Convolutional Analysis Operator Learning: Acceleration and Convergence

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

***Abstract*— Convolutional operator learning is gaining attention in many signal processing and computer vision applications. Learning kernels has mostly relied on so-called *patch-domain* approaches that extract and store many overlapping patches across training signals. Due to memory demands, patch-domain methods have limitations when learning kernels from large datasets – particularly with multi-layered structures, e.g., convolutional neural networks – or when applying the learned kernels to high-dimensional signal recovery problems. The so-called *convolution* approach does not store many overlapping patches, and thus overcomes the memory problems particularly with careful algorithmic designs; it has been studied within the "synthesis" signal model, e.g., convolutional dictionary learning. This paper proposes a new *convolutional analysis operator learning* (CAOL) framework that learns an analysis sparsifying regularizer with the convolution perspective, and develops a new convergent *Block Proximal Extrapolated Gradient method using a Majorizer* (BPEG-M) to solve the corresponding block multi-nonconvex problems. To learn diverse filters within the CAOL framework, this paper introduces an orthogonality constraint that enforces a tight-frame filter condition, and a regularizer that promotes diversity between filters. Numerical experiments show that, with sharp majorizers, BPEG-M significantly accelerates the CAOL convergence rate compared to the state-of-the-art block proximal gradient (BPG) method. Numerical experiments for sparse-view computational tomography show that a convolutional sparsifying regularizer learned via CAOL significantly improves reconstruction quality compared to a conventional edge-preserving regularizer. Using more and wider kernels in a learned regularizer better preserves edges in reconstructed images.**

***Index Terms*— Convolutional regularizer learning, convolutional dictionary learning, convolutional neural networks, unsupervised machine learning algorithms, nonconvex-nonsmooth optimization, block coordinate descent, inverse problems, X-ray computed tomography.**

I. Introduction

LEARNING convolutional operators from large datasets is a growing trend in signal/image processing, computer vision, and machine learning. The widely known *patch-domain* approaches for learning kernels (e.g., filter, dictionary, frame, and transform) extract patches from training signals for simple mathematical formulation and optimization, yielding (sparse) features of training signals [1]–[9]. Due to memory demands, using many overlapping patches across the training signals hinders using large datasets and building hierarchies on the features, e.g., deconvolutional neural networks [10], convolutional neural network (CNN) [11], and multi-layer convolutional sparse coding [12]. For similar reasons, the memory requirement of patch-domain approaches discourages learned kernels from being applied to large-scale inverse problems.

To moderate these limitations of the patch-domain approach, the so-called *convolution* perspective has been recently introduced by learning filters and obtaining (sparse) representations directly from the original signals without storing many overlapping patches, e.g., convolutional dictionary learning (CDL) [10], [13]–[17]. For large datasets, CDL using careful algorithmic designs [16] is more suitable for learning filters than patch-domain dictionary learning [1]; in addition, CDL can learn translation-invariant filters without obtaining highly redundant sparse representations [16]. The CDL method applies the convolution perspective for learning kernels within "synthesis" signal models. Within "analysis" signal models, however, there exist no prior frameworks using the convolution perspective for learning convolutional operators, whereas patch-domain approaches for learning analysis kernels are introduced in [3], [4], [6]–[8]. (See brief descriptions about synthesis and analysis signal models in [4, Sec. I].)

Researchers interested in dictionary learning have actively studied the structures of kernels learned by the patch-domain approach [3], [4], [6]–[8], [18]–[20]. In training CNNs (see Appendix A), however, there has been less study of filter structures having non-convex constraints, e.g., orthogonality and unit-norm constraints in Section III, although it is thought that diverse (i.e., incoherent) filters can improve performance for some applications, e.g., image recognition [9]. On the application side, researchers have applied (deep) NNs to signal/image recovery problems. Recent works combined model-based image reconstruction (MBIR) algorithm with image refining networks [21]–[30]. In these *iterative* NN methods, refining NNs should satisfy the non-expansiveness for fixed-point convergence [29]; however, their trainings lack consideration of filter diversity constraints, e.g., orthogonality constraint in Section III, and thus it is unclear whether the trained NNs are nonexpansive mapping [30].

This paper proposes *1)* a new *convolutional analysis operator learning* (CAOL) framework that learns an analysis sparsifying regularizer with the convolution perspective, and

Manuscript received January 24, 2019; revised June 25, 2019 and August 9, 2019; accepted August 9, 2019. This work was supported in part by the Keck Foundation, in part by NIH under Grant U01 EB018753, and in part by NSF under Grant IIS 1838179. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Xiaochun Cao. *(Corresponding author: Il Yong Chun.)*

I. Y. Chun was with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48019 USA. He is now with the Department of Electrical Engineering, University of Hawai'i at Mānoa, Honolulu, HI 96822 USA (e-mail: iychun@hawaii.edu).

J. A. Fessler is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48019 USA (e-mail: fessler@umich.edu).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TIP.2019.2937734

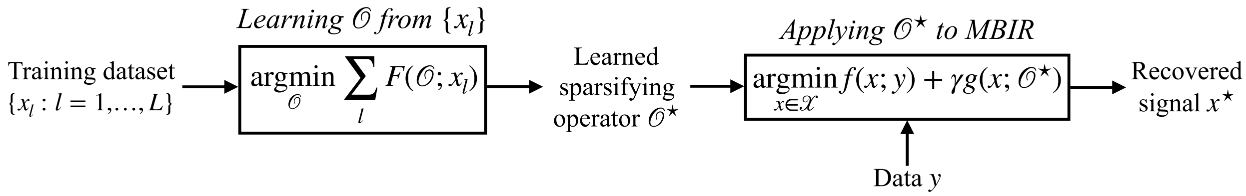


Fig. 1. A general flowchart from learning sparsifying operators $\mathcal{O}$ to solving inverse problems via MBIR using learned operators $\mathcal{O}^\star$; see Section II. For the lth training sample x_l, $F(\mathcal{O}; x_l)$ measures its sparse representation or sparsification errors, and sparsity of its representation generated by $\mathcal{O}$.

2) a new convergent *Block Proximal Extrapolated Gradient method using a Majorizer* (BPEG-M [16]) for solving block multi-nonconvex problems [31]. To learn diverse filters, we propose *a)* CAOL with an orthogonality constraint that enforces a tight-frame (TF) filter condition in convolutional perspectives, and *b)* CAOL with a regularizer that promotes filter diversity. BPEG-M with sharper majorizers converges significantly faster than the state-of-the-art technique, Block Proximal Gradient (BPG) method [31] for CAOL. This paper also introduces a new X-ray computational tomography (CT) MBIR model using a convolutional sparsifying regularizer learned via CAOL [32].

The remainder of this paper is organized as follows. Section II reviews how learned regularizers can help solve inverse problems. Section III proposes the two CAOL models. Section IV introduces BPEG-M with several generalizations, analyzes its convergence, and applies a momentum coefficient formula and restarting technique from [16]. Section V applies the proposed BPEG-M methods to the CAOL models, designs two majorization matrices, and describes memory flexibility and applicability of parallel computing to BPEG-M-based CAOL. Section VI introduces the CT MBIR model using a convolutional regularizer learned via CAOL [32], along with its properties, i.e., its mathematical relation to a convolutional autoencoder, the importance of TF filters, and its algorithmic role in signal recovery. Section VII reports numerical experiments that show *1)* the importance of sharp majorization in accelerating BPEG-M, and *2)* the benefits of BPEG-M-based CAOL – acceleration, convergence, and memory flexibility. Additionally, Section VII reports sparse-view CT experiments that show *3)* the CT MBIR using learned convolutional regularizers significantly improves the reconstruction quality compared to that using a conventional edge-preserving (EP) regularizer, and *4)* more and wider filters in a learned regularizer better preserves edges in reconstructed images. Finally, Appendix A mathematically formulates unsupervised training of CNNs via CAOL, and shows that its updates attained via BPEG-M correspond to the three important CNN operators. Appendix B introduces some potential applications of CAOL to image processing, imaging, and computer vision.

II. Backgrounds: MBIR Using *Learned* Regularizers

To recover a signal $x \in \mathbb{C}^{N'}$ from a data vector $y \in \mathbb{C}^m$, one often considers the following MBIR optimization problem (Appendix C provides mathematical notations): $\operatorname{argmin}_{x\in\mathcal{X}} f(x; y) + \gamma\, g(x)$, where $\mathcal{X}$ is a feasible set, $f(x; y)$ is data fidelity function that models imaging physics (or image formation) and noise statistics, $\gamma > 0$ is a regularization parameter, and $g(x)$ is a regularizer, such as total variation [33, §2–3]. However, when inverse problems are extremely ill-conditioned, the MBIR approach using hand-crafted regularizers $g(x)$ has limitations in recovering signals. Alternatively, there has been a growing trend in learning sparsifying regularizers (e.g., convolutional regularizers [16], [17], [32], [34], [35]) from training datasets and applying the learned regularizers to the following MBIR problem [33]:

$$\underset{x\in\mathcal{X}}{\operatorname{argmin}}\ f(x; y) + \gamma\, g(x; \mathcal{O}^\star), \tag{B1}$$

where a learned regularizer $g(x; \mathcal{O}^\star)$ quantifies consistency between any candidate x and training data that is encapsulated in some trained sparsifying operators $\mathcal{O}^\star$. The diagram in Fig. 1 shows the general process from training sparsifying operators to solving inverse problems via (B1). Such models (B1) arise in a wide range of applications. See some examples in Appendix B.

This paper describes multiple aspects of learning convolutional regularizers. The next section first starts with proposing a new convolutional regularizer.

III. CAOL: Models *Learning* Convolutional Regularizers

The goal of CAOL is to find a set of filters that "best" sparsify a set of training images. Compared to hand-crafted regularizers, learned convolutional regularizers can better extract "true" features of estimated images and remove "noisy" features with thresholding operators. We propose the following CAOL model:

$$\underset{D=[d_1,\ldots,d_K]}{\operatorname{argmin}}\ \min_{\{z_{l,k}\}} F(D, \{z_{l,k}\}) + \beta g(D),$$
$$F(D, \{z_{l,k}\}) := \sum_{l=1}^{L} \sum_{k=1}^{K} \frac{1}{2} \left\| d_k \circledast x_l - z_{l,k} \right\|_2^2 + \alpha \| z_{l,k} \|_0, \tag{P0}$$

where $\circledast$ denotes a convolution operator (see details about boundary conditions in the supplementary material), $\{x_l \in \mathbb{C}^N : l = 1, \ldots, L\}$ is a set of training images, $\{d_k \in \mathbb{C}^R : k = 1, \ldots, K\}$ is a set of convolutional kernels, $\{z_{l,k} \in \mathbb{C}^N : l = 1, \ldots, L, k = 1, \ldots, K\}$ is a set of sparse codes, and $g(D)$ is a regularizer or constraint that encourages filter diversity or incoherence, $\alpha > 0$ is a thresholding parameter controlling the sparsity of features $\{z_{l,k}\}$, and $\beta > 0$ is a regularization parameter for $g(D)$. We group the K filters into a matrix $D \in \mathbb{C}^{R\times K}$:

$$D := \left[\ d_1\ \ldots\ d_K\ \right]. \tag{1}$$

For simplicity, we fix the dimension for training signals, i.e., $\{x_l, z_{l,k} \in \mathbb{C}^N\}$, but the proposed model

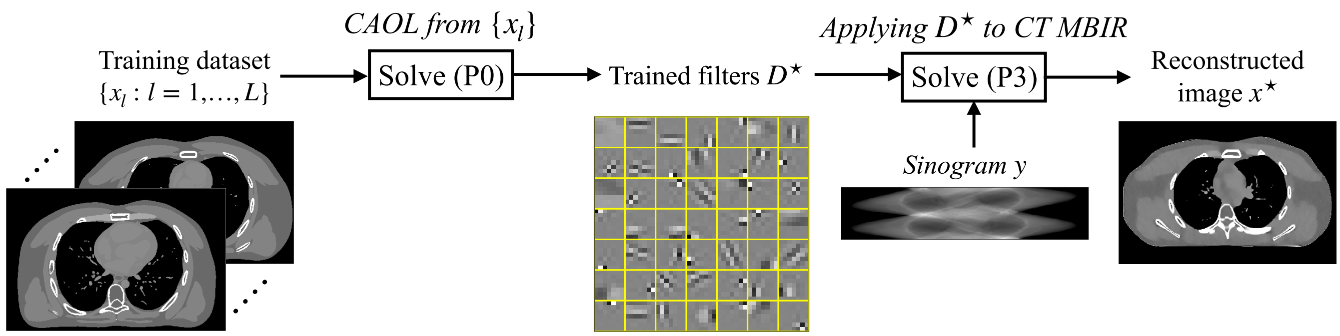


Fig. 2. A flowchart from CAOL (P0) to MBIR using a convolutional sparsifying regularizer learned via CAOL (P3) in sparse-view CT. See details of the CAOL process (P0) and its variants (P1)–(P2), and the CT MBIR process (P3) in Section III and Section VI, respectively.

(P0) can use training signals of different dimension, i.e., $\{x_l, z_{l,k} \in \mathbb{C}^{N_l}\}$. For sparse-view CT in particular, the diagram in Fig. 2 shows the process from CAOL (P0) to solving its inverse problem via MBIR using learned convolutional regularizers.

The following two subsections design the constraint or regularizer $g(D)$ to avoid redundant filters (without it, all filters could be identical).

A. CAOL With Orthogonality Constraint

We first propose a CAOL model with a nonconvex orthogonality constraint on the filter matrix D in (1):

$$\operatorname*{argmin}_{D} \min_{\{z_{l,k}\}} F(D, \{z_{l,k}\}) \quad \text{subj. to } DD^H = \frac{1}{R} \cdot I. \qquad \text{(P1)}$$

The orthogonality condition $DD^H = \frac{1}{R}I$ in (P1) enforces a TF condition on the filters $\{d_k\}$ in CAOL (P0). Proposition 3.1 below formally states this relation.

Proposition 3.1 (Tight-frame (TF) filters)**.** *Filters satisfying the orthogonality constraint $DD^H = \frac{1}{R}I$ in (P1) satisfy the following TF condition in a convolution perspective:*

$$\sum_{k=1}^{K} \|d_k \circledast x\|_2^2 = \|x\|_2^2, \quad \forall x \in \mathbb{C}^N, \tag{2}$$

for both circular and symmetric boundary conditions.

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

Proposition 3.1 corresponds to a TF result from patch-domain approaches; see Section S.I. (Note that the patch-domain approach in [6, Prop. 3] requires $R = K$.) However, we constrain the filter dimension to be $R \le K$ to have an efficient solution for CAOL model (P1); see Proposition 5.4 later. The following section proposes a more flexible CAOL model in terms of the filter dimensions R and K.

B. CAOL With Diversity Promoting Regularizer

As an alternative to the CAOL model (P1), we propose a CAOL model with a diversity promoting regularizer and a nonconvex norm constraint on the filters $\{d_k\}$:

$$\operatorname*{argmin}_{D} \min_{\{z_{l,k}\}} \quad F(D, \{z_{l,k}\}) + \frac{\beta}{2} \overbrace{\left\| D^H D - \frac{1}{R} \cdot I \right\|_{\mathrm{F}}^2}^{=:\, g_{\mathrm{div}}(D)},$$

$$\text{subject to} \qquad \|d_k\|_2^2 = \frac{1}{R}, \quad k = 1, \ldots, K. \qquad \text{(P2)}$$

In the CAOL model (P2), we consider the following:

- The constraint in (P2) forces the learned filters $\{d_k\}$ to have uniform energy. In addition, it avoids the "scale ambiguity" problem [36].
- The regularizer in (P2), $g_{\mathrm{div}}(D)$, promotes filter diversity, i.e., incoherence between d_k and $\{d_{k'} : k' \neq k\}$, measured by $|\langle d_k, d_{k'} \rangle|^2$ for $k \neq k'$.

When $R = K$ and $\beta \to \infty$, the model (P2) becomes (P1) since $D^H D = \frac{1}{R}I$ implies $DD^H = \frac{1}{R}I$ (for square matrices A and B, if $AB = I$ then $BA = I$). Thus (P2) generalizes (P1) by relaxing the off-diagonal elements of the equality constraint in (P1). (In other words, when $R = K$, the orthogonality constraint in (P1) enforces the TF condition and promotes the filter diversity.) One price of this generalization is the extra tuning parameter β.

(P1)–(P2) are challenging nonconvex optimization problems and block optimization approaches seem suitable. The following section proposes a new block optimization method with momentum and majorizers, to rapidly solve the multiple block multi-nonconvex problems proposed in this paper, while guaranteeing convergence to critical points.

IV. BPEG-M: Solving Block Multi-Nonconvex Problems With Convergence Guarantees

This section describes a new optimization approach, BPEG-M, for solving block multi-nonconvex problems like *a)* CAOL (P1)–(P2),[1] *b)* CT MBIR (P3) using learned convolutional regularizer via (P1) (see Section VI), and *c)* "hierarchical" CAOL (A1) (see Appendix A).

A. BPEG-M – Setup

We treat the variables of the underlying optimization problem either as a single block or multiple disjoint blocks. Specifically, consider the following *block multi-nonconvex* optimization problem:

$$\min \; F(x_1, \ldots, x_B) := f(x_1, \ldots, x_B) + \sum_{b=1}^{B} g_b(x_b), \tag{3}$$

where variable x is decomposed into B blocks $x_1, \ldots, x_B$ ($\{x_b \in \mathbb{R}^{n_b} : b = 1, \ldots, B\}$), f is assumed to be continuously differentiable, but functions $\{g_b : b = 1, \ldots, B\}$ are not necessarily differentiable. The function g_b can incorporate the

[1] A block coordinate descent algorithm can be applied to CAOL (P1); however, its convergence guarantee in solving CAOL (P1) is not yet known and might require stronger sufficient conditions than BPEG-M [37].

constraint $x_b \in \mathcal{X}_b$, by allowing any g_b to be extended-valued, e.g., $g_b(x_b) = \infty$ if $x_b \notin \mathcal{X}_b$, for $b = 1, \ldots, B$. It is standard to assume that both f and $\{g_b\}$ are closed and proper and the sets $\{\mathcal{X}_b\}$ are closed and nonempty. We do *not* assume that f, $\{g_b\}$, or $\{\mathcal{X}_b\}$ are convex. Importantly, g_b can be a nonconvex ℓ^p quasi-norm, $p \in [0, 1)$. The general block multi-convex problem in [16], [38] is a special case of (3).

The BPEG-M framework considers a more general concept than Lipschitz continuity of the gradient as follows:

Definition 4.1 (M-Lipschitz continuity)**.** *A function* $g : \mathbb{R}^n \to \mathbb{R}^n$ *is* M-Lipschitz continuous *on* $\mathbb{R}^n$ *if there exist a (symmetric) positive definite matrix* M *such that*

$$\|g(x) - g(y)\|_{M^{-1}} \le \|x - y\|_M, \quad \forall x, y,$$

where $\|x\|_M^2 := x^T M x$.

Lipschitz continuity is a special case of M-Lipschitz continuity with M equal to a scaled identity matrix with a Lipschitz constant of the gradient ∇f (e.g., for $f(x) = \frac{1}{2}\|Ax - b\|_2^2$, the (smallest) Lipschitz constant of ∇f is the maximum eigenvalue of $A^T A$). If the gradient of a function is M-Lipschitz continuous, then we obtain the following quadratic majorizer (i.e., surrogate function [39], [40]) at a given point y without assuming convexity:

Lemma 4.2 (Quadratic majorization (QM) via M-Lipschitz continuous gradients)**.** *Let* $f : \mathbb{R}^n \to \mathbb{R}$. *If* ∇f *is* M*-Lipschitz continuous, then*

$$f(x) \le f(y) + \langle \nabla f(y), x - y \rangle + \frac{1}{2}\|x - y\|_M^2, \quad \forall x, y \in \mathbb{R}^n.$$

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

Exploiting Definition 4.1 and Lemma 4.2, the proposed method, BPEG-M, is given as follows. To solve (3), we minimize a majorizer of F cyclically over each block $x_1, \ldots, x_B$, while fixing the remaining blocks at their previously updated variables. Let $x_b^{(i+1)}$ be the value of x_b after its ith update, and define

$$f_b^{(i+1)}(x_b) := f\Big(x_1^{(i+1)}, \ldots, x_{b-1}^{(i+1)}, x_b, x_{b+1}^{(i)}, \ldots, x_B^{(i)}\Big), \quad \forall b, i.$$

At the bth block of the ith iteration, we apply Lemma 4.2 to functional $f_b^{(i+1)}(x_b)$ with a $M_b^{(i+1)}$-Lipschitz continuous gradient, and minimize the majorized function.[2] Specifically, BPEG-M uses the updates

$$\begin{aligned} x_b^{(i+1)} &= \operatorname*{argmin}_{x_b} \langle \nabla_{x_b} f_b^{(i+1)}(\acute{x}_b^{(i+1)}), x_b - \acute{x}_b^{(i+1)} \rangle \\ &\qquad + \frac{1}{2}\Big\| x_b - \acute{x}_b^{(i+1)} \Big\|_{\widetilde{M}_b^{(i+1)}}^2 + g_b(x_b) \\ &= \operatorname*{argmin}_{x_b} \frac{1}{2}\bigg\| x_b - \bigg(\acute{x}_b^{(i+1)} - \Big(\widetilde{M}_b^{(i+1)}\Big)^{-1} \\ &\qquad\qquad \cdot \nabla_{x_b} f_b^{(i+1)}(\acute{x}_b^{(i+1)})\bigg) \bigg\|_{\widetilde{M}_b^{(i+1)}}^2 + g_b(x_b) \\ &= \operatorname{Prox}_{g_b}^{\widetilde{M}_b^{(i+1)}}\bigg(\underbrace{\acute{x}_b^{(i+1)} - \Big(\widetilde{M}_b^{(i+1)}\Big)^{-1} \nabla_{x_b} f_b^{(i+1)}(\acute{x}_b^{(i+1)})}_{\text{extrapolated gradient step using a majorizer of } f_b^{(i+1)}}\bigg), \end{aligned} \tag{4}$$

[2]The quadratically majorized function allows a unique minimizer if $g_b^{(i+1)}(x_b)$ is convex and $\mathcal{X}_b^{(i+1)}$ is a convex set (note that $M_b^{(i+1)} \succ 0$).

Algorithm 1 BPEG-M

Require: $\{x_b^{(0)} = x_b^{(-1)} : \forall b\}$, $\{E_b^{(i)} \in [0, 1], \forall b, i\}$, $i = 0$
while a stopping criterion is not satisfied **do**
  for $b = 1, \ldots, B$ **do**
    Calculate $M_b^{(i+1)}$, $\widetilde{M}_b^{(i+1)}$ by (6), and $E_b^{(i+1)}$ by (7)
    $\acute{x}_b^{(i+1)} = x_b^{(i)} + E_b^{(i+1)}\left(x_b^{(i)} - x_b^{(i-1)}\right)$
    $x_b^{(i+1)} = \ldots$
    $\operatorname{Prox}_{g_b}^{\widetilde{M}_b^{(i+1)}}\left(\acute{x}_b^{(i+1)} - \left(\widetilde{M}_b^{(i+1)}\right)^{-1} \nabla f_b^{(i+1)}(\acute{x}_b^{(i+1)})\right)$
  end for
  $i = i + 1$
end while

where

$$\acute{x}_b^{(i+1)} = x_b^{(i)} + E_b^{(i+1)}\left(x_b^{(i)} - x_b^{(i-1)}\right), \tag{5}$$

the proximal operator is defined by

$$\operatorname{Prox}_g^M(y) := \operatorname*{argmin}_x \frac{1}{2}\|x - y\|_M^2 + g(x),$$

$\nabla f_b^{(i+1)}(\acute{x}_b^{(i+1)})$ is the block-partial gradient of f at $\acute{x}_b^{(i+1)}$, an *upper-bounded majorization matrix* is updated by

$$\widetilde{M}_b^{(i+1)} = \lambda_b \cdot M_b^{(i+1)} \succ 0, \qquad \lambda_b > 1, \tag{6}$$

and $M_b^{(i+1)} \in \mathbb{R}^{n_b \times n_b}$ is a symmetric positive definite *majorization matrix* of $\nabla f_b^{(i+1)}$. In (5), the $\mathbb{R}^{n_b \times n_b}$ matrix $E_b^{(i+1)} \succeq 0$ is an *extrapolation matrix* that accelerates convergence in solving block multi-convex problems [16]. We design it in the following form:

$$E_b^{(i+1)} = e_b^{(i)} \cdot \frac{\delta(\lambda_b - 1)}{2(\lambda_b + 1)} \cdot \left(M_b^{(i+1)}\right)^{-1/2} \left(M_b^{(i)}\right)^{1/2}, \tag{7}$$

for some $\{0 \le e_b^{(i)} \le 1 : \forall b, i\}$ and $\delta < 1$, to satisfy condition (9) below. In general, choosing λ_b values in (6)–(7) to accelerate convergence is application-specific. Algorithm 1 summarizes these updates.

The majorization matrices $M_b^{(i)}$ and $\widetilde{M}_b^{(i+1)}$ in (6) influence the convergence rate of BPEG-M. A tighter majorization matrix (i.e., a matrix giving tighter bounds in the sense of Lemma 4.2) provided faster convergence rate [41, Lem. 1], [16, Fig. 2–3]. An interesting observation in Algorithm 1 is that there exists a tradeoff between majorization sharpness via (6) and extrapolation effect via (5) and (7). For example, increasing λ_b (e.g., $\lambda_b = 2$) allows more extrapolation but results in looser majorization; setting $\lambda_b \to 1$ results in sharper majorization but provides less extrapolation.

Remark 4.3. The proposed BPEG-M framework – with key updates (4)–(5) – generalizes the BPG method [31], and has several benefits over BPG [31] and BPEG-M introduced earlier in [16]:

- The BPG setup in [31] is a particular case of BPEG-M using a scaled identity majorization matrix M_b with a Lipschitz constant of $\nabla f_b^{(i+1)}(\acute{x}_b^{(i+1)})$. The BPEG-M framework can significantly accelerate convergence by allowing sharp majorization; see [16, Fig. 2–3] and Fig. 3. This generalization was first introduced for block multi-convex problems in [16], but the proposed BPEG-M in this paper addresses the more general problem, block multi-(non)convex optimization.

- BPEG-M is useful for controlling the tradeoff between majorization sharpness and extrapolation effect in different blocks, by allowing each block to use different λ_b values. If tight majorization matrices can be designed for a certain block b, then it could be reasonable to maintain the majorization sharpness by setting λ_b very close to 1. When setting $\lambda_b = 1 + \epsilon$ (e.g., ϵ is a machine epsilon) and using $E_b^{(i+1)} = 0$ (no extrapolation), solutions of the original and its upper-bounded problem become (almost) identical. In such cases, it is unnecessary to solve the upper bounded problem (4), and the proposed BPEG-M framework allows using the solution of $f_b^{(i+1)}(x_b)$ without QM; see Section V-B. This generalization was not considered in [31].
- The condition for designing the extrapolation matrix (7), i.e., (9) in Assumption 3, is more general than that in [16, (9)] (e.g., (10)). Specifically, the matrices $E_b^{(i+1)}$ and $M_b^{(i+1)}$ in (7) need not be diagonalized by the same basis.

The first two generalizations lead to the question, "Under the sharp QM regime (i.e., having tight bounds in Lemma 4.2), what is the best way in controlling $\{\lambda_b\}$ in (6)–(7) in Algorithm 1?" Our experiments show that, if sufficiently sharp majorizers are obtained for partial or all blocks, then giving more weight to sharp majorization provides faster convergence compared to emphasizing extrapolation; for example, $\lambda_b = 1 + \epsilon$ gives faster convergence than $\lambda_b = 2$.

B. BPEG-M – Convergence Analysis

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

Assumption 1) F is proper and lower bounded in $\mathrm{dom}(F)$, f is continuously differentiable, g_b is proper lower semicontinuous, $\forall b$.[3] (3) has a critical point $\bar{x}$, i.e., $0 \in \partial F(\bar{x})$, where $\partial F(x)$ denotes the limiting subdifferential of F at x (see [42, §1.9], [43, §8]).

Assumption 2) The block-partial gradients of f, $\nabla f_b^{(i+1)}$, are $M_b^{(i+1)}$-Lipschitz continuous, i.e.,

$$\left\| \nabla_{x_b} f_b^{(i+1)}(u) \; - \nabla_{x_b} f_b^{(i+1)}(v) \; \right\|_{\left(M_b^{(i+1)}\right)^{-1}} \leq \| u - v \|_{M_b^{(i+1)}}, \tag{8}$$

for $u, v \in \mathbb{R}^{n_b}$, and (unscaled) majorization matrices satisfy $m_b I_{n_b} \preceq M_b^{(i+1)}$ with $0 < m_b < \infty$, $\forall b, i$.

Assumption 3) The extrapolation matrices $E_b^{(i+1)} \succeq 0$ satisfy

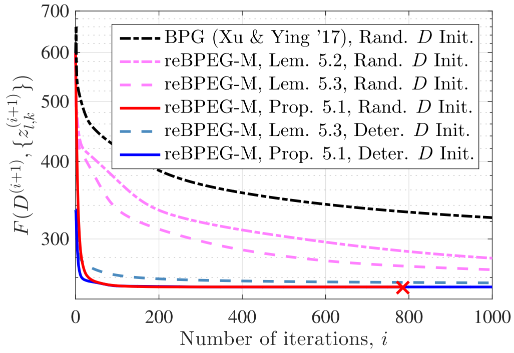
$$\left(E_b^{(i+1)} \right)^T M_b^{(i+1)} E_b^{(i+1)} \preceq \frac{\delta^2 (\lambda_b - 1)^2}{4(\lambda_b + 1)^2} \cdot M_b^{(i)}, \tag{9}$$

for any $\delta < 1$, $\forall b, i$.

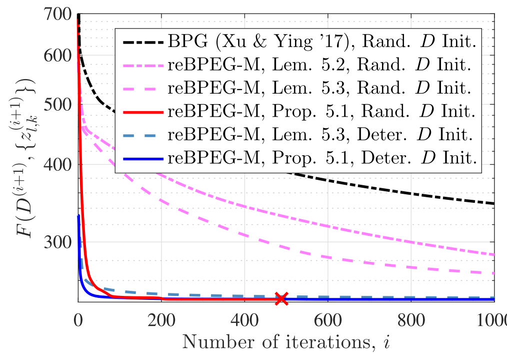
Condition (9) in Assumption 3 generalizes that in [16, Assumption 3]. If eigenspaces of $E_b^{(i+1)}$ and $M_b^{(i+1)}$ coincide (e.g., diagonal and circulant matrices), $\forall i$ [16, Assumption 3], (9) becomes

$$E_b^{(i+1)} \preceq \frac{\delta(\lambda_b - 1)}{2(\lambda_b + 1)} \cdot \left(M_b^{(i)} \right)^{1/2} \left(M_b^{(i+1)} \right)^{-1/2}, \tag{10}$$

as similarly given in [16, (9)]. This generalization allows one to consider arbitrary structures of $M_b^{(i)}$ across iterations.



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



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

Fig. 3. Cost minimization comparisons in CAOL (P1) with different BPG-type algorithms and datasets ($R = K = 49$ and $\alpha = 2.5 \times 10^{-4}$; solution (31) was used for sparse code updates; BPG (Xu & Ying '17) [31] used the maximum eigenvalue of Hessians for Lipschitz constants; the cross mark x denotes a termination point). A sharper majorization leads to faster convergence of BPEG-M; for all the training datasets considered in this paper, the majorization matrix in Proposition 5.1 is sharper than those in Lemmas 5.2–5.3.

Lemma 4.4 (Sequence bounds)**.** *Let $\{\widetilde{M}_b : b = 1, \ldots, B\}$ and $\{E_b : b = 1, \ldots, B\}$ be as in (6)–(7), respectively. The cost function decrease for the* ith *update satisfies*:

$$F_b(x_b^{(i)}) - F_b(x_b^{(i+1)}) \geq \frac{\lambda_b - 1}{4} \left\| x_b^{(i)} - x_b^{(i+1)} \right\|_{M_b^{(i+1)}}^2 - \frac{(\lambda_b - 1)\delta^2}{4} \left\| x_b^{(i-1)} - x_b^{(i)} \right\|_{M_b^{(i)}}^2 \tag{11}$$

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

Lemma 4.4 generalizes [31, Lem. 1] using $\{\lambda_b = 2\}$. Taking the majorization matrices in (11) to be scaled identities with Lipschitz constants, i.e., $M_b^{(i+1)} = L_b^{(i+1)} \cdot I$ and $M_b^{(i)} = L_b^{(i)} \cdot I$, where $L_b^{(i+1)}$ and $L_b^{(i)}$ are Lipschitz constants, the bound (11) becomes equivalent to that in [31, (13)]. Note that BPEG-M for block multi-convex problems in [16] can be viewed within BPEG-M in Algorithm 1, by similar reasons in [31, Rem. 2] – bound (11) holds for the block multi-convex problems by taking $E_b^{(i+1)}$ in (10) as $E_b^{(i+1)} \preceq \delta \cdot (M_b^{(i)})^{1/2} (M_b^{(i+1)})^{-1/2}$ in [16, Prop. 3.2].

[3] $F : \mathbb{R}^n \to (-\infty, +\infty]$ is proper if $\mathrm{dom} F \neq \emptyset$. F is lower bounded in $\mathrm{dom}(F) := \{x : F(x) < \infty\}$ if $\inf_{x \in \mathrm{dom}(F)} F(x) > -\infty$. F is lower semicontinuous at point x_0 if $\liminf_{x \to x_0} F(x) \geq F(x_0)$.

Proposition 4.5 (Square summability)**.** *Let* $\{x^{(i+1)} : i \geq 0\}$ *be generated by Algorithm 1. We have*

$$\sum_{i=0}^{\infty} \left\| x^{(i)} - x^{(i+1)} \right\|_2^2 < \infty. \tag{12}$$

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

Proposition 4.5 implies that

$$\left\| x^{(i)} - x^{(i+1)} \right\|_2^2 \to 0, \tag{13}$$

and (13) is used to prove the following theorem:

Theorem 4.6 (A limit point is a critical point)**.** *Under Assumptions 1–3, let* $\{x^{(i+1)} : i \geq 0\}$ *be generated by Algorithm 1. Then any limit point* $\bar{x}$ *of* $\{x^{(i+1)} : i \geq 0\}$ *is a critical point of (3). If the subsequence* $\{x^{(i_j+1)}\}$ *converges to* $\bar{x}$*, then*

$$\lim_{j \to \infty} F(x^{(i_j+1)}) = F(\bar{x}).$$

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

Finite limit points exist if the generated sequence $\{x^{(i+1)} : i \geq 0\}$ is bounded; see, for example, [44, Lem. 3.2–3.3]. For some applications, the boundedness of $\{x^{(i+1)} : i \geq 0\}$ can be satisfied by choosing appropriate regularization parameters, e.g., [16].

C. Restarting BPEG-M

BPEG-type methods [16], [31], [38] can be further accelerated by applying *1)* a momentum coefficient formula similar to those used in fast proximal gradient (FPG) methods [45]–[47], and/or *2)* an adaptive momentum restarting scheme [48], [49]; see [16]. This section applies these two techniques to further accelerate BPEG-M in Algorithm 1.

First, we apply the following increasing momentum-coefficient formula to (7) [45]:

$$e_b^{(i+1)} = \frac{\theta^{(i)} - 1}{\theta^{(i+1)}}, \quad \theta^{(i+1)} = \frac{1 + \sqrt{1 + 4(\theta^{(i)})^2}}{2}. \tag{14}$$

This choice guarantees fast convergence of FPG method [45]. Second, we apply a momentum restarting scheme [48], [49], when the following *gradient-mapping* criterion is met [16]:

$$\cos\Big(\Theta\Big(M_b^{(i+1)}\Big(\acute{x}_b^{(i+1)} - x_b^{(i+1)}\Big), x_b^{(i+1)} - x_b^{(i)}\Big)\Big) > \omega, \tag{15}$$

where the angle between two nonzero real vectors ϑ and ϑ' is $\Theta(\vartheta, \vartheta') := \langle \vartheta, \vartheta' \rangle / (\|\vartheta\|_2 \|\vartheta'\|_2)$ and $\omega \in [-1, 0]$. This scheme restarts the algorithm whenever the momentum, i.e., $x_b^{(i+1)} - x_b^{(i)}$, is likely to lead the algorithm in an unhelpful direction, as measured by the gradient mapping at the $x_b^{(i+1)}$-update. We refer to BPEG-M combined with the methods (14)–(15) as restarting BPEG-M (reBPEG-M). Section S.VI in the supplementary material summarizes the updates of reBPEG-M.

To solve the block multi-nonconvex problems proposed in this paper (e.g., (P1)–(P3)), we apply reBPEG-M (a variant of Algorithm 1; see Algorithm S.1), promoting fast convergence to a critical point.

V. Fast and Convergent CAOL via BPEG-M

This section applies the general BPEG-M approach to CAOL. The CAOL models (P1) and (P2) satisfy the assumptions of BPEG-M; see Assumption 1–3 in Section IV-B. CAOL models (P1) and (P2) readily satisfy Assumption 1 of BPEG-M. To show the continuously differentiability of f and the lower boundedness of F, consider that *1)* $\sum_l \sum_k \frac{1}{2} \left\| d_k \circledast x_l - z_{l,k} \right\|_2^2$ in (P0) is continuously differentiable with respect to D and $\{z_{l,k}\}$; *2)* the sequences $\{D^{(i+1)}\}$ are bounded, because they are in the compact set $\mathcal{D}_{(\mathrm{P1})} = \{D : DD^H = \frac{1}{R} I\}$ and $\mathcal{D}_{(\mathrm{P2})} = \{d_k : \|d_k\|_2^2 = \frac{1}{R}, \forall k\}$ in (P1) and (P2), respectively; and *3)* the positive thresholding parameter α ensures that the sequence $\{z_{l,k}^{(i+1)}\}$ is bounded (otherwise the cost would diverge). In addition, for both (P1) and (P2), the lower semicontinuity of regularizer g_b holds, $\forall b$. For D-optimization, the indicator function of the sets $\mathcal{D}_{(\mathrm{P1})}$ and $\mathcal{D}_{(\mathrm{P2})}$ is lower semicontinuous, because the sets are compact. For $\{z_{l,k}\}$-optimization, the ℓ^0-quasi-norm is a lower semicontinuous function. Assumptions 2 and 3 are satisfied with the majorization matrix designs in this section – see Sections V-A–V-B later – and the extrapolation matrix design in (7), respectively.

Since CAOL models (P1) and (P2) satisfy the BPEG-M conditions, we solve (P1) and (P2) by the reBPEG-M method with a two-block scheme, i.e., we alternatively update all filters D and all sparse codes $\{z_{l,k} : l = 1, \ldots, L, k = 1, \ldots, K\}$. Sections V-A and V-B describe details of D-block and $\{z_{l,k}\}$-block optimization within the BPEG-M framework, respectively. The BPEG-M-based CAOL algorithm is particularly useful for learning convolutional regularizers from large datasets because of its memory flexibility and parallel computing applicability, as described in Section V-C and Sections V-A–V-B, respectively.

A. Filter Update: D-Block Optimization

We first investigate the structure of the system matrix in the filter update for (P0). This is useful for *1)* accelerating majorization matrix computation in filter updates (e.g., Lemmas 5.2–5.3) and *2)* applying $R \times N$-sized adjoint operators (e.g., Ψ_l^H in (17) below) to an N-sized vector without needing the Fourier approach [16, Sec. V-A] that uses commutativity of convolution and Parseval's relation. Given the current estimates of $\{z_{l,k} : l = 1, \ldots, L, k = 1, \ldots, K\}$, the filter update problem of (P0) is equivalent to

$$\underset{\{d_k\}}{\operatorname{argmin}} \frac{1}{2} \sum_{k=1}^{K} \sum_{l=1}^{L} \left\| \Psi_l d_k - z_{l,k} \right\|_2^2 + \beta g(D), \tag{16}$$

where D is defined in (1), $\Psi_l \in \mathbb{C}^{N \times R}$ is defined by

$$\Psi_l := \left[\, P_{B_1} \hat{x}_l \ \ldots \ P_{B_R} \hat{x}_l \,\right], \tag{17}$$

$P_{B_r} \in \mathbb{C}^{N \times \hat{N}}$ is the rth (rectangular) selection matrix that selects N rows corresponding to the indices $B_r = \{r, \ldots, r + N - 1\}$ from $I_{\hat{N}}$, $\{\hat{x}_l \in \mathbb{C}^{\hat{N}} : l = 1, \ldots, L\}$ is a set of padded training data, $\hat{N} = N + R - 1$. Note that applying Ψ_l^H in (17) to a vector of size N is analogous to calculating cross-correlation between $\hat{x}_l$ and the vector, i.e., $(\Psi_l^H \hat{z}_{l,k})_r = \sum_{n=1}^{N} \hat{x}_{n+r-1}^* (\hat{z}_{l,k})_n$, $r = 1, \ldots, R$. In general, $\hat{(\cdot)}$ denotes a padded signal vector.

TABLE I

COMPUTATIONAL COMPLEXITY OF DIFFERENT MAJORIZATION MATRIX DESIGNS FOR THE FILTER UPDATE PROBLEM (16)

Lemma 5.2–5.3	Proposition 5.1
$O(LRN)$	$O(LR^2N)$

1) Majorizer Design: This subsection designs multiple majorizers for the D-block optimization and compares their required computational complexity and tightness. The next proposition considers the structure of Ψ_l in (17) to obtain the Hessian $\sum_{l=1}^L \Psi_l^H \Psi_l \in \mathbb{C}^{R \times R}$ in (16) for an arbitrary boundary condition.

Proposition 5.1 (Exact Hessian Matrix M_D)**.** *The following matrix $M_D \in \mathbb{C}^{R \times R}$ is identical to $\sum_{l=1}^L \Psi_l^H \Psi_l$:*

$$[M_D]_{r,r'} = \sum_{l=1}^{L} \langle P_{B_r} \hat{x}_l, P_{B_{r'}} \hat{x}_l \rangle, \quad r, r' = 1, \ldots, R. \tag{18}$$

A sufficiently large number of training signals (with $N \geq R$), L, can guarantee $M_D = \sum_{l=1}^L \Psi_l^H \Psi_l \succ 0$ in Proposition 5.1. The drawback of using Proposition 5.1 is its polynomial computational complexity, i.e., $O(LR^2 \, N)$ – see Table I. When L (the number of training signals) or N (the size of training signals) are large, the quadratic complexity with the size of filters – R^2 – can quickly increase the total computational costs when multiplied by L and N. (The BPG setup in [31] additionally requires $O(R^3)$ because it uses the eigendecomposition of (18) to calculate the Lipschitz constant.)

Considering CAOL problems (P0) themselves, different from CDL [13]–[17], the complexity $O(LR^2N)$ in applying Proposition 5.1 is reasonable. In BPEG-M-based CDL [16], [17], a majorization matrix for kernel update is calculated every iteration because it depends on updated sparse codes; however, in CAOL, one can precompute M_D via Proposition 5.1 (or Lemmas 5.2–5.3 below) without needing to change it every kernel update. The polynomial computational cost in applying Proposition 5.1 becomes problematic only when the training signals change. Examples include *1)* hierarchical CAOL, e.g., CNN in Appendix A, *2)* "adaptive-filter MBIR" particularly with high-dimensional signals [2], [6], [50], and *3)* online learning [51], [52]. Therefore, we also describe a more efficiently computable majorization matrix at the cost of looser bounds (i.e., slower convergence; see Fig 3). Applying Lemma S.1, we first introduce a diagonal majorization matrix M_D for the Hessian $\sum_l \Psi_l^H \Psi_l$ in (16):

Lemma 5.2 (Diagonal majorization matrix M_D)**.** *The following matrix $M_D \in \mathbb{C}^{R \times R}$ satisfies $M_D \succeq \sum_{l=1}^L \Psi_l^H \Psi_l$:*

$$M_D = \mathrm{diag}\left(\sum_{l=1}^{L} |\Psi_l^H| |\Psi_l| 1_R \right), \tag{19}$$

where $|\cdot|$ takes the absolute values of the elements of a matrix.

The majorization matrix design in Lemma 5.2 is more efficient to compute than that in Proposition 5.1, because no R^2-factor is needed for calculating M_D in Lemma 5.2, i.e., $O(LRN)$; see Table I. Designing M_D in Lemma 5.2 takes fewer calculations than [16, Lem. 5.1] using Fourier approaches, when $R < \log(\hat{N})$. Using Lemma S.2, we next design a potentially sharper majorization matrix than (19), while maintaining the cost $O(LRN)$:

Lemma 5.3 (Scaled identity majorization matrix M_D)**.** *The following matrix $M_D \in \mathbb{C}^{R \times R}$ satisfies $M_D \succsim \sum_{l=1}^L \Psi_l^H \Psi_l$:*

$$M_D = \sum_{r=1}^{R} \left| \sum_{l=1}^{L} \langle P_{B_1} \hat{x}_l, P_{B_r} \hat{x}_l \rangle \right| \cdot I_R, \tag{20}$$

for a circular boundary condition.

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

For all the training datasets used in this paper, we observed that the tightness of majorization matrices in Proposition 5.1 and Lemmas 5.2–5.3 for the Hessian $\sum_l \Psi_l^H \Psi_l$ is given by

$$\sum_{l=1}^{L} \Psi_l^H \Psi_l = (18) \preceq (20) \preceq (19). \tag{21}$$

(Note that (18)$\preceq$(19) always holds regardless of training data.) Fig. 3 illustrates the effects of the majorizer sharpness in (21) on CAOL convergence rates. As described in Section IV-A, selecting λ_D (see (22) and (26) below) controls the tradeoff between majorization sharpness and extrapolation effect. We found that using fixed $\lambda_D = 1 + \epsilon$ gives faster convergence than $\lambda_D = 2$; see Fig. 4 (this behavior is more obvious in solving the CT MBIR model in (P3) via BPEG-M – see [32, Fig. 3]). The results in Fig. 4 and [32, Fig. 3] show that, under the sharp majorization regime, maintaining sharper majorization is more critical in accelerating the convergence of BPEG-M than giving more weight to extrapolation.

Sections V-A2 and V-A3 below apply the majorization matrices designed in this section to proximal mappings of D-optimization in (P1) and (P2), respectively.

2) Proximal Mapping With Orthogonality Constraint: The corresponding proximal mapping problem of (16) using the orthogonality constraint in (P1) is given by

$$\{d_k^{(i+1)}\} = \operatorname*{argmin}_{\{d_k\}} \; \sum_{k=1}^{K} \frac{1}{2} \left\| d_k - \nu_k^{(i+1)} \right\|_{\widetilde{M}_D}^2 ,$$

$$\text{subject to } DD^H = \frac{1}{R} \cdot I, \tag{22}$$

where

$$\nu_k^{(i+1)} = \acute{d}_k^{(i+1)} - \widetilde{M}_D^{-1} \textstyle\sum_{l=1}^{L} \Psi_l^H \left(\Psi_l \acute{d}_k^{(i+1)} - z_{l,k} \right), \tag{23}$$

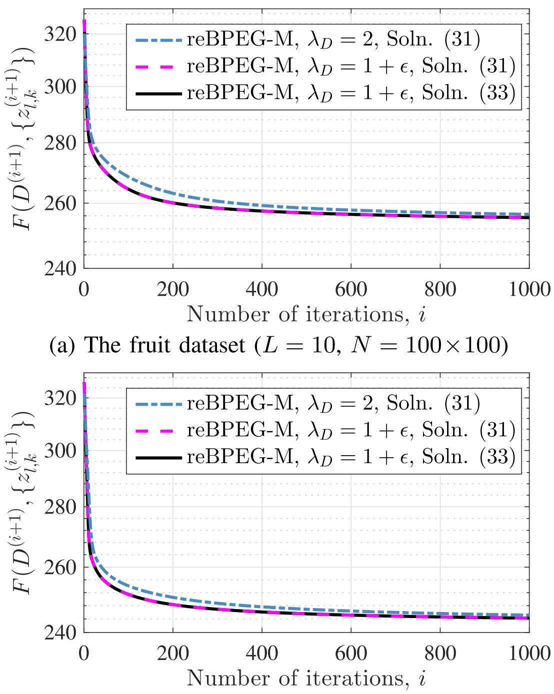
$$\acute{d}_k^{(i+1)} = d_k^{(i)} + E_D^{(i+1)} \left(d_k^{(i)} - d_k^{(i-1)} \right), \tag{24}$$

for $k = 1, \ldots, K$, and $\widetilde{M}_D = \lambda_D M_D$ by (6). One can parallelize over $k = 1, \ldots, K$ in computing $\{\nu_k^{(i+1)}\}$ in (23). The proposition below provides an optimal solution to (22):

Proposition 5.4. *Consider the following constrained minimization problem:*

$$\min_{D} \; \left\| \widetilde{M}_D^{1/2} D - \widetilde{M}_D^{1/2} \mathcal{V} \right\|_{\mathrm{F}}^2, \quad \text{subj. to } DD^H = \frac{1}{R} \cdot I, \tag{25}$$

where D is given as (1), $\mathcal{V} = [\nu_1^{(i+1)} \cdots \nu_K^{(i+1)}] \in \mathbb{C}^{R \times K}$, $\widetilde{M}_D = \lambda_D M_D$, and $M_D \in \mathbb{R}^{R \times R}$ is given by (18), (19),



(a) The fruit dataset ($L = 10$, $N = 100\times100$)

(b) The city dataset ($L = 10$, $N = 100\times100$)

Fig. 4. Cost minimization comparisons in CAOL (P1) with different BPEG-M algorithms and datasets (Lemma 5.2 was used for M_D; $R = K = 49$; deterministic filter initialization and random sparse code initialization). Under the sharp majorization regime, maintaining sharp majorization (i.e., $\lambda_D = 1+\epsilon$) provides faster convergence than giving more weight on extrapolation (i.e., $\lambda_D = 2$). (The same behavior was found in sparse-view CT application [32, Fig. 3].) There exist no differences in convergence between solution (31) and solution (33) using $\{\lambda_Z = 1+\epsilon\}$.

or (20). The optimal solution to (25) is given by

$$D^\star = \frac{1}{\sqrt{R}} \cdot U \left[I_R, 0_{R\times(K-R)} \right] V^H, \quad \textit{for } R \le K,$$

where $\widetilde{M}_D \mathcal{V}$ *has (full) singular value decomposition,* $\widetilde{M}_D \mathcal{V} = U \Lambda V^H$.

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

When using Proposition 5.1, $\widetilde{M}_D \nu_k^{(i+1)}$ of $\widetilde{M}_D \mathcal{V}$ in Proposition 5.4 simplifies to the following update:

$$\widetilde{M}_D \nu_k^{(i+1)} = (\lambda_D - 1)\, M_D \acute{d}_k^{(i+1)} + \sum_{l=1}^{L} \Psi_l^H z_{l,k}.$$

Similar to obtaining $\{\nu_k^{(i+1)}\}$ in (23), computing $\{\widetilde{M}_D \nu_k^{(i+1)} : k = 1, \ldots, K\}$ is parallelizable over k.

3) Proximal Mapping With Diversity Promoting Regularizer: The corresponding proximal mapping problem of (16) using the norm constraint and diversity promoting regularizer in (P2) is given by

$$\{d_k^{(i+1)}\} = \operatorname*{argmin}_{\{d_k\}} \sum_{k=1}^{K} \frac{1}{2} \left\| d_k - \nu_k^{(i+1)} \right\|_{\widetilde{M}_D}^2 + \frac{\beta}{2} g_{\text{div}}(D),$$
$$\text{subject to } \|d_k\|_2^2 = \frac{1}{R}, \quad k = 1, \ldots, K, \tag{26}$$

where $g_{\text{div}}(D)$, $\nu_k^{(i+1)}$, and $\acute{d}_k^{(i+1)}$ are given as in (P2), (23), and (24), respectively. We first decompose the regularization term $g_{\text{div}}(D)$ as follows:

$$\begin{aligned} g_{\text{div}}(D) &= \sum_{k=1}^{K} \sum_{k'=1}^{K} \left(d_k^H d_{k'} d_{k'}^H d_k - R^{-1} \right) \\ &= \sum_{k=1}^{K} d_k^H \Big(\sum_{k' \neq k} d_{k'} d_{k'}^H \Big) d_k + \left(d_k^H d_k - R^{-1} \right)^2 \\ &= \sum_{k=1}^{K} d_k^H \Gamma_k d_k, \end{aligned} \tag{27}$$

where the equality in (27) holds by using the constraint in (26), and the Hermitian matrix $\Gamma_k \in \mathbb{C}^{R\times R}$ is defined by

$$\Gamma_k := \sum_{k' \neq k} d_{k'} d_{k'}^H. \tag{28}$$

Using (27) and (28), we rewrite (26) as

$$d_k^{(i+1)} = \operatorname*{argmin}_{d_k} \frac{1}{2} \left\| d_k - \nu_k^{(i)} \right\|_{\widetilde{M}_D}^2 + \frac{\beta}{2} d_k^H \Gamma_k d_k,$$
$$\text{subject to } \|d_k\|_2^2 = \frac{1}{R}, \quad k = 1, \ldots, K. \tag{29}$$

This is a quadratically constrained quadratic program with $\{\widetilde{M}_D + \beta\Gamma_k \succ 0 : k = 1, \ldots, K\}$. We apply an accelerated Newton's method to solve (29); see Section S.IX. Similar to solving (22) in Section V-A2, solving (26) is a small-dimensional problem (K separate problems of size R).

B. Sparse Code Update: $\{z_{l,k}\}$-Block Optimization

Given the current estimate of D, the sparse code update problem for (P0) is given by

$$\operatorname*{argmin}_{\{z_{l,k}\}} \sum_{l=1}^{L} \sum_{k=1}^{K} \frac{1}{2} \left\| d_k \circledast x_l - z_{l,k} \right\|_2^2 + \alpha \left\| z_{l,k} \right\|_0. \tag{30}$$

This problem separates readily, allowing parallel computation with LK threads. An optimal solution to (30) is efficiently obtained by the well-known hard thresholding:

$$z_{l,k}^{(i+1)} = \mathcal{H}_{\sqrt{2\alpha}} \left(d_k \circledast x_l \right), \tag{31}$$

for $k = 1, \ldots, K$ and $l = 1, \ldots, L$, where

$$\mathcal{H}_a(x)_n := \begin{cases} 0, & |x_n| < a_n, \\ x_n, & |x_n| \ge a_n. \end{cases} \tag{32}$$

for all n. Considering λ_Z (in $\widetilde{M}_Z = \lambda_Z M_Z$) as $\lambda_Z \to 1$, the solution obtained by the BPEG-M approach becomes equivalent to (31). To show this, observe first that the BPEG-M-based solution (using $M_Z = I_N$) to (30) is obtained by

$$\begin{aligned} z_{l,k}^{(i+1)} &= \mathcal{H}_{\sqrt{\frac{2\alpha}{\lambda_Z}}} \Big(\zeta_{l,k}^{(i+1)} \Big), \\ \zeta_{l,k}^{(i+1)} &= \Big(1 - \lambda_Z^{-1} \Big) \cdot \acute{z}_{l,k}^{(i+1)} + \lambda_Z^{-1} \cdot d_k \circledast x_l, \\ \acute{z}_{l,k}^{(i+1)} &= z_{l,k}^{(i)} + E_Z^{(i+1)} \Big(z_{l,k}^{(i)} - z_{l,k}^{(i-1)} \Big). \end{aligned} \tag{33}$$

The downside of applying solution (33) is that it would require additional memory to store the corresponding extrapolated points – $\{\acute{z}_{l,k}^{(i+1)}\}$ – and the memory grows with N, L, and K. Considering the sharpness of the

TABLE II
COMPARISONS OF COMPUTATIONAL COMPLEXITY AND MEMORY USAGES BETWEEN CAOL AND PATCH-DOMAIN APPROACH

A. Computational complexity per BPEG-M iteration		
	Filter update	Sparse code update
CAOL (P1)	$O(LKRN)+O(R^2K)$	$O(LKRN)$
Patch-domain [6]†	$O(LR^2N)+O(R^3)$	$O(LR^2N)$

B. Memory usage for BPEG-M algorithm		
	Filter update	Sparse code update
CAOL (P1)	$O(LN)+O(RK)$	$O(LKN)$
Patch-domain [6]†	$O(LRN)+O(R^2)$	$O(LRN)$

† The patch-domain approach [6] considers the orthogonality constraint in (P1) with $R=K$; see Section III-A. The estimates consider all the extracted overlapping patches of size R with the stride parameter 1 and periodic boundaries, as used in convolution.

majorizer in (30), i.e., $M_Z = I_N$, and the memory issue, it is reasonable to consider the solution (33) with no extrapolation, i.e., $\{E_Z^{(i+1)} = 0\}$:

$$z_{l,k}^{(i+1)} = \mathcal{H}_{\sqrt{\frac{2\alpha}{\lambda_Z}}}\left((\lambda_Z - 1)^{-1} \lambda_Z \cdot z_{l,k}^{(i)} + \lambda_Z^{-1} \cdot d_k \circledast x_l \right)$$

becoming equivalent to (31) as $\lambda_Z \to 1$.

Solution (31) has two benefits over (33): compared to (33), (31) requires only half the memory to update all $z_{l,k}^{(i+1)}$ vectors and no additional computations related to $\acute{z}_{l,k}^{(i+1)}$. While having these benefits, empirically (31) has equivalent convergence rates as (33) using $\{\lambda_Z = 1+\epsilon\}$; see Fig. 4. Throughout the paper, we solve the sparse coding problems (e.g., (30) and $\{z_k\}$-block optimization in (P3)) via optimal solutions in the form of (31).

C. Lower Memory Use Than Patch-Domain Approaches

The convolution perspective in CAOL (P0) requires much less memory than conventional patch-domain approaches; thus, it is more suitable for learning filters from large datasets or applying the learned filters to high-dimensional MBIR problems. First, consider the training stage (e.g., (P0)). The patch-domain approaches, e.g., [1], [6], [7], require about R times more memory to store training signals. For example, 2D patches extracted by $\sqrt{R} \times \sqrt{R}$-sized windows (with "stride" one and periodic boundaries [6], [12], as used in convolution) require about R (e.g., $R = 64$ [1], [7]) times more memory than storing the original image of size $\sqrt{N} \times \sqrt{N}$. For L training images, their memory usage dramatically increases with a factor LRN. This becomes even more problematic in forming hierarchical representations, e.g., CNNs – see Appendix A. Unlike the patch-domain approaches, the memory use of CAOL (P0) only depends on the LN-factor to store training signals. As a result, the BPEG-M algorithm for CAOL (P1) requires about two times less memory than the patch-domain approach [6] (using BPEG-M). See Table II-B. (Both the corresponding BPEG-M algorithms use identical computations per iteration that scale with $LR^2 N$; see Table II-A.)

Second, consider solving MBIR problems. Different from the training stage, the memory burden depends on how one applies the learned filters. In [53], the learned filters are applied with the conventional convolutional operators – e.g., $\circledast$ in (P0) – and, thus, there exists no additional memory burden. However, in [2], [54], [55], the $\sqrt{R} \times \sqrt{R}$-sized learned kernels are applied with a matrix constructed by many overlapping patches extracted from the updated image at each iteration. In adaptive-filter MBIR problems [2], [6], [8], the memory issue pervades the patch-domain approaches.

VI. SPARSE-VIEW CT MBIR USING CONVOLUTIONAL REGULARIZER LEARNED VIA CAOL, AND BPEG-M

This section introduces a specific example of applying the learned convolutional regularizer, i.e., $F(D^\star, \{z_{l,k}\})$ in (P0), from a representative dataset to recover images in *extreme* imaging that collects highly undersampled or noisy measurements. We choose a sparse-view CT application since it has interesting challenges in reconstructing images that include Poisson noise in measurements, nonuniform noise or resolution properties in reconstructed images, and complicated (or no) structures in the system matrices. For CT, undersampling schemes can significantly reduce the radiation dose and cancer risk from CT scanning. The proposed approach can be applied to other applications (by replacing the data fidelity and spatial strength regularization terms in (P3) below).

We pre-learn TF filters $\{d_k^\star \in \mathbb{R}^K : k = 1, \ldots, K\}$ via CAOL (P1) with a set of high-quality (e.g., normal-dose) CT images $\{x_l : l = 1, \ldots, L\}$. To reconstruct a linear attenuation coefficient image $x \in \mathbb{R}^{N'}$ from post-log measurement $y \in \mathbb{R}^m$ [54], [56], we apply the learned convolutional regularizer to CT MBIR and solve the following block multi-nonconvex problem [32], [35]:

$$\begin{aligned} \operatorname*{argmin}_{x \ge 0} \; & \underbrace{\frac{1}{2} \|y - Ax\|_W^2}_{\text{data fidelity } f(x;\, y)} \\ & + \underbrace{\gamma \cdot \min_{\{z_k\}} \sum_{k=1}^{K} \frac{1}{2} \left\| d_k^\star \circledast x - z_k \right\|_2^2 + \alpha' \sum_{n=1}^{N'} \psi_j \phi((z_k)_n)}_{\text{learned convolutional regularizer } g(x,\, \{z_k\};\, \{d_k\})} . \end{aligned} \tag{P3}$$

Here, $A \in \mathbb{R}^{m \times N'}$ is a CT system matrix, $W \in \mathbb{R}^{m \times m}$ is a (diagonal) weighting matrix with elements $\{W_{l,l} = \rho_l^2 / (\rho_l + \sigma^2) : l = 1, \ldots, m\}$ based on a Poisson-Gaussian model for the pre-log measurements $\rho \in \mathbb{R}^m$ with electronic readout noise variance σ^2 [54]–[56], $\psi \in \mathbb{R}^{N'}$ is a pre-tuned spatial strength regularization vector [57] with non-negative elements $\{\psi_n = (\sum_{l=1}^m A_{l,n}^2 \, W_{l,l})^{1/2} / (\sum_{l=1}^m A_{l,n}^2)^{1/2} : n = 1, \ldots, N'\}$[4] that promotes uniform resolution or noise properties in the reconstructed image [54, Appx.], an indicator function $\phi(a)$ is equal to 0 if $a = 0$, and is 1 otherwise, $z_k \in \mathbb{R}^{N'}$ is unknown sparse code for the kth filter, and $\alpha' > 0$ is a thresholding parameter.

We solved (P3) via reBPEG-M in Section IV with a two-block scheme [32], and summarize the corresponding

[4]See details of computing $\{A_{l,j}^2 : \forall l, j\}$ in [32].

BPEG-M updates as

$$x^{(i+1)} = \left[\left(\widetilde{M}_A + \gamma I_R \right)^{-1} \cdot \left(\widetilde{M}_A \eta^{(i+1)} + \gamma \sum_{k=1}^{K} (P_f d_k^\star) \circledast \mathcal{H}_{\sqrt{2\alpha' \psi}} \left(d_k^\star \circledast x^{(i)} \right) \right) \right]_{\geq 0}, \quad (34)$$

where

$$\eta^{(i+1)} = \acute{x}^{(i+1)} - \widetilde{M}_A^{-1} A^T W \left(A \acute{x}^{(i+1)} - y \right),$$
$$\acute{x}^{(i+1)} = x^{(i)} + E_A^{(i+1)} \left(x^{(i)} - x^{(i-1)} \right), \quad (35)$$

$\widetilde{M}_A = \lambda_A M_A$ by (6), a diagonal majorization matrix $M_A \succeq A^T W A$ is designed by Lemma S.1, and $P_f \in \mathbb{C}^{R \times R}$ flips a column vector in the vertical direction (e.g., it rotates 2D filters by 180°). Interpreting the update (34) leads to the following two remarks:

Remark 6.1. When the convolutional regularizer learned via CAOL (P1) is applied to MBIR, it works as an autoencoding CNN:

$$\mathcal{M}(x) = \sum_{k=1}^{K} (P_f d_k^\star) \circledast \mathcal{H}_{\sqrt{2\alpha'_k}} \left(d_k^\star \circledast x \right) \quad (36)$$

(setting $\psi = 1_{N'}$ and generalizing α' to $\{\alpha'_k : k = 1, \ldots, K\}$ in (P3)). This is an explicit mathematical motivation for constructing architectures of iterative regression CNNs for MBIR, e.g., BCD-Net [28], [58]–[60] and Momentum-Net [29], [30]. Particularly when the learned filters $\{d_k^\star\}$ in (36) satisfy the TF condition, they are useful for compacting energy of an input signal x and removing unwanted features via the non-linear thresholding in (36).

Remark 6.2. Update (34) improves the solution $x^{(i+1)}$ by weighting between *a)* the extrapolated point considering the data fidelity, i.e., $\eta^{(i+1)}$ in (35), and *b)* the "refined" update via the (ψ-weighting) convolutional autoencoder, i.e., $\sum_k (P_f d_k^\star) \circledast \mathcal{H}_{\sqrt{2\alpha'\psi}}(d_k^\star \circledast x^{(i)})$.

VII. Results and Discussion

A. Experimental Setup

This section examines the performance (e.g., scalability, convergence, and acceleration) and behaviors (e.g., effects of model parameters on filters structures and effects of dimensions of learned filter on MBIR performance) of the proposed CAOL algorithms and models, respectively.

1) CAOL: We tested the introduced CAOL models/algorithms for four datasets: *1)* the fruit dataset with $L = 10$ and $N = 100 \times 100$ [10]; *2)* the city dataset with $L = 10$ and $N = 100 \times 100$ [14]; *3)* the CT dataset of $L = 80$ and $N = 128 \times 128$, created by dividing down-sampled 512×512 XCAT phantom slices [61] into 16 sub-images [13], [62] – referred to the CT-(i) dataset; *4)* the CT dataset of with $L = 10$ and $N = 512 \times 512$ from down-sampled 512×512 XCAT phantom slices [61] – referred to the CT-(ii) dataset. The preprocessing includes intensity rescaling to [0, 1] [10], [13], [14] and/or (global) mean substraction [1], [63, §2], as conventionally used in many sparse coding studies, e.g., [1], [10], [13], [14], [63]. For the fruit and city datasets, we trained $K = 49$ filters of size $R = 7 \times 7$. For the CT dataset (i), we trained filters of size $R = 5 \times 5$, with $K = 25$ or $K = 20$. For CT reconstruction experiments, we learned the filters from the CT-(ii) dataset; however, we did not apply mean subtraction because it is not modeled in (P3).

The parameters for the BPEG-M algorithms were defined as follows.[5] We set the regularization parameters α, β as follows:

- CAOL (P1): To investigate the effects of α, we tested (P1) with different α's in the case $R = K$. For the fruit and city datasets, we used $\alpha = 2.5 \times \{10^{-5}, 10^{-4}\}$; for the CT-(i) dataset, we used $\alpha = \{10^{-4}, 2 \times 10^{-3}\}$. For the CT-(ii) dataset (for CT reconstruction experiments), see details in [32, Sec. V1].
- CAOL (P2): Once α is fixed from the CAOL (P1) experiments above, we tested (P2) with different β's to see its effects in the case $R > K$. For the CT-(i) dataset, we fixed $\alpha = 10^{-4}$, and used $\beta = \{5 \times 10^6, 5 \times 10^4\}$.

We set $\lambda_D = 1 + \epsilon$ as the default. We initialized filters in either deterministic or random ways. The deterministic filter initialization follows that in [6, Sec. 3.4]. When filters were randomly initialized, we used a scaled one-vector for the first filter. We initialize sparse codes mainly with a deterministic way that applies (31) based on $\{d_k^{(0)}\}$. If not specified, we used the random filter and deterministic sparse code initializations. For BPG [31], we used the maximum eigenvalue of Hessians for Lipschitz constants in (16), and applied the gradient-based restarting scheme in Section IV-C. We terminated the iterations if the relative error stopping criterion (e.g., [16, (44)]) is met before reaching the maximum number of iterations. We set the tolerance value as 10^{-13} for the CAOL algorithms using Proposition 5.1, and 10^{-5} for those using Lemmas 5.2–5.3, and the maximum number of iterations to 2×10^4.

The CAOL experiments used the convolutional operator learning toolbox [64].

2) Sparse-View CT MBIR With Learned Convolutional Regularizer via CAOL: We simulated sparse-view sinograms of size 888×123 ('detectors or rays' $\times$ 'regularly spaced projection views or angles', where 984 is the number of full views) with GE LightSpeed fan-beam geometry corresponding to a monoenergetic source with 10^5 incident photons per ray and no background events, and electronic noise variance $\sigma^2 = 5^2$. We avoided an inverse crime in our imaging simulation and reconstructed images with a coarser grid with $\Delta_x = \Delta_y = 0.9766$ mm; see details in [32, Sec. V-A2].

For EP MBIR, we finely tuned its regularization parameter to achieve both good root mean square error (RMSE) and structural similarity index measurement [65] values. For the CT MBIR model (P3), we chose the model parameters $\{\gamma, \alpha'\}$ that showed a good tradeoff between the data fidelity term and the learned convolutional regularizer, and set $\lambda_A = 1 + \epsilon$. We evaluated the reconstruction quality by the RMSE (in a modified Hounsfield unit, HU, where air is 0 HU and water is 1000 HU) in a region of interest. See further details in [32, Sec. V-A2] and Fig. 6.

[5] The remaining BPEG-M parameters not described here are identical to those in [16, VII-A2].

The imaging simulation and reconstruction experiments used the Michigan image reconstruction toolbox [66].

B. CAOL With BPEG-M

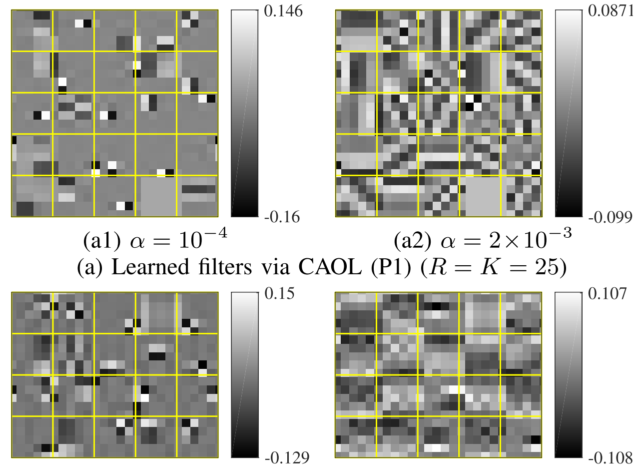
Under the sharp majorization regime (i.e., partial or all blocks have sufficiently tight bounds in Lemma 4.2), the proposed convergence-guaranteed BPEG-M can achieve significantly faster CAOL convergence rates compared with the state-of-the-art BPG algorithm [31] for solving block multi-nonconvex problems, by several generalizations of BPG (see Remark 4.3) and two majorization designs (see Proposition 5.1 and Lemma 5.3). See Fig. 3. In controlling the tradeoff between majorization sharpness and extrapolation effect of BPEG-M (i.e., choosing $\{\lambda_b\}$ in (6)–(7)), maintaining majorization sharpness is more critical than gaining stronger extrapolation effects to accelerate convergence under the sharp majorization regime. See Fig. 4.

While using about two times less memory (see Table II), CAOL (P0) learns TF filters corresponding to those given by the patch-domain TF learning in [6, Fig. 2]. See Section V-C and Fig. S.1 with deterministic $\{d_k^{(0)}\}$. Note that BPEG-M-based CAOL (P0) requires even less memory than BPEG-M-based CDL in [16], by using exact sparse coding solutions (e.g., (31) and (34)) without saving their extrapolated points. In particular, when tested with the large CT dataset of $\{L\!=\!40,\, N\!=\!512\!\times\!512\}$, the BPEG-M-based CAOL algorithm ran fine, while BPEG-M-based CDL [16] and patch-domain AOL [6] were terminated due to exceeding available memory.[6] In addition, the CAOL models (P1) and (P2) are easily parallelizable with K threads. Combining these results, the BPEG-M-based CAOL is a reasonable choice for learning filters from large training datasets. Finally, [34] shows theoretically how using many samples can improve CAOL, accentuating the benefits of the low memory usage of CAOL.

The effects of parameters for the CAOL models are shown as follows. In CAOL (P1), as the thresholding parameter α increases, the learned filters have more elongated structures; see Figs. 5(a) and S.2. In CAOL (P2), when α is fixed, increasing the filter diversity promoting regularizer β successfully lowers coherences between filters (e.g., $g_{\text{div}}(D)$ in (P2)); see Fig. 5(b).

In adaptive MBIR (e.g., [2], [6], [8]), one may apply adaptive image denoising [53], [67]–[71] to optimize thresholding parameters. However, if CAOL (P0) and testing the learned convolutional regularizer to MBIR (e.g., (P3)) are separated, selecting "optimal" thresholding parameters in (unsupervised) CAOL is challenging – similar to existing dictionary or analysis operator learning methods. Our strategy to select the thresholding parameter α in CAOL (P1) (with $R = K$) is given as follows. We first apply the first-order finite difference filters $\{d_k : \|d_k\|_2^2 = 1/R, \forall k\}$ (e.g., $\frac{1}{\sqrt{2R}}[1,-1]^T$ in 1D) to all training signals and find their sparse representations, and then find α_{est} that corresponds to the largest $95(\pm1)\%$ of non-zero elements of the sparsified training signals. This procedure defines the range $[\frac{1}{10}\alpha_{\text{est}}, \alpha_{\text{est}}]$ to select desirable $\alpha^\star$ and its corresponding filter $D^\star$. We next ran CAOL (P1) with multiple α values within this range. Selecting $\{\alpha^\star, D^\star\}$ depends on application. For CT MBIR, $D^\star$ that both has (short) first-order finite difference filters and captures diverse (particularly diagonal) features of training signals, gave good RMSE values and well preserved edges; see Fig. S.2(c) and [32, Fig. 2].

[6]Their double-precision MATLAB implementations were tested on 3.3 GHz Intel Core i5 CPU with 32 GB RAM.



(a1) $\alpha = 10^{-4}$ (a2) $\alpha = 2\times10^{-3}$

(a) Learned filters via CAOL (P1) ($R = K = 25$)

$g_{\text{div}}(D) = \mathbf{8.96\times10^{-6}}$ $g_{\text{div}}(D) = \mathbf{0.12}$

(b1) $\alpha = 10^{-4}$, $\beta = 5\times10^6$ (b2) $\alpha = 10^{-4}$, $\beta = 5\times10^4$

(b) Learned filters via CAOL (P2) ($R = 25, K = 20$)

Fig. 5. Examples of learned filters with different CAOL models and parameters (Proposition 5.1 was used for M_D; the CT-(i) dataset with a symmetric boundary condition).

C. Sparse-View CT MBIR With Learned Convolutional Sparsifying Regularizer (via CAOL) and BPEG-M

In sparse-view CT using only 12.5% of the full projections views, the CT MBIR (P3) using the learned convolutional regularizer via CAOL (P1) outperforms EP MBIR; it reduces RMSE by approximately 5.6–6.1HU. See the results in Figs. 6(c)–(e). The model (P3) can better recover high-contrast regions (e.g., bones) – see red arrows and magnified areas in Fig. 6(c)–(e). Nonetheless, the filters with $R = K = 5^2$ in the (ψ-weighting) autoencoding CNN, i.e., $\sum_k (P_f d_k^\star) \circledast \mathcal{H}_{\sqrt{2\alpha'\psi}}(d_k^\star \circledast (\cdot))$ in (36), can blur edges in low-contrast regions (e.g., soft tissues) while removing noise. See Fig. 6(d) – the blurry issues were similarly observed in [54], [55]. The larger dimensional kernels (i.e., $R = K = 7^2$) in the convolutional autoencoder can moderate this issue, while further reducing RMSE values; compare the results in Fig. 6(d)–(e). In particular, the larger dimensional convolutional kernels capture more diverse features – see [32, Fig. 2]) – and the diverse features captured in kernels are useful to further improve the performance of the proposed MBIR model (P3). (The importance of diverse features in kernels was similarly observed in CT experiments with the learned autoencoders having a fixed kernel dimension; see Fig. S.2(c).) The RMSE reduction over EP MBIR is comparable to that of CT MBIR (P3) using the $\{R, K = 8^2\}$-dimensional filters trained via the patch-domain AOL [7]; however, at each BPEG-M iteration, this MBIR model using the trained (non-TF) filters via patch-domain AOL [7] requires more computations than the proposed CT MBIR model (P3)

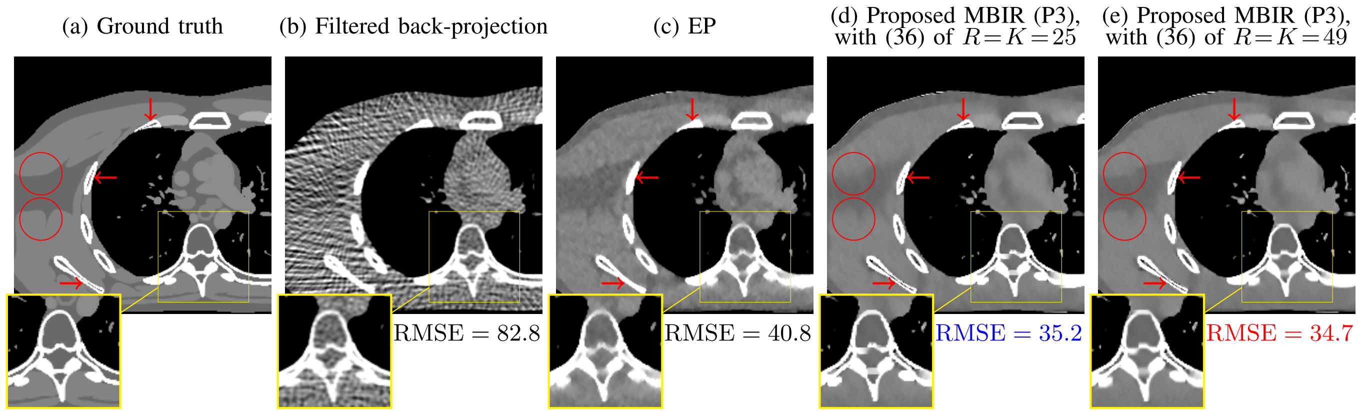


Fig. 6. Comparisons of reconstructed images from different reconstruction methods for sparse-view CT (123 views (12.5% sampling); for the MBIR model (P3), convolutional regularizers were trained by CAOL (P1) – see [32, Fig. 2]; display window is within [800, 1200] HU) [32]. The MBIR model (P3) using convolutional sparsifying regularizers trained via CAOL (P1) shows higher image reconstruction accuracy compared to the EP reconstruction; see red arrows and magnified areas. For the MBIR model (P3), the autoencoder (see Remark 6.1) using the filter dimension $R=K=49$ improves reconstruction accuracy of that using $R=K=25$; compare the results in (d) and (e). In particular, the larger dimensional filters improve the edge sharpness of reconstructed images; see circled areas. The corresponding error maps are shown in Fig. S.5 of the supplementary material.

using the learned convolutional regularizer via CAOL (P1). See related results and discussion in Fig. S.4 and Section S.X, respectively.

On the algorithmic side, the BPEG-M framework can guarantee the convergence of CT MBIR (P3). Under the sharp majorization regime in BPEG-M, maintaining the majorization sharpness is more critical than having stronger extrapolation effects – see [32, Fig. 3], as similarly shown in CAOL experiments (see Section VII-B).

VIII. Conclusion

Developing rapidly converging and memory-efficient CAOL engines is important, since it is a basic element in training CNNs in an unsupervised learning manner (see Appendix A). Studying structures of convolutional kernels is another fundamental issue, since it can avoid learning redundant filters or provide energy compaction properties to filters. The proposed BPEG-M-based CAOL framework has several benefits. First, the orthogonality constraint and diversity promoting regularizer in CAOL are useful in learning filters with diverse structures. Second, the proposed BPEG-M algorithm significantly accelerates CAOL over the state-of-the-art method, BPG [31], with our sufficiently sharp majorizer designs. Third, BPEG-M-based CAOL uses much less memory compared to patch-domain AOL methods [3], [4], [7], and easily allows parallel computing. Finally, the learned convolutional regularizer provides the autoencoding CNN architecture in MBIR, and outperforms EP reconstruction in sparse-view CT.

Similar to existing unsupervised synthesis or analysis operator learning methods, the biggest remaining challenge of CAOL is optimizing its model parameters. This would become more challenging when one applies CAOL to train CNNs (see Appendix A). Our first future work is developing "task-driven" CAOL that is particularly useful to train thresholding values. Other future works include further acceleration of BPEG-M in Algorithm 1, designing sharper majorizers requiring only $O(LRN)$ for the filter update problem of CAOL (P0), and applying the CNN model learned via (A1) to MBIR.

Appendix

A. Training CNN in a Unsupervised Manner via CAOL

This section mathematically formulates an unsupervised training cost function for classical CNN (e.g., LeNet-5 [11] and AlexNet [72]) and solves the corresponding optimization problem, via the CAOL and BPEG-M frameworks studied in Sections III–V. We model the three core modules of CNN: *1)* convolution, *2)* pooling, e.g., average [11] or max [63], and *3)* thresholding, e.g., RELU [73], while considering the TF filter condition in Proposition 3.1. Particularly, the orthogonality constraint in CAOL (P1) leads to a sharp majorizer, and BPEG-M is useful to train CNNs with convergence guarantees. Note that it is unclear how to train such diverse (or incoherent) filters described in Section III by the most common CNN optimization method, the stochastic gradient method in which gradients are computed by back-propagation. The major challenges include *a)* the non-differentiable hard thresholding operator related to ℓ^0-norm in (P0), *b)* the nonconvex filter constraints in (P1) and (P2), *c)* using the identical filters in both encoder and decoder (e.g., W and W^H in Section S.I), and *d)* vanishing gradients.

For simplicity, we consider a two-layer CNN with a single training image, but one can extend the CNN model (A1) (see below) to "deep" layers with multiple images. The first layer consists of *1c)* convolutional, *1t)* thresholding, and *1p)* pooling layers; the second layer consists of *2c)* convolutional and *2t)* thresholding layers. Extending CAOL (P1), we model two-layer CNN training as the following optimization problem:

$$
\begin{aligned}
\underset{\{d_k^{[1]}, d_{k,k'}^{[2]}\}}{\operatorname{argmin}} \; \min_{\{z_k^{[1]}, z_{k'}^{[2]}\}} \; & \sum_{k=1}^{K_1} \frac{1}{2} \left\| d_k^{[1]} \circledast x - z_k^{[1]} \right\|_2^2 + \alpha_1 \left\| z_k^{[1]} \right\|_0 \\
& + \frac{1}{2} \left\| \left(\sum_{k=1}^{K_1} \begin{bmatrix} d_{k,1}^{[2]} \circledast P z_k^{[1]} \\ \vdots \\ d_{k,K_2}^{[2]} \circledast P z_k^{[1]} \end{bmatrix} \right) - \begin{bmatrix} z_1^{[2]} \\ \vdots \\ z_{K_2}^{[2]} \end{bmatrix} \right\|_2^2 \\
& + \alpha_2 \sum_{k'=1}^{K_2} \left\| z_{k'}^{[2]} \right\|_0
\end{aligned}
$$

$$\text{subject to} \quad D^{[1]}\big(D^{[1]}\big)^H = \frac{1}{R_1}\cdot I, \qquad D_k^{[2]}\big(D_k^{[2]}\big)^H = \frac{1}{R_2}\cdot I, \quad k=1,\ldots,K_1, \tag{A1}$$

where $x \in \mathbb{R}^N$ is the training data, $\{d_k^{[1]} \in \mathbb{R}^{R_1} : k = 1,\ldots,K_1\}$ is a set of filters in the first convolutional layer, $\{z_k^{[1]} \in \mathbb{R}^N : k = 1,\ldots,K_1\}$ is a set of features after the first thresholding layer, $\{d_{k,k'}^{[2]} \in \mathbb{R}^{R_2} : k' = 1,\ldots,K_2\}$ is a set of filters for each of $\{z_k^{[1]}\}$ in the second convolutional layer, $\{z_{k'}^{[2]} \in \mathbb{R}^{N/\omega} : k = 1,\ldots,K_2\}$ is a set of features after the second thresholding layer, $D^{[1]}$ and $\{D_k^{[2]}\}$ are similarly given as in (1), $P \in \mathbb{R}^{N/\omega\times\omega}$ denotes an average pooling [11] operator (see its definition below), and ω is the size of pooling window. The superscripted number in the bracket of vectors and matrices denotes the $(\cdot)$th layer. Here, we model a simple average pooling operator $P \in \mathbb{R}^{(N/\omega)\times\omega}$ by a block diagonal matrix with row vector $\frac{1}{\omega}1_\omega^T \in \mathbb{R}^\omega$: $P := \frac{1}{\omega}\bigoplus_{j=1}^{N/\omega} 1_\omega^T$. We obtain a majorization matrix of $P^T P$ by $P^T P \preceq \mathrm{diag}(P^T P 1_N) = \frac{1}{\omega} I_N$ (using Lemma S.1). For 2D case, the structure of P changes, but $P^T P \preceq \frac{1}{\omega} I_N$ holds.

We solve the CNN training model in (A1) via the BPEG-M techniques in Section V, and relate the solutions of (A1) and modules in the two-layer CNN training. The symbols in the following items denote the CNN modules.

- *1c*) Filters in the first layer, $\{d_k^{[1]}\}$: Updating the filters is straightforward via the techniques in Section V-A2.
- *1t*) Features at the first layers, $\{z_k^{[1]}\}$: Using BPEG-M with the kth set of TF filters $\{d_{k,k'}^{[2]} : k'\}$ and $P^T P \preceq \frac{1}{\omega} I_N$ (see above), the proximal mapping for $z_k^{[1]}$ is

$$\min_{z_k^{[1]}} \frac{1}{2}\left\| d_k^{[1]} \circledast x - z_k^{[1]} \right\|_2^2 + \frac{1}{2\omega'}\left\| z_k^{[1]} - \zeta_k^{[k]} \right\|_2^2 + \alpha_1 \left\| z_k^{[1]} \right\|_0, \tag{37}$$

where $\omega' = \omega/\lambda_Z$ and $\zeta_k^{[k]}$ is given by (4). Combining the first two quadratic terms in (37) into a single quadratic term leads to an optimal update for (37):

$$z_k^{[1]} = \mathcal{H}_{\sqrt{2\frac{\omega'\alpha_1}{\omega'+1}}}\left(d_k^{[1]} \circledast x + \frac{1}{\omega'}\zeta_k^{[k]} \right), \quad k \in [K],$$

where the hard thresholding operator $\mathcal{H}_a(\cdot)$ with a thresholding parameter a is defined in (32).
- *1p*) Pooling, P: Applying the pooling operator P to $\{z_k^{[1]}\}$ gives input data – $\{P z_k^{[1]}\}$ – to the second layer.
- *2c*) Filters in the second layer, $\{d_{k,k'}^{[2]}\}$: We update the kth set filters $\{d_{k,k'}^{[2]} : \forall k'\}$ in a sequential way. Updating the kth set filters is straightforward via the techniques in Section V-A2.
- *2t*) Features at the second layers, $\{z_{k'}^{[2]}\}$: The corresponding update is given by

$$z_{k'}^{[2]} = \mathcal{H}_{\sqrt{2\alpha_2}}\left(\sum_{k=1}^{K_1} d_{k,k'}^{[1]} \circledast P z_k^{[1]} \right), \quad k' \in [K_2].$$

Considering the introduced mathematical formulation of training CNNs [11] via CAOL, BPEG-M-based CAOL has potential to be a basic engine to rapidly train CNNs with big data (i.e., training data consisting of many (high-dimensional) signals).

B. Examples of $\{f(x;y), \mathcal{X}\}$ in MBIR Model (B1) Using Learned Regularizers

This section introduces some potential applications of using MBIR model (B1) using learned regularizers in imaging processing, imaging, and computer vision. We first consider quadratic data fidelity function in the form of $f(x;y) = \frac{1}{2}\|y - Ax\|_W^2$. Examples include

- Image debluring (with $W = I$ for simplicity), where y is a blurred image, A is a blurring operator, and $\mathcal{X}$ is a box constraint;
- Image denoising (with $A = I$), where y is a noisy image corrupted by additive white Gaussian noise (AWGN), W is the inverse covariance matrix corresponding to AWGN statistics, and $\mathcal{X}$ is a box constraint;
- Compressed sensing (with $\{W = I, \mathcal{X} \in \mathbb{C}^{N'}\}$ for simplicity) [74], [75], where y is a measurement vector, and A is a compressed sensing operator, e.g., subgaussian random matrix, bounded orthonormal system, subsampled isometries, certain types of random convolutions;
- Image inpainting (with $W = I$ for simplicity), where y is an image with missing entries, A is a masking operator, and $\mathcal{X}$ is a box constraint;
- Light-field photography from focal stack data with $f(x;y) = \sum_c \|y_c - \sum_s A_{c,s} x_s\|_2^2$, where y_c denotes measurements collected at the cth sensor, $A_{c,s}$ models camera imaging geometry at the sth angular position for the cth detector, x_s denotes the sth sub-aperture image, $\forall c, s$, and $\mathcal{X}$ is a box constraint [29], [76].

Examples that use nonlinear data fidelity function include image classification using the logistic function [77], magnetic resonance imaging considering unknown magnetic field variation [78], and positron emission tomography [59].

C. 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-quasi-norm, i.e., the number of nonzeros of a vector. The Frobenius norm of a matrix is denoted by $\|\cdot\|_{\mathrm{F}}$. $(\cdot)^T$, $(\cdot)^H$, and $(\cdot)^*$ indicate the transpose, complex conjugate transpose (Hermitian transpose), and complex conjugate, respectively. $\mathrm{diag}(\cdot)$ denotes the conversion of a vector into a diagonal matrix or diagonal elements of a matrix into a vector. $\bigoplus$ denotes the matrix direct sum of matrices. $[C]$ denotes the set $\{1,2,\ldots,C\}$. Distinct from the index i, we denote the imaginary unit $\sqrt{-1}$ by i. 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

The authors thank Xuehang Zheng for providing CT imaging simulation setup, and Dr. Jonghoon Jin for constructive feedback on CNNs.

REFERENCES

[1] 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.

[2] M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[3] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, “Constrained overcomplete analysis operator learning for cosparse signal modelling,” *IEEE Trans. Signal Process.*, vol. 61, no. 9, pp. 2341–2355, May 2013.

[4] S. Hawe, M. Kleinsteuber, and K. Diepold, “Analysis operator learning and its application to image reconstruction,” *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2138–2150, Jun. 2013.

[5] 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.

[6] J. F. Cai, S. Huang, H. Ji, Z. Shen, and G. Ye, “Data-driven tight frame construction and image denoising,” *Appl. Comput. Harmon. Anal.*, vol. 37, no. 1, pp. 89–105, Jul. 2014.

[7] S. Ravishankar and Y. Bresler, “ℓ_0 sparsifying transform learning with efficient optimal updates and convergence guarantees,” *IEEE Trans. Image Process.*, vol. 63, no. 9, pp. 2389–2404, May 2015.

[8] L. Pfister and Y. Bresler, “Learning sparsifying filter banks,” *Proc. SPIE*, vol. 9597, pp. 959703–1–959703–10, Aug. 2015.

[9] 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, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., 2nd ed. Berlin, Germany: Springer-Verlag, 2012, ch. 22, pp. 561–580.

[10] 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.

[11] 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.

[12] V. Papyan, Y. Romano, and M. Elad, “Convolutional neural networks analyzed via convolutional sparse coding,” *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 2887–2938, Jan. 2017.

[13] H. Bristow, A. Eriksson, and S. Lucey, “Fast convolutional sparse coding,” in *Proc. IEEE CVPR*, Portland, OR, USA, Jun. 2013, pp. 391–398.

[14] 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.

[15] B. Wohlberg, “Efficient algorithms for convolutional sparse representations,” *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 301–315, Jan. 2016.

[16] I. Y. Chun and J. A. Fessler, “Convolutional dictionary learning: Acceleration and convergence,” *IEEE Trans. Image Process.*, vol. 27, no. 4, pp. 1697–1712, Apr. 2018.

[17] 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. Int. Conf. Sampling Theory Appl. (SampTA)*, Tallinn, Estonia, Jul. 2017, pp. 460–464.

[18] D. Barchiesi and M. D. Plumbley, “Learning incoherent dictionaries for sparse approximation using iterative projections and rotations,” *IEEE Trans. Signal Process.*, vol. 61, no. 8, pp. 2055–2065, Apr. 2013.

[19] C. Bao, J.-F. Cai, and H. Ji, “Fast sparsity-based orthogonal dictionary learning for image restoration,” in *Proc. IEEE ICCV*, Sydney, NSW, Australia, Dec. 2013, pp. 3384–3391.

[20] S. Ravishankar and Y. Bresler, “Learning overcomplete sparsifying transforms for signal processing,” in *Proc. IEEE ICASSP*, Vancouver, BC, Canada, May 2013, pp. 3088–3092.

[21] Y. Yang, J. Sun, H. Li, and Z. Xu, “Deep ADMM-net for compressive sensing MRI,” in *Proc. NIPS*, Long Beach, CA, USA, Dec. 2016, pp. 10–18.

[22] K. Zhang, W. Zuo, S. Gu, and L. Zhang, “Learning deep CNN denoiser prior for image restoration,” in *Proc. IEEE CVPR*, Honolulu, HI, USA, Jul. 2017, pp. 3929–3938.

[23] Y. Chen and T. Pock, “Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, Jun. 2017.

[24] H. Chen *et al.*, “LEARN: Learned experts’ assessment-based reconstruction network for sparse-data CT,” *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1333–1347, Jun. 2018.

[25] D. Wu, K. Kim, G. E. Fakhri, and Q. Li, “Iterative low-dose CT reconstruction with priors trained by artificial neural network,” *IEEE Trans. Med. Imag.*, vol. 36, no. 12, pp. 2479–2486, Sep. 2017.

[26] Y. Romano, M. Elad, and P. Milanfar, “The little engine that could: Regularization by denoising (RED),” *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1804–1844, Oct. 2017.

[27] G. T. Buzzard, S. H. Chan, S. Sreehari, and C. A. Bouman, “Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium,” *SIAM J. Imag. Sci.*, vol. 11, no. 3, pp. 2001–2020, Sep. 2018.

[28] Y. Chun and J. A. Fessler, “Deep BCD-net using identical encoding-decoding CNN structures for iterative image recovery,” in *Proc. IEEE IVMSP Workshop*, Zagori, Greece, Jun. 2018, pp. 1–5.

[29] I. Y. Chun, Z. Huang, H. Lim, and J. A. Fessler, “Momentum-net: Fast and convergent iterative neural network for inverse problems,” Jul. 2019, *arXiv:1907.11818*. [Online]. Available: https://arxiv.org/abs/1907.11818

[30] I. Y. Chun, H. Lim, Z. Huang, and J. A. Fessler, “Fast and convergent iterative signal recovery using trained convolutional neural networkss,” in *Proc. Allerton Conf. Commun., Control, Comput.*, Allerton, IL, USA, Oct. 2018, pp. 155–159.

[31] Y. Xu and W. Yin, “A globally convergent algorithm for nonconvex optimization based on block coordinate update,” *J. Sci. Comput.*, vol. 72, no. 2, pp. 700–734, Aug. 2017.

[32] I. Y. Chun and J. A. Fessler, “Convolutional analysis operator learning: Application to sparse-view CT,” in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, USA, Oct. 2018, pp. 1631–1635.

[33] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, “Solving inverse problems using data-driven models,” *Acta Numerica*, vol. 28, pp. 1–174, May 2019.

[34] I. Y. Chun, D. Hong, B. Adcock, and J. A. Fessler, “Convolutional analysis operator learning: Dependence on training data and compressed sensing recovery guarantees,” *IEEE Signal Process. Lett.*, vol. 26, no. 8, pp. 1137–1141, Jun. 2019. [Online]. Available: http://arxiv.org/abs/1902.08267

[35] C. Crockett, D. Hong, I. Y. Chun, and J. A. Fessler, “Incorporating handcrafted filters in convolutional analysis operator learning for ill-posed inverse problems,” in *Proc. IEEE Workshop CAMSAP*, Jul. 2019.

[36] R. Remi and K. Schnass, “Dictionary identification—Sparse matrix-factorization via ℓ_1-minimization,” *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3523–3539, Jul. 2010.

[37] P. Tseng, “Convergence of a block coordinate descent method for nondifferentiable minimization,” *J. Optim. Theory Appl.*, vol. 109, no. 3, pp. 475–494, Jun. 2001.

[38] 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.

[39] K. Lange, D. R. Hunter, and I. Yang, “Optimization transfer using surrogate objective functions,” *J. Comput. Graph. Statist.*, vol. 9, no. 1, pp. 1–20, Mar. 2000.

[40] M. W. Jacobson and J. A. Fessler, “An expanded theoretical treatment of iteration-dependent majorize-minimize algorithms,” *IEEE Trans. Image Process.*, vol. 16, no. 10, pp. 2411–2422, Oct. 2007.

[41] 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.

[42] A. Y. Kruger, “On Fréchet subdifferentials,” *J. Math. Sci.*, vol. 116, no. 3, pp. 3325–3358, Jul. 2003.

[43] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, vol. 317. Berlin, Germany: Springer-Verlag, 2009.

[44] C. Bao, H. Ji, and Z. Shen, “Convergence analysis for iterative data-driven tight frame construction scheme,” *Appl. Comput. Harmon. Anal.*, vol. 38, no. 3, pp. 510–523, May 2015.

[45] 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, 2009.

[46] Y. Nesterov. (2007). Gradient Methods for Minimizing Composite Objective Function. CORE Discussion Papers-2007/76, UCL, Louvain-la-Neuve, Belgium. [Online]. Available: http://www.uclouvain.be/cps/ucl/doc/core/documents/Composit.pdf

[47] P. Tseng. (May 2008). *On Accelerated Proximal Gradient Methods for Convex-Concave Optimization*. [Online]. Available: http://www.mit.edu/~dimitrib/PTseng/papers/apgm.pdf

[48] B. O'Donoghue and E. Candès, "Adaptive restart for accelerated gradient schemes," *Found. Comput. Math.*, vol. 15, no. 3, pp. 715–732, Jun. 2015.

[49] P. Giselsson and S. Boyd, "Monotonicity and restart in fast gradient methods," in *Proc. IEEE CDC*, Los Angeles, CA, USA, Dec. 2014, pp. 5058–5063.

[50] Q. Xu, H. Yu, X. Mou, L. Zhang, J. Hsieh, and G. Wang, "Low-dose X-ray CT reconstruction via dictionary learning," *IEEE Trans. Med. Imag.*, vol. 31, no. 9, pp. 1682–1697, Sep. 2012.

[51] J. Liu, C. Garcia-Cardona, B. Wohlberg, and W. Yin, "Online convolutional dictionary learning," in *Proc. IEEE ICIP*, Beijing, China, Sep. 2017, pp. 1707–1711.

[52] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. ICML*, Montreal, QC, Canada, Jun. 2009, pp. 689–696.

[53] L. Pfister and Y. Bresler, "Automatic parameter tuning for image denoising with learned sparsifying transforms," in *Proc. IEEE ICASSP*, Mar. 2017, pp. 6040–6044.

[54] I. Y. Chun, X. Zheng, Y. Long, and J. A. Fessler, "Sparse-view X-ray CT reconstruction using ℓ_1 regularization with learned sparsifying transform," in *Proc. Fully Three-Dimensional Image Reconstruct. Radiol. Nucl. Med.*, Xi'an, China, Jun. 2017, pp. 115–119.

[55] X. Zheng, I. Y. Chun, Z. Li, Y. Long, and J. A. Fessler, "Sparse-view X-ray CT reconstruction using ℓ_1 prior with learned transform," Feb. 2019, *arXiv:1711.00905*. [Online]. Available: https://arxiv.org/abs/1711.00905

[56] I. Y. Chun and T. Talavage, "Efficient compressed sensing statistical X-ray/CT reconstruction from fewer measurements," in *Proc. Int. Image Fully 3D Image Recognit. Rad. Nuc. Med.*, Lake Tahoe, CA, USA, Jun. 2013, pp. 30–33.

[57] J. A. Fessler and W. L. Rogers, "Spatial resolution properties of penalized-likelihood image reconstruction: Space-invariant tomographs," *IEEE Trans. Image Process.*, vol. 5, no. 9, pp. 1346–1358, Sep. 1996.

[58] I. Y. Chun, X. Zheng, Y. Long, and J. A. Fessler, "BCD-net for low-dose CT reconstruction: Acceleration, convergence, and generalization," in *Proc. Med. Image Comput. Comput. Assist. Interven.*, Shenzhen, China, to be published.

[59] H. Lim, I. Y. Chun, Y. K. Dewaraja, and J. A. Fessler, "Improved low-count quantitative PET reconstruction with a variational neural network," May 2019, *arXiv:1906.02327*. [Online]. Available: https://arxiv.org/abs/1906.02327

[60] H. Lim, J. A. Fessler, Y. K. Dewaraja, and I. Y. Chun, "Application of trained deep BCD-Net to iterative low-count PET image reconstruction," in *Proc. IEEE NSS-MIC*, Sydney, NSW, Australia, pp. 1–4.

[61] W. P. Segars, M. Mahesh, T. J. Beck, E. C. Frey, and B. M. Tsui, "Realistic CT simulation using the 4D XCAT phantom," *Med. Phys.*, vol. 35, no. 8, pp. 3800–3808, Jul. 2008.

[62] 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.

[63] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. IEEE ICCV*, Kyoto, Japan, Oct. 2009, pp. 2146–2153.

[64] I. Y. Chun. *CONVOLT: CONVolutional Operator Learning Toolbox (for MATLAB)*, 2019. [Online]. Available: https://github.com/mechatoz/convolt

[65] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[66] J. A. Fessler. *Michigan Image Reconstruction Toolbox (MIRT) for MATLAB*, 2016. [Online]. Available: http://web.eecs.umich.edu/~fessler

[67] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 613–627, May 1995.

[68] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *J. Amer. Statist. Assoc.*, vol. 90, no. 432, pp. 1200–1224, 1995.

[69] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Trans. Image Process.*, vol. 9, no. 9, pp. 1532–1546, Sep. 2000.

[70] T. Blu and F. Luisier, "The SURE-LET approach to image denoising," *IEEE Trans. Image Process.*, vol. 16, no. 11, pp. 2778–2786, Nov. 2007.

[71] H. Liu, R. Xiong, J. Zhang, and W. Gao, "Image denoising via adaptive soft-thresholding based on non-local samples," in *Proc. IEEE CVPR*, Boston, MA, USA, Jun. 2015, pp. 484–492.

[72] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1097–1105.

[73] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, Haifa, Israel, Jun. 2010, pp. 807–814.

[74] I. Y. Chun and B. Adcock, "Compressed sensing and parallel acquisition," *IEEE Trans. Inf. Theory*, vol. 63, no. 8, pp. 4860–4882, Aug. 2017. [Online]. Available: https://arxiv.org/abs/1601.06214

[75] I. Y. Chun and B. Adcock, "Uniform recovery from subgaussian multi-sensor measurements," *Appl. Comput. Harmon. Anal.*, to be published. [Online]. Available: http://arxiv.org/abs/1610.05758

[76] C. J. Blocker, Y. Chun, and J. A. Fessler, "Low-rank plus sparse tensor models for light-field reconstruction from focal stack data," in *Proc. IEEE IVMSP Workshop*, Zagori, Greece, Jun. 2018, pp. 1–5.

[77] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, "Supervised dictionary learning," in *Proc. NIPS*, Vancouver, BC, Canada, Dec. 2009, pp. 1033–1040.

[78] J. A. Fessler, "Model-based image reconstruction for MRI," *IEEE Signal Process. Mag.*, vol. 27, no. 4, pp. 81–89, Jul. 2010.

Il Yong Chun (S'14–M'16) received the B.Eng. degree in electrical engineering from Korea University in 2009, and the Ph.D. degree in electrical engineering from Purdue University in 2015. During his Ph.D. degree, he was with Intel Labs, the Samsung Advanced Institute of Technology, and Neuroscience Research Institute, as a Research Intern or a Visiting Lecturer. From 2015 to 2016, he was a Postdoctoral Research Associate of mathematics at Purdue University. From 2016 to 2019, he was a Research Fellow with the Department of Electrical Engineering and Computer Science, University of Michigan. He is currently a tenure-track Assistant Professor with the Department of Electrical Engineering, University of Hawai'i at Mānoa (UHM). His research interests include machine learning and AI with "big data," optimization, compressed sensing, and adaptive signal processing, applied to medical imaging, computational photography, and neuroscience.

Jeffrey A. Fessler (F'06) received the B.S.E.E. degree from Purdue University in 1985, the M.S.E.E. degree from Stanford University in 1986, the M.S. degree in statistics from Stanford University in 1989, and the Ph.D. degree in electrical engineering in 1990. From 1985 to 1988, he was a National Science Foundation Graduate Fellow at Stanford. From 1991 to 1992, he was a Department of Energy Alexander Hollaender Postdoctoral Fellow at the Division of Nuclear Medicine. From 1993 to 1995, he was an Assistant Professor of nuclear medicine and the bioengineering program. He is currently a Professor with the Department of Electrical Engineering and Computer Science, Department of Radiology, and Department of Biomedical Engineering. He is also the William L. Root Professor with the Department of Electrical Engineering and Computer Science, University of Michigan. His research interest includes statistical aspects of imaging problems. He has supervised doctoral research in PET, SPECT, X-ray CT, MRI, and optical imaging problems. He became a fellow of the IEEE in 2006, for contributions to the theory and practice of image reconstruction. He was a recipient of the Francois Erbsmann Award for his IPMI93 presentation and the Edward Hoffman Medical Imaging Scientist Award in 2013. He has chaired the IEEE T-MI Steering Committee and the ISBI Steering Committee. He was the Co-Chair of the 1997 SPIE Conference on Image Reconstruction and Restoration, the Technical Program Co-Chair of the 2002 IEEE International Symposium on Biomedical Imaging (ISBI), and the General Chair of ISBI 2007. He has served as an Associate Editor for the IEEE TRANSACTIONS ON MEDICAL IMAGING, the IEEE SIGNAL PROCESSING LETTERS, the IEEE TRANSACTIONS ON IMAGE PROCESSING, and the IEEE TRANSACTIONS ON COMPUTATIONAL IMAGING. He is currently serving as an Associate Editor for the *SIAM Journal on Imaging Sciences*.