Convolutional Dictionary Learning: Acceleration and Convergence

Il Yong Chun, *Member, IEEE*, and Jeffrey A. Fessler, *Fellow, IEEE*

***Abstract*— Convolutional dictionary learning (CDL or sparsifying CDL) has many applications in image processing and computer vision. There has been growing interest in developing efficient algorithms for CDL, mostly relying on the augmented Lagrangian (AL) method or the variant alternating direction method of multipliers (ADMM). When their parameters are properly tuned, AL methods have shown fast convergence in CDL. However, the parameter tuning process is not trivial due to its data dependence and, in practice, the convergence of AL methods depends on the AL parameters for nonconvex CDL problems. To moderate these problems, this paper proposes a new practically feasible and convergent *Block Proximal Gradient method using a Majorizer* (BPG-M) for CDL. The BPG-M-based CDL is investigated with different block updating schemes and majorization matrix designs, and further accelerated by incorporating some momentum coefficient formulas and restarting techniques. All of the methods investigated incorporate a boundary artifacts removal (or, more generally, sampling) operator in the learning model. Numerical experiments show that, without needing any parameter tuning process, the proposed BPG-M approach converges more stably to desirable solutions of lower objective values than the existing state-of-the-art ADMM algorithm and its memory-efficient variant do. Compared with the ADMM approaches, the BPG-M method using a multi-block updating scheme is particularly useful in single-threaded CDL algorithm handling large data sets, due to its lower memory requirement and no polynomial computational complexity. Image denoising experiments show that, for relatively strong additive white Gaussian noise, the filters learned by BPG-M-based CDL outperform those trained by the ADMM approach.**

***Index Terms*— Convolutional dictionary learning, convolutional sparse coding, block proximal gradient method, majorization matrix design, convergence guarantee.**

I. Introduction

ADAPTIVE sparse representations can model intricate redundancies of complex structured images in a wide range of applications. "Learning" sparse representations from large datasets, such as (sparsifying) dictionary learning, is a growing trend. Patch-based dictionary learning is a well-known technique for obtaining sparse representations of training signals [1]–[5]. The learned dictionaries from patch-based techniques have been applied to various image processing and computer vision problems, i.e., image inpainting, denoising, deblurring, compression, classification, etc. (see [1]–[5] and references therein). However, patch-based dictionary learning has three fundamental limitations. Firstly, learned basis elements often are shifted versions of each other (i.e., *translation-variant* dictionaries) and underlying structure of the signal may be lost, because each patch—rather than an entire image—is synthesized (or reconstructed) individually. Secondly, sparse representation for a whole image is highly redundant because neighboring and even overlapping patches are sparsified independently. Thirdly, using many overlapping patches across the training and test signals hinders using "big data"—i.e., training data with the large number of signals or high-dimensional signals; for example, see [6, §3.2] or Section VII-B—and discourages the learned dictionary from being applied to large-scale inverse problems, respectively.

Convolutional dictionary learning (CDL or sparsifying CDL), motivated by the perspective of modeling receptive fields in human vision [7], [8] and convolutional neural networks [9]–[11], can overcome the problems of patch-based dictionary learning [6], [12]–[22]. In particular, signals displaying translation invariance in any dimension (e.g., natural images and sounds) are better represented using a CDL approach [13]. In addition, CDL is a basic element in training deconvolutional neural networks [23]; its sparse coding step (e.g., see Section IV-B) is closely related to convolutional neural networks [24]. Learned convolutional dictionaries have been applied to various image processing and computer vision problems, e.g., image inpainting, denoising, classification, recognition, detection, etc. (see [6], [12]–[14], [18], [22]).

CDL in 2D (and beyond) has two major challenges. The first concern lies in its optimization techniques: *1)* computational complexity, *2) and memory-inefficient algorithm (particularly augmented Lagrangian (AL) method),* and *3)* convergence guarantees. In terms of computational complexity, the most recent advances include algorithmic development with AL method (e.g., alternating direction method of multipliers, ADMM [25], [26]) [6], [14]–[21] and fast proximal gradient (FPG) method [27] (e.g., fast iterative shrinkage-thresholding algorithm, FISTA [28]). Although AL methods have shown fast convergence in [6], [14], and [18] (and faster than the continuation-type approach in [12] and [14]), they require tricky parameter tuning processes for acceleration and (stable)

Manuscript received November 29, 2016; revised May 31, 2017 and August 22, 2017; accepted September 7, 2017. Date of publication October 9, 2017; date of current version January 12, 2018. This work was supported in part by the Keck Foundation and in part by the UM-SJTU Seed Grant. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Abd-Krim K. Seghouane. *(Corresponding author: Il Yong Chun.)*

The authors are with the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48019 USA (email: iychun@umich.edu; fessler@umich.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. The prefix "S" indicates the numbers in section, equation, figure, and table in the supplementary material. The total size of the files is 1.29 MB. Contact chun.ilyong@gmail.com for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2761545

convergence, due to their dependence on training data (specifically, preprocessing of training data, types of training data, and the number and size of training data and filters). In particular, in the AL frameworks, the number of AL parameters to be tuned increases as CDL models become more sophisticated, e.g., *a)* for the CDL model including a boundary truncation (or, more generally, sampling) operator [6], one needs to tune four additional AL parameters; *b)* for the CDL model using adaptive contrast enhancement (CDL-ACE), six additional AL parameters should be tuned [22]! The FPG method introduced in [27] is still not free from the parameter selection problem. The method uses a backtracking scheme because it is impractical to compute the Lipschitz constant of the tremendous-sized system matrix of 2D CDL. Another limitation of the AL approaches is that they require larger amount of memory as CDL models become more sophisticated (see examples above), because one often needs to introduce more auxiliary variables. This drawback can be particularly problematic for some applications, e.g., image classification [29], because their performance improves as the number of training images increases for learning filters [29, Fig. 3]. In terms of theoretical aspects, there exists no known convergence analysis (even for local convergence) noting that CDL is a nonconvex problem. Without a convergence theorem, it is possible that the iterates could diverge.

The second problem is boundary effects associated with the convolution operators. Bristow *et al.* [14] experimentally showed that neglecting boundary effects might be acceptable in learning small-sized filters under periodic boundary condition. However, this is not necessarily true as illustrated in [6, §4.2] with examples using 11-by-11 filters: high-frequency components still exist on image boundaries of synthesized images (i.e., $\sum_k d_k \circledast z_{l,k}$ in (1)). Neglecting them can be unreasonable for learning larger filters or for general boundary conditions. As pointed out in [13], if one does not properly handle the boundary effects, the CDL model can be easily influenced by the boundary effects even if one uses non-periodic boundary conditions for convolutions (e.g., the reflective boundary condition [14, §2]). Specifically, the synthesis errors (i.e., the ℓ^2 data fitting term in (1) without the truncation operator P_B) close to the boundaries might grow much larger compared to those in the interior, because sparse code pixels near the boundaries are convolved less than those in the interior. To remove the boundary artifacts, the formulation in [6] used a boundary truncation operator that was also used in image deblurring problem in [30] and [31]. The truncation operator is inherently considered in the local patch-based CDL framework [21]. In the big data setup, it is important to learn decent filters with less training data (but not necessarily decreasing the number of training signals—see above). The boundary truncation operator in [6], [30], and [31] can be generalized to a sampling operator that reduces the amount of each training signal [6, §4.3]. Considering the sampling operator, the CDL model in [6] learns filters leading better image synthesis accuracy than that without it, e.g., [14]; see Fig. 1.

In this paper, we consider the CDL model in [6] that avoids boundary artifacts or, more generally, incorporates

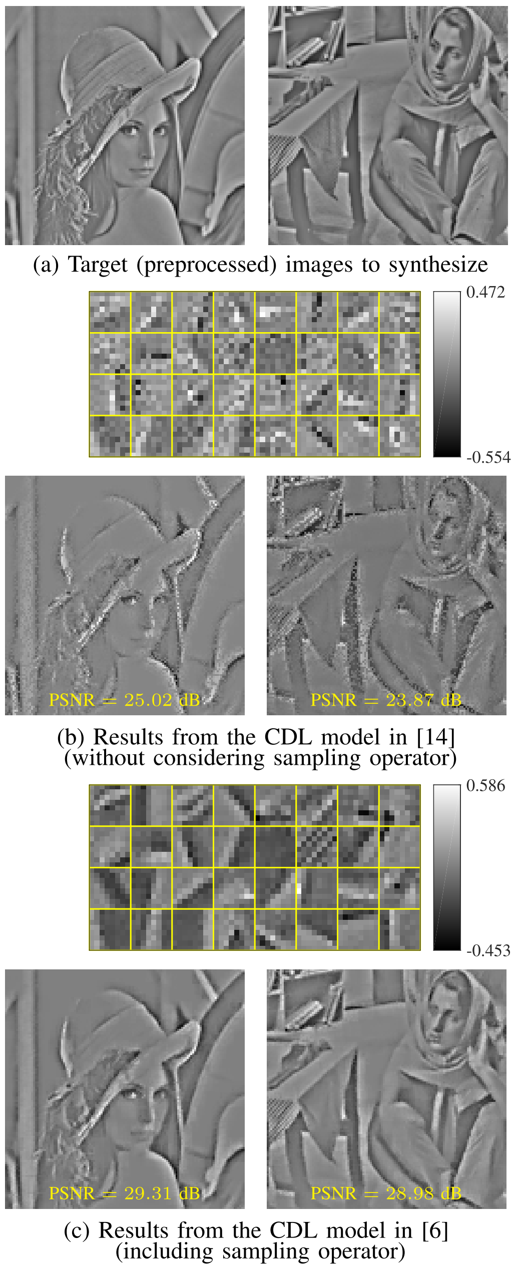

(a) Target (preprocessed) images to synthesize
0.472
-0.554
PSNR = 25.02 dB
PSNR = 23.87 dB
(b) Results from the CDL model in [14] (without considering sampling operator)
0.586
-0.453
PSNR = 29.31 dB
PSNR = 28.98 dB
(c) Results from the CDL model in [6] (including sampling operator)

Fig. 1. Examples of learned filters and synthesized images from sparse datasets with different CDL models (32 8 × 8-sized filters were learned from two *sparse* 128 × 128-sized training images; sparse images were generated by ≈ 60% random sampling—see, for example, [6, Fig. 5]; the experiments are based on the dataset, initialization, parameters, and preprocessing method used in [32, ver. 0.0.7, "demo_cbpdndlmd.m"]). In sparse data settings, including a sampling operator in CDL (e.g., P_B in (1)) allows to learn filters leading better image synthesis performance (note that the results in (c) correspond to those in [6, §4.3]). Note that the image synthesis accuracy in training affects the performance in testing models, e.g., image denoising—see Fig. 5.

sampling operators[1] We propose a new practically feasible and convergent block proximal gradient (BPG [33]) algorithmic framework, called *Block Proximal Gradient method using*

[1]We do not consider the boundary handling CDL model in [19, §3.1], because of its inconsistency in boundary conditions. Its constraint trick in [19, eq. (13)] casts zero-boundary on sparse codes [19, eq. (14)]; however, its CDL algorithm solves the model with the Parseval tricks [6], [14], [18] using periodic boundary condition. See more potential issues in [19, §3.1].

Majorizer (BPG-M), and accelerate it with two momentum coefficient formulas and two restarting techniques. For the CDL model [6], we introduce two block updating schemes within the BPG-M framework: *a) two-block* scheme and *b) multi-block* scheme. In particular, the proposed multi-block BPG-M approach has several benefits over the ADMM approach [6] and its memory-efficient variant (see below): *1)* guaranteed local convergence (or global convergence if some conditions are satisfied) without difficult parameter tuning processes, *2)* lower memory usage, *3) no polynomial computational complexity (particularly, no quadratic complexity with the number of training images)*, and *4)* (empirically) reaching lower objective values. Specifically, for small datasets, the BPG-M approach converges more stably to a "desirable" solution of lower objective values with comparable computational time[2] for larger datasets (i.e., datasets with the larger number of training images or larger sized training images), it stably converges to the desirable solution with better memory flexibility and/or lower computational complexity. Section III introduces the BPG-M, analyzes its convergence, and develops acceleration methods. Section IV applies the proposed BPG-M methods to two-block CDL. Section V proposes multi-block CDL using BPG-M that is particularly useful for single-thread computation. Sections IV–V include computationally efficient majorization matrices and efficient proximal mapping methods. Section VI summarizes CDL-ACE [22] and the corresponding image denoising model using learned filters. Section VII reports numerical experiments that show the benefits—convergence stability, memory flexibility, no polynomial (specifically, quadratic and cubic) complexity, and reaching lower objective values—of the BPG-M framework in CDL, and illustrate the effectiveness of a tight majorizer design and the accelerating schemes on BPG-M convergence rate in CDL. Furthermore, Section VII reports image denoising experiments that show, for relatively strong additive white Gaussian (AWGN) noise, i.e., SNR $= 10$ dB, the learned filters by BPG-M-based CDL improve image denoising compared to the filters trained by ADMM [6].

Throughout the paper, we compare the proposed BPG-M methods mainly to Heide et al.'s ADMM framework in [6] using their suggested ADMM parameters, and its memory-efficient variant applying the linear solver in [18, §III-B] to solve the sub-problem [6, (10)] (see Section VII-A.1 for details). These ADMM frameworks can be viewed as a *theoretically stable* block coordinate descent (BCD, e.g., [33]) method using two blocks if sufficient inner (i.e., ADMM) iterations are used to ensure descent for each block update, whereas methods that use a single inner iteration for each block update may not be guaranteed to descend—see, for example, [19, Fig. 2, AVA-MD].

II. CDL Model and Existing AL-Based Optimization Methods

The CDL problem corresponds to the following joint optimization problem [6] (mathematical notations are provided in Appendix):

$$\min_{\{d_k\},\{z_{l,k}\}} \sum_{l=1}^{L} \frac{1}{2} \left\| y_l - P_B \sum_{k=1}^{K} d_k \circledast z_{l,k} \right\|_2^2 + \alpha \sum_{k=1}^{K} \|z_{l,k}\|_1$$
$$\text{s.t.} \quad \|d_k\|_2^2 \le 1, \quad k = 1, \ldots, K, \tag{1}$$

where $\{d_k \in \mathbb{C}^D : k = 1, \ldots, K\}$ is a set of synthesis convolutional filters to be learned, $\{y_l \in \mathbb{C}^N : l = 1, \ldots, L\}$ is a set of training data, $\circledast$ denotes a circular convolution operator, $\{z_{l,k} \in \mathbb{C}^N : l = 1, \ldots, L, k = 1, \ldots, K\}$ is a set of sparse codes, $P_B \in \mathbb{C}^{N \times \tilde{N}}$ is a projection matrix with $|B| = N$ and $N < \tilde{N}$, and B is a list of distinct indices from the set $\{1, \ldots, \tilde{N}\}$ that correspond to truncating the boundaries of the padded convolution $\sum_{k=1}^{K} d_k \circledast z_{l,k}$. Here, D is the filter size, K is the number of convolution operators, N is the dimension of training data, $\tilde{N}$ is the dimension after convolution with padding,[3] and L is the number of training images. Note that D is much smaller than $\tilde{N}$ in general. Using P_B to eliminate boundary artifacts is useful because CDL can be sensitive to the convolution boundary conditions; see Section I for details. In sparse data settings, one can generalize B to $\{B_l : |B_l| = S_l < N, l = 1, \ldots, L\}$, where B_l contains the indices of (randomly) collected samples from y_l [6, §4.3], or the indices of the non-zero elements of the lth sparse signal, for $l = 1, \ldots, L$.

Using Parseval's relation [6], [14], [16], [17], problem (1) is equivalent to the following joint optimization problem (in the frequency domain):

$$\min_{\{d_k\},\{\tilde{z}_l\}} \sum_{l=1}^{L} \frac{1}{2} \left\| y_l - P_B \left[\Phi^{-1} \mathrm{diag}(\Phi P_S^T d_1) \Phi \right.\right.$$
$$\left.\left. \cdots \; \Phi^{-1} \mathrm{diag}(\Phi P_S^T d_K) \Phi \right] \tilde{z}_l \right\|_2^2 + \alpha \|\tilde{z}_l\|_1$$
$$\text{s.t.} \quad \|d_k\|_2^2 \le 1, \qquad k = 1, \ldots, K, \tag{2}$$

where Φ denotes the $\tilde{N}$-point 2D (unnormalized) discrete Fourier transform (DFT), $P_S^T \in \mathbb{C}^{\tilde{N} \times D}$ is zero-padding matrix, S is a list of indices that correspond to a small support of the filter with $|S| = D$ (again, $D \ll \tilde{N}$), where $\tilde{z}_l = [\tilde{z}_{l,1}^H, \ldots, \tilde{z}_{l,K}^H]^H \in \mathbb{C}^{K\tilde{N}}$, and $\{\tilde{z}_{l,k} \in \mathbb{C}^{\tilde{N}} : l = 1, \ldots, L, k = 1, \ldots, K\}$ denotes sparse codes. In general, $\tilde{(\cdot)}$ and $\hat{(\cdot)}$ denote a padded signal vector and a transformed vector in frequency domain, respectively.

AL methods are powerful, particularly for non-smooth optimization. An AL method was first applied to CDL with Parseval's theorem in [14], but without handling boundary artifacts. In [6], ADMM was first applied to solve (1). A similar spatial domain ADMM framework was introduced in [18] and [19]. These AL methods alternate between updating the dictionary $\{d_k\}$ (the filters) and updating the sparse codes $\{\tilde{z}_l\}$ (i.e., a two-block update), using AL (or ADMM) methods for each inner update. In [6], each filter and sparse code update consists of multiple iterations before switching to the other, whereas [14], [18] explored merging all the updates into a single set

[2]Throughout the paper, "desirable" solutions mean that *1)*; the learned filters capture structures of training images; *2)* the corresponding sparse codes are sufficiently sparse; *3)* the filters and sparse codes can properly synthesize training images through convolutional operators.

[3]The convolved signal has size of $\tilde{N} = (N_\mathrm{h}+K_\mathrm{h}-1) \times (N_\mathrm{w}+K_\mathrm{h}-1)$, where the original signal has size $N = N_\mathrm{h} \times N_\mathrm{w}$, the filter has size $K = K_\mathrm{h} \times K_\mathrm{w}$, and w and h denote the width and height, respectively.

of iterations. This single-set-of-iterations scheme based on AL method can be unstable because each filter and sparse code update no longer ensures monotone descent of the cost function. To improve its stability, one can apply the increasing ADMM parameter scheme [14, eq. (23)], the adaptive ADMM parameter selection scheme controlling primal and dual residual norms [25, §3.4.1], [18, §III-D], or interleaving schemes [14, Algorithm 1], [18, §V-B], [19]. However, it is difficult to obtain theoretical convergence guarantees (even for local convergence) for the AL algorithms using the single-set-of-iterations scheme; in addition, the techniques for reducing instability further complicate theoretical guarantees.

The following section introduces a practical BPG-M method consisting of a single set of updates that guarantees convergence for solving multi-convex problems like CDL.

III. Convergent Fast BPG-M and Adaptive Restarting

A. BPG-M – Setup

Consider the optimization problem

$$\min_{x \in \mathcal{X}} \; F(x_1, \ldots, x_B) := f(x_1, \ldots, x_B) + \sum_{b=1}^{B} r_b(x_b) \tag{3}$$

where variable x is decomposed into B blocks $x_1, \ldots, x_B$, the set $\mathcal{X}$ of feasible points is assumed to be closed and *block multi-convex* subset of $\mathbb{R}^n$, f is assumed to be a differentiable and *block multi-convex* function, and r_b are extended-value convex functions for $b = 1, \ldots, B$. A set $\mathcal{X}$ is called *block multi-convex* if its projection to each block of variable is convex, i.e., for each b and any fixed $B-1$ blocks $x_1, \ldots, x_{b-1}, x_{b+1}, \ldots, x_B$, the set

$$\begin{aligned} &\mathcal{X}_b(x_1, \ldots, x_{b-1}, x_{b+1}, \ldots, x_B) \\ &:= \left\{ x_b \in \mathbb{R}^{n_b} : (x_1, \ldots, x_{b-1}, x_b, x_{b+1}, \ldots, x_B) \in \mathcal{X} \right\} \end{aligned}$$

is convex. A function f is called *block multi-convex* if for each b, f is a convex function of x_b, when all the other blocks are fixed. In other words, when all blocks are fixed except one block, (3) over the free block is a convex problem. Extended-value means $r_b(x_b) = \infty$ if $x_b \notin \mathrm{dom}(r_b)$, for $b = 1, \ldots, B$. In particular, r_b can be indicator functions of convex sets. We use $r_1, \ldots, r_B$ to enforce individual constraints of $x_1, \ldots, x_B$, when they are present. Importantly, r_b can include nonsmooth functions.

In this paper, we are particularly interested in adopting the following quadratic *majorizer* (i.e., surrogate function) model of the composite function $\varrho(u) = \varrho_1(u) + \varrho_2(u)$ at a given point v to the block multi-convex problem (3):

$$\begin{aligned} \tilde{\varrho}_M(u, v) &= \psi_M(u; v) + \varrho_2(u), \\ \psi_M(u; v) &= \varrho_1(v) + \langle \nabla \varrho_1(v), u - v \rangle + \frac{1}{2} \|u - v\|_M^2 \end{aligned} \tag{4}$$

where $\varrho_1(u)$ and $\varrho_2(u)$ are two convex functions defined on the convex set $\mathcal{U}$, $\varrho_1(u)$ is differentiable, the majorizer $\psi_M(u; v)$ satisfies the following two conditions

$$\varrho_1(v) = \psi_M(v; v) \quad \text{and} \quad \varrho_1(u) \le \psi_M(u; v), \quad \forall u \in \mathcal{U}, \; \forall v,$$

and $M = M^T \succ 0$ is so-called *majorization matrix*. The majorizer $\tilde{\rho}_M(u, v)$ has the following unique minimizer

$$u^\star = \operatorname*{argmin}_{u \in \mathcal{U}} \frac{1}{2} \left\| u - \left(v - M^{-1} \nabla \varrho_1(v) \right) \right\|_M^2 + \varrho_2(u).$$

Note that decreasing the majorizer $\tilde{\varrho}_M(u, v)$ ensures a monotone decrease of the original cost function $\varrho(u)$. For example, a majorizer for $\varrho_1(u) = 1/2\|y - Au\|_2^2$ is given by

$$\psi_M(u; v) = \frac{1}{2} \left\| u - \left(v - M_A^{-1} A^T (Av - y) \right) \right\|_{M_A}^2, \tag{5}$$

where $A \in \mathbb{R}^{m \times n}$ and $M_A \in \mathbb{R}^{n \times n}$ is any majorization matrix for the Hessian $A^T A$ (i.e. $M_A \succeq A^T A$). Other examples include when ϱ_1 has Lipschitz-continuous gradient, or ϱ_1 is twice continuously differentiable and can be approximated with a majorization matrix $M \succ 0$ for the Hermitian $\nabla^2 \varrho_1(u) \succeq 0$, $\forall u \in \mathcal{U}$.

Based on majorizers of the form (4), the proposed method, BPG-M, is given as follows. To solve (3), we minimize F cyclically over each block $x_1, \ldots, x_B$, while fixing the remaining blocks at their previously updated values. Let $x_b^{(i+1)}$ be the value of x_b after its ith update, and

$$f_b^{(i)}(x_b) := f(x_1^{(i+1)}, \ldots, x_{b-1}^{(i+1)}, x_b, x_{b+1}^{(i)}, \ldots, x_B^{(i)}), \tag{6}$$

for all b, i. At the bth step of the ith iteration, we consider the updates

$$\begin{aligned} &x_b^{(i+1)} \\ &= \operatorname*{argmin}_{x_b \in \mathcal{X}_b^{(i)}} \langle \nabla f_b^{(i)}(\acute{x}_b^{(i)}), x_b - \acute{x}_b^{(i)} \rangle + \frac{1}{2} \left\| x_b - \acute{x}_b^{(i)} \right\|_{M_b^{(i)}}^2 + r_b(x_b) \\ &= \operatorname*{argmin}_{x_b \in \mathcal{X}_b^{(i)}} \frac{1}{2} \left\| x_b - \left(\acute{x}_b^{(i)} - \left(M_b^{(i)} \right)^{-1} \nabla f_b^{(i)}(\acute{x}_b^{(i)}) \right) \right\|_{M_b^{(i)}}^2 \\ &\qquad + r_b(x_b) \\ &= \mathrm{Prox}_{r_b} \left(\acute{x}_b^{(i)} - \left(M_b^{(i)} \right)^{-1} \nabla f_b^{(i)}(\acute{x}_b^{(i)}); M_b^{(i)} \right), \end{aligned}$$

where

$$\begin{aligned} \acute{x}_b^{(i)} &= x_b^{(i)} + W_b^{(i)} \left(x_b^{(i)} - x_b^{(i-1)} \right), \\ \mathcal{X}_b^{(i)} &= \mathcal{X}_b(x_1^{(i+1)}, \ldots, x_{b-1}^{(i+1)}, x_{b+1}^{(i)}, \ldots, x_B^{(i)}), \end{aligned}$$

$\nabla f_b^{(i)}(\acute{x}_b^{(i)})$ is the block-partial gradient of f at $\acute{x}_b^{(i)}$, $M_b^{(i)} \in \mathbb{R}^{n_b \times n_b}$ is a symmetric positive definite majorization matrix of $\nabla f_b^{(i)}(x_b)$, and the proximal operator is defined by

$$\mathrm{Prox}_r(y; M) := \operatorname*{argmin}_x \frac{1}{2} \|x - y\|_M^2 + r(x).$$

The $\mathbb{R}^{n_b \times n_b}$ matrix $W_b^{(i)} \succeq 0$, upper bounded by (9) below, is an *extrapolation matrix* that significantly accelerates convergence, in a similar manner to the extrapolation weight introduced in [33]. Algorithm 1 summarizes these updates.

B. BPG-M – Convergence Analysis

This section analyzes the convergence of Algorithm 1 under the following assumptions.

Algorithm 1 Block Proximal Gradient Method Using a Majorizer $\{M_b : b = 1, \ldots, B\}$ (BPG-M)

```
**Require:** $\{x_b^{(1)} = x_b^{(0)} : b = 1, \ldots, B\}$, $i = 1$
  **while** a stopping criterion is not satisfied **do**
    **for** $b = 1, \ldots, B$ **do**
      Calculate $M_b^{(i)}, W_b^{(i)}$ for $f_b^{(i)}(x_b)$ in (6)
      $\acute{x}_b^{(i)} = x_b^{(i)} + W_b^{(i)} \left( x_b^{(i)} - x_b^{(i-1)} \right)$
      $x_b^{(i+1)} = \mathrm{Prox}_{r_b}\left( \acute{x}_b^{(i)} - \left( M_b^{(i)} \right)^{-1} \nabla f_b^{(i)}(\acute{x}_b^{(i)}); M_b^{(i)} \right)$
    **end for**
    $i = i + 1$
  **end while**
```

Assumption 1) F in (3) is continuous in $\mathrm{dom}(F)$ and $\inf_{x \in \mathrm{dom}(F)} F(x) > -\infty$, and (3) has a Nash point (see Definition 3.1).

Assumption 2) The majorization matrix $M_b^{(i)}$ obeys $\beta I \preceq M_b^{(i)} \preceq M_b$ with $\beta > 0$ and a nonsingular matrix M_b, and

$$f_b^{(i)}(x_b^{(i+1)}) \le f_b^{(i)}(\acute{x}_b^{(i)}) + \langle \nabla f_b^{(i)}(\acute{x}_b^{(i)}), x_b^{(i+1)} - \acute{x}_b^{(i)} \rangle + \frac{1}{2} \left\| x_b^{(i+1)} - \acute{x}_b^{(i)} \right\|_{M_b^{(i+1)}}^2. \tag{7}$$

Assumption 3) The majorization matrices $M_b^{(i)}$ and extrapolation matrices $W_b^{(i)}$ are diagonalized by the same basis, $\forall i$.

The CDL problem (1) or (2) straightforwardly satisfies the continuity and the lower-boundedness of F in Assumption 1. To show this, consider that *1)* the sequence $\{d_k^{(i+1)}\}$ is in the bounded set $\mathcal{D} = \{d_k : \|d_k\|_2^2 \le 1, k = 1, \ldots, K\}$; *2)* the positive regularization parameter α ensures that the sequence $\{z_{l,k}^{(i+1)}\}$ (or $\{\tilde{z}_{l,k}^{(i+1)}\}$) is bounded (otherwise the cost would diverge). This applies to both the two-block and the multi-block BPG-M frameworks; see Section IV and V, respectively. Note that one must carefully design $M_b^{(i+1)}$ to ensure that Assumption 2 is satisfied; Sections IV-A.1 and IV-B.1 describe our designs for CDL. Using a tighter majorization matrix $M_b^{(i)}$ (i.e., tighter bound in (7)) is expected to accelerate the algorithm [34, Lemma 1]. Some examples that satisfy Assumption 3 include diagonal and circulant matrices (that are decomposed by canonical and Fourier basis, respectively). Assumptions 1–2 guarantee sufficient decrease of the objective function values.

We now recall the definition of a Nash point (or block coordinate-wise minimizer):

Definition 3.1 (A Nash Point [33, (2.3)—(2.4)]). *A Nash point (or block coordinate-wise minimizer) $\bar{x}$ is a point satisfying the Nash equilibrium condition. The Nash equilibrium condition of (3) is*

$$F(\bar{x}_1, \ldots, \bar{x}_{b-1}, \bar{x}_b, \bar{x}_{b+1}, \ldots, \bar{x}_B) \le F(\bar{x}_1, \ldots, \bar{x}_{b-1}, x_b, \bar{x}_{b+1}, \ldots, \bar{x}_B), \quad \forall x_b \in \bar{\mathcal{X}}_b, b \in [B],$$

which is equivalent to the following condition:

$$\langle \nabla_{x_b} f(\bar{x}) + \bar{g}_b, x_b - \bar{x}_b \rangle \ge 0, \quad \text{for all } x_b \in \bar{\mathcal{X}}_b \text{ and for some } \bar{g}_b \in \partial r_b(\bar{x}_b), \tag{8}$$

where $\bar{\mathcal{X}}_b = \mathcal{X}_b(\bar{x}_1, \ldots, \bar{x}_{b-1}, \bar{x}_{b+1}, \ldots, \bar{x}_B)$ and $\partial r(x_b)$ is the limiting subdifferential (see [35, §1.9], [36, §8]) of r at x_b.

In general, the Nash equilibrium condition (8) is weaker than the first-order optimality condition. For problem (3), a Nash point is not necessarily a critical point, but a critical point must be a Nash point [33, Remark 2.2].[4] This property is particularly useful to show convergence of limit points to a critical point, if one exists; see Remark 3.4.

Proposition 3.2 (Square Summability of $\|x^{(i+1)} - x^{(i)}\|_2$). *Under Assumptions 1–3, let $\{x^{(i+1)}\}$ be the sequence generated by Algorithm 1 with*

$$0 \preceq W_b^{(i)} \preceq \delta \left(M_b^{(i)} \right)^{-1/2} \left(M_b^{(i-1)} \right)^{1/2} \tag{9}$$

for $\delta < 1$ for all $b = 1, \ldots, B$ and i. Then

$$\sum_{i=1}^{\infty} \left\| x^{(i+1)} - x^{(i)} \right\|_2^2 < \infty.$$

Proof: See Section S.II of the supplementary material.

Proposition 3.2 implies that

$$\left\| x^{(i+1)} - x^{(i)} \right\|_2^2 \to 0. \tag{10}$$

Theorem 3.3 (A Limit Point Is a Nash Point). *If the assumptions in Proposition 3.2 hold, then any limit point of $\{x^{(i)}\}$ is a Nash point, i.e., it satisfies (8).*

Proof: See Section S.III of the supplementary material.

Remark 3.4. Theorem 3.3 implies that, if there exists a stationary point for (3), then any limit point of $\{x^{(i)}\}$ is a stationary point. One can further show global convergence under some conditions: if $\{x^{(i)}\}$ is bounded and the stationary points are isolated, then $\{x^{(i)}\}$ converges to a stationary point [33, Corollary 2.4].[5]

We summarize some important properties of the proposed BPG-M in CDL:

Summary 3.5. The proposed BPG-M approach exploits a majorization matrix rather than using a Lipschitz constant; therefore, it can be practically applied to CDL without any parameter tuning process (except the regularization parameter). The BPG-M guarantees the local convergence in (1) or (2), i.e., if there exists a critical point, any limit point of the BPG-M sequence is a critical point (it also guarantees the global convergence if some further conditions are satisfied; see Remark 3.4 for details). Note that this is the first convergence guarantee in CDL. The convergence rate of the BPG-M method depends on the tightness of the majorization matrix in (4); see, for example, Fig. 2. The next section describes variants of BPG-M that further accelerate its convergence.

[4]Given a feasible set $\mathcal{X}$, a point $\bar{x} \in \mathrm{dom}(f) \cup \mathcal{X}$ is a critical point (or stationary point) of f if $f'(\bar{x}; d) \ge 0$ for any feasible direction d at $\bar{x}$, where $f'(\bar{x}; d)$ denotes directional derivate ($f'(x; d) = d^T \nabla f(x)$ for differentiable f). If x is an interior point of $\mathcal{X}$, then the condition is equivalent to $0 \in \partial F(\bar{x})$.

[5]Due to the difficulty of checking the isolation condition, Xu & Yin [33] introduced a better tool to show global convergence based on Kurdyka-Łojasiewicz property.

C. Restarting Fast BPG-M

This section proposes a technique to accelerate BPG-M. By including *1)* a momentum coefficient formula similar to those used in FPG methods [28], [37], [38], and *2)* an adaptive momentum restarting scheme [39], [40], this section focuses on computationally efficient majorization matrices, e.g., diagonal or circulant majorization matrices.

Similar to [5], we apply some increasing momentum-coefficient formulas $w^{(i)}$ to the extrapolation matrix updates $W_b^{(i)}$ in Algorithm 1:

$$w^{(i)} = \frac{\theta^{(i-1)} - 1}{\theta^{(i)}}, \quad \theta^{(i)} = \frac{1 + \sqrt{1 + 4(\theta^{(i-1)})^2}}{2}, \quad \text{or} \quad (11)$$

$$w^{(i)} = \frac{\theta^{(i-1)} - 1}{\theta^{(i)}}, \quad \theta^{(i)} = \frac{i+2}{2}. \quad (12)$$

These choices guarantee fast convergence of FPG in [38] and [28]. The momentum coefficient update rule in (11) was applied to block coordinate updates in [33], [41]. For diagonal majorization matrices $M_b^{(i)}$, $M_b^{(i-1)}$, the extrapolation matrix update is given by

$$\left(W_b^{(i)}\right)_{j,j} = \delta \cdot \min\left\{ w^{(i)}, \left(\left(M_b^{(i)}\right)^{-1} M_b^{(i-1)} \right)_{j,j}^{1/2} \right\}, \quad (13)$$

where $\delta < 1$ appeared in (9), for $j = 1, \ldots, n$. (Alternatively, $(W_b^{(i)})_{j,j} = \min\{w^{(i)}, \delta((M_b^{(i)})^{-1} M_b^{(i-1)})_{j,j}^{1/2}\}$.) For circulant majorization matrices $M_b^{(i)} = \Phi_{n_b}^H \operatorname{diag}(\hat{m}_b^{(i)}) \Phi_{n_b}$, $M_b^{(i-1)} = \Phi_{n_b}^H \operatorname{diag}(\hat{m}_b^{(i-1)}) \Phi_{n_b}$, we have the extrapolation matrix updates as follows:

$$W_b^{(i)} = \left(\Phi_{n_b}^H\right)^{1/2} \widehat{W}_b^{(i)} \Phi_{n_b}^{1/2} \quad (14)$$

where Φ_{n_b} is a unitary DFT matrix of size $n_b \times n_b$ and $\widehat{W}_b^{(i)} \in \mathbb{R}^{n_b \times n_b}$ is a diagonal matrix with entries

$$\left(\widehat{W}_b^{(i)}\right)_{j,j} = \delta \cdot \min\left\{ w^{(i)}, \left(\left(\hat{m}_{b,j}^{(i)}\right)^{-1} \hat{m}_{b,j}^{(i-1)} \right)^{1/2} \right\},$$

for $j = 1, \ldots, n$. We refer to BPG-M combined with the modified extrapolation matrix updates (13)–(14) using momentum coefficient formulas (11)–(12) as *Fast BPG-M* (FBPG-M). Note that convergence of FBPG-M is guaranteed because (9) in Proposition 3.2 still holds.

To further accelerate FBPG-M, we apply the adaptive momentum restarting scheme introduced in [39] and [40]. This technique restarts the algorithm by resetting the momentum back to zero and taking the current iteration as the new starting point, when a restarting criterion is satisfied. The *non-monotonicity* restarting scheme (referred to *reO*) can be used to make whole objective non-increasing [5], [39], [40]. The restarting criterion for this method is given by

$$\begin{aligned} &F(\bar{x}_1, \ldots, \bar{x}_{b-1}, \bar{x}_b, \bar{x}_{b+1}, \ldots, \bar{x}_B) \\ &\leq F(\bar{x}_1, \ldots, \bar{x}_{b-1}, x_b, \bar{x}_{b+1}, \ldots, \bar{x}_B), \quad \forall x_b \in \bar{\mathcal{X}}_b, b \in [B], \end{aligned} \quad (15)$$

However, evaluating the objective in each iteration is computationally expensive and can become an overhead as one increases the number of filters and the size of training datasets.

Algorithm 2 Restarting Fast Block Proximal Gradient Using a Diagonal Majorizer $\{M_b : b = 1, \ldots, B\}$ and Gradient-Mapping Scheme (reG-FBPG-M)

```
Require: $\{x_b^{(1)} = x_b^{(0)} : b = 1, \ldots, B\}$, $\theta^{(1)} = \theta^{(0)} = 1$,
  $\delta \in [0,1), \omega \in [-1,0]$, $i = 1$
  while a stopping criterion is not satisfied do
    Update $w^{(i)}$ using either (11) or (12)
    for $b = 1, \ldots, B$ do
      Calculate $M_b^{(i)}$ for $f_b^{(i)}(x_b)$ in (6)
      Calculate $W_b^{(i)}$ by (13) with $M_b^{(i)}, M_b^{(i-1)}, w^{(i)}$
      $\acute{x}_b^{(i)} = x_b^{(i)} + W_b^{(i)} \left( x_b^{(i)} - x_b^{(i-1)} \right)$
      $x_b^{(i+1)} = \mathrm{Prox}_{r_b}\Big( \acute{x}_b^{(i)} - \left(M_b^{(i)}\right)^{-1} \nabla f_b^{(i)}(\acute{x}_b^{(i)}); M_b^{(i)} \Big)$
      if $\cos\Big(\Theta\Big(M_b^{(i)}\Big(\acute{x}_b^{(i)} - x_b^{(i+1)}\Big), x_b^{(i+1)} - x_b^{(i)}\Big)\Big) > \omega$
      then
        $\acute{x}_b^{(i)} = x_b^{(i)}$
        $x_b^{(i+1)} = \mathrm{Prox}_{r_b}\Big( \acute{x}_b^{(i)} - \left(M_b^{(i)}\right)^{-1} \nabla f_b^{(i)}(\acute{x}_b^{(i)}); M_b^{(i)} \Big)$
      end if
    end for
    $i = i + 1$
  end while
```

Therefore, we introduce a *gradient-mapping* scheme (referred to *reG*) that restarts the algorithm when the following criterion is met:

$$\cos\left(\Theta\left(M_b^{(i)} \left(\acute{x}_b^{(i)} - x_b^{(i+1)} \right), x_b^{(i+1)} - x_b^{(i)} \right) \right) > \omega, \quad (16)$$

where the angle between two nonzero real vectors ϑ and ϑ' is

$$\Theta(\vartheta, \vartheta') := \frac{\langle \vartheta, \vartheta' \rangle}{\|\vartheta\|_2 \|\vartheta'\|_2},$$

and $\omega \in [-1, 0]$. The gradient-mapping scheme restarts the algorithm whenever the momentum, i.e., $x_b^{(i+1)} - x_b^{(i)}$, is likely to lead the algorithm in a bad direction, as measured by the gradient mapping (which is a generalization of the gradient, i.e., $M_b^{(i)}(\acute{x}_b^{(i)} - x_b^{(i+1)})$) at the $x_b^{(i+1)}$-update. The gradient-mapping criterion (16) is a relaxed version of the gradient-based restarting technique introduced in [39] and [40]. Compared to those in [39] and [40], the relaxed criterion often provides a faster convergence at the early iterations in practice [42].

To solve the multi-convex optimization problem (2), we apply Algorithm 2, promoting stable and fast convergence. We minimize (2) by the proposed BPG-M using the two-block and multi-block schemes; see Section IV and Section V, respectively—each section presents efficiently computable separable majorizers and introduces efficient proximal mapping methods.

IV. Convergent CDL: FBPG-M With Two-Block Update

Based on the FBPG-M method in the previous section, we first solve (2) by the two-block scheme, i.e., similar to the AL methods, we alternatively update filters $\{d_k : k =$

$1,\ldots,K\}$ and sparse codes $\{\tilde{z}_l : l = 1,\ldots,L\}$. The two-block scheme is particularly useful with parallel computing, because proximal mapping problems are separable (see Sections IV-A.2 and IV-B.2) and some majorization matrices computations are parallelizable.

A. Dictionary (Filter) Update

1) Separable Majorizer Design: Using the current estimates of the $\{\tilde{z}_l : l = 1,\ldots,L\}$, the filter update problem for (2) is given by

$$\begin{aligned}\min_{\{d_k\}} \quad & \frac{1}{2}\sum_{l=1}^{L} \Big\| y_l - P_B \Big[\Phi^{-1}\mathrm{diag}(\Phi P_S^T d_1)\Phi \\ & \cdots\ \Phi^{-1}\mathrm{diag}(\Phi P_S^T d_K)\Phi\Big]\tilde{z}_l \Big\|_2^2 \\ \text{s.t.} \quad & \|d_k\|_2^2 \le 1, \qquad k = 1,\ldots,K,\end{aligned}$$

which can be rewritten as follows:

$$\begin{aligned}\min_{\{d_k\}} \quad & \frac{1}{2}\left\| \begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix} - \Psi \begin{bmatrix} d_1 \\ \vdots \\ d_K \end{bmatrix} \right\|_2^2 \\ \text{s.t.} \quad & \|d_k\|_2^2 \le 1, \qquad k = 1,\ldots,K,\end{aligned} \tag{17}$$

where

$$\Psi := \left(I_L \otimes P_B \Phi^{-1}\right) \widehat{Z} \left(I_K \otimes \Phi P_S^T\right), \tag{18}$$

$$\widehat{Z} := \begin{bmatrix} \mathrm{diag}(\hat{z}_{1,1}) & \cdots & \mathrm{diag}(\hat{z}_{1,K}) \\ \vdots & \ddots & \vdots \\ \mathrm{diag}(\hat{z}_{L,1}) & \cdots & \mathrm{diag}(\hat{z}_{L,K}) \end{bmatrix}. \tag{19}$$

and $\{\hat{z}_{l,k} = \Phi \tilde{z}_{l,k} : l = 1,\ldots,L, k = 1,\ldots,K\}$. We now design block separable majorizer for the Hessian matrix $\Psi^H\Psi \in \mathbb{R}^{KD\times KD}$ of the cost function in (17). Using $\Phi^{-H}P_B^T P_B \Phi^{-1} \preceq \tilde{N}^{-1} I$ and $\Phi^H = \tilde{N}\Phi^{-1}$, $\Psi^H\Psi$ is bounded by

$$\begin{aligned}\Psi^H\Psi &\preceq \left(I_K \otimes P_S\Phi^{-1}\right)\widehat{Z}^H\widehat{Z}\left(I_K \otimes \Phi P_S^T\right) \\ &= (I_K \otimes P_S)\, Q_\Psi^H Q_\Psi \left(I_K \otimes P_S^T\right)\end{aligned} \tag{20}$$

where $\widehat{Z}^H\widehat{Z}$ is given according to (19), $Q_\Psi^H Q_\Psi \in \mathbb{C}^{\tilde{N}K\times\tilde{N}K}$ is a block matrix with submatrices $\{[Q_\Psi^H Q_\Psi]_{k,k'} \in \mathbb{C}^{\tilde{N}\times\tilde{N}} : k,k' = 1,\ldots,K\}$:

$$[Q_\Psi^H Q_\Psi]_{k,k'} := \Phi^{-1}\sum_{l=1}^{L}\mathrm{diag}(\hat{z}_{l,k}^* \odot \hat{z}_{l,k'})\Phi. \tag{21}$$

Based on the bound (20), our first diagonal majorization matrix for $\Psi^H\Psi$ is given as follows:

Lemma 4.1 (Block Diagonal Majorization Matrix M_Ψ with Diagonals I). *The following block diagonal matrix $M_\Psi \in \mathbb{R}^{KD\times KD}$ with diagonal blocks satisfies $M_\Psi \succeq \Psi^H\Psi$:*

$$M_\Psi = \mathrm{diag}\left((I_K \otimes P_S)\,|Q_\Psi^H Q_\Psi|\left(I_K \otimes P_S^T\right)1_{KD}\right),$$

where $Q_\Psi^H Q_\Psi$ is defined in (21) and $|A|$ denotes the matrix consisting of the absolute values of the elements of A.

Proof: See Section S.V-A of the supplementary material.

We compute $|Q_\Psi^H Q_\Psi|$ by taking the absolute values of elements of the first row (or column) of each circulant submatrix $[Q_\Psi^H Q_\Psi]_{k,k'}$ for $k,k' = 1,\ldots,K$. Throughout the paper, we apply this simple trick to efficiently compute the element-wise absolute value of the circulant matrices (because circulant matrices can be fully specified by a single vector). The computational complexity for the majorization matrix in Lemma 4.1 involves $\mathcal{O}(K^2\ L\tilde{N})$ operations for $\widehat{Z}^H\widehat{Z}$ and approximately $\mathcal{O}(K^2\tilde{N}\log\tilde{N})$ operations for $Q_\Psi^H Q_\Psi$. The permutation trick for a block matrix with diagonal blocks in [14] (see details in [43, Remark 3]) allows parallel computation of $\widehat{Z}^H\widehat{Z}$ over $j = 1,\ldots,\tilde{N}$, i.e., each thread requires $\mathcal{O}(K^2\ L)$ operations. Using Proposition 4.2 below, we can substantially reduce the latter number of operations at the cost of looser bounds (i.e., slower convergence).

Proposition 4.2. *The following block diagonal matrix $M_{Q_\Psi} \in \mathbb{R}^{\tilde{N}K\times\tilde{N}K}$ satisfies $M_{Q_\Psi} \succeq Q_\Psi^H Q_\Psi$:*

$$M_{Q_\Psi} = \bigoplus_{k=1}^{K}\Phi^{-1}\Sigma_k\Phi, \tag{22}$$

$$\Sigma_k = \sum_{l=1}^{L}\mathrm{diag}(|\hat{z}_{l,k}|^2) + \sum_{k'\neq k}\left|\sum_{l=1}^{L}\mathrm{diag}(\hat{z}_{l,k}^*\odot\hat{z}_{l,k'})\right|, \tag{23}$$

for $k = 1,\ldots,K$.

Proof: See Section S.IV-A of the supplementary material.

We now substitute (22) into (20). Unfortunately, the resulting $KD\times KD$ block-diagonal matrix below is inconvenient for inverting:

$$\Psi^H\Psi \preceq \begin{bmatrix} P_S\Phi^{-1}\Sigma_1\Phi P_S^T & & \\ & \ddots & \\ & & P_S\Phi^{-1}\Sigma_K\Phi P_S^T \end{bmatrix}.$$

Using some bounds for the block diagonal matrix intertwined with P_S and P_S^T above, the following two lemmas propose two separable majorization matrices for $\Psi^H\Psi$.

Lemma 4.3 (Block Diagonal Majorization Matrix M_Ψ with Scaled Identities). *The following block diagonal matrix $M_\Psi \in \mathbb{R}^{KD\times KD}$ with scaled identity blocks satisfies $M_\Psi \succeq \Psi^H\Psi$:*

$$\begin{aligned} M_\Psi &= \bigoplus_{k=1}^{K}[M_\Psi]_{k,k}, \\ [M_\Psi]_{k,k} &= \max_{j=1,\ldots,\tilde{N}}\left\{(\Sigma_k)_{j,j}\right\}\cdot I_K, \quad k\in[K],\end{aligned}$$

where diagonal matrices $\{\Sigma_k\}$ are as in (23).

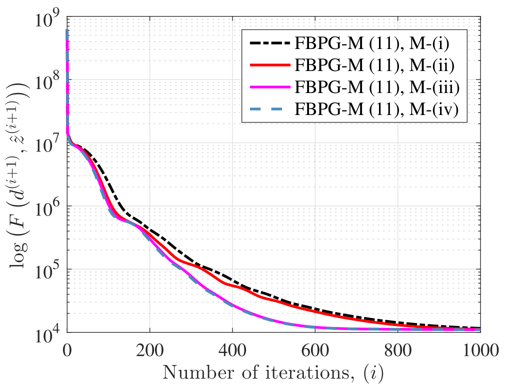
Proof: See Section S.V-B of the supplementary material.

Lemma 4.4 (Block Diagonal Majorization Matrix M_Ψ with Diagonals II). *The following block diagonal matrix $M_\Psi \in \mathbb{R}^{KD\times KD}$ with diagonal blocks satisfies $M_\Psi \succeq \Psi^H\Psi$:*

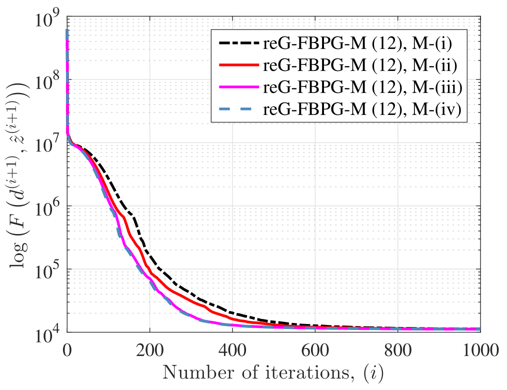
$$\begin{aligned} M_\Psi &= \bigoplus_{k=1}^{K}[M_\Psi]_{k,k}, \\ [M_\Psi]_{k,k} &= \mathrm{diag}\left(P_S\left|\Phi^{-1}\Sigma_k\Phi\right|P_S^T 1_K\right), \quad k\in[K],\end{aligned}$$

where diagonal matrices $\{\Sigma_k\}$ are as in (23).

Proof: See Section S.V-C of the supplementary material.



(a) Cost minimization with FBPG-M (11) using different majorizers



(b) Cost minimization with reG-FBPG-M (12) using different majorizers

Fig. 2. Cost minimization behavior for different majorizer designs (the fruit dataset). As the majorizer changes from M-(i) to M-(iv), we expect to have a tighter majorizer of the Hessian. (See Table I for details of majorization matrix design.) As expected, tighter majorizers lead to faster convergence.

The majorization matrix designs in Lemma 4.3 and 4.4 reduce the number of operations $\mathcal{O}(K^2\tilde{N}\log\tilde{N})$ to $\mathcal{O}(K\tilde{N})$ and $\mathcal{O}(K\tilde{N}\log\tilde{N})$, respectively. If parallel computing is applied over $k = 1,\ldots,K$, each thread requires $\mathcal{O}(\tilde{N})$ and $\mathcal{O}(\tilde{N}\log N)$ operations for Lemma 4.3 and 4.4, respectively. However, the majorization matrix in Lemma 4.1 is tighter than those in Lemma 4.3–4.4 because those in Lemma 4.3–4.4 are designed based on another bound. Fig. 2 verifies that the tighter majorizer leads to faster convergence. Table II summarizes these results.

2) Proximal Mapping: Because all of our majorization matrices are block diagonal, using (5) the proximal mapping problem (17) simplifies to separate problems for each filter:

$$d_k^{(i+1)} = \underset{d_k}{\operatorname{argmin}}\ \frac{1}{2}\left\| d_k - \nu_k^{(i)} \right\|^2_{\left[M_\Psi^{(i)}\right]_{k,k}} \quad \text{s.t.} \quad \|d_k\|_2^2 \leq 1, \quad k = 1,\ldots,K, \tag{24}$$

where

$$\nu^{(i)} = \acute{d}^{(i)} - \left(M_\Psi^{(i)}\right)^{-1}\left(\Psi^{(i)}\right)^H\left(\Psi^{(i)}\acute{d}^{(i)} - y\right),$$

we construct $\Psi^{(i)}$ using (18) with updated sparse codes $\{\tilde{z}_l^{(i)} : l = 1,\ldots,L\}$, $M_\Psi^{(i)}$ is a designed block diagonal majorization

TABLE I

NAME CONVENTIONS FOR BPG-M ALGORITHMS AND MAJORIZATION MATRIX DESIGNS

BPG-M names†	Momentum coeff.	Restarting scheme
FBPG-M	(11) or (12)	·
reO-BPG-M	·	Objective – (15)
reG-BPG-M	·	Gradient – (16)
reG-FBPG-M	(11) or (12)	Gradient – (16)

Majorizer names	Majorization matrix for filter updates	Majorization matrix for sparse coding
M-(i), two-block	Lemma 4.3	Lemma 4.7
M-(ii), two-block	Lemma 4.4	Lemma 4.7
M-(iii), two-block	Lemma 4.4	Lemma 4.5
M-(iv), two-block	Lemma 4.1	Lemma 4.5
M-(v), multi-block	Lemma 5.1	Lemma 5.2

†The non-monotonicity restarting scheme based on the objective is referred to as *reO*. The gradient-mapping restarting scheme is referred to as *reG*.

matrix for $(\Psi^{(i)})^H\Psi^{(i)}$, y is a concatenated vector with $\{y_l\}$, and $\nu^{(i)}$ is a concatenated vector with $\{\nu_k^{(i)} \in \mathbb{R}^K : k = 1,\ldots,K\}$. When $\{[M_\Psi^{(i)}]_{k,k}\}$ is a scaled identity matrix (i.e., Lemma 4.3), the optimal solution is simply the projection of $\nu_k^{(i)}$ onto the ℓ^2 unit ball. If $\{[M_\Psi^{(i)}]_{k,k}\}$ is a diagonal matrix (Lemma 4.1 and 4.4), the proximal mapping requires an iterative scheme. We apply accelerated Newton's method to efficiently obtain the optimal solution to (24); see details in Section S.VI.

B. Sparse Code Update

1) Separable Majorizer Design: Given the current estimates of the $\{\lambda_k = \Phi P_S^T d_k : k = 1,\ldots,K\}$, the sparse code update problem for (2) becomes L separate optimization problems:

$$\min_{\tilde{z}_l}\ \frac{1}{2}\|y_l - \Gamma\tilde{z}_l\|_2^2 + \alpha\|\tilde{z}_l\|_1, \tag{25}$$

for $l = 1,\ldots,L$, where

$$\Gamma := P_B\left[\ \Phi^{-1}\mathrm{diag}(\lambda_1)\Phi\ \cdots\ \Phi^{-1}\mathrm{diag}(\lambda_K)\Phi\ \right]. \tag{26}$$

We now seek a block separable majorizer for the Hessian matrix $\Gamma^H\Gamma \in \mathbb{C}^{\tilde{N}K\times\tilde{N}K}$ of the quadratic term in (25). Using $\Phi^{-H}P_B^T P_B\Phi^{-1} \preceq \tilde{N}^{-1}I$ and $\Phi^H = \tilde{N}\Phi^{-1}$, $\Gamma^H\Gamma$ is bounded as follows:

$$\Gamma^H\Gamma \preceq \left(I_K \otimes \Phi^{-1}\right)\Lambda^H\Lambda\left(I_K\otimes\Phi\right) = Q_\Gamma^H Q_\Gamma \tag{27}$$

where $\Lambda^H\Lambda$ is given according to

$$\Lambda := \left[\mathrm{diag}(\lambda_1),\cdots,\mathrm{diag}(\lambda_K)\right], \tag{28}$$

and $Q_\Gamma^H Q_\Gamma \in \mathbb{C}^{\tilde{N}K\times\tilde{N}K}$ is a block matrix with submatrices $\{[Q_\Gamma^H Q_\Gamma]_{k,k'} \in \mathbb{C}^{\tilde{N}\times\tilde{N}} : k,k' = 1,\ldots,K\}$:

$$[Q_\Gamma^H Q_\Gamma]_{k,k'} = \Phi^{-1}\mathrm{diag}(\lambda_k^*\odot\lambda_{k'})\Phi. \tag{29}$$

The following lemma describes our first diagonal majorization matrix for $\Gamma^H\Gamma$.

TABLE II
COMPARISON OF COMPUTATIONAL COMPLEXITY IN COMPUTING DIFFERENT MAJORIZATION MATRICES

Majorizer[†]	A. Computations for majorization matrix in filter updates[‡]
M-(i), two-block	Multi-thread: $\mathcal{O}(K^2 L) + \mathcal{O}(\tilde{N})$ Single-thread: $\mathcal{O}(K^2 L\tilde{N}) + \mathcal{O}(K\tilde{N})$
M-(ii), two-block	Multi-thread: $\mathcal{O}(K^2 L) + \mathcal{O}(\tilde{N}\log\tilde{N})$ Single-thread: $\mathcal{O}(K^2 L\tilde{N}) + \mathcal{O}(K\tilde{N}\log\tilde{N})$
M-(iii), two-block	Multi-thread: $\mathcal{O}(K^2 L) + \mathcal{O}(\tilde{N}\log\tilde{N})$ Single-thread: $\mathcal{O}(K^2 L\tilde{N}) + \mathcal{O}(K\tilde{N}\log\tilde{N})$
M-(iv), two-block	Multi-thread: $\mathcal{O}(K^2 L) + \mathcal{O}(K^2\tilde{N}\log\tilde{N})$ Single-thread: $\mathcal{O}(K^2 L\tilde{N}) + \mathcal{O}(K^2\tilde{N}\log\tilde{N})$
M-(v), multi-block	$\mathcal{O}(K(L\tilde{N} + \tilde{N}\log\tilde{N}))$

Majorizer[†]	B. Computations for majorization matrix in sparse code updates[‡]
M-(i), two-block	Multi-thread: $\mathcal{O}(K^2) + \mathcal{O}(\tilde{N}\log\tilde{N})$ Single-thread: $\mathcal{O}(K^2\tilde{N}) + \mathcal{O}(K\tilde{N}\log\tilde{N})$
M-(ii), two-block	Multi-thread: $\mathcal{O}(K^2) + \mathcal{O}(\tilde{N}\log\tilde{N})$ Single-thread: $\mathcal{O}(K^2\tilde{N}) + \mathcal{O}(K\tilde{N}\log\tilde{N})$
M-(iii), two-block	Multi-thread: $\mathcal{O}(K^2) + \mathcal{O}(K^2\tilde{N}\log\tilde{N})$ Single-thread: $\mathcal{O}(K^2\tilde{N}) + \mathcal{O}(K^2\tilde{N}\log\tilde{N})$
M-(iv), two-block	Multi-thread: $\mathcal{O}(K^2) + \mathcal{O}(K^2\tilde{N}\log\tilde{N})$ Single-thread: $\mathcal{O}(K^2\tilde{N}) + \mathcal{O}(K^2\tilde{N}\log\tilde{N})$
M-(v), multi-block	$\mathcal{O}(K\tilde{N}D)$ or $\mathcal{O}(K(\tilde{N}\log\tilde{N}))$

[†]The majorizer name follows the name convention in Table I.
[‡]For two-block BPG-M, the values in *multi-thread* denote computational costs at each thread when parallel computing is applied. The computational costs for multi-block BPG-M are estimated when multi-core processing is not used.

Lemma 4.5 (Block Diagonal Majorization Matrix M_Γ with Diagonals I). *The following block diagonal matrix $M_\Gamma \in \mathbb{R}^{\tilde{N}K\times\tilde{N}K}$ with diagonal blocks satisfies $M_\Gamma \succeq \Gamma^H\Gamma$:*

$$M_\Gamma = \mathrm{diag}\left(|Q_\Gamma^H Q_\Gamma| 1_{\tilde{N}K}\right),$$

where $Q_\Gamma^H Q_\Gamma$ is defined in (29).

Proof: See Section S.V-A of the supplementary material.

Computing the majorization matrix in Lemma 4.5 involves $\mathcal{O}(K^2\tilde{N})$ operations for $\Lambda^H\Lambda$ and approximately $\mathcal{O}(K^2\tilde{N}\log\tilde{N})$ operations for $Q_\Gamma^H Q_\Gamma$. Again, applying the permutation trick in [14] and [43, Remark 3] allows computing $\Lambda^H\Lambda$ by parallelization over $j = 1,\ldots,\tilde{N}$, i.e., each thread requires $\mathcal{O}(K^2)$ operations. Similar to the filter update case, Proposition 4.6 below substantially reduces the computational cost $\mathcal{O}(K^2\tilde{N}\log\tilde{N})$ at the cost of slower convergence.

Proposition 4.6. *The following block diagonal matrix $M_{Q_\Gamma} \in \mathbb{R}^{\tilde{N}K\times\tilde{N}K}$ satisfies $M_{Q_\Gamma} \succeq Q_\Gamma^H Q_\Gamma$.*

$$M_{Q_\Gamma} = \bigoplus_{k=1}^{K} \Phi^{-1}\Sigma_k'\Phi, \tag{30}$$

$$\Sigma_k' = \mathrm{diag}(|\lambda_k|^2) + \sum_{k'\neq k} \left|\mathrm{diag}(\lambda_k^* \odot \lambda_{k'})\right|, \tag{31}$$

for $k = 1,\ldots,K$.

Proof: See Section S.IV-B of the supplementary material.

Lemma 4.7 (Block Diagonal Majorization Matrix M_Γ with Diagonals II). *The following block diagonal matrix $M_\Gamma \in \mathbb{R}^{\tilde{N}K\times\tilde{N}K}$ with diagonal blocks satisfies $M_\Gamma \succeq \Gamma^H\Gamma$:*

$$M_\Psi = \bigoplus_{k=1}^{K} [M_\Psi]_{k,k},$$

$$[M_\Psi]_{k,k} = \mathrm{diag}\left(P_S \left|\Phi^{-1}\Sigma_k\Phi\right| P_S^T 1_K\right), \quad k \in [K],$$

where diagonal matrices $\{\Sigma_k'\}$ are as in (31).

Proof: See Section S.V-C of the supplementary material.

The majorization matrix in Lemma 4.7 reduces the cost $\mathcal{O}(K^2\tilde{N}\log\tilde{N})$ (of computing that in Lemma 4.5) to $\mathcal{O}(K\tilde{N}\log\tilde{N})$. Parallelization can further reduce computational complexity to $\mathcal{O}(\tilde{N}\log\tilde{N})$. However, similar to the majorizer designs in the filter update, the majorization matrix in Lemma 4.5 is expected to be tighter than those in Lemma 4.7 because the majorization matrix in Lemma 4.4 is designed based on another bound. Fig. 2 illustrates that tighter majorizers lead to faster convergence. Table II summarizes these results.

2) Proximal Mapping: Using (5), the corresponding proximal mapping problem of (25) is given by:

$$\tilde{z}_l^{(i+1)} = \underset{\tilde{z}_l}{\mathrm{argmin}}\ \frac{1}{2}\left\|\tilde{z}_l - \zeta_l^{(i)}\right\|_{M_\Gamma^{(i)}}^2 + \alpha\|\tilde{z}_l\|_1 \tag{32}$$

where

$$\zeta_l^{(i)} = \acute{z}_l^{(i)} - \left(M_\Gamma^{(i)}\right)^{-1}\left(\Gamma^{(i)}\right)^H\left(\Gamma^{(i)}\acute{z}_l^{(i)} - y_l\right),$$

we construct $\Gamma^{(i)}$ using (26) with updated kernels $\{d_k^{(i+1)} : k = 1,\ldots,K\}$, $M_\Gamma^{(i)}$ is a designed majorization matrix for $(\Gamma^{(i)})^H\Gamma^{(i)}$, and $\zeta_l^{(i)}$ is a concatenated vector with $\{\zeta_{l,k}^{(i)} \in \mathbb{R}^{\tilde{N}} : k = 1,\ldots,K\}$, for $l = 1,\ldots,L$. Using the circulant majorizer in Proposition 4.6 would require an iterative method for proximal mapping. For computational efficiency in proximal mapping, we focus on diagonal majorizers, i.e., Lemma 4.5 and 4.7. Exploiting the structure of diagonal majorization matrices, the solution to (32) is efficiently computed by soft-shrinkage:

$$\left(\tilde{z}_{l,k}^{(i+1)}\right)_j = \mathrm{softshrink}\left(\left(\zeta_{l,k}^{(i)}\right)_j, \alpha\left(\left[M_\Gamma^{(i)}\right]_{k,k}\right)_{j,j}^{-1}\right),$$

for $k = 1,\ldots,K$, $j = 1,\ldots,\tilde{N}$, where the soft-shrinkage operator is defined by $\mathrm{softshrink}(a,b) := \mathrm{sign}(a)\max(|a| - b, 0)$.

Note that one does not need to use $\Gamma^{(i)}$ in (26) (or $(\Gamma^{(i)})^H$) directly. If the filter size D is smaller than $\log\tilde{N}$, it is more efficient to use (circular) convolutions, by considering that the computational complexities for $d_k \circledast z_{l,k}$ and $\Phi^{-1}\mathrm{diag}(\lambda_k)\Phi\tilde{z}_{l,k}$ are $\mathcal{O}(\tilde{N}D)$ and $\mathcal{O}(\tilde{N}\log\tilde{N})$, respectively. This scheme analogously applies to $\Psi^{(i)}$ in (18) in the filter update.

V. Accelerated Convergent CDL: FBPG-M With Multi-Block Update

This section establishes a multi-block BPG-M framework for CDL that is particularly useful for single-thread computation mainly due to *1)* more efficient majorization matrix computations and *2)* (possibly) tighter majorizer designs than those in the two-block methods. In single-thread computing, it is desired to reduce the computational cost for majorizers, by noting that without parallel computing, the computational cost—but disregarding majorizer computation costs—in the two-block scheme is $\mathcal{O}(KL(\tilde{N}\log\tilde{N} + D/L + \tilde{N}))$ and identical to that of the multi-block approach. While guaranteeing convergence, the multi-block BPG-M approach accelerates the convergence rate of the two-block BPG-M methods in the previous section, with a possible reason that the majorizers of the multi-block scheme are tighter than those of the two-block scheme.

We update $2 \cdot K$ blocks sequentially; at the kth block, we sequentially update the kth filter—d_k—and the set of kth sparse codes for each training image—$\{z_{l,k} : l = 1, \ldots, L\}$ (referred to the kth *sparse code set*). One could alternatively randomly shuffle the K blocks at the beginning of each cycle [44] to further accelerate convergence. The mathematical decomposing trick used in this section (specifically, (33) and (36)) generalizes a sum of outer products of two vectors in [4] and [45].

*A. k*th *Dictionary (Filter) Update*

We decompose the $\{d_k : k = 1, \ldots, K\}$-update problem (17) into K d_k-update problems as follows:

$$\min_{d_k} \ \frac{1}{2}\left\| \left(\begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix} - \sum_{k' \neq k} \Psi_{k'} d_{k'} \right) - \Psi_k d_k \right\|_2^2 \quad \text{s.t.} \ \ \|d_k\|_2^2 \leq 1, \tag{33}$$

where the kth submatrix of $\Psi = [\Psi_1 \cdots \Psi_K]$ (18) is defined by

$$\Psi_k := \begin{bmatrix} P_B \Phi^{-1} \mathrm{diag}(\hat{z}_{1,k}) \Phi P_S^T \\ \vdots \\ P_B \Phi^{-1} \mathrm{diag}(\hat{z}_{L,k}) \Phi P_S^T \end{bmatrix}, \tag{34}$$

$\{\hat{z}_{l,k} = \Phi \tilde{z}_{l,k} : l = 1, \ldots, L\}$, and we use the most recent estimates of all other filters and coefficients in (33) and (34), for $k = 1, \ldots, K$.

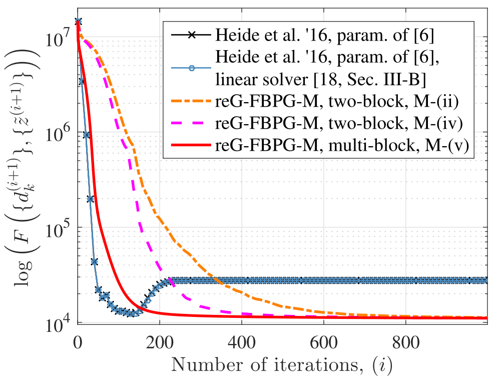
1) Separable Majorizer Design: The following lemma introduces a majorization matrix for $\Psi_k^H \Psi_k$:

Lemma 5.1 (Diagonal Majorization Matrix M_{Ψ_k}). *The following diagonal matrix $M_{\Psi_k} \in \mathbb{R}^{D \times D}$ satisfies $M_{\Psi_k} \succeq \Psi_k^H \Psi_k$:*

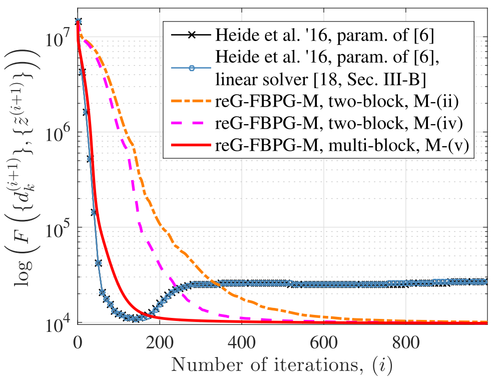
$$M_{\Psi_k} = \mathrm{diag}\left(P_S \left| \Phi^{-1} \sum_{l=1}^{L} \mathrm{diag}(|\hat{z}_{l,k}|^2) \Phi \right| P_S^T 1_D \right).$$

Proof: See Section S.V-D of the supplementary material.

The design in Lemma 5.1 is expected to be tighter than those in Lemma 4.3 and 4.4, because we use fewer bounds in designing it. Fig. 3 supports this expectation through convergence rate; additionally, Fig. 3 illustrates that the majorization matrix in Lemma 5.1 is expected to be tighter than that in Lemma 4.1. Another benefit of the majorization matrix in Lemma 5.1 is lower computational complexity than those in the two-block approaches (in single-thread computing). As shown in Table II-A, it allows up to $2K$ times faster the majorizer computations in the multi-block scheme (particularly, that in Lemma 4.1).



(a) The fruit dataset



(b) The city dataset

Fig. 3. Comparison of cost minimization between different CDL algorithms (the small datasets; for ADMM [6], the number of whole iterations is the product of the number of inner iterations and that of outer iterations; reG-FBPG-M used the momentum-coefficient formula (11)). The multi-block framework significantly improves the convergence rate over the two-block schemes, with a possible reason that the majorizer in the multi-block update, i.e., M-(v), is tighter than those in the two-block update, i.e., M-(i)–M-(iv).

2) Proximal Mapping: Using (5), the corresponding proximal mapping problem of (33) is given by

$$d_k^{(i+1)} = \underset{d_k}{\mathrm{argmin}} \ \frac{1}{2}\left\| d_k - \nu_k^{(i)} \right\|_{M_{\Psi_k}}, \quad \text{s.t.} \ \|d_k\|_2^2 \leq 1, \tag{35}$$

where

$$\nu_k^{(i)} = \acute{d}_k^{(i)} - \left(M_{\Psi_k}^{(i)}\right)^{-1} \left(\Psi_k^{(i)}\right)^H \left(\Psi_k^{(i)} \acute{d}_k^{(i)} - \check{y}\right),$$

$$\check{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix} - \sum_{k' \neq k} \Psi_{k'} d_{k'},$$

we construct $\Psi_k^{(i)}$ using (34) with the updated kth sparse code set $\{\tilde{z}_{l,k}^{(i)} : l = 1, \ldots, L\}$, and $M_{\Psi_k}^{(i)}$ is a designed diagonal majorization matrix for $(\Psi_k^{(i)})^H \Psi_k^{(i)}$. Similar to Section IV-A.2, we apply the accelerated Newton's method in Section S.VI to efficiently solve (35).

B. *k*th *Sparse Code Set Update*

We decompose the $\{\tilde{z}_{l,k} : k = 1, \ldots, K\}$-update problem (25) into K $\tilde{z}_{l,k}$-update problems as follows:

$$\min_{\tilde{z}_{l,k}} \frac{1}{2} \left\| \left(y_l - \sum_{k' \neq k} \Gamma_{k'} \tilde{z}_{l,k'} \right) - \Gamma_k \tilde{z}_{l,k} \right\|_2^2 + \alpha \|\tilde{z}_{l,k}\|_1, \quad (36)$$

where the kth submatrix of $\Gamma = [\Gamma_1 \cdots \Gamma_K]$ (26) is defined by

$$\Gamma_k := P_B \Phi^{-1} \mathrm{diag}(\lambda_k) \Phi, \quad (37)$$

$\{\lambda_k = \Phi P_S^T d_k\}$, we use the most recent estimates of all other filters and coefficients in (36) and (34), for $k = 1, \ldots, K$. Using (36), we update the kth set of sparse codes $\{\tilde{z}_{l,k} : l = 1, \ldots, L\}$, which is easily parallelizable over $l = 1, \ldots, L$. Note, however, that this parallel computing scheme does not provide computational benefits over that in the two-block approach. Specifically, in each thread, the two-block scheme requires $\mathcal{O}(K\tilde{N}(\log \tilde{N} + 1))$; and the multi-block requires K times the cost $\mathcal{O}(\tilde{N}(\log \tilde{N} + 1))$, i.e., $\mathcal{O}(K\tilde{N}(\log \tilde{N} + 1))$.

1) Separable Majorizer Design: Applying Lemma S.3, our diagonal majorization matrix for $\Gamma_k^H \Gamma_k$ is given in the following lemma:

Lemma 5.2 (Diagonal Majorization Matrix M_{Γ_k}). *The following diagonal matrix $M_{\Gamma_k} \in \mathbb{R}^{\tilde{N} \times \tilde{N}}$ satisfies $M_{\Gamma_k} \succeq \Gamma_k^H \Gamma_k$:*

$$M_{\Gamma_k} = \mathrm{diag}\left(|\Gamma_k^H| |\Gamma_k| 1_{\tilde{N}} \right).$$

The design in Lemma 5.2 is expected to be tighter than those in Lemma 4.5 and 4.7, because only a single bound is used in designing it. Fig. 3 supports our expectation through convergence rate per iteration. In addition, the design in Lemma 5.2 requires lower computation costs than those in the two-block schemes (in a single processor computing). Specifically, it reduces complexities of computing those in multi-block scheme (particularly, that in Lemma 4.5) by up to a factor of $K(1/D + 1)$; see Table II-B.

2) Proximal Mapping: Using (5), the corresponding proximal mapping problem of (36) is given by:

$$\tilde{z}_{l,k}^{(i+1)} = \underset{\tilde{z}_{l,k}}{\mathrm{argmin}} \ \frac{1}{2} \left\| \tilde{z}_{l,k} - \zeta_{l,k}^{(i)} \right\|_{\Gamma_k^{(i)}}^2 + \alpha \|\tilde{z}_{l,k}\|_1 \quad (38)$$

where

$$\zeta_{l,k}^{(i)} = \acute{z}_{l,k}^{(i)} - \left(M_{\Gamma_k}^{(i)} \right)^{-1} \left(\Gamma_k^{(i)} \right)^H \left(\Gamma_k^{(i)} \acute{z}_{l,k}^{(i)} - \check{y}_l \right),$$
$$\check{y}_l = y_l - \sum_{k' \neq k} \Gamma_{k'} \tilde{z}_{l,k'},$$

we construct $\Gamma_k^{(i)}$ using (37) with the updated kth filter $d_k^{(i+1)}$, and $M_{\Gamma_k}^{(i)}$ is a designed diagonal majorization matrix for $(\Gamma_k^{(i)})^H \Gamma_k^{(i)}$. Similar to Section IV-B.2, problem (38) is solved by the soft-shrinkage operator.

To efficiently compute $\sum_{k' \neq k} \Psi_{k'} d_{k'}$ in (33) and $\sum_{k' \neq k} \Gamma_{k'} \tilde{z}_{l,k'}$ in (36) at the kth iteration, we update and store $\{\Gamma_k \tilde{z}_{l,k} : l = 1, \ldots, L\}$—which is identical to $\Psi_k d_k$—with newly estimated $d_k^{(i+1)}$ and $\{\tilde{z}_{l,k}^{(i+1)} : l = 1, \ldots, L\}$, and simply take sum in $\sum_{k' \neq k} \Psi_{k'} d_{k'}$ and $\sum_{k' \neq k} \Gamma_{k'} \tilde{z}_{l,k'}$. Similar to $\Gamma^{(i)}$ in (26) and $\Psi^{(i)}$ in (18), one can perform $\Gamma_k^{(i)}$ in (37) (or $(\Gamma_k^{(i)})^H$) and $\Psi_k^{(i)}$ in (34) in a spatial domain—see Section IV-A.2.

VI. CDL-ACE: Application of CDL to Image Denoising

Applying learned filters by CDL to some inverse problems is not straightforward due to *model mismatch* between training and testing stages. CDL conventionally learns features from preprocessed training datasets (by, for example, the techniques in Section VII-A); however, such nonlinear preprocessing techniques are not readily incorporated when solving inverse problems [46].

The most straightforward approach in resolving the model mismatch is to learn filters from non-preprocessed training data, as noted in [6, §5.2]. An alternative approach is to model (linear) contrast enhancement methods in CDL—similar to CDL-ACE in [22]—and apply them to solving inverse problems. The CDL-ACE model is given by [22]

$$\begin{aligned} \min_{\{d_k\},\{z_{l,k}\},\{\rho_l\}} \ & \sum_{l=1}^{L} \frac{1}{2} \left\| y_l - \left(P_B \sum_{k=1}^{K} d_k \circledast z_{l,k} \right) - \rho_l \right\|_2^2 \\ & + \alpha \sum_{k=1}^{K} \left\| z_{l,k} \right\|_1 + \frac{\gamma}{2} \|C\rho_l\|_2^2 \\ \text{s.t.} \ & \|d_k\|_2^2 \leq 1, \quad k = 1, \ldots, K, \end{aligned} \quad (39)$$

where $\{\rho_l \in \mathbb{R}^N : l = 1, \ldots, L\}$ is a set of low-frequency component vectors and we design $C \in \mathbb{R}^{N' \times N}$ for adaptive contrast enhancement of $\{y_l\}$ (see below). Considering particular boundary conditions (e.g. periodic or reflective) for $\{\rho_l\}$, we rewrite (39) as follows [22]:

$$\begin{aligned} \min_{\{d_k\},\{z_{l,k}\}} \ & \sum_{l=1}^{L} \frac{1}{2} \left\| \tilde{y}_l - R\left(P_B \sum_{k=1}^{K} d_k \circledast z_{l,k} \right) \right\|_2^2 + \alpha \sum_{k=1}^{K} \left\| z_{l,k} \right\|_1 \\ \text{s.t.} \ & \|d_k\|_2^2 \leq 1, \quad k = 1, \ldots, K, \end{aligned} \quad (40)$$

where $\{\tilde{y}_l := R y_l : l = 1, \ldots, L\}$ and

$$R := \left(\gamma C^T C \right)^{1/2} \left(\gamma C^T C + I \right)^{-1/2}. \quad (41)$$

The matrix R in (41) can be viewed as a simple form of a contrast enhancing transform (without divisive normalization by local variances), e.g., $R^T R y = y - (\gamma C^T C + I)^{-1} y$, where $(\gamma C^T C + I)^{-1}$ is a low-pass filter. To solve (40), AL methods would now require six additional AL parameters to tune and consume more memory (than the ADMM approach in [6] solving (1)); however, BPG-M methods are free from the additional parameter tuning processes and memory issues.

To denoise a measured image $b \in \mathbb{R}^n$ corrupted by AWGN ($\sim \mathcal{N}(0, \sigma^2)$), we solve the following optimization problem with the filters $\{d_k^\star : k = 1, \ldots, K\}$ learned via the CDL models, i.e., (1) or, optimally, (40), [22]:

$$\begin{aligned} \left\{ \{a_k^\star\}, \rho^\star \right\} = \underset{\{a_k\},\rho}{\mathrm{argmin}} \ & \frac{1}{2} \left\| b - \left(P_B \sum_{k=1}^{K} d_k^\star \circledast a_k \right) - \rho \right\|_2^2 \\ & + \alpha' \sum_{k=1}^{K} \|a_k\|_1 + \gamma' \|C\rho\|_2^2, \end{aligned} \quad (42)$$

and synthesize the denoised image by $P_B \sum_{k=1}^{K} d_k^\star \circledast a_k^\star + \rho^\star$, where $\{a_k \in \mathbb{R}^n\}$ is a set of sparse codes, $\rho \in \mathbb{R}^n$ is a low-frequency component vectors, and $C \in \mathbb{R}^{n' \times n}$ is a regularization transform modeled in the CDL model (39). Using the reformulation techniques in (39)–(40), we rewrite (42) as a convex problem and solve it through FPG method using a diagonal majorizer (designed by a technique similar to Lemma 4.7) and adaptive restarting [22].

VII. Results and Discussion

A. Experimental Setup

Table I gives the naming conventions for the proposed BPG-M algorithms and designed majorizers.

We tested all the introduced CDL algorithms for two types of datasets: preprocessed and non-preprocessed. The preprocessed datasets include the fruit and city datasets with $L = 10$ and $N = 100 \times 100$ [6], [12], and the CT dataset with $L = 10$ and $N = 512 \times 512$ from down-sampled 512×512 XCAT phantom slices [47]—referred to the CT-(i) dataset. The preprocessing includes local contrast normalization [23, Adaptive Deconvolutional Networks Toolbox], [48, §2], [12] and intensity rescaling to [0, 1] [12], [14], [23], and [6]. The non-preprocessed dataset [6, §5.2] consists of XCAT phantom images of $L = 80$ and $N = 128 \times 128$, created by dividing down-sampled 512×512 XCAT phantom slices [47] into 16 sub-images [7], [14]; we refer this to the CT-(ii) dataset. Both the preprocessed and non-preprocessed datasets contain zero-mean training images (i.e., by subtracting the mean from each training image [23, Adaptive Deconvolutional Networks Toolbox], [48, §2]; note that subtracting the mean can be omitted for the preprocessed datasets), as conventionally used in many (convolutional) dictionary learning studies, e.g., [5], [6], [12], [14], [23], [48]. For image denoising experiments, we additionally trained filters by CDL-ACE (40) through the BPG-M method, and the non-preprocessed city datasets (however, note that we do not apply the mean subtraction step because it is not modeled in (40). For all the CDL experiments, we trained filters of $D = 11 \times 11$ and $K = 100$ [6], [19].

The parameters for the algorithms were defined as follows. For CDL (1) using both the preprocessed and non-preprocessed datasets, we set the regularization parameters as $\alpha = 1$ [6]. For CDL-ACE (40) using the non-preprocessed dataset, we set $\alpha = 0.4$. We used the same (normally distributed) random initial filters and coefficients for each training dataset to fairly compare different CDL algorithms. We set the tolerance value, tol in (44), as 10^{-4}. Specific details regarding the algorithms are described below.

Comparing convergence rates in Fig. 3 and execution time in Table III-C, we normalized the initial filters such that $\{\|d_k\|_2^2 \le 1 : k = 1, \ldots, K\}$ (we empirically observed that the normalized initial filters improve convergence rates of the multi-block algorithms, but marginally improve convergence rates of the two-block algorithms—for both ADMMs and BPG-M). The execution time in Table III was recorded by (double precision) MATLAB implementations based on Intel Core i5 with 3.30 GHz CPU and 32 GB RAM.

1) ADMM [6]: We first selected ADMM parameters as suggested in the corresponding MATLAB code of [6]: the ADMM parameters were selected by considering the maximum value of $\{y_m : m = 1, \ldots, M\}$, similar to [31]. We used 10 inner iterations (i.e., $\mathrm{Iter}_{\mathrm{ADMM}}$ in Table III-A) for each kernel and sparse code update [6] and set the maximum number of outer iterations to 100. We terminated the iterations if either of the following stopping criteria are met before reaching the maximum number of iterations [6]:

$$F(d^{(i+1)}, \tilde{z}^{(i)}) \ge F(d^{(i)}, \tilde{z}^{(i)}) \quad \text{and} \quad F(d^{(i+1)}, \tilde{z}^{(i+1)}) \ge F(d^{(i)}, \tilde{z}^{(i)}), \tag{43}$$

or

$$\frac{\|d^{(i+1)} - d^{(i)}\|_2}{\|d^{(i+1)}\|_2} < \mathrm{tol} \quad \text{and} \quad \frac{\|\tilde{z}^{(i+1)} - \tilde{z}^{(i)}\|_2}{\|\tilde{z}^{(i+1)}\|_2} < \mathrm{tol}, \tag{44}$$

where d and $\tilde{z}$ are concatenated vectors from $\{d_k\}$ and $\{\tilde{z}_{l,k}\}$, respectively. These rules were applied at the outer iteration loop [6]. For the experiments in Figs. 3 and S.2, and Table III-C, we disregarded the objective-value-based termination criterion (43). For a memory-efficient variant of ADMM [6], we replaced the direct solver in [6, eq. (11)] with the iterative method in [18, §III-B] to solve the linear system [6, eq. (10)], and tested it with the same parameter sets above.

2) BPG-M Algorithms: We first selected the parameter δ in (13) as $1 - \varepsilon$, where ε is the (double) machine epsilon value, similar to [5]. For the gradient-mapping restarting, we selected the parameter ω in (16) as $\cos(95^\circ)$, similar to [42]. For the accelerated Newton's method, we set the initial point $\varphi_k^{(0)}$ to 0, the tolerance level for $|\varphi_k^{(i'+1)} - \varphi_k^{(i')}|$ to 10^{-6}, and the maximum number of iterations to 10, for $k = 1, \ldots, K$. The maximum number of BPG-M iterations was set to Iter = 1000. We terminated the iterations if the relative error stopping criterion (44) was met before reaching the maximum number of iterations.

3) Image Denoising with Learned Filters via CDL: For image denoising applications, we corrupted a test image with relatively strong AWGN, i.e., SNR = 10 dB. We denoised the noisy image through the following methods (all the parameters were selected as suggested in [22], giving the best peak signal-to-noise ratio (PSNR) values): *1)* adaptive Wiener filtering with 3×3 window size; *2)* total variation (TV) with MFISTA using its regularization parameter 0.8σ and maximum number of iterations 200 [49]; *3)* image denoiser (42) with 100 (empirically) convergent filters trained by CDL model (1) (i.e., Fig. S.2(b)) and preprocessed training data, $\alpha' = 2.5\sigma$, the first-order finite difference for C in (42) [46], and $\gamma' = 10\sigma$; and *4)* (42) with 100 learned filters by CDL-ACE (39), $\alpha' = \alpha \cdot 5.5\sigma$, and $\gamma' = \gamma \cdot 5.5\sigma$. For (42), the stopping criteria is set similar to (44) (with tol $= 10^{-3}$) before reaching the maximum number of iterations 100.

B. BPG-M Versus ADMM [6] and Its Memory-Efficient Variant for CDL (1)

The BPG-M methods guarantee convergence without difficult parameter tuning processes. Figs. 3 and S.1–S.2 show that the BPG-M methods converge more stably than ADMM [6]

TABLE III

COMPARISON OF COMPUTATIONAL COMPLEXITY, EXECUTION TIME, AND MEMORY REQUIREMENT FROM DIFFERENT SINGLE-THREADED CDL ALGORITHMS

Algorithms	A. Computations for updating whole blocks in a single (outer) iteration[a]
ADMM in Heide et al. [6]	Case $K > L$ [6]: $\mathcal{O}\Big(KL^2\tilde{N} + (\mathrm{Iter}_{\mathrm{ADMM}} - 1) \cdot KL\tilde{N} + \mathrm{Iter}_{\mathrm{ADMM}} \cdot KL(\tilde{N}\log\tilde{N} + \tilde{N})\Big)$ Case $K \le L$ [6]: $\mathcal{O}\Big(K^3\tilde{N} + (\mathrm{Iter}_{\mathrm{ADMM}} - 1) \cdot K^2 L + \mathrm{Iter}_{\mathrm{ADMM}} \cdot KL(\tilde{N}\log\tilde{N} + \tilde{N}))\Big)$
ADMM in Heide et al. [6] w. linear solver [18, §III-B]	$\mathcal{O}\Big(\mathrm{Iter}_{\mathrm{ADMM}} \cdot \big(KL^2\tilde{N} + KL(\tilde{N}\log\tilde{N} + \tilde{N})\big)\Big)$
reG-FBPG-M, multi-block	$\mathcal{O}\Big(K \cdot L(\tilde{N}\log\tilde{N} + D/L + \tilde{N})\Big)$

Algorithms	B. Execution time[b] (hours : minutes)			
	Fruit	City	CT-(i)	CT-(ii)
ADMM in Heide et al. [6]	0 : 45	0 : 45	·	·
ADMM in Heide et al. [6] w. linear solver [18, §III-B]	0 : 53	0 : 53	·	44 : 12
reG-FBPG-M, multi-block	0 : 54	0 : 55	36 : 09	17 : 25

Algorithms	C. Total dimension of variables to be stored[c]	
	Filter update	Sparse code update
ADMM in Heide et al. [6]	$2\tilde{N}(2L+K)$	$2\tilde{N}L(2K+1) + K^2\tilde{N}$
ADMM in Heide et al. [6] w. linear solver [18, §III-B]	$2\tilde{N}(2L+K)$	$2\tilde{N}L(2K+1) + KL\tilde{N}$
reG-FBPG-M, multi-block	$D(3K+2)$	$\tilde{N}(2KL+K+2) + KLN$

[a]Here, $\mathrm{Iter}_{\mathrm{ADMM}}$ denotes the number of (inner) ADMM iterations in each block update in the ADMM framework in [6]. For fair comparison with ADMM [6], one should multiply the cost of a single iteration in reG-FBPG-M by $\mathrm{Iter}_{\mathrm{ADMM}}$.
[b]The symbol · means that the execution time cannot be recorded due to exceeding available memory.
[c]For the ADMM approach in [6], one must store previously updated filters $\{d_k^{(i)}\}$ and sparse codes $\{\tilde{z}_{l,k}^{(i)}\}$, because it outputs them instead of the current estimates $\{d_k^{(i+1)}\}$ and $\{\tilde{z}_{l,k}^{(i+1)}\}$, when it is terminated by the objective function criterion (43). The additional dimension on the second line of each method corresponds to the following: for ADMM [6], it is the dimension of variables to solve the linear systems [6, (10)] through the Parseval tricks in [6]; for ADMM [6] using linear solver [18, §III-B], it is the largest dimension to apply [18, §III-B] to solve [6, (10)]; for multi-block reG-FBPG-M, it is the dimension of $\{\Gamma_k \tilde{z}_{l,k} : l = 1, \ldots, L\}$ or $\Psi_k d_k$ for faster computation of $\sum_{k' \neq k} \Psi_{k'} d_{k'}$ in (33) and $\sum_{k' \neq k} \Gamma_{k'} \tilde{z}_{l,k'}$ in (36) (see details in Section V-B2).

and the memory-efficient variant of ADMM [6]. When the ADMM parameters are poorly chosen, for example simply using 1, the ADMM algorithm fails (see Table IV). The objective function termination criterion (43) can stabilize ADMM; however, note that terminating the algorithm with (43) is not a natural choice, because the monotonic decrease in objective function values is not guaranteed [33]. For the small datasets (i.e., the fruit and city datasets), the execution time

TABLE IV

COMPARISONS OF OBJECTIVE VALUES WITH DIFFERENT CONVOLUTIONAL DICTIONARY LEARNING ALGORITHMS

Algorithms[†]	Objective values	
	Fruit	City
ADMM [6], param. of [6]	12594	11057
ADMM [6], param. of [6], no termination by (43)	27415	26375
ADMM [6], param. 1	53504	53907
ADMM [6] w. linear solver [18, §III-B], param. of [6], no termination by (43)	27405	26372
FBPG-M, two-block, (11)	11013	9585
reO-BPG-M, two-block	11039	9606
reG-BPG-M, two-block	11081	9674
reG-FBPG-M, two-block, (11)	11068	9644
reG-FBPG-M, two-block, (12)	11149	9648
reG-FBPG-M, multi-block, (11)	10980	9698

[†]The two-block BPG-M algorithms used the M-(iv) majorizer.

of reG-FBPG-M using the multi-block scheme is comparable to that of the ADMM approach [6] and its memory-efficient variant; see Table III-B. Based on the numerical experiments in [19], for small datasets particularly with the small number of training images, the state-of-the-art ADMM approach in [19, AVA-MD] using the single-set-of-iterations scheme [18] (or [14]) can be faster than multi-block reG-FBPG-M; however, it lacks theoretical convergence guarantees and can result in non-monotone minimization behavior—see Section II and [19, Fig. 2, AVA-MD].

The proposed BPG-M-based CDL using the multi-block scheme is especially useful to large datasets having large image size or many images (compared to ADMM [6] and its memory-efficient variant applying linear solver [18, §III-B]):

- The computational complexity of BPG-M depends mainly on the factor $K \cdot L \cdot \tilde{N}\log\tilde{N}$; whereas that of ADMM [6] depends not only on the factor $K \cdot L \cdot \tilde{N}\log\tilde{N}$, but also on the approximated factors $K \cdot L^2 \cdot \tilde{N} \cdot \mathrm{Iter}_{\mathrm{ADMM}}^{-1}$ (for $K > L$) or $K^3 \cdot \tilde{N} \cdot \mathrm{Iter}_{\mathrm{ADMM}}^{-1}$ (for $K \le L$). The memory-efficient variant of ADMM [6] requires even higher computational complexity than ADMM [6]: it depends both on the factors $K \cdot L \cdot \tilde{N}\log\tilde{N}$ and $K \cdot L^2 \cdot \tilde{N}$. See Table III-A–B.
- The multi-block reG-FBPG-M method requires much less memory than ADMM [6] and its variant. In the filter updates, it only depends on the parameter dimensions of filters (i.e., K, D); however, ADMM requires the amount of memory depending on the dimensions of training images and the number of filters (i.e., $\tilde{N}, L, K$). In the sparse code updates, the multi-block reG-FBPG-M method requires about half the memory of ADMM. Additionally, there exists no K^2 factor dependence in multi-block reG-FBPG-M. The memory-efficient variant of ADMM [6] removes the K^2 factor dependence, but still requires higher memory than multi-block reG-FBPG-M. See Table III-C.

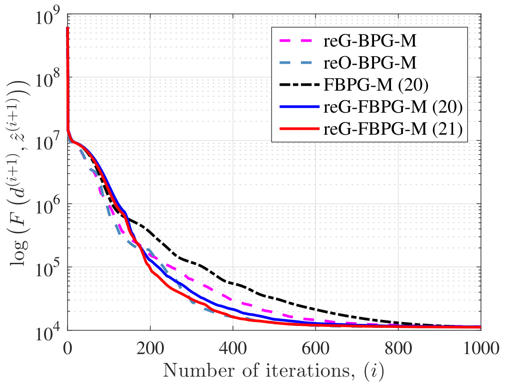
Table III-B shows that the ADMM approach in [6] and/or its memory-efficient variant fail to run CDL for the larger datasets (i.e., CT-(i) and CT-(ii)), due to its high memory usage. By not caching the inverted matrices [6, eq. (11)] computed at the beginning of each block update, the memory-efficient variant of ADMM [6] avoids the K^2 factor dependence in memory requirement. However, its computational cost now depends on the factor L^2 multiplied with K and $\tilde{N}$; this product becomes a serious computational bottleneck as L—the number of training images—grows. See the CT-(ii) column in Table III-B. (Note that single-set-of-iterations ADMM [19, AVA-MD] obeys the same trends.) Heide et al.'s report that their ADMM can handle the large dataset (of $L = 10$, $N = 1000 \times 1000$) that the patch-based method (i.e., K-SVD [4] using all patches) cannot, due to its high memory usage [6, §3.2, using 132 GB RAM machine]. Combining these results, the BPG-M-based CDL algorithm (particularly using the multi-block scheme) is a reasonable choice to learn dictionary from large datasets. Especially, the multi-block BPG-M method is well-suited to CDL with large datasets consisting of a large number of (relatively small-dimensional) signals—for example, the datasets are often generated by dividing (large) images into many sub-images [6], [14], [18].

For the non-preproccsed dataset (i.e., CT-(ii)), the proposed BPG-M algorithm (specifically, reG-FBPG-M using the multi-block scheme) properly converges to desirable solutions, i.e., the resultant filters and sparse codes (of sparsity 5.25%) properly synthesize training images. However, the memory-efficient variant of ADMM [6] does not converge to the desirable solutions. Compare the results in Fig. S.1(d) to those in Fig. S.2(c).

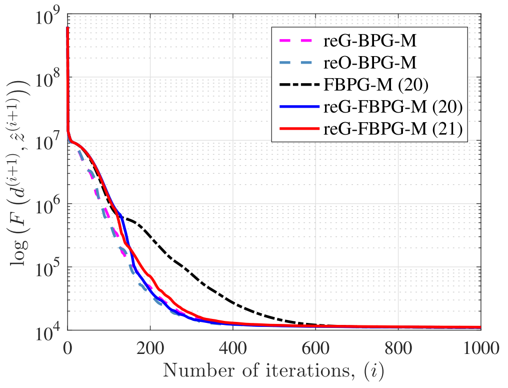
Figs. 2 and S.1–S.2 illustrate that all the proposed BPG-M algorithms converge to desirable solutions and reach lower objective function values than the ADMM approach in [6] and its memory-efficient variant (see Table IV and compare Fig. S.1(d) to Fig. S.2(c)). In particular, the tighter majorizer enables the BPG-M algorithms to converge faster; see Figs. 2–3. Interestingly, the restarting schemes (15)–(16) provide significant convergence acceleration over the momentum coefficient formula (11). The combination of reG (16) and momentum coefficient formulas (11)–(12), i.e., reG-FBPG-M, can be useful in accelerating the convergence rate of reG-BPG-M, particularly when majorizers are insufficiently tight. Fig. 4 supports these assertions. Most importantly, all the numerical experiments regarding the BPG-M methods are in good agreement with our theoretical results on the convergence analysis, e.g., Theorem 3.3 and Remark 3.4. Finally, the results in Table IV concur with the existing empirical results of comparison between BPG and BCD in [33] and [5], noting that ADMM in [6] is BCD-type method.

C. Application of Learned Filters by CDL to Image Denoising

The filters learned via convergent BPG-M-based CDL (1) show better image denoising performance than the (empirically) convergent ones trained by ADMM-based CDL (1) in [6]; it improves PSNR by approximately 1.6 dB. Considering that the BPG-M methods reach lower objective values than ADMM of [6], this implies that the filters of lower objective values can improve the CDL-based image denoiser (42). The learned filters by CDL-ACE (40) further improve image denoising compared to those trained by BPG-M-based CDL (1), by resolving the model mismatch; it improves PSNR by approximately 0.2 dB. Combining these, the CDL-based image denoiser using the learned filters by CDL-ACE (40) outperforms the TV denoising model. All these assertions are supported by Fig. 5. Finally, the CDL-ACE model (39) better captures structures of non-preprocessed training images than the CDL model (1); see [22, Fig. 2].



(a) Cost minimization with different accelerated BPG-M algorithms using M-(ii)



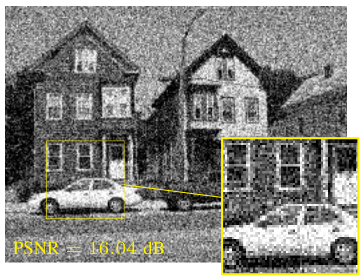
(b) Cost minimization with different accelerated BPG-M algorithms using M-(iii)

Fig. 4. Comparison of cost minimization between different accelerated BPG-M CDL algorithms (the fruit dataset).

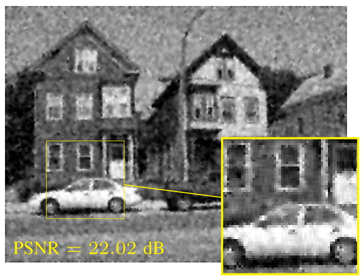
VIII. Conclusion

Developing convergent and stable algorithms for nonconvex problems is important and challenging. In addition, parameter tuning is a known challenge for AL methods. This paper has considered both algorithm acceleration and the above two important issues for CDL.

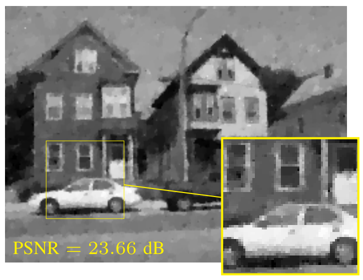
The proposed BPG-M methods have several benefits over the ADMM approach in [6] and its memory-efficient variant. First, the BPG-M algorithms guarantee local convergence (or global convergence if some conditions are satisfied) without additional parameter tuning (except regularization parameter). The BPG-M methods converge stably and empirically to a "desirable" solution regardless of the datasets. Second, particularly with the multi-block framework, they are useful for larger datasets due to their lower memory requirement and no polynomial computational complexity (specifically, no $\mathcal{O}(L^2\ K\tilde{N})$, $\mathcal{O}(K^2\ L)$, and $\mathcal{O}(K^3\tilde{N})$ complexity). Third,



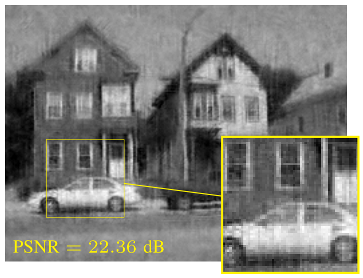
(a) Noisy image



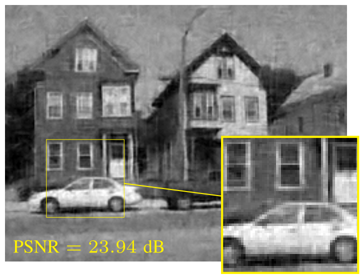
(b) Wiener filtering



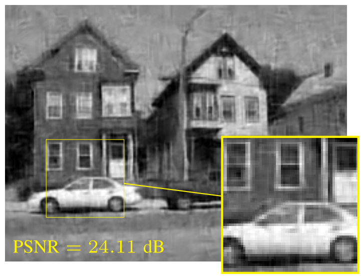
(c) TV denoiser



(d) CDL image denoiser (42) using learned filters via ADMM CDL (1)



(e) CDL image denoiser (42) using learned filters via BPG-M CDL (1)



(f) CDL image denoiser (42) using learned filters via BPG-M CDL (40)

Fig. 5. Comparison of denoised images from different image denoising models (image is corrupted by AWGN with SNR = 10 dB; for ADMM [6], we used (empirically) convergent learned filters; for BPG-M, we used the two-block reG-FBPG-M method using (12)). The image denoising model (42) using the learned filters by BPG-M-based CDL—(e)—shows better image denoising performance compared to (b) Wiener filtering, (c) TV denoising, and (d) that using the learned filters by ADMM-based CDL. The filters trained by CDL-ACE further improves (e)-image denoiser.

they empirically achieve lower objective values. Among the proposed BPG-M algorithms, the reG-FBPG-M scheme—i.e., BPG-M using gradient-mapping-based restarting and momentum coefficient formulas—is practically useful by due to its fast convergence rate and no requirements in objective value evaluation. The CDL-based image denoiser using learned filters via BPG-M-based CDL-ACE [22] outperforms Wiener filtering, TV denoising, and filters trained by the conventional ADMM-based CDL [6]. The proposed BPG-M algorithmic framework is a reasonable choice towards stable and fast convergent algorithm development in CDL with big data (i.e., training data with the large number of signals or high-dimensional signals).

There are a number of avenues for future work. First, in this paper, the global convergence guarantee in Remark 3.4 requires a stringent condition in practice. Future work will explore the more general global convergence guarantee based on the Kurdyka-Łojasiewicz property. Second, we expect to further accelerate BPG-M by using the stochastic gradient method while guaranteeing its convergence (the stochastic ADMM [50] improves the convergence rate of ADMM on convex problems, and is applied to convolutional sparse coding for image super-resolution [51]). Applying the proposed CDL algorithm to multiple-layer setup is an interesting topic for future work [24]. On the application side, we expect that incorporating normalization by local variances into CDL-ACE will further improve solutions to inverse problems.

Appendix
Notation

We use $\|\cdot\|_p$ to denote the ℓ^p-norm and write $\langle\cdot,\cdot\rangle$ for the standard inner product on $\mathbb{C}^N$. The weighted ℓ^2-norm with a Hermitian positive definite matrix A is denoted by $\|\cdot\|_A = \left\|A^{1/2}(\cdot)\right\|_2$. $\|\cdot\|_0$ denotes the ℓ^0-norm, i.e., the number of nonzeros of a vector. $(\cdot)^T$, $(\cdot)^H$, and $(\cdot)^*$ indicate the transpose, complex conjugate transpose (Hermitian transpose), and complex conjugate, respectively. $\mathrm{diag}(\cdot)$ and $\mathrm{sign}(\cdot)$ denote the conversion of a vector into a diagonal matrix or diagonal elements of a matrix into a vector and the sign function, respectively. $\otimes$, $\odot$, and $\bigoplus$ denote Kronecker product for two matrices, element-wise multiplication in a vector or a matrix, and the matrix direct sum of square matrices, respectively. $[C]$ denotes the set $\{1, 2, \ldots, C\}$. For self-adjoint matrices $A, B \in \mathbb{C}^{N\times N}$, the notation $B \preceq A$ denotes that $A - B$ is a positive semi-definite matrix.

Acknowledgment

We thank Dr. Donghwan Kim and Dr. Jonghoon Jin for constructive feedback.

References

[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Rev.*, vol. 51, no. 1, pp. 34–81, Feb. 2009.
[2] A. Coates and A. Y. Ng, "Learning feature representations with K-means," in *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science), vol. 7700, 2nd ed., G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Germany: Springer-Verlag, 2012, ch. 22, pp. 561–580.
[3] J. Mairal, F. Bach, and J. Ponce, "Sparse modeling for image and vision processing," *Found. Trends Comput. Graph. Vis.*, vol. 8, nos. 2–3, pp. 85–283, Dec. 2014.
[4] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
[5] Y. Xu and W. Yin, "A fast patch-dictionary method for whole image recovery," *Inverse Problems Imag.*, vol. 10, no. 2, pp. 563–583, May 2016.
[6] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *Proc. IEEE CVPR*, Boston, MA, USA, Jun. 2015, pp. 5135–5143.

[7] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, Jun. 1996.
[8] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vis. Res.*, vol. 37, no. 23, pp. 3311–3325, Dec. 1997.
[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25. 2012, pp. 1097–1105.
[12] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE CVPR*, San Francisco, CA, USA, Jun. 2010, pp. 2528–2535.
[13] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, "Learning convolutional feature hierarchies for visual recognition," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 23. 2010, pp. 1090–1098.
[14] H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," in *Proc. IEEE CVPR*, Portland, OR, USA, Jun. 2013, pp. 391–398.
[15] B. Wohlberg, "Efficient convolutional sparse coding," in *Proc. 39th IEEE ICASSP*, Florence, Italy, May 2014, pp. 7173–7177.
[16] B. Kong and C. C. Fowlkes. (2014). "Fast convolutional sparse coding (FCSC)," Dept. Comput. Sci., Univ. California, Irvine, CA, USA, Tech. Rep. 3. [Online]. Available: http://vision.ics.uci.edu/papers/KongF_TR_2014/KongF_TR_2014.pdf
[17] H. Bristow and S. Lucey. (Jun. 2014). "Optimization methods for convolutional sparse coding." [Online]. Available: https://arxiv.org/abs/1406.2407
[18] B. Wohlberg, "Efficient algorithms for convolutional sparse representations," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 301–315, Jan. 2016.
[19] B. Wohlberg, "Boundary handling for convolutional sparse representations," in *Proc. 23rd IEEE ICIP*, Phoenix, AZ, USA, Sep. 2016, pp. 1833–1837.
[20] M. Šorel and F. Šroubek, "Fast convolutional sparse coding using matrix inversion lemma," *Digit. Signal Process.*, vol. 55, pp. 44–51, Aug. 2016.
[21] V. Papyan, J. Sulam, and M. Elad. (2016). "Working locally thinking globally—Part II: Stability and algorithms for convolutional sparse coding." [Online]. Available: https://arxiv.org/abs/1607.02009
[22] I. Y. Chun and J. A. Fessler, "Convergent convolutional dictionary learning using adaptive contrast enhancement (CDL-ACE): Application of CDL to image denoising," in *Proc. 12th Int. Conf. Sampling Theory Appl. (SampTA)*, Tallinn, Estonia, Jul. 2017, pp. 460–464.
[23] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Proc. IEEE CVPR*, Colorado Springs, CO, USA, Jun. 2011, pp. 2018–2025.
[24] V. Papyan, Y. Romano, and M. Elad. (2016). "Convolutional neural networks analyzed via convolutional sparse coding." [Online]. Available: https://arxiv.org/abs/1607.08194
[25] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.
[26] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014.
[27] R. Chalasani, J. C. Principe, and N. Ramakrishnan, "A fast proximal method for convolutional sparse coding," in *Proc. IEEE IJCNN*, Dallas, TX, USA, Aug. 2013, pp. 1–5.
[28] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009.
[29] B. Chen, J. Li, B. Ma, and G. Wei, "Convolutional sparse coding classification model for image classification," in *Proc. 23rd IEEE ICIP*, Phoenix, AZ, USA, Sep. 2016, pp. 1918–1922.
[30] A. Matakos, S. Ramani, and J. A. Fessler, "Accelerated edge-preserving image restoration without boundary artifacts," *IEEE Trans. Image Process.*, vol. 22, no. 5, pp. 2019–2029, May 2013.
[31] M. S. C. Almeida and M. A. T. Figueiredo, "Deconvolving images with unknown boundaries using the alternating direction method of multipliers," *IEEE Trans. Image Process.*, vol. 22, no. 8, pp. 3074–3086, Aug. 2013.
[32] B. Wohlberg. (2017). *SParse Optimization Research COde (SPORCO)*. [Online]. Available: http://purl.org/brendt/software/sporco
[33] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. Imag. Sci.*, vol. 6, no. 3, pp. 1758–1789, Sep. 2013.
[34] J. A. Fessler, N. H. Clinthorne, and W. L. Rogers, "On complete-data spaces for PET reconstruction algorithms," *IEEE Trans. Nucl. Sci.*, vol. 40, no. 4, pp. 1055–1061, Aug. 1993.
[35] A. Y. Kruger, "On Fréchet subdifferentials," *J. Math. Sci.*, vol. 116, no. 3, pp. 3325–3358, Jul. 2003.
[36] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, vol. 317. Berlin, Germany: Springer-Verlag, 2009.
[37] Y. Nesterov, "Gradient methods for minimizing composite objective function," Univ. Catholique de Louvain, "Louvain-la-Neuve, Belgium, CORE Discussion Papers 2007/76, 2007. [Online]. Available: http://www.uclouvain.be/cps/ucl/doc/core/documents/Composit.pdf
[38] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," *SIAM J. Optim.*, May 2008. [Online]. Available: http://www.mit.edu/~dimitrib/PTseng/papers/apgm.pdf
[39] B. O'Donoghue and E. J. Candès, "Adaptive restart for accelerated gradient schemes," *Found. Comput. Math.*, vol. 15, no. 3, pp. 715–732, Jun. 2015.
[40] P. Giselsson and S. Boyd, "Monotonicity and restart in fast gradient methods," in *Proc. 53rd IEEE CDC*, Los Angeles, CA, USA, Dec. 2014, pp. 5058–5063.
[41] Y. Xu and W. Yin. (2014). "A globally convergent algorithm for nonconvex optimization based on block coordinate update." [Online]. Available: https://arxiv.org/abs/1410.1386
[42] M. J. Muckley, D. C. Noll, and J. A. Fessler, "Fast parallel MR image reconstruction via B1-based, adaptive restart, iterative soft thresholding algorithms (BARISTA)," *IEEE Trans. Med. Imag.*, vol. 34, no. 2, pp. 578–588, Feb. 2015.
[43] I. Y. Chun, S. Noh, D. J. Love, T. M. Talavage, S. Beckley, and S. J. Kisner, "Mean squared error based excitation pattern design for parallel transmit and receive SENSE MRI image reconstruction," *IEEE Trans. Comput. Imag.*, vol. 2, no. 4, pp. 424–439, Dec. 2016.
[44] P. Richtárik and M. Takáč, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function," *Math. Program.*, vol. 144, nos. 1–2, pp. 1–38, Apr. 2014.
[45] S. Ravishankar, R. R. Nadakuditi, and J. Fessler, "Efficient sum of outer products dictionary learning (SOUP-DIL) and its application to inverse problems," *IEEE Trans. Comput. Imag.*, to be published.
[46] A. Serrano, F. Heide, D. Gutierrez, G. Wetzstein, and B. Masia, "Convolutional sparse coding for high dynamic range imaging," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 153–163, May 2016.
[47] W. P. Segars, M. Mahesh, T. J. Beck, E. C. Frey, and B. W. M. Tsui, "Realistic CT simulation using the 4D XCAT phantom," *Med. Phys.*, vol. 35, no. 8, pp. 3800–3808, Jul. 2008.
[48] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. ICCV*, Kyoto, Japan, Sep. 2009, pp. 2146–2153.
[49] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2419–2434, Nov. 2009.
[50] L. W. Zhong and J. T. Kwok, "Fast stochastic alternating direction method of multipliers," in *Proc. IEEE ICML*, Beijing, China, Jun. 2014, pp. 46–54.
[51] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, "Convolutional sparse coding for image super-resolution," in *Proc. IEEE ICCV*, Santiago, Chile, Dec. 2015, pp. 1823–1831.

Convolutional Dictionary Learning: Acceleration and Convergence (Supplementary Material)

Il Yong Chun and Jeffrey A. Fessler

In this supplementary material for [1], we provide mathematical proofs or detailed descriptions to support several arguments in the main manuscript. We use the prefix "S" for the numbers in section, equation, figure, and table in the supplementary material.

S.I. Useful Lemmas and Their Proofs

Lemma S.1. *Let $\varrho_1(u)$ and $\varrho_2(u)$ be two convex functions defined on the convex set $\mathcal{U}$, $\varrho_1(u)$ be differentiable, and $M \succ 0$. Let $\varrho(u) = \varrho_1(u) + \varrho_2(u)$ and $u^\star = \mathrm{argmin}_{u \in \mathcal{U}} \langle \nabla \varrho_1(v), u - v \rangle + \frac{1}{2}\|u - v\|_M^2 + \varrho_2(u)$. If*

$$\varrho_1(u^\star) \leq \varrho_1(u) + \langle \nabla \varrho_1(v), u^\star - v \rangle + \frac{1}{2}\|u^\star - v\|_M^2, \quad \text{(S.1)}$$

then we have

$$\varrho(u) - \varrho(u^\star) \leq \frac{1}{2}\|u^\star - v\|_M^2 + (v-u)^T M (u^\star - v). \quad \text{(S.2)}$$

Proof. The following proof is an extension of that given in [2]. The first-order optimality condition for $u^\star = \mathrm{argmin}_{u \in \mathcal{U}} \langle \nabla \varrho_1(v), u - v \rangle + \frac{1}{2}\|u - v\|_M^2 + \varrho_2(u)$ is given by

$$\langle \nabla \varrho_1(v) + M(u^\star - v) + g^\star, u - u^\star \rangle \geq 0, \quad \text{for any } u \in \mathcal{U} \quad \text{(S.3)}$$

and for some $g \in \partial \varrho_2(u^\star)$. For any $u \in \mathcal{U}$, we obtain

$$\begin{aligned}
&\varrho(u) - \varrho(u^\star) \\
&\geq \varrho(u) - \left(\varrho_1(v) + \langle \nabla \varrho_1(v), u^\star - v \rangle + \frac{1}{2}\|u^\star - v\|_L^2 \right) \\
&\quad - \varrho_2(u^\star) \\
&= \varrho_1(u) - \varrho_1(v) - \langle \nabla \varrho_1(v), u - v \rangle + \langle \nabla \varrho_1(v), u - u^\star \rangle \\
&\quad + \varrho_2(u) - \varrho_2(u^\star) - \frac{1}{2}\|u^\star - v\|_2^2 \\
&\geq \varrho_2(u) - \varrho_2(u^\star) - \langle g^\star, u - u^\star \rangle - (u^\star - v)^T M (u - u^\star) \\
&\quad - \frac{1}{2}(u^\star - v)^T M (u^\star - v) \\
&\geq -(u^\star - v)^T M (u - u^\star) - \frac{1}{2}(u^\star - v)^T M (u^\star - v) \\
&= \frac{1}{2}\|u^\star - v\|_L^2 + (v - u)^T M (u^\star - v)
\end{aligned}$$

where the first inequality comes from (S.1), the second inequality is obtained by convexity of ϱ_1 (i.e., $\langle \nabla \varrho_1(v), u - v \rangle \leq \varrho_1(u) - \varrho_1(v)$) and (S.3), and the last inequality is obtained by the convexity of ϱ_2 and the fact $g^\star \in \partial \varrho_2(u^\star)$ (i.e., $\langle g^\star, u - u^\star \rangle \leq \varrho_2(u) - \varrho_2(u^\star) \leq 0$). This completes the proof. □

Lemma S.2. *If the diagonal elements of a Hermitian matrix A are nonnegative (e.g., if A is positive semidefinite), then $A \preceq \mathrm{diag}(|A|1)$, where $|A|$ denotes the matrix consisting of the absolute values of the elements of A.*

Proof. Let $E = \mathrm{diag}(|A|1) - A$. We seek to apply the property that, if a Hermitian matrix is diagonally dominant with nonnegative diagonal entries, then it is positive semidefinite. We first show that E is diagonally dominant. For $j = k$, we have

$$E_{j,j} = \sum_k |A_{j,k}| - A_{j,j} = \sum_{k \neq j} |A_{j,k}| \quad \text{(S.4)}$$

due to the assumption of $A_{j,j} \geq 0$ in Lemma S.2. For $j \neq k$, $\sum_{k \neq j} |E_{j,k}| = \sum_{k \neq j} |A_{j,k}| = E_{j,j}$ where the first equality uses $E_{j,k} = -A_{j,k}$ and the second equality uses (S.4). It is straightforward to show that E is a Hermitian matrix (due to $E_{j,k} = -A_{j,k}$) with nonnegative diagonal entries (due to (S.4)). Combining these results completes the proof. □

Lemma S.3. *For a complex-valued matrix A, $A^H A \preceq \mathrm{diag}(|A^H||A|1)$.*

Proof. Let $E = \mathrm{diag}(|A^H||A|1) - A^H A$. The jth diagonal element of $\mathrm{diag}(|A^H||A|1)$ is $\sum_l |A_{l,j}| \sum_k |A_{l,k}|$. We again seek to apply the property that, if a Hermitian matrix is diagonally dominant with nonnegative diagonal entries, then it is positive semidefinite. For $j = k$, observe that

$$\begin{aligned}
E_{j,j} &= \sum_l |A_{l,j}| \sum_k |A_{l,k}| - \sum_l |A_{l,j}|^2 \\
&= \sum_l |A_{l,j}| \left(|A_{l,j}| + \sum_{k \neq j} |A_{l,k}| \right) - |A_{l,j}|^2 \\
&= \sum_l |A_{l,j}| \sum_{k \neq j} |A_{l,k}| \quad \text{(S.5)} \\
&\geq 0
\end{aligned}$$

establishing nonnegative diagonal elements. For $j \neq k$, it follows from the triangle inequality that

$$\sum_{k \neq j} |E_{j,k}| = \sum_{k \neq j} \left| \sum_l A_{l,k}^* A_{l,j} \right| \leq \sum_{k \neq j} \sum_l |A_{l,k}||A_{l,j}|. \quad \text{(S.6)}$$

Combining (S.5) and (S.6) gives $E_{j,j} - \sum_{k \neq j} |E_{j,k}| \geq 0$, $\forall j$. Combining these results completes the proof. □

Supplementary material dated September 27, 2017.

Il Yong Chun and Jeffrey A. Fessler are with the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48019 USA (email: iychun@umich.edu; fessler@umich.edu).

S.II. Proof of Proposition 3.2

The following proof extends that given in [2]. Let $F_b^{(i)} := f_b^{(i)} + r_b$ and $\varrho_1 = f_b^{(i)}$, $\varrho_2 = r_b$, $u = x_b^{(i)}$, $v = \acute{x}_b^{(i+1)}$, $u^\star = x_b^{(i+1)}$, and $M = M_b^{(i)}$, by applying Lemma S.1 to (7). We first obtain the following bounds for $F_b^{(i)}(x_b^{(i)}) - F_b^{(i)}(x_b^{(i+1)})$:

$$\begin{aligned}
&F_b^{(i)}(x_b^{(i)}) - F_b^{(i)}(x_b^{(i+1)}) \\
&\geq \frac{1}{2}\left\|\acute{x}_b^{(i)} - x_b^{(i+1)}\right\|^2_{M_b^{(i)}} \\
&\quad + \left(\acute{x}_b^{(i)} - x_b^{(i+1)}\right)^T M_b^{(i)} \left(x_b^{(i)} - \acute{x}_b^{(i)}\right) \\
&= \frac{1}{2}\left\|x_b^{(i)} - x_b^{(i+1)}\right\|^2_{M_b^{(i)}} - \frac{1}{2}\left\|W_b^{(i)}\left(x_b^{(i-1)} - x_b^{(i)}\right)\right\|^2_{M_b^{(i)}} \\
&\geq \frac{1}{2}\left\|x_b^{(i)} - x_b^{(i+1)}\right\|^2_{M_b^{(i)}} - \frac{\delta^2}{2}\left\|x_b^{(i-1)} - x_b^{(i)}\right\|^2_{M_b^{(i-1)}}
\end{aligned}$$

where the first inequality is obtained by using (S.2) in Lemma S.1, the first equality uses the symmetry of $M_b^{(i)}$, and the second inequality holds by

$$\left(W_b^{(i)}\right)^T M_b^{(i)} W_b^{(i)} \preceq \delta^2 M_b^{(i-1)}$$

due to Assumption 3 and (9). Summing the following inequality of $F(x^{(i)}) - F(x^{(i+1)})$

$$\begin{aligned}
&F(x^{(i)}) - F(x^{(i+1)}) \\
&= \sum_{b=1}^{B} F_b^{(i)}(x_b^{(i)}) - F_b^{(i)}(x_b^{(i+1)}) \\
&\geq \sum_{b=1}^{B} \frac{1}{2}\left\|x_b^{(i)} - x_b^{(i+1)}\right\|^2_{M_b^{(i)}} - \frac{\delta^2}{2}\left\|x_b^{(i-1)} - x_b^{(i)}\right\|^2_{M_b^{(i-1)}}
\end{aligned}$$

over $i = 1, \ldots, \mathrm{Iter}$, we have

$$\begin{aligned}
&F(x^{(0)}) - F(x^{(\mathrm{Iter}+1)}) \\
&\geq \sum_{i=1}^{\mathrm{Iter}} \sum_{b=1}^{B} \frac{1}{2}\left\|x_b^{(i)} - x_b^{(i+1)}\right\|^2_{M_b^{(i)}} - \frac{\delta^2}{2}\left\|x_b^{(i-1)} - x_b^{(i)}\right\|^2_{M_b^{(i-1)}} \\
&\geq \sum_{i=1}^{\mathrm{Iter}} \sum_{b=1}^{B} \frac{1-\delta^2}{2}\left\|x_b^{(i)} - x_b^{(i+1)}\right\|^2_{M_b^{(i)}} \\
&\geq \sum_{i=1}^{\mathrm{Iter}} \frac{(1-\delta^2)\beta}{2}\left\|x^{(i)} - x^{(i+1)}\right\|^2_2.
\end{aligned}$$

Due to the lower boundedness of F in Assumption 1, taking $\mathrm{Iter} \to \infty$ completes the proof.

S.III. Proof of Theorem 3.3

Let $\bar{x}$ be a limit point of $\{x^{(i)}\}$ and $\{x^{(i_j)}\}$ be the subsequence converging to $\bar{x}$. Note that $\bar{x} \in \mathcal{X}$ (due to the closedness of $\mathcal{X}$), and $M_b^{(i_j)} \to M_b^{(i)}$ (taking another subsequence if necessary) as $j \to \infty$ since $\{M_b^{(i)}\}$ is bounded, for $b \in [B]$. Using (10), $\{x^{(i_j+\iota)}\}$ converges to $\bar{x}$ for any $\iota \geq 0$.

Now we observe that

$$\begin{aligned}
x_b^{(i_j+1)} = \operatorname*{argmin}_{x_b \in \mathcal{X}_b^{(i_j)}} &\langle \nabla f_b^{(i_j)}(\acute{x}_b^{(i_j)}), x_b - \acute{x}_b^{(i_j)} \rangle \\
&+ \frac{1}{2}\left\|x_b - \acute{x}_b^{(i_j)}\right\|^2_{M_b^{(i_j)}} + r_b(x_b). \quad \text{(S.7)}
\end{aligned}$$

Note that the convex proximal minimization is continuous in the sense that the output point $x_b^{(i_j+1)}$ continuously depends on the input point $\acute{x}_b^{(i_j)}$ [3]. Using the fact that $x_b^{(i_j+1)} \to \bar{x}_b$ and $\acute{x}_b^{(i_j)} \to \bar{x}_b$ as $j \to \infty$, (S.7) becomes

$$\bar{x}_b = \operatorname*{argmin}_{x_b \in \bar{\mathcal{X}}_b} \langle \nabla_{x_b} f_b(\bar{x}), x_b - \bar{x}_b \rangle + \frac{1}{2}\|x_b - \bar{x}_b\|^2_{\bar{M}_b} + r_b(x_b). \quad \text{(S.8)}$$

Thus, $\bar{x}_b$ satisfies the first-order optimality condition (see (S.3)) of (S.8):

$$\langle \nabla_{x_b} f_b(\bar{x}) + \bar{g}_b, x_b - \bar{x}_b \rangle \geq 0, \quad \text{for any } x_b \in \bar{\mathcal{X}}_b$$

and for some $\bar{g} \in \partial r_b(\bar{x}_b)$, which is equivalent to the Nash equilibrium condition (8). This completes the proof.

S.IV. Proofs of Propositions 4.2 and 4.6

A. Proof of Proposition 4.2

To show that $M_{Q_\Psi} \succeq Q_\Psi^H Q_\Psi$, we use $\Sigma \succeq \widehat{Z}^H \widehat{Z}$ satisfying

$$(I_K \otimes \Phi^{-1})\Sigma(I_K \otimes \Phi) \succeq (I_K \otimes \Phi^{-1})\widehat{Z}^H \widehat{Z}(I_K \otimes \Phi)$$

where $\Sigma \in \mathbb{C}^{\tilde{N}D \times \tilde{N}K}$ is a block diagonal matrix with diagonal matrices $\{\Sigma_k : k = 1, \ldots, K\}$. We seek to apply the property that, if a Hermitian matrix is diagonally dominant with nonnegative diagonal entries, then it is positive semidefinite. Noting that $\Sigma - \widehat{Z}^H \widehat{Z} \succeq 0$ is a Hermitian matrix, this property can be applied to show $\Sigma \succeq \widehat{Z}^H \widehat{Z}$. Observe that the diagonal elements of $[\Sigma - \widehat{Z}^H \widehat{Z}]_{k,k}$ are nonnegative because

$$\sum_{k' \neq k} \left| \sum_{l=1}^{L} \overline{(\hat{z}_{l,k})_i} \cdot (\hat{z}_{l,k'})_i \right| \geq 0, \qquad k \in [K].$$

It now suffices to show that

$$\left| \left([\Sigma - \widehat{Z}^H \widehat{Z}]_{k,k}\right)_{i,i} \right| \geq \sum_{k' \neq k} \left| \left([\Sigma - \widehat{Z}^H \widehat{Z}]_{k,k'}\right)_{i,i} \right|.$$

This is true because the left terms and the right terms are identical, given by

$$\sum_{k' \neq k} \left| \sum_{l=1}^{L} \overline{(\hat{z}_{l,k})_i} \cdot (\hat{z}_{l,k'})_i \right|$$

for all $k = 1, \ldots, K$ and $i = 1, \ldots, \tilde{N}$. Combining these results completes the proof.

B. Proofs of Proposition 4.6

Using the similar technique in Section S.IV-A, the majorization matrix for $\Lambda^H \Lambda$ is given by a block diagonal matrix with diagonal blocks $\{\Sigma'_k\}$ given in (31). Substituting this majorization matrix into (27) completes the proof.

S.V. Proofs of Lemmas 4.1 & 4.5, 4.3, 4.4 & 4.7, and 5.1

A. Proofs of Lemmas 4.1 & 4.5

Note that $Q_\Psi^H Q_\Psi$ is a Hermitian matrix with nonnegative diagonal entries because its diagonal submatrices are given by

$$[Q_\Psi^H Q_\Psi]_{k,k} = \Phi^{-1} \sum_{l=1}^{L} \mathrm{diag}(|\hat{z}_{l,k}|^2)\Phi, \qquad k \in [K],$$

which imply that $[Q_\Psi^H Q]_{k,k}$ is a circulant matrix with (identical) positive diagonal entries, and

$$[Q_\Psi^H Q_\Psi]_{k,k'} = [Q_\Psi^H Q_\Psi]_{k',k}^H, \qquad k \neq k' \in [K]$$

where $[Q_\Psi^H Q_\Psi]_{k,k'}$ is given as (21). Applying Lemma S.2 to the Hermitian matrix $Q_\Psi^H Q_\Psi$ completes the proof.

Repeating the similar procedure leads to a result in Lemma 4.5.

B. Proof of Lemma 4.3

Observe that for any $x \in \mathbb{C}^{\tilde{N}}$

$$\begin{aligned} x^H \Phi^{-1} \Sigma_k \Phi x = x^H \widetilde{\Phi}^H \Sigma_k \widetilde{\Phi} x &= \sum_{i=1}^{\tilde{N}} (\Sigma_k)_{i,i} |y_i|^2 \\ &\leq \max_{i=1,\ldots,\tilde{N}} \{(\Sigma_k)_{i,i}\} \sum_{i=1}^{\tilde{N}} |y_i|^2 \\ &= \max_{i=1,\ldots,\tilde{N}} \{(\Sigma_k)_{i,i}\} \cdot x^H I_{\tilde{N}} x \end{aligned}$$

where we use $y = \widetilde{\Phi} x$ and $\|y\|_2^2 = \|x\|_2^2$ and $\widetilde{\Phi}$ denotes unitary DFT. This completes the proof.

C. Proofs of Lemmas 4.4 & 4.7

The results directly follow by noting that $\{\Phi^{-1}\Sigma_k\Phi\}$ and $\{\Phi^{-1}\Sigma_k'\Phi\}$ are Hermitian (circulant) matrices with nonnegative diagonal entries, and applying Lemma S.2.

D. Proof of Lemma 5.1

Using $P_B^T P_B \preceq I$, we have

$$\Psi_k^H \Psi_k \preceq \Phi^{-1} \sum_{l=1}^{L} \mathrm{diag}(|\hat{z}_{l,k}|^2)\Phi.$$

Observe that the Hermitian matrix $\Phi^{-1} \sum_{l=1}^{L} \mathrm{diag}(|\hat{z}_{l,k}|^2)\Phi$ is positive semidefinite. Applying Lemma S.2 completes the proof.

S.VI. Accelerated Newton's Method to Solve (24)

The optimal solution to (24) can be obtained by the classical approach for solving a quadratically constrained quadratic program (see, for example, [4, Ex. 4.22]):[1]

$$d_k^{(i+1)} = \left(\left[M_\Psi^{(i)} \right]_{k,k} + \varphi_k I_K \right)^{-1} \left[M_\Psi^{(i)} \right]_{k,k} \nu_k^{(i)} \qquad \text{(S.9)}$$

where the Lagrangian parameter is determined by $\varphi_k = \max\{0, \varphi_k^\star\}$ and $\varphi_k^\star$ is the largest solution of the nonlinear equation $f(\varphi_k) = 1$, where

$$f(\varphi_k) = \left\| \left(\left[M_\Psi^{(i)} \right]_{k,k} + \varphi_k I_K \right)^{-1} \left[M_\Psi^{(i)} \right]_{k,k} \nu_k^{(i)} \right\|_2^2, \qquad \text{(S.10)}$$

which is the so-called *secular* equation, for $k = 1, \ldots, K$. More specifically, the algorithm goes as follows. If $\|\nu_k^{(i)}\|_2 \leq 1$, then $d_k^{(i+1)} = \nu_k^{(i)}$ is the optimal solution. Otherwise, one can obtain the optimal solution $d_k^{(i+1)}$ through (S.9) with the Lagrangian parameter $\varphi_k = \varphi_k^\star$, where $\varphi_k^\star$ is optimized by solving the secular equation $f(\varphi_k) = 1$ and $f(\varphi_k)$ is given as (S.10). To solve $f(\varphi_k) = 1$, we first rewrite (S.10) by

$$f(\varphi_k) = \sum_{j=1}^{K} \frac{\left(\left[M_\Psi^{(i)} \right]_{k,k} \right)_{j,j}^2 \left(\nu_k^{(i)} \right)_j^2}{\left(\varphi_k + \left(\left[M_\Psi^{(i)} \right]_{k,k} \right)_{j,j} \right)^2}. \qquad \text{(S.11)}$$

where $\{([M_\Psi^{(i)}]_{k,k})_{j,j} > 0 : j = 1, \ldots, K\}$. Noting that $f(0) > 1$ and $f(\varphi_k)$ monotonically decreases to zero as $\varphi_k \to \infty$, the nonlinear equation $f(\varphi_k) = 1$ has exactly one nonnegative solution $\varphi_k^\star$. The optimal solution $\varphi_k^\star$ can be determined by using the classical Newton's method. To solve the secular equation $f(\varphi_k) = 1$ faster, we apply the accelerated Newton's method in [5]:

$$\varphi_k^{(\iota+1)} = \varphi_k^{(\iota)} - 2 \frac{f(\varphi_k^{(\iota)})}{f'(\varphi_k^{(\iota)})} \left(\sqrt{f(\varphi_k^{(\iota)})} - 1 \right) \qquad \text{(S.12)}$$

where $f(\varphi_k)$ is given as (S.11),

$$f'(\varphi_k) = -2 \sum_{j=1}^{K} \frac{\left(\left[M_\Psi^{(i)} \right]_{k,k} \right)_{j,j}^2 \left(\nu_k^{(i)} \right)_j^2}{\left(\varphi_k + \left(\left[M_\Psi^{(i)} \right]_{k,k} \right)_{j,j} \right)^3},$$

and $\varphi_k^{(0)} = 0$. Note that (S.12) approaches the optimal solution $\varphi_k^\star$ faster than the classical Newton's method.

[1]Note that (S.9) can be also applied to a circulant majorizer M_Ψ, e.g.,

$$[M_\Psi]_{k,k} = \Phi_K^{-1} E_k \Phi_K,$$

where $E_k = \mathrm{diag}\big(|\Phi_K P_S \Phi^{-1} \Sigma_k \Phi P_S^T \Phi_K^{-1}| 1_K\big)$ and Φ_K is a (unnormalized) DFT matrix of size $K \times K$. Unfortunately, an efficient scheme to compute the circulant majorizer is unknown, due to the difficulty in deriving the symbolic expression for the matrix of $|\cdot|$ (i.e., the matrix inside $|\cdot|$ is no longer circulant).

References

[1] I. Y. Chun and J. Fessler, "Convolutional dictionary learning: Acceleration and convergence," to appear in *IEEE Trans. Image Process.*

[2] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. Imaging Sci.*, vol. 6, no. 3, pp. 1758–1789, Sep. 2013.

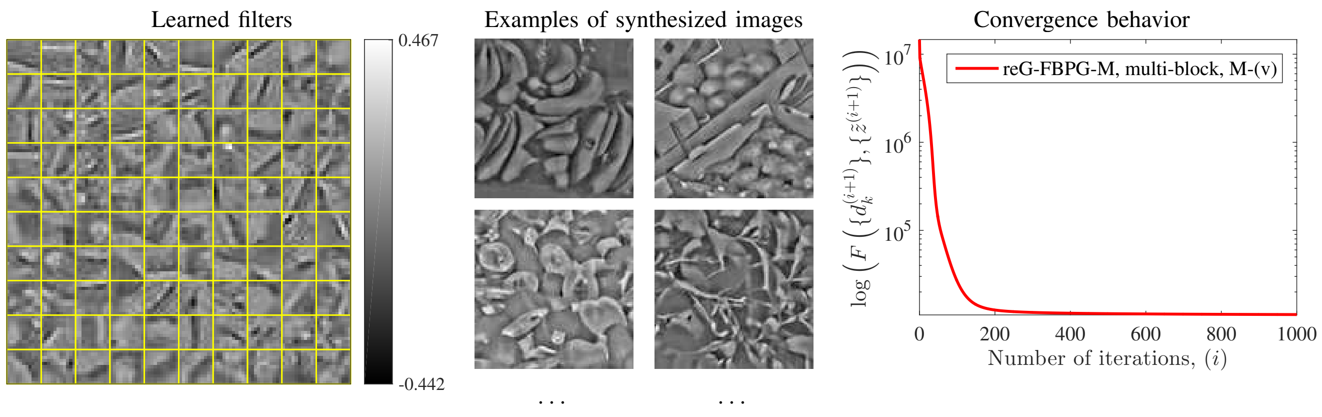
[3] R. T. Rockafellar, "Monotone operators and the proximal point algorithm," *SIAM J. Control Optim.*, vol. 14, no. 5, pp. 877–898, Aug. 1976.

[4] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY: Cambridge University Press, 2004.

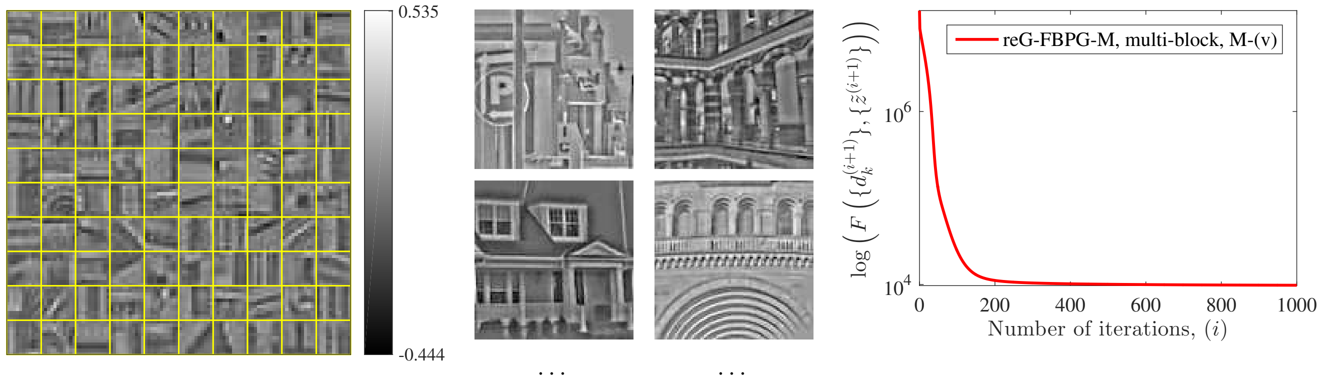
[5] C. H. Reinsch, "Smoothing by spline functions. II," *Numer. Math.*, vol. 16, no. 5, pp. 451–454, Feb. 1971.

[6] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *Proc.* 2015 *IEEE CVPR*, Boston, MA, Jun. 2015, pp. 5135–5143.

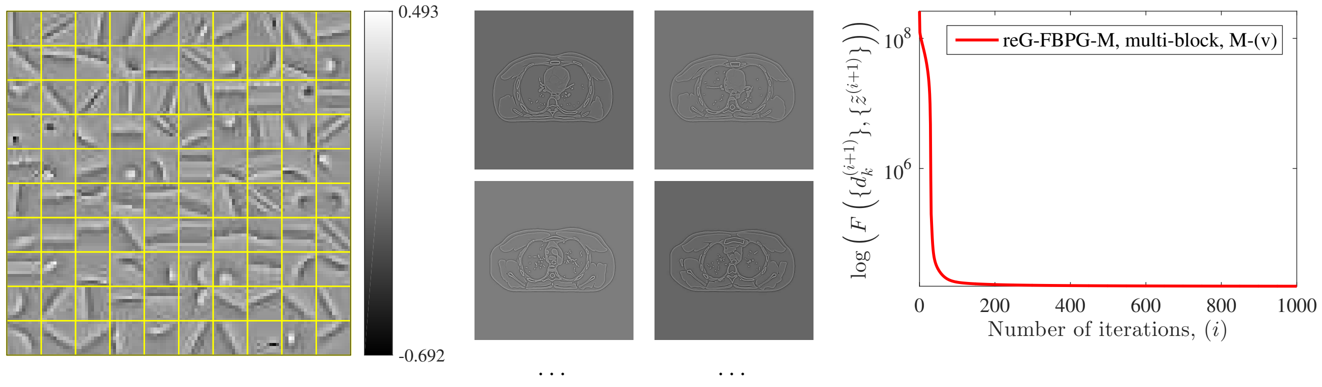
[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. & Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jan. 2011.



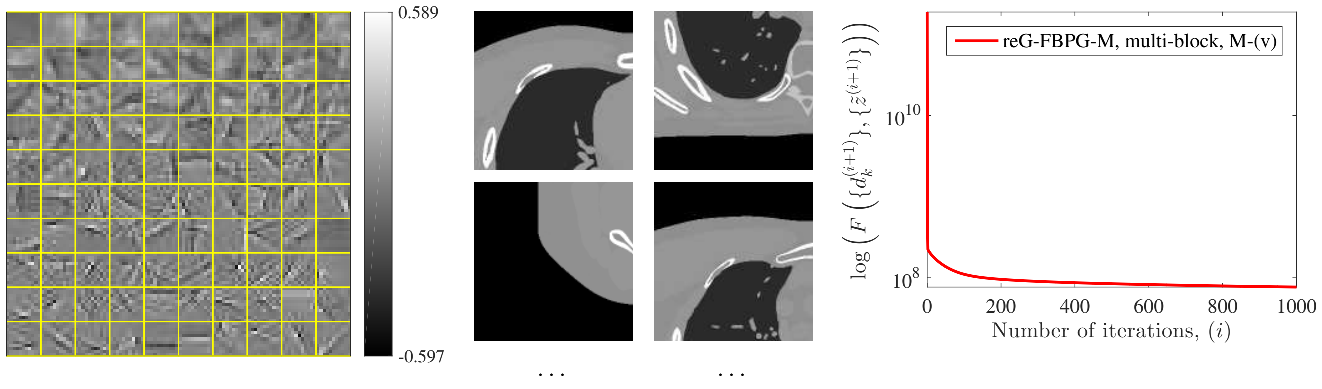
(a) The fruit dataset ($L = 10$, $N = 100 \times 100$)



(b) The city dataset ($L = 10$, $N = 100 \times 100$)

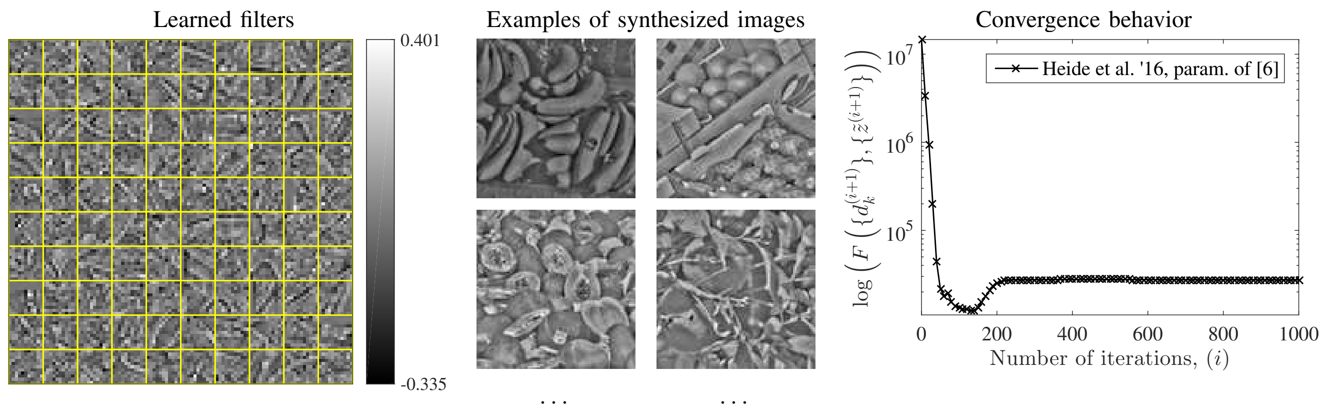


(c) The CT-(i) dataset ($L = 10$, $N = 512 \times 512$)

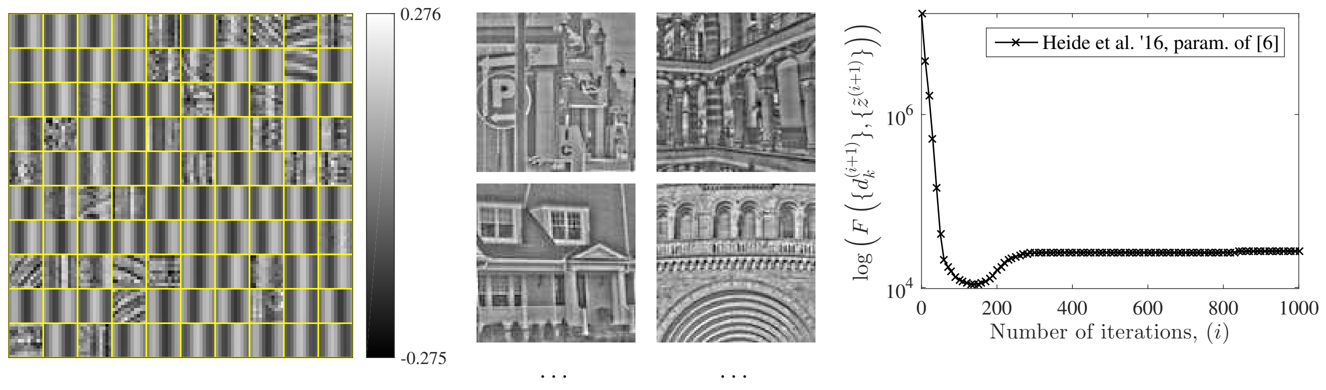


(d) The CT-(ii) dataset ($L = 80$, $N = 128 \times 128$)

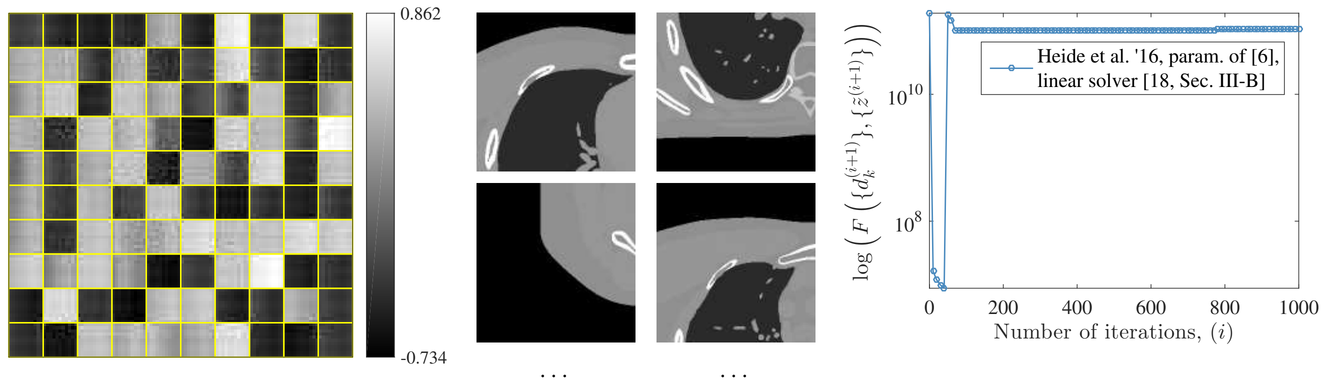
Fig. S.1. Examples of CDL results by the proposed reG-FBPG-M algorithm (using the multi-block scheme and M-(v)) from different datasets. While guaranteeing convergence, the BPG-M methods provide desirable solutions: *1)* the learned (Gabor-like) filters capture structures of training images; *2)* the corresponding sparse codes have sparsity less than 1% (for (d), approximately 5%); *3)* the resultant filters and sparse codes properly synthesize the training images. The sparsity is measured by $\sum_{l=1}^{L}(\|\tilde{z}_l\|/\tilde{N}K)$ in percentages. The synthesized images mean $\{P_B \sum_{k=1}^{K} d_k^\star \circledast z_{l,k}^\star : l = 1, \ldots, L\}$.



(a) The fruit dataset ($L = 10$, $N = 100 \times 100$)



(b) The city dataset ($L = 10$, $N = 100 \times 100$)



(c) The CT-(ii) dataset ($L = 80$, $N = 128 \times 128$)

Fig. S.2. Examples of CDL results by the ADMM algorithm [6] from different datasets (the number of whole iterations is the product of the number of inner iterations and that of outer iterations). Although using the suggested ADMM parameters in [6], ADMM and its memory-efficient variant (see Section VII-A1 in the main paper) show unstable convergence and the (empirically) convergent filters fail to capture structures of training images and do not have Gabor-like shapes. In addition, the sparse codes at the termination point have the sparsity of 100%. Because the system matrices in each filter and sparse code update keep changing with the dependency of updated filters of sparse codes, one may want to develop adaptive ADMM parameter control schemes, e.g., selection schemes based on primal and dual residual norms [7, §3.4.1], [8, §III-D] or condition number [9], [10, §IV-C].

[8] B. Wohlberg, "Efficient algorithms for convolutional sparse representations," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 301–315, Jan. 2016.
[9] T. Goldstein and S. Osher, "The split Bregman method for L1-regularized problems," *SIAM J. Imaging Sci.*, vol. 2, no. 2, pp. 323–343, Apr. 2009.
[10] S. Ramani and J. A. Fessler, "Parallel MR image reconstruction using augmented Lagrangian methods," *IEEE Trans. Med. Imag.*, vol. 30, no. 3, pp. 694–706, Mar. 2011.