,\beta]}\left(\frac{\boldsymbol{E}_r^\mathsf{T} \boldsymbol{u}_r}{\|\boldsymbol{u}_r\|^2}\right)\right). \tag{9}$$

It is noteworthy that the combination of $\mathrm{round}(\cdot)$ and $\mathrm{clamp}_{[\alpha,\beta]}(\cdot)$ 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.

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:

$$\boldsymbol{U}^0 = \mathrm{round}(\mathrm{clamp}_{[\alpha,\beta]}(\tilde{\boldsymbol{U}}\boldsymbol{\Sigma}^{\frac{1}{2}})), \tag{10}$$

$$\boldsymbol{V}^0 = \mathrm{round}(\mathrm{clamp}_{[\alpha,\beta]}(\boldsymbol{\Sigma}^{\frac{1}{2}}\tilde{\boldsymbol{V}})). \tag{11}$$

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 $\boldsymbol{U}, \boldsymbol{V}$ using the truncated SVD method, provided by (10) and (11)
while *stopping criterion not satisfied* **do**
  $\boldsymbol{A} \leftarrow \boldsymbol{X}\boldsymbol{V}$, $\boldsymbol{B} \leftarrow \boldsymbol{V}^\top \boldsymbol{V}$
  for $r = 1, \ldots, R$ **do**
    $U_{:r} \leftarrow \text{round}\left(\text{clamp}_{[\alpha,\beta]}\left(\frac{A_{:r} - \sum_{s \neq r} B_{sr} U_{:s}}{\|V_{:r}\|^2}\right)\right)$
  end
  $\boldsymbol{A} \leftarrow \boldsymbol{X}^\top \boldsymbol{U}$; $\boldsymbol{B} \leftarrow \boldsymbol{U}^\top \boldsymbol{U}$
  for $r = 1, \ldots, R$ **do**
    $V_{:r} \leftarrow \text{round}\left(\text{clamp}_{[\alpha,\beta]}\left(\frac{A_{:r} - \sum_{s \neq r} B_{sr} V_{:s}}{\|U_{:r}\|^2}\right)\right)$
  end
end
return $(\boldsymbol{U}, \boldsymbol{V})$

Essentially, this means we first low-rank approximate $\boldsymbol{X}$ and then project the elements of the resulting factor matrices into $\mathbb{Z}_{[\alpha,\beta]}$. Algorithm 1 provides the pseudocode for the proposed BCD algorithm for IMF.

The following theorem establishes the convergence of the proposed Algorithm 1.

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 for the proof. □

3.5 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 [4].

Evaluation. For comparison, we experimented with the widely-used Kodak dataset [10], 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 PSNR and 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 [4]. 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 [5]. This differs from the IMF algorithm where factors are first reshaped into factor maps and then compressed losslessly as images using WebP.

4 Experiments

In this section, the proposed IMF-based compression method is assessed against SVD-based and JPEG [] methods. The performance is reported qualitatively and also based on two criteria, namely

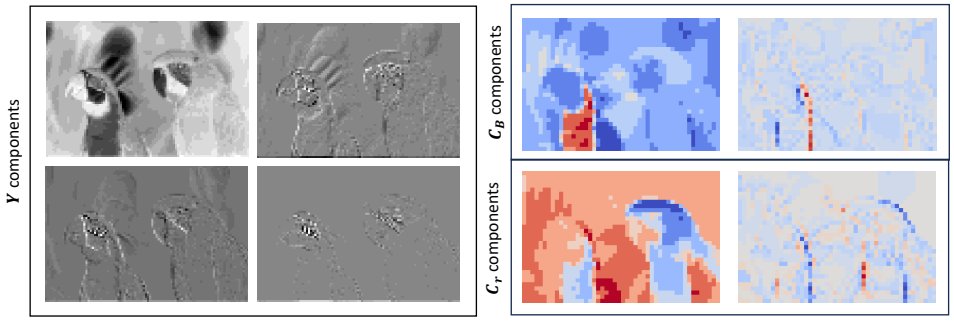


Figure 2 IMF components of the `kodim23` image from the Kodak dataset, corresponding to luma (Y), blue-difference (Cb), and red-difference (Cr) chroma.

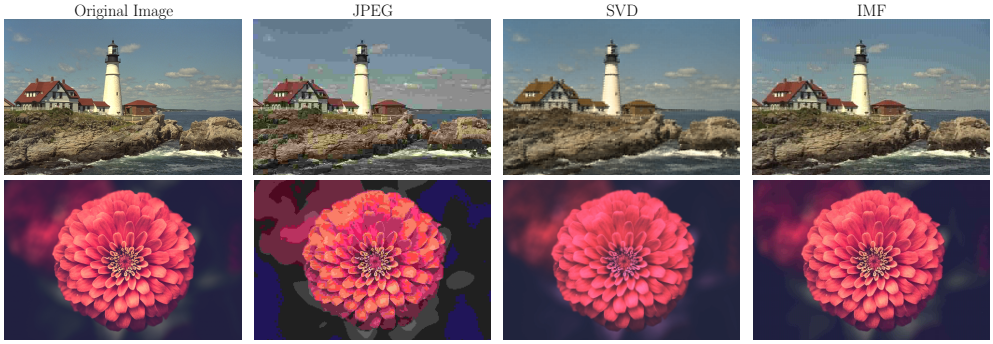


Figure 3 Qualitative performance comparison between various compression methods. The top row shows performance on kodim21 from the Kodak dataset in the bpp value of 0.3, and the bottom row shows one example of the Clic dataset in the bpp values of 0.14, 0.12, and 0.1 for JPEG, SVD, and IMF, respectively.

rate-distortion performance and image classification performance. Moreover, ablation studies are presented to investigate the effect of different hyperparameters in IMF.

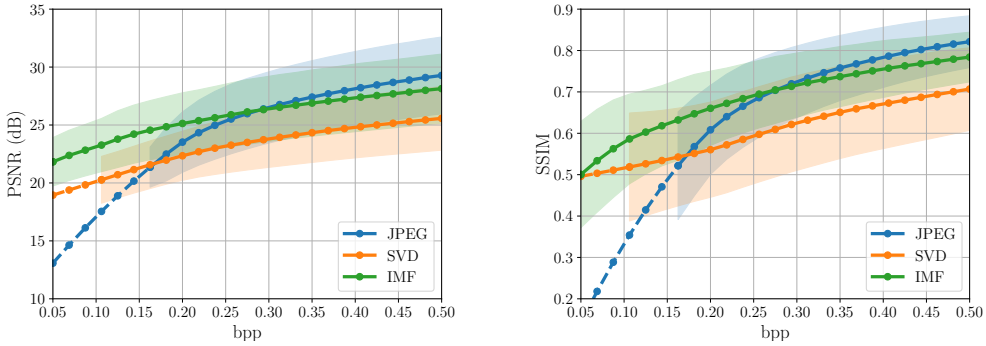
4.1 Qualitative Performance

Qualitative performance is shown on an image selected from the Kodak [] dataset. Fig.2 depicts the first IMF components U_Y, U_{Cb}, and U_{Cr} which are extracted following the procedure elaborated in Section 3.5. It is evident in the figure that the IMF components with higher energy maintain the overall texture of the image in each channel, while components with lower energy focus more on subtle changes.

The qualitative comparison is made in Fig.**??** on an image selected from the Kodak [] dataset. The images compressed by the considered compression methods are shown in different bits per pixel (bpp) values, a standard compression measure in the literature []. As can be seen, the IMF-based compression method is capable of maintaining quality compressed images in bpp values as low as ?, outperforming JPEG and SVD-based methods which suffer from maintaining balanced colors as soon as bpp drops below ? and ?, respectively. The artifacts in color are visible, e.g. by JPEG in bpp values starting ?.

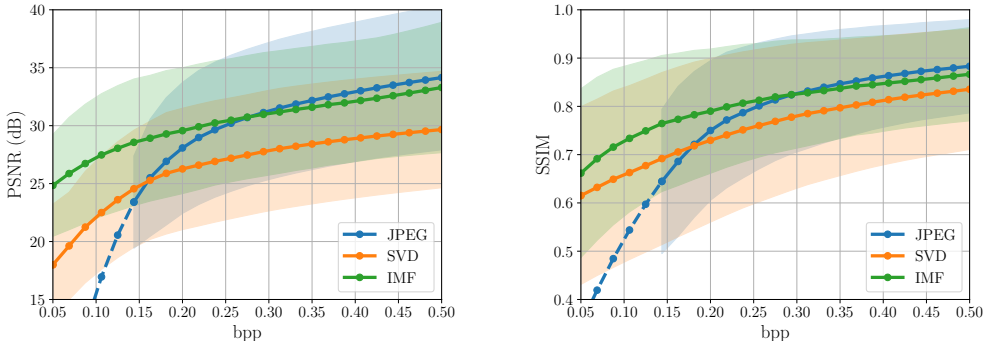
4.2 Rate-Distortion Performance

Peak signal-to-noise ratio (PSNR), as well as structural similarity index measure (SSIM) [], are reported versus bpp for the considered methods. The considered datasets are Clic [] and Kodak, consisting of respectively 32 and 24 colored images of various sizes. For each image compressed by any of the considered methods, bpp, PSNR, and SSIM are calculated. Then, for each compression method, PSNR and SSIM values are interpolated linearly in the fixed bpp values in the range $(0.05, 0.5)$. In the following plots, the average performance over all images is reported, along with



(a) **(b)**

Figure 4 Rate-distortion performance on the Kodak dataset. In panels (a) and (b), the average PSNR and SSIM are plotted against bpp, respectively.



(a) **(b)**

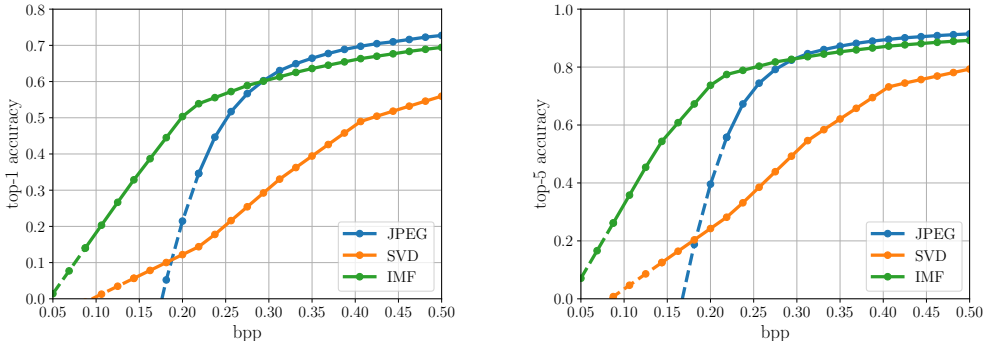
Figure 5 Rate-distortion performance on the CLIC dataset. In panels (a) and (b), the average PSNR and SSIM are plotted against bpp, respectively.

the standard deviation presented as shadows. For the missing bpp values, the average is extrapolated quadratically and is shown by dashed lines.

In Fig.4, the compression performance on the Kodak dataset is reported. In this figure, the proposed IMF-based method outperforms the SVD-based method, which can be attributed to the quantization errors that SVD is prone to during encoding and decoding, deteriorating its performance in both criteria. In this view, the quantization-free property of IMF effectively guarantees higher performance in different bpp values. It is also evident that the IMF-based method outperforms JPEG in low bpp values. The same performance regime can also be concluded for all the mentioned compression methods on the Clic dataset in Fig.5.

4.3 ImageNet Classification Performance

As another criterion, classification performance is investigated on the images compressed by the considered compression methods. This criterion focuses on the higher-level information required for object recognition and classification embedded in each image. Furthermore, it highlights the importance of image compression where various vision tasks such as classification are the main objective—rather than maintaining the perceived image quality—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 colored images in 1000 classes, is considered for this classification task done by a ResNet-50 classifier [], pre-trained on the original ImageNet dataset. The classification performance comparison is made in Fig.6, showing



(a) **(b)**

Figure 6 Impact of different compression methods on ImageNet classification accuracy. Panels (a) and (b) show the validation top-1 and top-5 accuracy plotted against bits per pixel (bpp), respectively. A ResNet-50 model pre-trained on the original ImageNet images is evaluated using validation images compressed by different methods.

the higher compression performance of IMF for classification, reaching top-1 accuracy of 70% in bpp values as low as 0.2.

4.4 Ablation Studies

In this section, ablation studies are performed, focusing on various hyper-parameters in the IMF-based compression method and their effect on the compression performance.

Patchification. In Fig.7a, patchification effect with different patch sizes is investigated. First, it can be concluded that patchification has a positive effect on performance. This performance boost is mainly due to the fact that in each patch (local) pattern variation is limited and hence IMF has more representation power to reconstruct the original pattern in each patch with fewer components. 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 Fig.7a.

Factor bounds. As elaborated in Section 3.5, during the IMF optimization, IMF components are constrained into a bound $(-\alpha, \alpha - 1)$. Fig.7b studies the compression performance versus different bounds. According to the results, the bound $(-16, 15)$ leads to the best performance since the value distribution of IMF components lies mostly in this bound. Hence, dedicating fewer bits to represent this narrower bound compared to the other bounds results in higher compression rates without sacrificing performance.

BCD iteration. The next parameter to study is the inner iteration number required in IMF BCD updates. According to the numerical results on various datasets, the objective value in the IMF optimization drops drastically after 2 iterations, while more iteration numbers have marginal improvement. This observation is shown for the Kodak dataset in Fig.7c. This feature makes the IMF computationally efficient since with a limited number of iterations a high compression performance can be achieved.

Color space. The compression performance of IMF is studied with two color spaces, namely RGB and YCbCr, in Fig.7d. Although the compression performance remains unchanged in terms of PSNR, qualitative results reported in Fig.? indicate that YCbCr color space can maintain natural colors and brightness more effectively. Consistently, the JPEG method employs this color space as well.

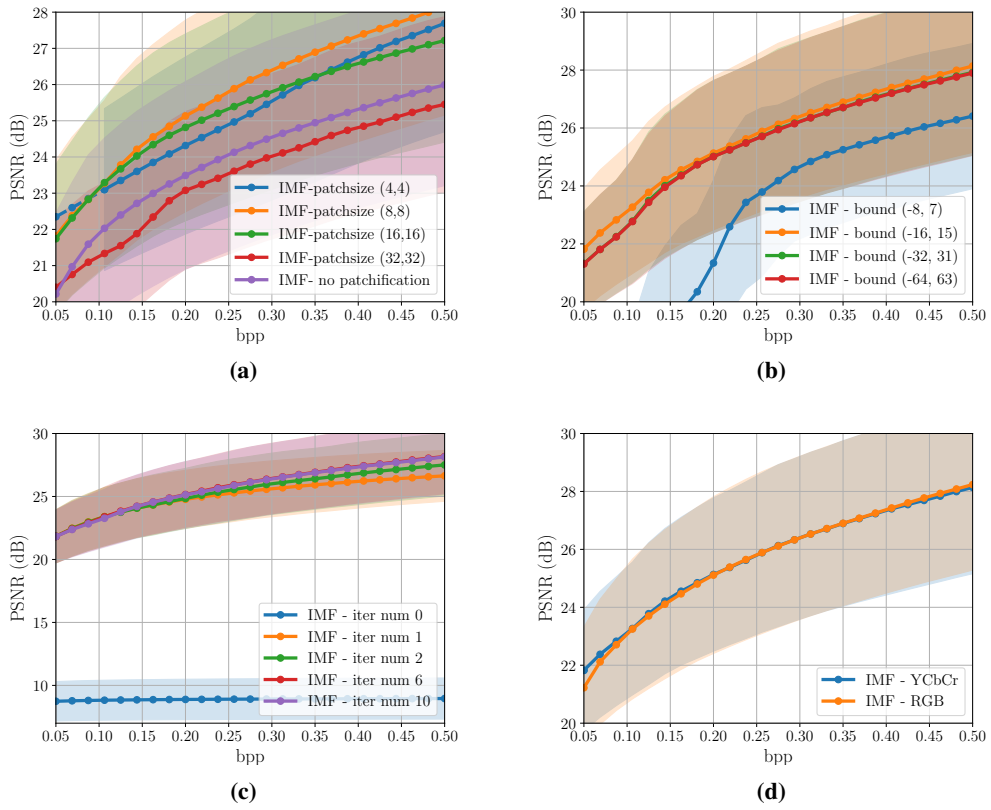


Figure 7 Ablation experiments for the IMF compression method. In all cases, we plot PSNR as a function of bits per pixel (bpp) on the Kodak dataset. (a) Compares IMF compression performance without patchification and different patch sizes. (b) Compares IMF compression performance for different bound values of factor matrices. (c) Compares IMF compression performance for different numbers of BCD iterations. (d) Compares IMF compression performance between RGB and YCbCr color space transform.

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

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

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

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

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

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

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

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

[11] 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.

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

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

[14] 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.

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

A Proof of Theorem 2

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

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

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 unconstrained nonconvex nonsmooth minimization problems. One of the common algorithms applied to such a problem class is the well-known forward-backward-splitting (FBS) algorithm []. 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) []. Specifically, we focus on the proximal alternating linearized minimization (PALM) algorithm [3], 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 $\boldsymbol{U}^0 \in \mathbb{R}^{M\times R}$, $\boldsymbol{V}^0 \in \mathbb{R}^{N\times R}$
2. For each iteration $k = 0, 1, \ldots$

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

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 $\operatorname{prox}_{\gamma}^{\varphi}(\boldsymbol{x}) \coloneqq \arg\min_{\boldsymbol{w}\in\mathbb{R}^n}\left\{\varphi(\boldsymbol{w}) + \frac{\gamma}{2}\|\boldsymbol{w}-\boldsymbol{x}\|_2^2\right\}$, ∇_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. [3, 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*

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

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 $M > 0$:*

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

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 [3])**.** *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 Algorithm 1. 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 $\boldsymbol{E}_r^{\mathsf{T}} \boldsymbol{u}_r / \|\boldsymbol{u}_r\|^2$ presented in (8) and (9), 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 $\mathrm{prox}_{c_k}^f$ and $\mathrm{prox}_{d_k}^g$ can efficiently be implemented by the operators round and $\mathrm{clamp}_{[\alpha,\beta]}$ in (8) and (9). The equivalence of these steps is proven in the following lemma.

Lemma 1 (prox implementation)**.** *Consider the operators* round *and* $\mathrm{clamp}_{[\alpha,\beta]}$ *defined in (8) and (9). Then* $\mathrm{prox}_{c_k}^f(\boldsymbol{W}) = \mathrm{round}(\mathrm{clamp}_{[\alpha,\beta]}(\boldsymbol{W}))$ *and* $\mathrm{prox}_{d_k}^f(\boldsymbol{Z}) = \mathrm{round}(\mathrm{clamp}_{[\alpha,\beta]}(\boldsymbol{Z}))$ *for any $\boldsymbol{W} \in \mathbb{R}^{M \times R}$, $\boldsymbol{Z} \in \mathbb{R}^{N \times R}$, and* $\mathrm{round}(\mathrm{clamp}_{[\alpha,\beta]}(\cdot))$ *being an elementwise operator on the input matrices.*

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

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

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

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

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 to be met by problems (12) and (3) in proposition 1, proposition 2 can be travially invoked to establish the convergence of Algorithm 1 to a locally optimal point of problem (3). □