Information Sciences 722 (2026) 122646

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Quantization-aware matrix factorization for low bit rate image compression

Pooya Ashtari [a,*,1], Pourya Behmandpoor [a,1], Fateme Nateghi Haredasht [b], Jonathan H. Chen [b], Panagiotis Patrinos [a], Sabine Van Huffel [a]

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

ARTICLE INFO

Keywords:
Matrix factorization
Low-rank approximation
Quantization
Image compression

ABSTRACT

Lossy image compression is essential for efficient transmission and storage. Traditional compression methods mainly rely on discrete cosine transform (DCT) or singular value decomposition (SVD), both of which represent image data in continuous domains and, therefore, necessitate carefully designed quantizers. Notably, these methods consider quantization as a separate step, which prevents quantization errors from being incorporated into the compression process and degrades the reconstruction quality, particularly in SVD-based methods. To address this issue, we introduce a quantization-aware matrix factorization (QMF) to develop a novel lossy image compression method. QMF provides a low-rank representation of the image data as a product of two smaller matrices, with elements constrained to bounded integer values, thereby effectively integrating quantization with low-rank approximation. We propose an efficient, provably convergent iterative algorithm for QMF using a block coordinate descent scheme, with subproblems having closed-form solutions. Our experiments demonstrate that our method consistently outperforms JPEG at low bit rates below 0.25 bits per pixel. We also demonstrated that our method has an improved capability to preserve visual semantics compared to JPEG at low bit rates by evaluating an ImageNet pre-trained classifier on compressed images. The project is available at https://github.com/pashtari/qmf.

1. Introduction

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

* Corresponding author.
E-mail address: pooya.ashtari@esat.kuleuven.be (P. Ashtari).
[1] Pooya Ashtari and Pourya Behmandpoor contributed equally to this work.

https://doi.org/10.1016/j.ins.2025.122646
Received 27 January 2025; Received in revised form 25 August 2025; Accepted 26 August 2025

Available online 5 September 2025
0020-0255/© 2025 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

quantization errors can significantly degrade compression performance, the quantizers must be carefully crafted to minimize this impact, which further complicates codec design.

Another promising paradigm relies on low-rank approximation techniques, with singular value decomposition (SVD) being a notable example. SVD is recognized as the deterministically optimal transform for energy compaction [6]. In practice, current SVD-based methods [6–8] can represent image data only with factors that contain floating-point elements, necessitating a quantization step prior to any byte-level processing. The quantization step often introduces errors, which result in suboptimal compression performance.

Motivated by this, we introduce quantization-aware matrix factorization (QMF) and, based on it, develop an effective lossy image compression method. Unlike traditional compression methods, the proposed approach integrates quantization into the optimization process rather than treating it as a separate step before byte-level processing. Our QMF formulation provides a low-rank representation of image data as the product of two smaller factor matrices. The quantization is integrated via introducing constraints in the optimization process, where the elements of the factor matrices are constrained to *bounded integer* values. These elements, with discrete values represented as bounded integers, can be directly stored using standard integral data types—such as `int8` and `int16` supported by programming languages—and losslessly processed, making QMF arguably better suited than SVD for image compression. Another advantage of QMF is that the reshaped factor matrices can be treated as 8-bit grayscale images, allowing any lossless image compression standard to be seamlessly integrated into the proposed framework. We propose an efficient iterative algorithm for QMF using a block coordinate descent (BCD) scheme, where each column of a factor matrix is taken as a block and updated one at a time using a closed-form solution.

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

2. Related work

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

Learned image compression (LIC) Recently, learned image compression (LIC) has gained attention for potentially outperforming traditional methods by leveraging deep neural networks. Ballé et al. [13] pioneered this area with an end-to-end trainable convolutional neural network based on variational autoencoders. Cheng et al. [14] incorporated a simplified attention module and discretized Gaussian mixture likelihoods for achieving a more accurate and flexible entropy model. Liu et al. [15] combined transformers and CNNs to exploit the local modeling ability of convolutions and the global modeling ability of the attention mechanism. Yang and Mandt [16] introduced diffusion models into LIC, using a denoising decoder to iteratively reconstruct a compressed image. Unlike LIC methods that require training on large datasets of images, our QMF is a single-image compression method that optimizes specifically for each input image. While both QMF and some LIC methods integrate quantization into their core algorithms rather than treating it as a separate step, LIC methods involve complex deep neural networks for encoding and decoding, resulting in significantly higher computational requirements. The proposed QMF method offers a more lightweight alternative with much faster encoding and decoding times, making it particularly suitable for resource-constrained environments where deep learning-based approaches may be impractical.

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

Integer matrix factorization There are applications where meaningful representation of data as discrete factor matrices is crucial. While typical low-rank techniques like SVD and nonnegative matrix factorization (NMF) are inappropriate for such applications, integer matrix factorization (IMF) ensures the integrality of factors to achieve this goal. Lin et al. [18] investigates IMF to effectively handle discrete data matrices for cluster analysis and pattern discovery. Dong et al. [19] introduce an alternative least squares method for IMF, verifying its effectiveness with some data mining applications. However, the application of IMF in image compression remains unexplored.

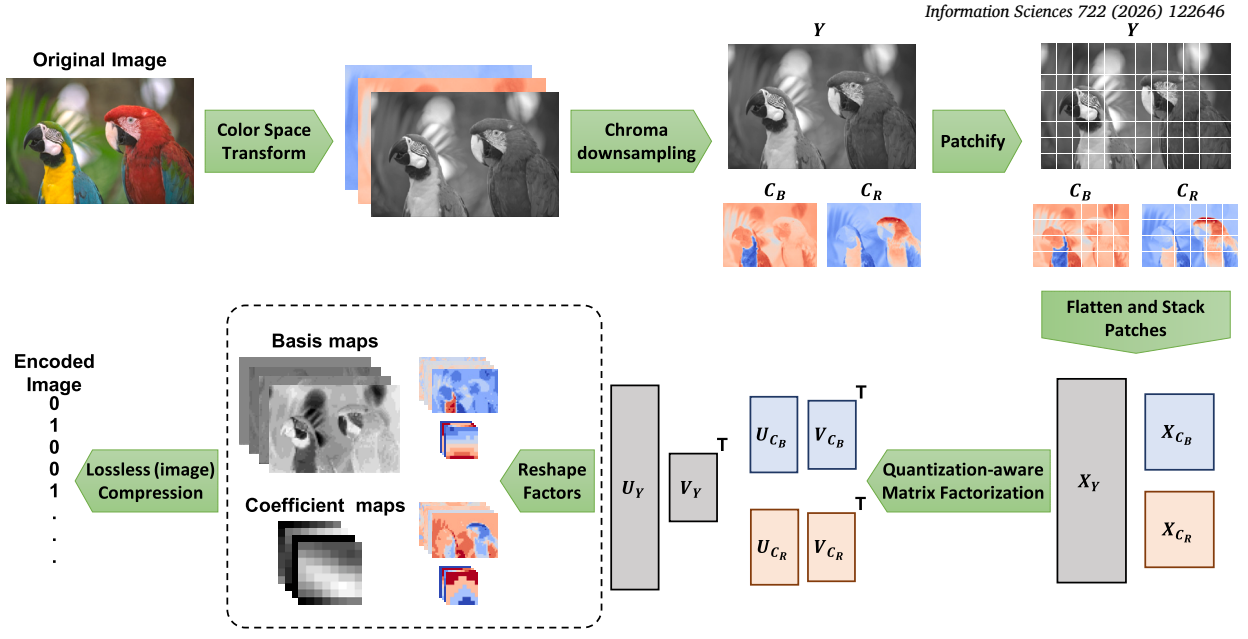


Fig. 1. An illustration of the encoder for our image compression method.

While existing IMF methods generally constrain factor elements to the entire set of integer values, we propose quantized matrix factorization (QMF), which minimizes the objective function over a bounded interval of integers, thereby modeling a uniformly quantized domain. Furthermore, we introduce a block coordinate descent (BCD)-based algorithm to solve the QMF problem, which is both computationally efficient and provably convergent. This work investigates the potential of QMF for image compression, arguing that it can serve as a powerful tool for this purpose. It is noted that while [18] and [19] propose iterative algorithms for integer matrix factorization, their settings differ from ours. Lin et al. [18] allow unbounded or binary-unbounded factors, which are not compatible with compression pipelines. Dong et al. [19] solve an integer matrix approximation problem via complex subroutines involving constrained least-squares and enumeration, making it computationally intensive for image compression tasks.

3. Method

3.1. Overall encoding framework

The proposed compression method follows a *transform coding* paradigm, but it does not involve a separate quantization step. Fig. 1 illustrates an overview of our encoding pipeline based on quantization-aware matrix factorization (QMF). 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 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 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.

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 QMF. Note that this patchification technique differs from the block splitting in JPEG, where each block is subject to DCT individually and processed independently. This patchification technique help capture the locality and spatial dependencies of neighboring pixels.

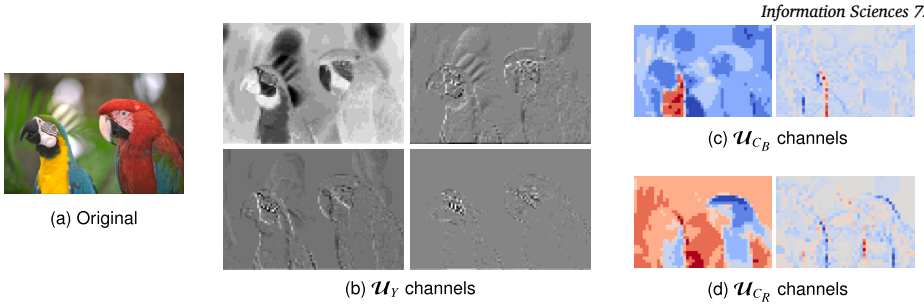


Fig. 2. The channels of QMF basis maps for the `kodim23` image from Kodak. (a) shows the original image. The QMF basis maps corresponding to luma (b), blue-difference (c), and red-difference chroma (d) are shown. The channels of basis map with higher energy maintain the overall texture of the original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

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 [20] aims to approximate a given matrix $\boldsymbol{X} \in \mathbb{R}^{M\times N}$ by

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

where $\boldsymbol{U} \in \mathbb{R}^{M\times R}$ and $\boldsymbol{V} \in \mathbb{R}^{N\times R}$ are *factor matrices* (or simply *factors*), $R \le \min(M, N)$ 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 $\boldsymbol{U}\boldsymbol{V}^\top$, as well as the constraints on the factor matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, various formulations and variants have been proposed for different purposes [21,22,18]. In Section 3.3, we introduce and elaborate on our variant, termed quantization-aware matrix factorization (QMF), and argue why it is well-suited and effective for image compression.

Lossless compression QMF yields factor matrices $\boldsymbol{U}_Y \in \{0,\dots,255\}^{\frac{HW}{64}\times R}$ and $\boldsymbol{V}_Y \in \{0,\dots,255\}^{64\times R}$; $\boldsymbol{U}_{C_B} \in \{0,\dots,255\}^{\frac{HW}{256}\times R}$ and $\boldsymbol{V}_{C_B} \in \{0,\dots,255\}^{64\times R}$; and $\boldsymbol{U}_{C_R} \in \{0,\dots,255\}^{\frac{HW}{256}\times R}$ and $\boldsymbol{V}_{C_R} \in \{0,\dots,255\}^{64\times R}$ that correspond to $\boldsymbol{X}_Y$, $\boldsymbol{X}_{C_B}$, and $\boldsymbol{X}_{C_R}$. Since these matrices have elements constrained to integer values (allowing seamless integration of quantization with their optimization process), they can be directly encoded using any standard lossless data compression method, such as zlib [23]. In contrast, other lossy image compression methods typically require a separate quantization step, introducing errors that cannot be incorporated or considered during the compression process.

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

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

As each channel of a *factor map* can be treated as an 8-bit grayscale image, we can encode it by any standard lossless image compression method such as PNG. For images with a resolution of $H, W \gg 64$, which are most common nowadays, the *basis maps* ($\mathcal{U}$) are significantly larger than the *coefficient maps* ($\mathcal{V}$), accounting for the majority of the storage space. Interestingly, in practice, the QMF *basis maps* turn out to be meaningful images, each capturing some visual semantic of the image (see Fig. 2 for an example). Therefore, our QMF approach can effectively leverage the power of existing lossless image compression algorithms, offering a significant advantage over current methods. However, in this work, we take the first approach and use the zlib library [23] to encode factor matrices, creating a stand-alone codec that is independent from other image compression methods.

3.2. Decoding

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

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

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

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

3.3. Quantization-aware matrix factorization (QMF)

The main building block of our method is quantization-aware matrix factorization (QMF), which is responsible for the lossy compression of matrices obtained through patchification. QMF 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 $\boldsymbol{U}\boldsymbol{V}^\top$, while ensuring, as an integrated quantization step, that the elements of the factor matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are integers within a specified interval $[\alpha, \beta]$ with integer endpoints, i.e., $\alpha, \beta \in \mathbb{Z}$. Formally, the QMF problem can be expressed as:

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

where $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm; $R \le \min(M, N)$ represents the *rank*; and $\mathbb{Z}_{[\alpha,\beta]} \triangleq [\alpha, \beta] \cap \mathbb{Z}$ denotes the set of integers within $[\alpha, \beta]$. Without constraints on the factors, the problem would have an analytic solution through singular value decomposition (SVD), as addressed by the Eckart–Young–Mirsky theorem [20]. If only a nonnegativity constraint were applied (without integrality), variations of nonnegative matrix factorization (NMF) would emerge [21,24]. The QMF problem (3) poses a challenging integer program, with finding its global minima known to be NP-hard [19,25]. Only a few iterative algorithms [19,18] have been proposed to find a "good solution" for some QMF variants in contexts other than image compression. Unlike [18], which considers unbounded or binary-unbounded factors, our QMF formulation constrains both factors to bounded integers, enabling integration with lossless compression. The method in [19], while related, involves heavy least-squares and enumeration procedures, in contrast to our efficient closed-form BCD updates. Therefore, we propose an efficient iterative algorithm for the QMF problem (3) in Section 3.4.

The existing lossy image compression methods based on SVD and NMF approach the problem as an optimization task, followed by a separate quantization step. The optimization focuses on finding factors that minimize the reconstruction error. Before any byte-level processing, a quantization step is applied to project the floating-point elements of the resulting factors onto a set of discrete values. However, because the quantization is performed separately from the optimization, the quantization errors cannot be incorporated into the compression process. This separation leads to suboptimal compression performance (as demonstrated in Section 4) and additional complications in designing quantizers.

In contrast, our QMF formulation, through the *constrained* optimization in (3), produces integer factor matrices that minimize the reconstruction error while ensuring that all elements are discrete. These integer factor matrices can be directly stored and processed losslessly without introducing roundoff errors. The reason for limiting the feasible region to $[\alpha, \beta]$ in our QMF 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 QMF formulation is well-suited for image compression, effectively integrating the factorization and quantization steps into a single, efficient compression process. Note that due to the integrality constraints on the factor matrices, exact reconstruction of the original matrix is not possible even when the rank reaches its maximum possible value (i.e., $R = \min(M, N)$). As such, the proposed method does not support lossless compression.

3.4. Block coordinate descent scheme for QMF

We propose an efficient algorithm for QMF using the block coordinate descent (BCD) scheme (aka alternating optimization). The pseudocode is provided in Algorithm 1. Starting with some initial parameter values, this approach involves sequentially minimizing

the cost function with respect to a single column of a factor at a time, while keeping the other columns of that factor and the entire other factor fixed. This process is repeated until a stopping criterion is met, such as when the change in the cost function value falls below a predefined threshold or the maximum number of iterations is reached. Formally, this involves solving one of the following subproblems at a time:

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

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

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

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

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

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

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

Proof. See Appendix A for the proof. □

It is noteworthy that the combination of $\mathrm{round}(\cdot)$ and $\mathrm{clamp}_{[\alpha,\beta]}(\cdot)$ in (6) and (7) can be interpreted as the element-wise projector to $\mathbb{Z}_{[\alpha,\beta]}$. In addition, updates (6) and (7) are presented in Algorithm 1 at steps 7 and 8, and steps 13 and 14, respectively. In Theorem 2, the convergence of Algorithm 1 is established.

Theorem 2 *(Convergence). Let* $(U_{:r}^k)_{k\in\mathbb{N}}$ *and* $(V_{:r}^k)_{k\in\mathbb{N}}$ *for* $r \in \{1, \ldots, R\}$ *be sequences generated by the proposed Algorithm 1. Then all sequences are bounded and convergent to a locally optimal point of optimization problem* (3).

Proof. See Appendix B for the proof. □

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

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

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

Essentially, this means that instead of performing a constrained optimization, we first low-rank approximate $\boldsymbol{X}$ and then satisfy the constraints by projecting the elements of the resulting factor matrices onto $\mathbb{Z}_{[\alpha,\beta]}$.

4. Experiments

4.1. Setup

QMF configuration In our QMF implementation, we used a default patch size of 8×8. The default factor bounds were set to $[-16, 15]$. Unless otherwise specified, the number of BCD iterations was set to 10 although our ablation studies in Section 4.6 suggest that even 2 iterations may suffice in practice (see Fig. 6b). For lossless compression of factors, we encoded and decoded each column of a factor separately using the zlib library [23]. We also tested other lossless compression methods, such as zstd [26] and Huffman coding [27], which demonstrated a comparable performance. However, as zlib is well-established, simple, and offers fast performance in Python, the experimental results are reported for this compression method.

Baseline codecs We compared our QMF method against JPEG and SVD baselines. For JPEG compression, we used the Pillow library [28]. Our SVD baseline follows the same framework as the proposed method (described in Section 3.1) but substitutes truncated SVD for QMF, followed by uniform quantization of the SVD factor matrices before lossless compression using zlib [23]. We incorporated this baseline (SVD + post-hoc quantization) to provide a direct comparison between our joint optimization approach (QMF) and a separate quantization approach. Notably, QMF produces integer factors that can be directly encoded with zlib, eliminating the need for a post-hoc quantization step.

Datasets To validate the effectiveness of our method, we conducted experiments using the widely-used **Kodak** dataset [29], consisting of 24 lossless images with a resolution of 768×512. To evaluate the robustness of our method in a higher-resolution setting, we also experimented with the **CLIC 2024** validation dataset [30], which contains 30 high-resolution, high-quality images. Additionally, we assessed the compression methods by their ability to retain visual semantics. This was achieved by evaluating a pre-trained ImageNet classifier on compressed images from the **ImageNet** validation set [31], consisting of 50,000 images with a resolution of 224×224 across 1000 classes.

Metrics To evaluate the rate-distortion performance of methods on the Kodak and CLIC 2024 datasets, we measured the bit rate in bits per pixel (bpp) and assessed the quality of the reconstructed images using peak signal-to-noise ratio (PSNR) and multiscale structural similarity index measure (MS-SSIM). Then, these metrics were plotted as functions of bit rate for each method to illustrate their rate-distortion performance. To control the quality of the reconstructed images in QMF and SVD, similar to JPEG, we defined a quality factor $Q \in [0, 1]$, where 0 represents the highest compression and 1 represents no compression. To determine the factorization rank R in Algorithm 1, we used $R = \max\{\text{round}(Q \times \min\{M, N\}), 1\}$.

More precisely, to construct a rate-distortion curve for each method on each dataset, we evaluated various qualities Q for each image. For each quality, we first measured the PSNR/MS-SSIM values at the corresponding bit rate. Next, PSNR/MS-SSIM values were interpolated at evenly spaced bit rates ranging from 0.05 bpp to 0.5 bpp using LOESS (locally estimated scatterplot smoothing) [32]. Finally, the interpolated values were averaged over all images at each of these bit rates.

4.2. Rate-distortion performance

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

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

CLIC 2024 A similar trend is observed with the CLIC 2024 dataset, as shown in Figs. 3c and 3d. Here, the PSNR and MS-SSIM results further confirm the competitive performance of QMF at low bit rates. For example, at a bit rate of 0.15 bpp, QMF achieves a PSNR of over 25 dB, compared to approximately 22 dB for both SVD and JPEG, and MS-SSIM of around 0.88, compared to 0.78 for SVD and 0.75 for JPEG. The superiority of QMF over JPEG is more pronounced in terms of MS-SSIM than PSNR, with QMF outperforming JPEG up to 0.25 bpp for PSNR and 0.35 bpp for MS-SSIM. This demonstrates the robustness of QMF in preserving visual quality across different datasets.

4.3. Qualitative performance

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

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

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

In the vegetables image (fourth row), JPEG yields visible *color distortion* (marked by the cyan boxes), while SVD introduces significant blurriness. QMF, however, effectively preserves the color fidelity and detail.

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

However, in the parrots image (third row), JPEG achieves a higher PSNR of 33.25 dB at approximately 0.4 bpp, whereas QMF attains only 30.67 dB at the same bit rate. This reflects the rate-distortion curves in Fig. 3, where JPEG outperforms QMF in reconstruction quality at high bit rates (> 0.4 bpp). Nevertheless, at these high PSNR levels (> 30 dB), visual differences between JPEG and QMF are typically imperceptible to the human eye.

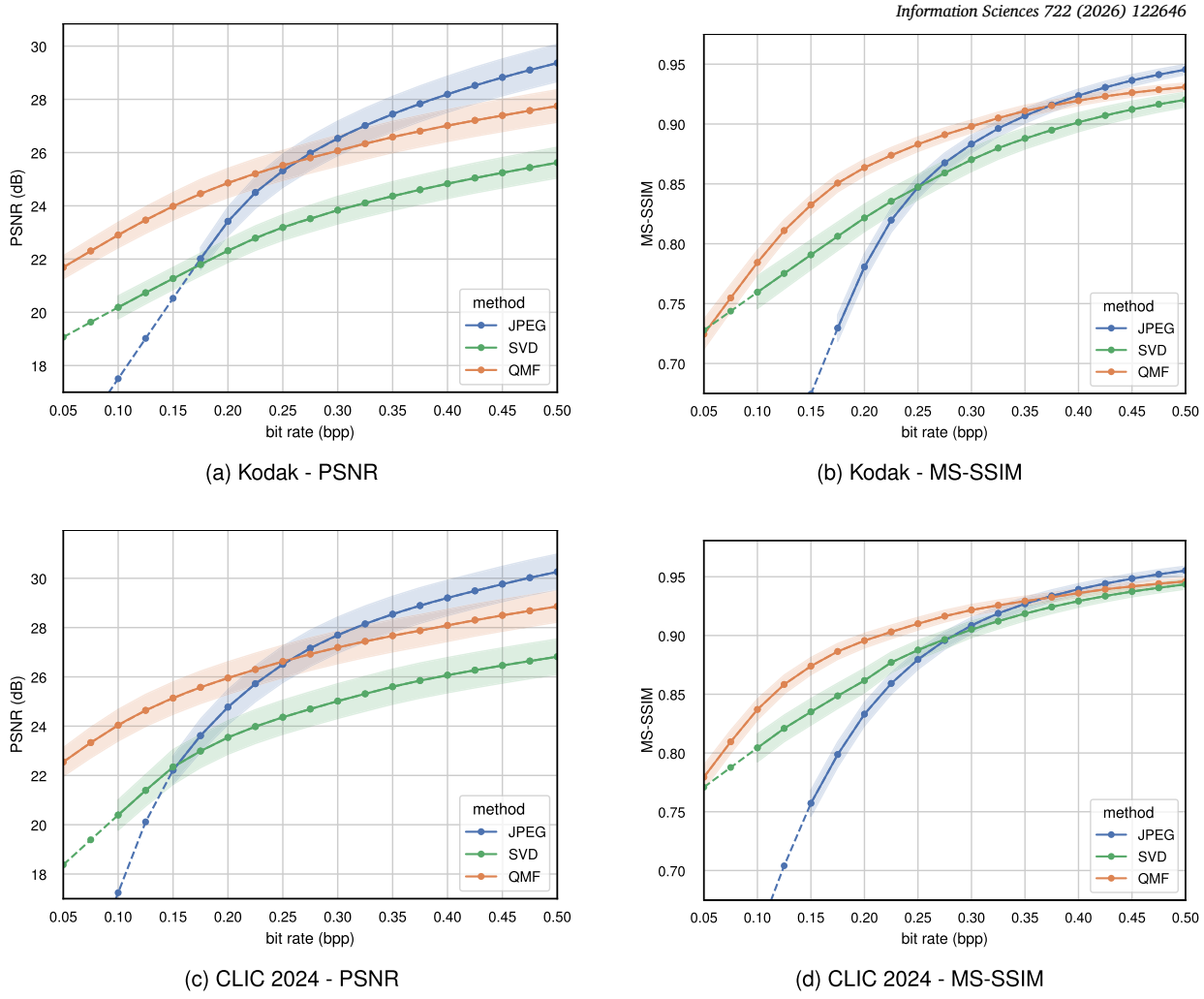


Fig. 3. Rate-distortion performance on the Kodak (top panels) and CLIC 2024 (bottom panels) datasets. The average PSNR (left panels) and average MS-SSIM (right panels) for each method are plotted as functions of bit rate. Shaded areas represent standard errors. Dashed lines indicate extrapolated values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

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

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

4.4. Run time

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

Although JPEG compression consistently has a lower encoding time, QMF and SVD have a significant advantage in decoding speed over JPEG, with SVD being the fastest. Specifically, QMF decodes more than twice as fast as JPEG on the CLIC 2024 dataset across all bit rates. Overall, QMF is preferable for applications requiring high image quality at low bit rates, especially in scenarios where compressed images are frequently accessed or displayed while encoding occurs less often. Examples include web browsing,

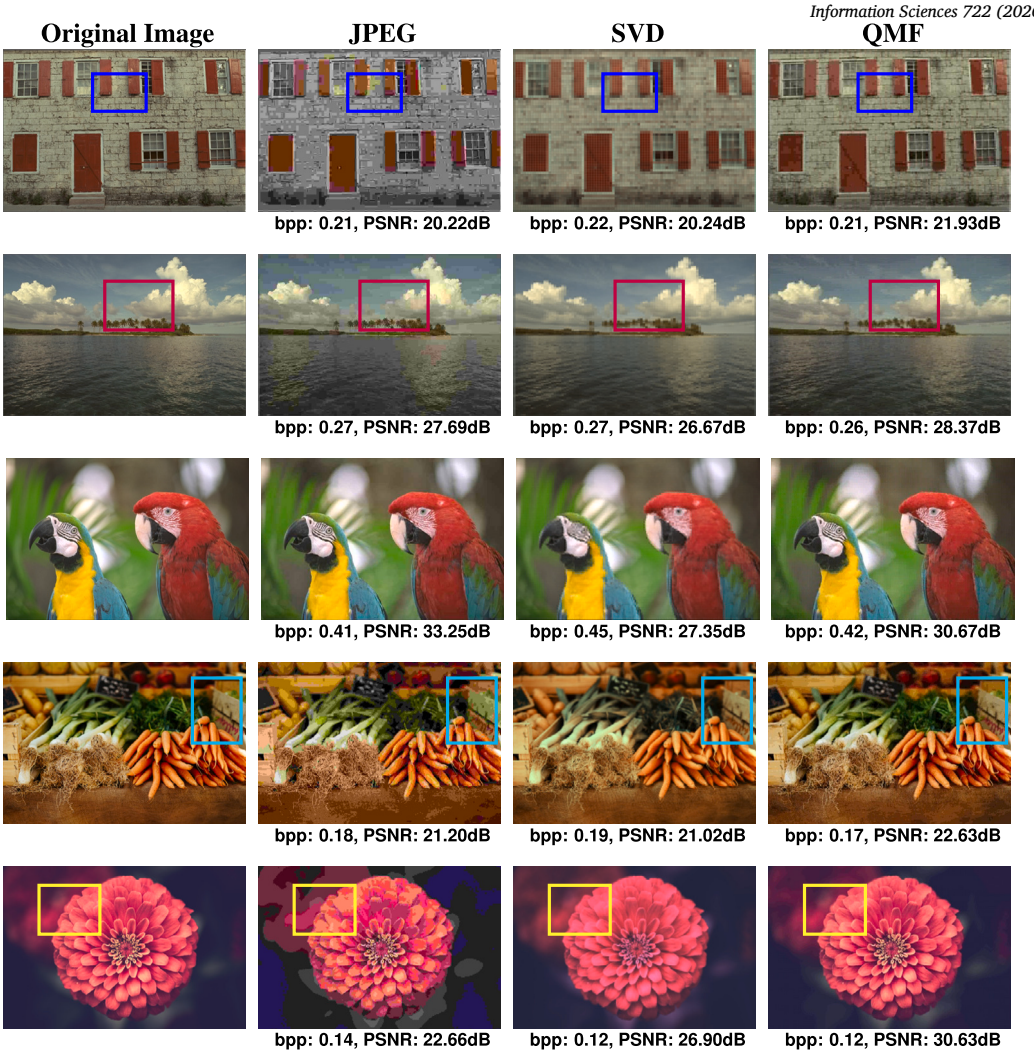


Fig. 4. Qualitative performance comparison on example images from the Kodak (top three rows) and the CLIC 2024 (bottom two rows) datasets. Each column shows the original image, JPEG, SVD, and QMF compression results respectively. The bit rate and PSNR values for each compressed image is reported. The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

image hosting, mobile applications, satellite imagery for maps, and interactive gaming applications. These use cases often involve large numbers of images or thumbnails, where faster decoding times are essential for seamless user experiences, even if encoding takes more time or demands higher computational resources.

From a theoretical perspective, a single BCD iteration in QMF has time complexity $O(MNR)$. For an image of size $H \times W$ and patch size $p \times p$, with $M = HW/p^2$ and $N = p^2$, this simplifies to $O(HWR)$ per iteration for fixed p. As only a small, fixed number of BCD iterations is required in practice (see Section 4), QMF encoding time complexity remains linear in the number of image pixels. Decoding QMF involves only a single matrix multiplication between the factor matrices, requiring exactly $MNR = HWR$ operations, again linear in image size. For JPEG, the main computational cost in both encoding and decoding is the 2D discrete cosine transform (DCT) on each $p \times p$ patch. With fast algorithms (e.g., FFT), each DCT requires $O(p^2 \log p)$ operations, and with HW/p^2 patches, the total is $O(HW \log p)$. Thus, JPEG's complexity also scales linearly with the number of pixels.

The empirical results in Table 1 align with this theoretical analysis, where QMF's slower encoding time compared to JPEG is attributed to the sequential BCD iterations, while its faster decoding may stem from the cheap, parallelizable matrix multiplication.

4.5. ImageNet classification performance

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

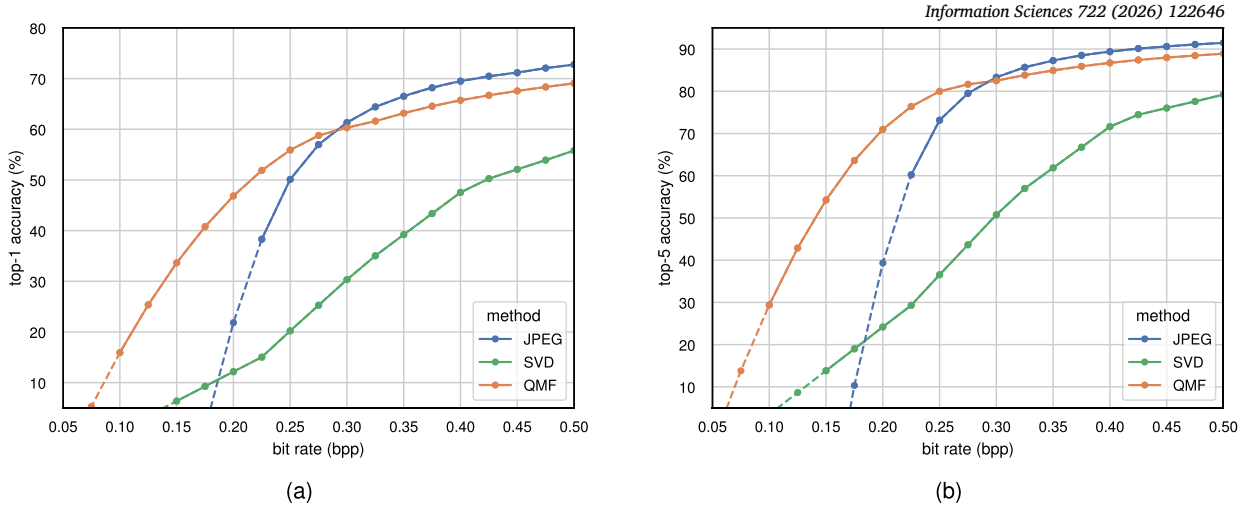


Fig. 5. Impact of different compression methods on ImageNet classification accuracy. A ResNet-50 classifier pre-trained on the original ImageNet images is evaluated using validation images compressed by different methods. Panels (a) and (b) show top-1 and top-5 accuracy plotted as a function of bit rate,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

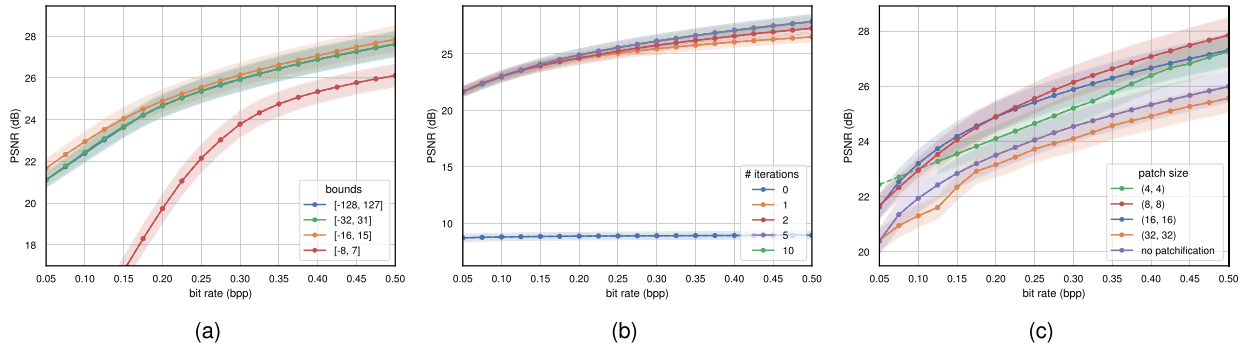


Fig. 6. Ablation studies for QMF. The average PSNR on the Kodak dataset is plotted as a function of bit rate under various experimental conditions: (a) varying the bounds $[\alpha, \beta]$ for the elements of the factor matrices, (b) changing the number of BCD iterations, and (c) adjusting the patch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

image quality, and we compress images before classification to minimize resource requirements, such as memory and communication bandwidth.

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

4.6. Ablation studies

We conducted ablation studies to investigate the impact of factor bounds, the number of BCD iterations, and patch size on the compression performance of our QMF method. All experiments in this section were performed using the Kodak dataset. We followed the QMF configuration described in Section 4.1 and varied only the parameters under ablation one at a time.

Factor bounds Fig. 6a shows the average PSNR as a function of bit rate for QMF using various factor bounds $[\alpha, \beta]$ in Algorithm 1. The results indicate that the interval $[-16, 15]$ yields the optimal performance, showing moderate improvement over both $[-32, 31]$ and $[-128, 127]$, while significantly outperforming $[-8, 7]$. In fact, constraining the factor elements within a sufficiently narrow range can reduce the bit allocation needed, leading to a lower bit rate after lossless compression in the final stage of our framework. However, the tighter bounds on the factor elements constrain the feasible region, which comes at the cost of higher reconstruction error. Our experiments showed that using the bounds $[-16, 15]$ provides the best trade-off between compression ratio and reconstruction quality. Note that in all these cases, the factor elements are represented as the `int8` data type.

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

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

5. Discussion

All our comparative results (Fig. 3 and Fig. 5) consistently show that our QMF method outperforms JPEG in both maintaining image quality and preserving visual semantics at low bit rates (< 0.25 bpp), while its performance advantage diminishes and even reverses at higher bit rates. Moreover, QMF consistently demonstrates superior performance compared to SVD across all bit rates. This superiority can be attributed to the integration of quantization with low-rank approximation in QMF, which enables more accurate reconstruction. In contrast, the high sensitivity of SVD to quantization errors, which arise during a separate quantization step, significantly degrades the reconstruction quality.

The performance degradation of QMF compared to JPEG at higher bit rates can be attributed to two key factors: 1) In QMF, as the bit rate is controlled by adjusting the rank, higher bit rates require increased rank, leading to a linear growth in the number of factor matrix elements to be losslessly compressed. However, the reduction in reconstruction error quickly saturates, resulting in a less favorable rate-distortion trade-off at high bit rates compared to JPEG. 2) The nonconvex nature of the QMF optimization problem means that as rank increases, the problem's dimensionality grows, making it more likely to converge to suboptimal local minima. JPEG avoids this issue by operating on fixed-size patches with closed-form DCT solutions, giving it an advantage at high bit rates.

While QMF is outperformed by JPEG at approximately 0.25 bpp in PSNR (see Figs. 3a and 3c), the ImageNet classification results (Fig. 5) and MS-SSIM curves (Figs. 3b and 3d) indicate a take-over point at around 0.3 bpp and over 0.35 bpp, respectively. These slight shifts imply that QMF's superiority over JPEG at low bit rates in preserving visual semantics is more pronounced than its advantage in maintaining image quality.

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

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

6. Conclusion

This work presents a novel lossy image compression method based on quantization-aware matrix factorization (QMF). By representing image data as the product of two smaller factor matrices with elements constrained to bounded integer values, the proposed QMF approach effectively integrates quantization with low-rank approximation. In contrast, traditional compression methods such as JPEG and SVD consider quantization as a separate step, where quantization errors cannot be incorporated into the compression process. The reshaped factor matrices in QMF are compatible with existing lossless compression standards, enhancing the overall flexibility and efficiency of our method. Our proposed iterative algorithm, utilizing a block coordinate descent scheme, has proven to be both efficient and convergent. Experimental results demonstrate that the QMF method significantly outperforms JPEG in terms of PSNR and SIMM at low bit rates (< 0.25 bpp) and maintains better visual semantic information. This advantage underscores the potential of QMF to set a new standard in lossy image compression, bridging the gap between factorization and quantization.

CRediT authorship contribution statement

Pooya Ashtari: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Pourya Behmandpoor:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration,

Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Fateme Nateghi Haredasht:** Writing – review & editing, Validation. **Jonathan H. Chen:** Supervision. **Panagiotis Patrinos:** Supervision. **Sabine Van Huffel:** Supervision, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Pooya Ashtari reports financial support was provided by Flanders AI Research Program. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research received funding from Flanders AI Research Program. Sabine Van Huffel and Pooya Ashtari are affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium.

Appendix A. 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:

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

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 (A.1) can be decoupled into M optimizations as follows:

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

The objective functions $q_i(u_i^r)$ in (A.2) are single-variable quadratic problems. Hence, the global optimum in each decoupled optimization problem can be achieved by finding the minimum of each quadratic problem and then projecting it onto the set $\mathbb{Z}_{[\alpha,\beta]}$. The minimum of each quadratic function in (A.2), denoted by $\bar{u}_i^r$, can be simply found by

$$\nabla_{u_i^r}q_i(u_i^r)=0\implies \bar{u}_i^r=\textstyle\sum_{j=1}^{N}e_{ij}^r v_j^r/\sum_{j=1}^{N}v_j^{r2}, \tag{A.3}$$

where ∇_x is the partial derivative with respect to x. Since q_i has a constant curvature (second derivative) and $q_i(\bar{u}_i^r+d)$ is nondecreasing with increasing $|d|$, the value in the set $\mathbb{Z}_{[\alpha,\beta]}$ which is closest to $\bar{u}_i^r$ is the global minimizer of (A.2). This value can be reached by projecting $\bar{u}_i^r$ onto the set $\mathbb{Z}_{[\alpha,\beta]}$, namely $u_i^{r\star}=\mathrm{clamp}_{[\alpha,\beta]}(\mathrm{round}(\bar{u}_i^r))$, which is presented for all $i\in\{1,\dots,M\}$ in a compact form in (6).

Since $\boldsymbol{u}_r^\star\triangleq(u_1^{r\star},\dots,u_M^{r\star})$ is the global optimum of optimization (A.1), it is evident that

$$\|\boldsymbol{E}_r-\boldsymbol{u}_r^\star\boldsymbol{v}_r^{\mathsf{T}}\|_{\mathrm{F}}\le\|\boldsymbol{E}_r-\boldsymbol{u}_r\boldsymbol{v}_r^{\mathsf{T}}\|_{\mathrm{F}}. \tag{A.4}$$

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

Appendix B. Proof of Theorem 2

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

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

where

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

$$g(V_{:r}) \triangleq \delta_{[a,b]}(V_{:r}) + \delta_{\mathbb{Z}}(V_{:r}),$$

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 (B.1) are equivalent.

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

The PALM algorithm can be summarized as follows:

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

$$\begin{aligned} &(a)\ \boldsymbol{U}^{k+1} \in \operatorname{prox}^{f}_{c_k}\left(\boldsymbol{U}^k - \frac{1}{c_k}\nabla_{\boldsymbol{U}} f_0(\boldsymbol{U}^k, \boldsymbol{V}^k)\right), \\ &(b)\ \boldsymbol{V}^{k+1} \in \operatorname{prox}^{g}_{d_k}\left(\boldsymbol{V}^k - \frac{1}{d_k}\nabla_{\boldsymbol{V}} f_0(\boldsymbol{U}^{k+1}, \boldsymbol{V}^k)\right), \end{aligned} \tag{B.2}$$

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

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

Proposition 1 *(Meeting required assumptions). The assumptions necessary for the convergence of iterates in* (B.2) *are satisfied by the functions involved in the problem* (B.1)*, specifically:*

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

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

   where $\|\cdot\|$ denotes the ℓ_2-norm of the vectorized input with the proper dimension (here, with the input in $\mathbb{R}^{MR\times 1}$). The similar Lipschitz continuity is evident for $\nabla_{\boldsymbol{V}} f_0(\boldsymbol{U}, \boldsymbol{V})$ as well with modulus $L_2(\boldsymbol{U}) = \|\boldsymbol{U}\boldsymbol{U}^T\|_{\mathrm{F}}$.
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 f_0 is twice differentiable, hence, its full gradient $\nabla f_0(\boldsymbol{U}, \boldsymbol{V})$ is Lipschitz continuous on the bounded set $\boldsymbol{U} \in [a, b]^{M\times R}$, $\boldsymbol{V} \in [a, b]^{N\times R}$. Namely, with $M > 0$:*

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

   where $(\cdot,\cdot)$ denotes the concatenation of the two arguments.
5. *The sets $[a, b]$ and integer numbers are semi-algebraic; so are their indicator functions. The function f_0 is also polynomial, hence it is semi-algebraic. The sum of these functions results in a semi-algebraic function Ψ in* (B.1)*, hence Ψ is a Kurdyka-Łojasiewicz (KL) function.*

By Proposition 1, the optimization problem (B.1) can be solved by the iterates in (B.2), due to the following proposition:

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

It is noted that the so-called *forward* steps $\boldsymbol{U}^k - \frac{1}{c_k}\nabla_{\boldsymbol{U}} f_0(\boldsymbol{U}^k, \boldsymbol{V}^k)$ and $\boldsymbol{V}^k - \frac{1}{d_k}\nabla_{\boldsymbol{V}} f_0(\boldsymbol{U}^{k+1}, \boldsymbol{V}^k)$ in the prox operators in (B.2) are replaced by the simple closed-form solutions $\boldsymbol{E}_r\boldsymbol{v}_r/\|\boldsymbol{v}_r\|^2$ and $\boldsymbol{E}_r^{\mathrm{T}}\boldsymbol{u}_r/\|\boldsymbol{u}_r\|^2$ in Algorithm 1 at steps 7 and 13 (cf. (6) and (7)), respectively.

In the case where the iterates (B.2) are extended to multi-block updates, each block represents one column. This is thanks to the special form of the functions $f_0(\cdot, \boldsymbol{V}^k)$ and $f_0(\boldsymbol{U}^{k+1}, \cdot)$ being quadratic functions, each having a global optimal point, which ensures a descent in each forward step. Furthermore, the proximal operators $\text{prox}^f_{c_k}$ and $\text{prox}^g_{d_k}$ can efficiently be implemented by the operators round and $\text{clamp}_{[\alpha,\beta]}$ in (6) and (7) (and equivalently in Algorithm 1 at steps 8 and 14). The equivalence of these steps is proven in the following lemma.

Lemma 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* $\text{prox}^g_{d_k}(\boldsymbol{Z}) = \text{round}(\text{clamp}_{[\alpha,\beta]}(\boldsymbol{Z}))$ *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 $\boldsymbol{W} \in \mathbb{R}^{M\times R}$:

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

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

$$\text{round}(\boldsymbol{W}) = \underset{\boldsymbol{U}\in\mathbb{Z}^{M\times R}}{\arg\min}\{\|\boldsymbol{U}-\boldsymbol{W}\|^2_F\}. \tag{B.3}$$

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

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

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 $\|\boldsymbol{U}-\boldsymbol{W}\|^2_a + \|\boldsymbol{U}-\boldsymbol{W}\|^2_b$ with $\|\boldsymbol{U}-\boldsymbol{A}\|^2_a + \|\boldsymbol{U}-\boldsymbol{B}\|^2_b$ 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, (B.3) is invoked. It is noted that in the implementation, $\text{round}(\text{clamp}_{[\alpha,\beta]}(\cdot)) = \text{clamp}_{[\alpha,\beta]}(\text{round}(\cdot))$ due to the integrality of the bounds $\alpha, \beta \in \mathbb{Z}$. A similar proof can be trivially followed for $\text{prox}^g_{d_k}(\boldsymbol{Z}) = \text{round}(\text{clamp}_{[\alpha,\beta]}(\boldsymbol{Z}))$ as well. □

Now that the equivalence of iterates (B.2) 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 (B.1) and (3), Proposition 2 can be trivially invoked to establish the convergence of Algorithm 1 to a locally optimal point of problem (3).

Data availability

The code repository address has been provided in the abstract.

References

[1] G.K. Wallace, The JPEG still picture compression standard, Commun. ACM 34 (1991) 30–44.
[2] A. Skodras, C. Christopoulos, T. Ebrahimi, The JPEG 2000 still image compression standard, IEEE Signal Process. Mag. 18 (2001) 36–58.
[3] V.K. Goyal, Theoretical foundations of transform coding, IEEE Signal Process. Mag. 18 (2001) 9–21.
[4] N. Ahmed, T. Natarajan, K.R. Rao, Discrete cosine transform, IEEE Trans. Comput. 100 (1974) 90–93.
[5] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, Image coding using wavelet transform, IEEE Trans. Image Process. 1 (1992) 20–25.
[6] H. Andrews, C. Patterson, Singular value decomposition (SVD) image coding, IEEE Trans. Commun. 24 (1976) 425–432.
[7] H. Prasantha, H. Shashidhara, K.B. Murthy, Image compression using SVD, in: International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), vol. 3, IEEE, 2007, pp. 143–145.
[8] J. Hou, L.-P. Chau, N. Magnenat-Thalmann, Y. He, Sparse low-rank matrix approximation for data compression, IEEE Trans. Circuits Syst. Video Technol. 27 (2015) 1043–1054.
[9] J.M. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, IEEE Trans. Signal Process. 41 (1993) 3445–3462.
[10] G. Developers, WebP compression techniques, https://developers.google.com/speed/webp, 2011. (Accessed 17 May 2024).

[11] J. Lainema, M.M. Hannuksela, V.K.M. Vadakital, E.B. Aksu, HEVC still image coding and high efficiency image file format, in: International Conference on Image Processing (ICIP), IEEE, 2016, pp. 71–75.
[12] M.M. Hannuksela, J. Lainema, V.K.M. Vadakital, The high efficiency image file format standard [standards in a nutshell], IEEE Signal Process. Mag. 32 (2015) 150–156.
[13] J. Ballé, D. Minnen, S. Singh, S.J. Hwang, N. Johnston, Variational image compression with a scale hyperprior, in: 6th International Conference on Learning Representations (ICLR), OpenReview.net, 2018.
[14] Z. Cheng, H. Sun, M. Takeuchi, J. Katto, Learned image compression with discretized Gaussian mixture likelihoods and attention modules, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7939–7948.
[15] J. Liu, H. Sun, J. Katto, Learned image compression with mixed transformer-CNN architectures, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 14388–14397.
[16] R. Yang, S. Mandt, Lossy image compression with conditional diffusion models, Adv. Neural Inf. Process. Syst. 36 (2024).
[17] X. Yuan, R. Haimi-Cohen, Image compression based on compressive sensing: end-to-end comparison with JPEG, IEEE Trans. Multimed. 22 (2020) 2889–2904.
[18] M.M. Lin, B. Dong, M.T. Chu, Integer matrix factorization and its application, Tech. Rep., 2005.
[19] B. Dong, M.M. Lin, H. Park, Integer matrix approximation and data mining, J. Sci. Comput. 75 (2018) 198–224.
[20] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (1936) 211–218.
[21] D. Lee, H.S. Seung, Algorithms for non-negative matrix factorization, Adv. Neural Inf. Process. Syst. 13 (2000).
[22] C.H. Ding, T. Li, M.I. Jordan, Convex and semi-nonnegative matrix factorizations, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2008) 45–55.
[23] P. Deutsch, J.-L. Gailly, Zlib compressed data format specification version 3.3, Technical Report, 1996.
[24] N. Gillis, Nonnegative Matrix Factorization, SIAM, 2020.
[25] P. van Emde Boas, Another NP-complete problem and the complexity of computing short vectors in a lattice, Technical Report, Department of Mathematics, University of Amsterdam, 1981.
[26] Y. Collet, contributors, Zstandard - fast real-time compression algorithm, https://facebook.github.io/zstd/, 2016.
[27] D.A. Huffman, A method for the construction of minimum-redundancy codes, Proc. IRE 40 (1952) 1098–1101.
[28] A. Clark, Pillow (PIL Fork) documentation, https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf, 2015.
[29] E. Kodak, Kodak lossless true color image suite, https://r0k.us/graphics/kodak/, 1993.
[30] Organizers, Challenge on learned image compression, https://www.compression.cc/, 2024.
[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (2015) 211–252.
[32] W.S. Cleveland, S.J. Devlin, Locally weighted regression: an approach to regression analysis by local fitting, J. Am. Stat. Assoc. 83 (1988) 596–610.
[33] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Conference on Computer Vision and Pattern Recognition, IEEE, 2016, pp. 770–778.
[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.
[35] Y. Nesterov, Efficiency of coordinate descent methods on huge-scale optimization problems, SIAM J. Optim. 22 (2012) 341–362.
[36] H. Attouch, J. Bolte, B.F. Svaiter, Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods, Math. Program. 137 (2013) 91–129.
[37] A. Beck, L. Tetruashvili, On the convergence of block coordinate descent type methods, SIAM J. Optim. 23 (2013) 2037–2060.
[38] J. Bolte, S. Sabach, M. Teboulle, Proximal alternating linearized minimization for nonconvex and nonsmooth problems, Math. Program. 146 (2014) 459–494.