SIAM J. Imaging Sciences
Vol. 11, No. 2, pp. 1589–1628

First- and Second-Order Methods for Online Convolutional Dictionary Learning[*]

Jialin Liu[†], Cristina Garcia-Cardona[‡], Brendt Wohlberg[§], and Wotao Yin[†]

Abstract. Convolutional sparse representations are a form of sparse representation with a structured, translation-invariant dictionary. Most convolutional dictionary learning algorithms to date operate in batch mode, requiring simultaneous access to all training images during the learning process, which results in very high memory usage and severely limits the training data size that can be used. Very recently, however, a number of authors have considered the design of online convolutional dictionary learning algorithms that offer far better scaling of memory and computational cost with training set size than batch methods. This paper extends our prior work, improving a number of aspects of our previous algorithm; proposing an entirely new one, with better performance, that supports the inclusion of a spatial mask for learning from incomplete data; and providing a rigorous theoretical analysis of these methods.

Key words. convolutional sparse coding, convolutional dictionary learning, online dictionary learning, stochastic gradient descent, recursive least squares

AMS subject classifications. 94A08, 68U10, 94A12, 90C26

DOI. 10.1137/17M1145689

1. Introduction.

1.1. Sparse representations and dictionary learning.

Sparse signal representation aims to represent a given signal by a linear combination of only a few elements of a fixed set of signal components [36]. For example, we can approximate an N-dimensional signal $\mathbf{s} \in \mathbb{R}^N$ as

$$\mathbf{s} \approx D\mathbf{x} = \mathbf{d}_1 x_1 + \cdots + \mathbf{d}_M x_M \;, \tag{1}$$

where $D = [\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_M] \in \mathbb{R}^{N\times M}$ is the *dictionary* with M *atoms* and $\mathbf{x} = [x_1, x_2, \dots, x_M]^T \in \mathbb{R}^M$ is the sparse representation. The problem of computing the sparse representation $\mathbf{x}$ given $\mathbf{s}$ and D is referred to as *sparse coding*. Among a variety of formulations of this problem, we focus on basis pursuit denoising (BPDN) [9]:

$$\min_{\mathbf{x}} \,(1/2)\, \|D\mathbf{x} - \mathbf{s}\|_2^2 + \lambda \|\mathbf{x}\|_1 \;. \tag{2}$$

*Received by the editors September 1, 2017; accepted for publication (in revised form) February 12, 2018; published electronically June 12, 2018.
http://www.siam.org/journals/siims/11-2/M114568.html

Funding: The work of the authors was supported by the U.S. Department of Energy through the LANL/LDRD Program. The work of the first and fourth authors was supported in part by NSF grant DMS-1720237 and ONR grant N000141712162.

[†]Department of Mathematics, UCLA, Los Angeles, CA 90095 (danny19921123@gmail.com, wotaoyin@math.ucla.edu).
[‡]CCS Division, Los Alamos National Laboratory, Los Alamos, NM 87545 (cgarciac@lanl.gov).
[§]Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545 (brendt@lanl.gov).

Sparse representations have been used in a wide variety of applications, including denoising [16, 36], superresolution [69, 75], classification [67], and face recognition [66]. A key issue when solving sparse coding problems as in (2) is how to choose the dictionary D. Early work on sparse representations used a fixed basis [45], such as wavelets [39] or discrete cosine transform [26], but learned dictionaries can provide better performance [2, 16].

Dictionary learning aims to learn a good dictionary D for a given distribution of signals. If $\mathbf{s}$ is a random variable, the dictionary learning problem can be formulated as

$$\min_{D\in C} \mathbb{E}_{\mathbf{s}}\left\{ \min_{\mathbf{x}} \frac{1}{2}\left\|D\mathbf{x}-\mathbf{s}\right\|_2^2 + \lambda\left\|\mathbf{x}\right\|_1 \right\}, \tag{3}$$

where $C = \{D \mid \|\mathbf{d}_m\|_2^2 \le 1, \forall m\}$ is the constraint set, which is necessary to resolve the scaling ambiguity between D and $\mathbf{x}$.

Batch dictionary learning methods (e.g., [18, 17, 2, 68]) sample a batch of training signals $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_K\}$ before training and minimize an objective function such as

$$\min_{D\in C,\mathbf{x}} \sum_{k=1}^{K}\left\{ \frac{1}{2}\left\|D\mathbf{x}_k-\mathbf{s}_k\right\|_2^2 + \lambda\left\|\mathbf{x}_k\right\|_1 \right\}. \tag{4}$$

These methods require simultaneous access to all the training samples during training.

In contrast, *online dictionary learning methods* process training samples in a streaming fashion. Specifically, let $\mathbf{s}^{(t)}$ be the chosen sample at the t^{th}, training step. The framework of online dictionary learning is

$$\begin{aligned} \mathbf{x}^{(t)} &= \mathrm{SC}\left(D^{(t-1)}; \mathbf{s}^{(t)}\right), \\ D^{(t)} &= D\text{-update}\left(\{D^{(\tau)}\}_{\tau=0}^{t-1}, \{\mathbf{x}^{(\tau)}\}_{\tau=1}^{t}, \{\mathbf{s}^{(\tau)}\}_{\tau=1}^{t}\right), \end{aligned} \tag{5}$$

where SC denotes sparse coding, for instance, (2), and D-update computes a new dictionary $D^{(t)}$ given the past information $\{D^{(\tau)}\}_{\tau=0}^{t-1}, \{\mathbf{x}^{(\tau)}\}_{\tau=1}^{t}, \{\mathbf{s}^{(\tau)}\}_{\tau=1}^{t}$. While each outer iteration of a batch dictionary learning algorithm involves computing the coefficient maps $\mathbf{x}_k$ for all training samples, online learning methods compute the coefficient map $\mathbf{x}^{(t)}$ for only one, or a small number, of the training sample $\mathbf{s}^{(t)}$ at each iteration, the other coefficient maps $\{\mathbf{x}^{(\tau)}\}_{\tau=1}^{t-1}$ used in the D-update having been computed in previous iterations. Thus, these algorithms can be implemented for large sets of training data or dynamically generated data. Online D-update methods and the corresponding online dictionary learning algorithms can be divided into two classes:

Class I*: First-order algorithms* [55, 35, 1] are inspired by stochastic gradient descent (SGD), which only uses first-order information, the gradient of the loss function, to update the dictionary D.

Class II*: Second-order algorithms.* These algorithms are inspired by recursive least squares (RLS) [47, 15], iterative IRLS [34, 56], kernel RLS [21], second-order stochastic approximation (SA) [37, 51, 71, 48, 74, 30], etc. They use previous information $\{D^{(\tau)}\}_{\tau=0}^{t-1}, \{\mathbf{x}^{(\tau)}\}_{\tau=1}^{t}, \{\mathbf{s}^{(\tau)}\}_{\tau=1}^{t}$ to construct a surrogate function $\mathcal{F}^{(t)}(D)$ to estimate the true loss function of D and then update D by minimizing this surrogate function. These surrogate functions involve both first-order and second-order information, i.e., the gradient and Hessian of the loss function, respectively.

The most significant difference between the two classes is that Class I algorithms only need access to information from the current step, t, i.e., $D^{(t-1)}, \mathbf{x}^{(t)}, \mathbf{s}^{(t)}$, while Class II algorithms use the entire history up to step t, i.e., $\{D^{(\tau)}\}_{\tau=0}^{t-1}$, $\{\mathbf{x}^{(\tau)}\}_{\tau=1}^{t}$, $\{\mathbf{s}^{(\tau)}\}_{\tau=1}^{t}$. However, as we discuss in section 4 below, it is possible to store this information in aggregate form so that the memory requirements do not scale with t.

1.2. Convolutional form. *Convolutional sparse coding (CSC)* [31, 70] [62, sec. II], a highly structured sparse representation model, has recently attracted increasing attention for a variety of imaging inverse problems [24, 33, 72, 43, 61, 73]. CSC aims to represent a given signal $\mathbf{s} \in \mathbb{R}^N$ as a sum of convolutions,

$$\mathbf{s} \approx \mathbf{d}_1 * \mathbf{x}_1 + \cdots + \mathbf{d}_M * \mathbf{x}_M \,, \tag{6}$$

where dictionary atoms $\{\mathbf{d}_m\}_{m=1}^M$ are linear filters and the representation $\{\mathbf{x}_m\}_{m=1}^M$ is a set of *coefficient maps*, each map $\mathbf{x}_m$ having the same size N as the signal $\mathbf{s}$. Since we implement the convolutions in the frequency domain for computational efficiency, it is convenient to adopt *circular boundary conditions* for the convolution operation.

Given $\{\mathbf{d}_m\}$ and $\mathbf{s}$, the maps $\{\mathbf{x}_m\}$ can be obtained by solving the convolutional basis pursuit denoising (CBPDN) ℓ_1-minimization problem

$$\min_{\{\mathbf{x}_m\}} \frac{1}{2} \left\| \sum_{m=1}^M \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \right\|_2^2 + \lambda \sum_{m=1}^M \|\mathbf{x}_m\|_1 \,. \tag{7}$$

The corresponding dictionary learning problem is called *convolutional dictionary learning (CDL)*. Specifically, given a set of K training signals $\{\mathbf{s}_k\}_{k=1}^K$, CDL is implemented via minimization of the function

$$\begin{gathered}\min_{\{\mathbf{d}_m\},\{\mathbf{x}_{k,m}\}} \frac{1}{2} \sum_{k=1}^K \left\| \sum_{m=1}^M \mathbf{d}_m * \mathbf{x}_{k,m} - \mathbf{s}_k \right\|_2^2 + \lambda \sum_{k=1}^K \sum_{m=1}^M \|\mathbf{x}_{k,m}\|_1 \\ \text{subject to } \|\mathbf{d}_m\|_2 \le 1, \ \forall m \in \{1, \dots, M\} \,,\end{gathered} \tag{8}$$

where the coefficient maps $\mathbf{x}_{k,m}$, $k \in \{1, \dots, K\}$, $m \in \{1, \dots, M\}$, represent $\mathbf{s}_k$ and the norm constraint avoids the scaling ambiguity between $\mathbf{d}_m$ and $\mathbf{x}_{k,m}$.

The *masked CDL problem* [25, 59], which is able to learn the convolutional dictionary from training signals with missing samples, is a variant of (8):

$$\begin{gathered}\min_{\{\mathbf{d}_m\},\{\mathbf{x}_{k,m}\}} \frac{1}{2} \sum_{k=1}^K \left\| \sum_{m=1}^M W \odot (\mathbf{d}_m * \mathbf{x}_{k,m} - \mathbf{s}_k) \right\|_2^2 + \lambda \sum_{k=1}^K \sum_{m=1}^M \|\mathbf{x}_{k,m}\|_1 \\ \text{subject to } \|\mathbf{d}_m\|_2 \le 1, \ \forall m \in \{1, \dots, M\} \,,\end{gathered} \tag{9}$$

where the *masking matrix* W is usually a $\{0, 1\}$-valued matrix that masks unknown or unreliable pixels and operator $\odot$ denotes pointwise multiplication.

Complexity of batch CDL. Most current CDL algorithms [8, 25, 24, 62, 54, 59, 22, 10] are batch learning methods that alternatively minimize over $\{\mathbf{x}_{k,m}\}$ and $\{\mathbf{d}_m\}$, dealing with the entire training set at each iteration. When K is large, the $\mathbf{d}_m$ update subproblem is computationally expensive; e.g., the single-step complexity and memory usage are both $\mathcal{O}(KMN\log(N))$ for one of the current state-of-the-art methods [54, 22]. For example, for a medium-sized problem with $K = 40, N = 256 \times 256, M = 64$, we have $KMN\log(N) \approx 10^9$, which is computationally very expensive.

1.3. Contribution of this article. The goal of the present work is to develop online convolutional dictionary learning methods for training data sets that are much larger than those that are presently feasible. We develop online methods for CDL in two directions: first-order method and second-order method. The contribution of this article includes the following:

1. An efficient first-order online CDL method (Algorithm 1) that provides a new framework with lower learning time and memory requirements than our previous state-of-the-art online method [32].
2. An efficient second-order online CDL method (Algorithm 2) that improves the algorithm proposed in [32] and proves its convergence.
3. An online CDL method for masked images, which is, to the best of our knowledge, the first online algorithm able to learn dictionaries from a partially masked training set.
4. An analysis of the forgetting factor[1] used in Algorithm 2.
5. An analysis of the stopping condition in the D-update.
6. An analysis of the effects of circular boundary conditions on dictionary learning.

Relationship with other works. Recently, two other works on online CDL [13, 57] have appeared. Both of them study second-order SA methods. They use the same framework as [37] but different methods to update D: [13] uses projected coordinate descent, and [57] uses the iterated Sherman–Morrison update [62]. Our previous work [32] uses the frequency-domain fast iterative shrinkage-thresholding algorithm (FISTA) to update D, with a *forgetting factor* technique inspired by [47, 38] to correct the surrogate function, and uses "region sampling" to reduce the memory cost. In this paper, the second-order SA algorithm, Algorithm 2, improves the algorithm in [32] by introducing two additional techniques: an improved stopping condition for FISTA and image splitting. The former technique greatly reduces the number of inner-loop iterations of the D-update. The latter, compared with "region sampling," fully utilizes all the information in the training set. With these techniques, Algorithm 2 converges faster than the algorithm in [32].

The method in [13] is designed for a different problem than (8) and is therefore not directly comparable with our methods. The other recent paper on online CDL [57], which appeared while we were completing this work, proposed an algorithm that uses the same framework as our previous work [32] and is therefore expected to offer similar performance to our initial method.

2. Preliminaries. Here we introduce our notation. The signal is denoted by $\mathbf{s} \in \mathbb{R}^N$ and the dictionaries by $\mathbf{d} = (\mathbf{d}_1\ \mathbf{d}_2\ \ldots\ \mathbf{d}_M)^T \in \mathbb{R}^{ML}$, where the dictionary kernels (or filters) are $\mathbf{d}_m \in \mathbb{R}^L$. The coefficient maps are denoted by $\mathbf{x} = (\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_M)^T \in \mathbb{R}^{MN}$, where

[1]This technique is used in previous works [47, 38, 51, 48] but not theoretically analyzed.

$\mathbf{x}_m \in \mathbb{R}^N$ is the coefficient map corresponding to $\mathbf{d}_m$. In addition to the *vector form*, $\mathbf{x}$, of the coefficient maps, we define an *operator form* X. First, we define a linear operator X_m on $\mathbf{d}_m$ such that $X_m\mathbf{d}_m = \mathbf{d}_m * \mathbf{x}_m$ and let $X \triangleq (X_1\ X_2 \cdots X_M)$. Then we have

$$X\mathbf{d} \triangleq \sum_{m=1}^{M} X_m\mathbf{d}_m = \sum_{m=1}^{M} \mathbf{d}_m * \mathbf{x}_m \approx \mathbf{s}\ . \tag{10}$$

Hence, $X : \mathbb{R}^{ML} \to \mathbb{R}^N$, a linear operator defined from the dictionary space to the signal space, is the operator form of $\mathbf{x}$.

2.1. Problem settings. Now we reformulate (8) into a more general form. (The masked problem (9) will be discussed in section 5.) Usually, the signal is sampled from a large training set, but we consider the training signal $\mathbf{s}$ as a random variable following the distribution $\mathbf{s} \sim P_S(\mathbf{s})$. Our goal is to optimize the dictionary $\mathbf{d}$. Given $\mathbf{s}$, the loss function l to evaluate $\mathbf{d}, \mathbf{x}$ is defined as

$$l(\mathbf{d}, \mathbf{x}; \mathbf{s}) = (1/2)\left\|X\mathbf{d} - \mathbf{s}\right\|_2^2 + \lambda\left\|\mathbf{x}\right\|_1\ . \tag{11}$$

Given $\mathbf{s}$, the loss function f to evaluate $\mathbf{d}$ and the corresponding minimizer are, respectively,

$$f(\mathbf{d}; \mathbf{s}) \triangleq \min_{\mathbf{x}} l(\mathbf{d}, \mathbf{x}; \mathbf{s}) \quad \text{and} \quad \mathbf{x}^*(\mathbf{d}; \mathbf{s}) \triangleq \arg\min_{\mathbf{x}} l(\mathbf{d}, \mathbf{x}; \mathbf{s})\ . \tag{12}$$

A general CDL problem can be formulated as

$$\min_{\mathbf{d}\in \mathrm{C}} \mathbb{E}_{\mathbf{s}}[f(\mathbf{d}; \mathbf{s})]\ , \tag{13}$$

where C is the constraint set of $\mathrm{C} = \{\mathbf{d} \mid \left\|\mathbf{d}_m\right\|^2 \leq 1, \forall m\}$.

2.2. Two online frameworks. Now we consider the CDL problem (13) when the training signals $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(t)}, \ldots$ arrive in a streaming fashion. Inspired by online methods for standard dictionary learning problems, we propose two online frameworks for the CDL problem (13). One is a first-order method based on projected SGD [55, 35, 1]:

$$\mathbf{d}^{(t)} = \mathrm{Proj}_{\mathrm{C}}\left(\mathbf{d}^{(t-1)} - \eta^{(t)}\nabla f\left(\mathbf{d}^{(t-1)}; \mathbf{s}^{(t)}\right)\right)\ . \tag{14}$$

The other is a second-order method, which is inspired by least squares estimator for dictionary learning [37, 47, 51, 71, 48, 74, 30]. A naive least squares estimator can be written as

$$\mathbf{d}^{(t)} = \arg\min_{\mathbf{d}\in\mathrm{C}}\left\{\underbrace{\min_{\mathbf{x}} \ell(\mathbf{d}, \mathbf{x}, \mathbf{s}^{(1)}) + \cdots + \min_{\mathbf{x}} \ell(\mathbf{d}, \mathbf{x}, \mathbf{s}^{(t)})}_{\text{Objective function on training samples } F^{(t)}(\mathbf{d})}\right\}\ .$$

This is not practical because the inner minimizer of $\mathbf{x}$ depends on $\mathbf{d}$, which is unknown. To solve this problem, we can fix $\mathbf{d}$ when we minimize over $\mathbf{x}$, i.e.,

$$\mathbf{x}^{(t)} = \arg\min_{\mathbf{x}} \ell(\mathbf{d}^{(t-1)}, \mathbf{x}; \mathbf{s}^{(t)}) \tag{15a}$$

$$\mathbf{d}^{(t)} = \arg\min_{\mathbf{d}\in \mathrm{C}} \left\{ \underbrace{\ell(\mathbf{d}, \mathbf{x}^{(1)}, \mathbf{s}^{(1)}) + \cdots + \ell(\mathbf{d}, \mathbf{x}^{(t)}, \mathbf{s}^{(t)})}_{\text{Surrogate function } \mathcal{F}^{(t)}(\mathbf{d})} \right\} . \tag{15b}$$

Direct application of these methods to the CDL problem is very computationally expensive, but we propose a number of techniques to reduce the time and memory usage. The details are discussed in sections 3 and 4, respectively.

2.3. Techniques to calculate operator X.

Before introducing our algorithms for (13), we consider a basic problem and two computational techniques that are used in this section as well as in sections 3 and 4.

With $\mathbf{s}$ and $\mathbf{x}$ fixed, the basic problem is

$$\min_{\mathbf{d}\in\mathbb{R}^{ML}} l(\mathbf{d}, \mathbf{x}; \mathbf{s}) + \iota_{\mathrm{C}}(\mathbf{d}) \ , \tag{16}$$

where $\iota_{\mathrm{C}}(\cdot)$ is the indicator function[2] of set C. To solve this problem, we can apply projected gradient descent (GD) [5]

$$\mathbf{d}^{(t)} = \mathrm{Proj}_{\mathrm{C}} \left(\mathbf{d}^{(t-1)} - \eta^{(t)} X^T \left(X\mathbf{d}^{(t-1)} - \mathbf{s} \right) \right) \ , \tag{17}$$

where (t) is the iteration index and $X^T (X\mathbf{d} - \mathbf{s})$ is the gradient of l with respect to $\mathbf{d}$. Since X is a linear operator from $\mathbb{R}^{ML}$ to $\mathbb{R}^N$, the cost of directly computing (17) is $\mathcal{O}(NML)$. However, we can exploit the sparsity or the structure of operator X to yield a more efficient computation that greatly reduces the time complexity.

2.3.1. Computing with sparsity property.

The first option is to utilize the sparsity of X. Specifically, X is saved as a triple array (i, j, v), which records the indices (i, j) and values v of the nonzero elements of X, so that only the nonzero entries in X contribute to the computational time. This triple array is commonly referred as a *coordinate list* and is a standard way of representing a sparse matrix.

Let us compute the nonzero entries of operator X. The operator form X_m of the N-dimensional vectors $\mathbf{x}_m = ((x_m)_1, \ldots, (x_m)_N)^T$ can be written as

$$X_m = \begin{bmatrix} (x_m)_1 & (x_m)_N & (x_m)_{N-1} & \cdots & (x_m)_{N-L+2} \\ (x_m)_2 & (x_m)_1 & (x_m)_N & \cdots & (x_m)_{N-L+3} \\ (x_m)_3 & (x_m)_2 & (x_m)_1 & \cdots & (x_m)_{N-L+4} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (x_m)_N & (x_m)_{N-1} & (x_m)_{N-2} & \cdots & (x_m)_{N-L+1} \end{bmatrix} ,$$

[2]The indicator function is defined as: $\iota_{\mathrm{C}}(\mathbf{d}) = \begin{cases} 0, & \text{if } \mathbf{d}\in\mathrm{C} \\ +\infty, & \text{otherwise} \end{cases}$.

where each column is a circular shift of $\mathbf{x}_m$ and L is the dimension of each dictionary kernel. Thus, the densities of $\mathbf{x}_m$ and X_m are the same. Assuming the density of vector $\mathbf{x}$ is ρ, the number of nonzero entries of operator X is $NML\rho$, giving a single-step complexity of $\mathcal{O}(NML\rho)$ for computing (17).

2.3.2. Computing in the frequency domain.

Another option is to utilize the structure of X. It is well known that convolving two signals of the same size corresponds to the pointwise multiplication of their frequency representations. Our method below takes advantage of this property. First, we zero-pad each $\mathbf{d}_m$ from $\mathbb{R}^L$ to $\mathbb{R}^N$ to match the size of $\mathbf{s}$. Then the basic problem can be written as

$$\min_{\mathbf{d}\in\mathbb{R}^{MN}} l(\mathbf{d},\mathbf{x};\mathbf{s}) + \iota_{\mathrm{C_{PN}}}(\mathbf{d}) \,, \tag{18}$$

where the set $\mathrm{C_{PN}}$ is defined as

$$\mathrm{C_{PN}} \triangleq \{\mathbf{d}_m \in \mathbb{R}^N : (I-P)\mathbf{d}_m = 0, \|\mathbf{d}_m\|^2 \le 1\} \,. \tag{19}$$

Operator P preserves the desired support of $\mathbf{d}_m$ and masks the remaining part to zeros. Projected GD (17) has an equivalent form:

$$\mathbf{d}^{(t)} = \mathrm{Proj}_{\mathrm{C_{PN}}}\left(\mathbf{d}^{(t-1)} - \eta^{(t)}\frac{\partial l}{\partial \mathbf{d}}(\mathbf{d}^{(t-1)},\mathbf{x};\mathbf{s})\right) \,. \tag{20}$$

Then, using the Plancherel formula, we can write the loss function[3] l as

$$l(\mathbf{d},\mathbf{x};\mathbf{s}) = \underbrace{\left\|\sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\right\|_2^2}_{\|X\mathbf{d}-\mathbf{s}\|^2} = \underbrace{\left\|\sum_m \hat{\mathbf{d}}_m \odot \hat{\mathbf{x}}_m - \hat{\mathbf{s}}\right\|_2^2}_{\|\hat{X}\hat{\mathbf{d}}-\hat{\mathbf{s}}\|^2} \,, \tag{21}$$

where $\hat{\cdot}$ denotes the corresponding quantity in the frequency domain and $\odot$ means pointwise multiplication. Therefore, we have $\hat{\mathbf{d}} \in \mathbb{C}^{MN}$, and $\hat{X} = (\hat{X}_1\ \hat{X}_2 \cdots \hat{X}_M)$ is a linear operator. Define the loss function in the frequency domain

$$\hat{l}(\hat{\mathbf{d}},\hat{\mathbf{x}};\hat{\mathbf{s}}) = (1/2)\left\|\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}\right\|^2 \,, \tag{22}$$

which is a real-valued function defined in the complex domain. The Cauchy–Riemann condition [3] implies that (22) is not differentiable unless it is constant. However, the *conjugate cogradient*[4] [49]

$$\frac{\partial \hat{l}}{\partial \hat{\mathbf{d}}}(\hat{\mathbf{d}},\hat{\mathbf{x}};\hat{\mathbf{s}}) \triangleq \hat{X}^H(\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}) \tag{23}$$

exists and can be used for minimizing (22) by gradient descent.

[3]We ignore the term $\lambda\|\mathbf{x}\|_1$ in l here because $\mathbf{x}$ is fixed in this problem.

[4]The conjugate cogradient of function $f(x) : \mathbb{C}^n \to \mathbb{R}$ is defined as $\frac{\partial f}{\partial \Re(x)} + i\frac{\partial f}{\partial \Im(x)}$, where $\Re(x)$, $I\Im(x)$ are the real part and imaginary part of x. The derivation of (22) is given in Appendix A.

Since each item $\hat{X}_m$ in $\hat{X}$ is diagonal, the gradient is easy to compute, with a complexity of $\mathcal{O}(NM)$ instead of $\mathcal{O}(NML)$. Based on (23), we have the following modified gradient descent:

$$\mathbf{d}^{(t)} = \mathrm{Proj}_{\mathrm{C}_{\mathrm{PN}}}\left(\mathrm{IFFT}\left(\hat{\mathbf{d}}^{(t-1)} - \eta^{(t)}\frac{\partial \hat{l}}{\partial \hat{\mathbf{d}}}\left(\hat{\mathbf{d}}^{(t-1)}, \hat{\mathbf{x}}; \hat{\mathbf{s}}\right)\right)\right) . \tag{24}$$

To compute (24), we transform $\mathbf{d}^{(t)}$ into its frequency-domain counterpart $\hat{\mathbf{d}}^{(t)}$, perform gradient descent in the frequency domain, return to the spatial domain, and project the result onto the set C_{PN}.

In our modified method (24), the iterate $\mathbf{d}^{(t)}$ is transformed between the frequency and spatial domains because the gradient is cheaper to compute in the frequency domain, but projection is cheaper to compute in the spatial domain.

Equivalence of (20) and (24). We can prove

$$\hat{X}^H(\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}) = \mathrm{FFT}\left(X^T(X\mathbf{d} - \mathbf{s})\right) \quad \forall \mathbf{x}, \mathbf{d}, \mathbf{s} , \tag{25}$$

which means that the conjugate cogradient of $\hat{l}$ is equivalent to the gradient of l. Thus, modified GD (24) coincides with standard GD (20) using the conjugate cogradient.

A proof of (25) given in Appendix B. A similar result is also given in [44] under the name "conjugate symmetry."

3. First-order method: Algorithm 1.

Recall the projected SGD step (14)

$$\mathbf{d}^{(t)} = \mathrm{Proj}_{\mathrm{C}}\left(\mathbf{d}^{(t-1)} - \eta^{(t)}\nabla f(\mathbf{d}^{(t-1)}; \mathbf{s}^{(t)})\right) ,$$

where parameter $\eta^{(t)}$ is the *step size*[5]. Given the definition of f in (12), $\nabla f(\mathbf{d}^{(t-1)}; \mathbf{s}^{(t)})$ is the partial derivative with respect to $\mathbf{d}$ at the optimal $\mathbf{x}$ [38, 11], i.e., $\nabla f(\mathbf{d}; \mathbf{s}) = \frac{\partial l}{\partial \mathbf{d}}(\mathbf{d}, \mathbf{x}^*(\mathbf{d}, \mathbf{s}); \mathbf{s})$, where $\mathbf{x}^*$ is defined by (12).

Thus, to compute the gradient $\nabla f(\mathbf{d}^{(t-1)}; \mathbf{s}^{(t)})$, we should first compute the coefficient maps $\mathbf{x}^{(t)}$ of the t^{th} training signal $\mathbf{s}^{(t)}$ with dictionary $\mathbf{d}^{(t-1)}$, which is given by (15a). Then we can compute the gradient as

$$\nabla f\left(\mathbf{d}^{(t-1)}; \mathbf{s}^{(t)}\right) = \frac{\partial l}{\partial \mathbf{d}}\left(\mathbf{d}^{(t-1)}, \mathbf{x}^{(t)}; \mathbf{s}^{(t)}\right) = \left(X^{(t)}\right)^T\left(X^{(t)}\mathbf{d}^{(t-1)} - \mathbf{s}^{(t)}\right) .$$

Based on the discussion in section 2.3, we can perform gradient descent either in the spatial domain or in the frequency domain. In the frequency domain, the conjugate cogradient of $\nabla\hat{f}$ is

$$\nabla\hat{f}(\hat{\mathbf{d}}^{(t-1)}; \hat{\mathbf{s}}^{(t)}) = \frac{\partial \hat{l}}{\partial \hat{\mathbf{d}}}\left(\hat{\mathbf{d}}^{(t-1)}, \hat{\mathbf{x}}^{(t)}; \hat{\mathbf{s}}^{(t)}\right) = \left(\hat{X}^{(t)}\right)^H\left(\hat{X}^{(t)}\hat{\mathbf{d}}^{(t-1)} - \hat{\mathbf{s}}^{(t)}\right) .$$

The full algorithm is summarized in Algorithm 1.

[5]Some authors refer to it as the *learning rate*.

Algorithm 1. Online Convolutional Dictionary Learning (Modified SGD).

```
Initialize: Initialize $\mathbf{d}^{(0)}$ with a random dictionary.
1 for $t = 1, \dots, T$ do
2     Sample a signal $\mathbf{s}^{(t)}$.
3     Solve convolutional sparse coding problem (15a) to obtain $\mathbf{x}^{(t)}$.
4     if Option I then
5         Update dictionary in the spatial domain with sparse matrix $X^{(t)}$:
              $$\mathbf{d}^{(t)} = \mathrm{Proj}_{\mathrm{C}}\left(\mathbf{d}^{(t-1)} - \eta^{(t)} \left(X^{(t)}\right)^T \left(X^{(t)}\mathbf{d}^{(t-1)} - \mathbf{s}^{(t)}\right)\right)$$
6     else if Option II then
7         Update dictionary in the frequency domain:
              $$\hat{\mathbf{x}}^{(t)} = \mathrm{FFT}(\mathbf{x}^{(t)})$$
              $$\mathbf{d}^{(t)} = \mathrm{Proj}_{\mathrm{C}_{\mathrm{PN}}}\left(\mathrm{IFFT}\left(\hat{\mathbf{d}}^{(t-1)} - \eta^{(t)} \left(\hat{X}^{(t)}\right)^H \left(\hat{X}^{(t)}\hat{\mathbf{d}}^{(t-1)} - \hat{\mathbf{s}}^{(t)}\right)\right)\right)$$
8     end
9 end
Output: $\mathbf{d}^{(T)}$
```

Table 1

Single-step complexity and memory usage of Algorithm 1. N: signal dimension; M: number of dictionary kernels; L: size of each kernel; ρ: average density of the coefficient maps.

Scheme	Single-step complexity	Memory usage
Spatial (dense matrix)	$T_{\mathrm{CBPDN}} + \mathcal{O}(NML)$	$\mathcal{O}(NML)$
Spatial (sparse matrix)	$T_{\mathrm{CBPDN}} + \mathcal{O}(NML\rho)$	$\mathcal{O}(NML\rho)$
Frequency update	$T_{\mathrm{CBPDN}} + \mathcal{O}(NM\log(N)) + \mathcal{O}(NM)$	$\mathcal{O}(MN)$

Complexity analysis of Algorithm 1. We list the single-step complexity and the memory usage of different options in Table 1. Both the frequency-domain update and the sparse matrix technique reduce single-step complexities. The comparison between these two computational techniques depends on the sparsity of $X^{(t)}$ and the dictionary kernel size L. In section 6.1, we will numerically compare these methods.

Convergence of Algorithm 1. Algorithm 1, by (25), is equivalent to the standard projected SGD. Thus, by properly choosing step sizes $\eta^{(t)}$, Algorithm 1 converges to a stationary point [23]. A diminishing step size rule $\eta^{(t)} = a/(b+t)$ is used in other dictionary learning works [1, 37]. The convergence performances with different step sizes are numerically tested in section 6.1.

4. Second-order method: Algorithm 2.

In this section, we first introduce some details of directly applying the second-order stochastic approximation method (15) to CDL problems, then we discuss some issues and our resolutions.

Aggregating the true loss function $f(\mathbf{d};\mathbf{s}^{(t)})$ on the tth sample $\mathbf{s}^{(t)}$, the objective function on the first t training samples is

$$F^{(t)}(\mathbf{d}) = \frac{1}{t}\left(\sum_{\tau=1}^{t} f\left(\mathbf{d};\mathbf{s}^{(\tau)}\right)\right) \approx F(\mathbf{d}) = \mathbb{E}_{\mathbf{s}}[f(\mathbf{d};\mathbf{s})]\ . \tag{26}$$

The central limit theorem tells us that $F^{(t)} \to F$ as $t \to \infty$. However, as discussed in section 2.2, $F^{(t)}$ is not computationally tractable. To update $\mathbf{d}$ efficiently, we introduce the *surrogate function* $\mathcal{F}^{(t)}$ of $F^{(t)}$. Given $\mathbf{s}^{(t)}$, $\mathbf{x}^{(t)}$ is computed by CBPDN (12) using the latest dictionary $\mathbf{d}^{(t-1)}$, then a surrogate of $f(\mathbf{d};\mathbf{s}^{(t)})$ is given as

$$\mathbf{x}^{(t)} = \arg\min_{\mathbf{x}} \ell(\mathbf{d}^{(t-1)},\mathbf{x};\mathbf{s}^{(t)}), \qquad f^{(t)}(\mathbf{d}) \triangleq l\left(\mathbf{d},\mathbf{x}^{(t)};\mathbf{s}^{(t)}\right). \tag{27}$$

The surrogate function of $F^{(t)}$ is defined as

$$\mathcal{F}^{(t)}(\mathbf{d}) = \frac{1}{t}\left(f^{(1)}(\mathbf{d}) + \cdots + f^{(t)}(\mathbf{d})\right)\ . \tag{28}$$

Then, at the tth step, the dictionary is updated as

$$\mathbf{d}^{(t)} = \arg\min_{\mathbf{d}\in\mathbb{R}^{ML}} \mathcal{F}^{(t)}(\mathbf{d}) + \iota_{\mathrm{C}}(\mathbf{d})\ . \tag{29}$$

Solving subproblem (29). To solve (29), we apply FISTA [4], which needs to compute a gradient at each step. The gradient for the surrogate function can be computed as

$$\nabla\mathcal{F}^{(t)}(\mathbf{d}) = \frac{1}{t}\left(\sum_{\tau=1}^{t}\left(X^{(\tau)}\right)^T X^{(\tau)}\right)\mathbf{d} - \frac{1}{t}\left(\sum_{\tau=1}^{t}\left(X^{(\tau)}\right)^T \mathbf{s}^{(\tau)}\right)\ .$$

We cannot follow this formula directly since the cost increases linearly in t. Instead, we perform the recursive updates

$$A^{(t)} = A^{(t-1)} + (X^{(t)})^T X^{(t)}\ , \quad \mathbf{b}^{(t)} = \mathbf{b}^{(t-1)} + (X^{(t)})^T \mathbf{s}^{(t)}\ , \tag{30}$$

where $(X^{(t)})^T X^{(t)}$ is the Hessian matrix of $f^{(t)}$. These updates, which have a constant cost per step, yield $\nabla\mathcal{F}^{(t)}(\mathbf{d}) = (A^{(t)}\mathbf{d} - \mathbf{b}^{(t)})/t$. The matrix $A^{(t)}/t$, the Hessian matrix of the surrogate function $\mathcal{F}^{(t)}$, accumulates the Hessian matrices of all the past loss functions. This is why we call this method the *second-order stochastic approximation* method.

Practical issues

- Inaccurate loss function: The surrogate function $\mathcal{F}^{(t)}$ involves old loss functions $f^{(1)}$, $f^{(2)},\ldots$, which contain old information $\mathbf{x}^{(1)},\mathbf{x}^{(2)},\ldots$. For example, $\mathbf{x}^{(1)}$ is computed using $\mathbf{d}^{(0)}$ (cf. (27)).
- Large single-step complexity and memory usage: Handling a whole image $\mathbf{s}^{(t)}$ at each time is still a large-scale problem.
- FISTA is slow at solving subproblem (29): FISTA takes many steps to reach a sufficient accuracy.

To address these points, four modifications are given in this section.[6]

[6]Improvements I and II have been addressed in our previous work [32]. In the present article, we include their theoretical analysis and introduce the new enhancement of the stopping criterion (Improvement III).

4.1. Improvement I: Forgetting factor. At time t, the dictionary is the result of an accumulation of past coefficient maps $\mathbf{x}_m^{(\tau)}$, $\tau < t$, which were computed with the then-available dictionaries. A way to balance accumulated past contributions and the information provided by the new training samples is to compute a weighted combination of these contributions [47, 38, 51, 48]. This combination gives more weight to more recent updates since those are the result of a more extensively trained dictionary. Specifically, we consider the following weighted (or modified) surrogate function:

$$\mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}) = \frac{1}{\Lambda^{(t)}} \sum_{\tau=1}^{t} (\tau/t)^p f^{(\tau)}(\mathbf{d}) \ , \quad \Lambda^{(t)} = \sum_{\tau=1}^{t} (\tau/t)^p \ . \tag{31}$$

This function can be written in recursive form as

$$\Lambda^{(t)} = \alpha^{(t)} \Lambda^{(t-1)} + 1 \ , \tag{32}$$

$$\Lambda^{(t)} \mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}) = \alpha^{(t)} \Lambda^{(t-1)} \mathcal{F}_{\text{mod}}^{(t-1)}(\mathbf{d}) + f^{(t)}(\mathbf{d}) \ . \tag{33}$$

Here $\alpha^{(t)} \in (0,1)$ is a *forgetting factor*, which has its own time evolution,

$$\alpha^{(t)} = (1 - 1/t)^p, \tag{34}$$

regulated by the *forgetting exponent* $p > 0$. As t increases, the factor $\alpha^{(t)}$ increases ($\alpha^{(t)} \to 1$ as $t \to \infty$), reflecting the increasing accuracy of the past information as the training progresses. The dictionary update (29) is modified correspondingly to

$$\mathbf{d}^{(t)} = \mathop{\arg\min}_{\mathbf{d} \in \mathbb{R}^{ML}} \mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}) + \iota_{\text{C}}(\mathbf{d}) \ . \tag{35}$$

This technique has been used in some previous dictionary learning works, as we mentioned before, but was not theoretically analyzed. In this paper, we prove in Propositions 1 and 2 that $F_{\text{mod}}^{(t)} \to F$ as $t \to \infty$, where $F_{\text{mod}}^{(t)}$ is a weighted approximation of F:

$$F_{\text{mod}}^{(t)}(\mathbf{d}) = \frac{1}{\Lambda^{(t)}} \left(\sum_{\tau=1}^{t} (\tau/t)^p f(\mathbf{d}; \mathbf{s}^{(\tau)}) \right) \ . \tag{36}$$

Moreover, in Theorem 1, $\mathcal{F}_{\text{mod}}^{(t)}$, the surrogate of $F_{\text{mod}}^{(t)}$, is also proved to be convergent on the current dictionary, i.e., $\mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}^{(t)}) - F_{\text{mod}}^{(t)}(\mathbf{d}^{(t)}) \to 0$.

Effect of the forgetting exponent p. A small p tends to lead to a stable algorithm since all the training signals are given nearly equal weights and $F_{\text{mod}}^{(t)}$ is a stochastic approximation of F with small variance. Propositions 1 and 2 give theoretical explanations of this phenomenon. However, a small p leads to an inaccurate surrogate loss function $\mathcal{F}_{\text{mod}}^{(t)}$ since it gives large weights to old information. In the extreme case, as $p \to 0$, the modified surrogate function (31) reduces to the standard one (28). Section 6.2.1 reports the related numerical results.

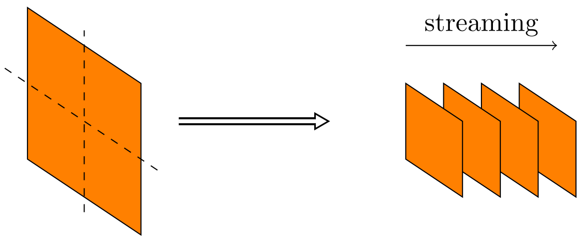


Figure 1. *An example of image splitting:* $N = 256 \times 256 \rightarrow \tilde{N} = 128 \times 128$.

4.2. Improvement II: Image splitting. Both the single-step complexity and memory usage are related to the signal dimension N. For a typical imaging problem, $N = 256 \times 256$ or greater, which is large. To reduce the complexities, we use small regions[7] instead of the whole signal. Specifically, as illustrated in Figure 1, we split a signal $\mathbf{s}^{(t)} \in N$ into small regions $\mathbf{s}^{(t)}_{\text{split},1}, \mathbf{s}^{(t)}_{\text{split},2}, \dots \in \tilde{N}$, with $\tilde{N} < N$, and treat them as if they were distinct signals. In this way, the training signal sequence becomes

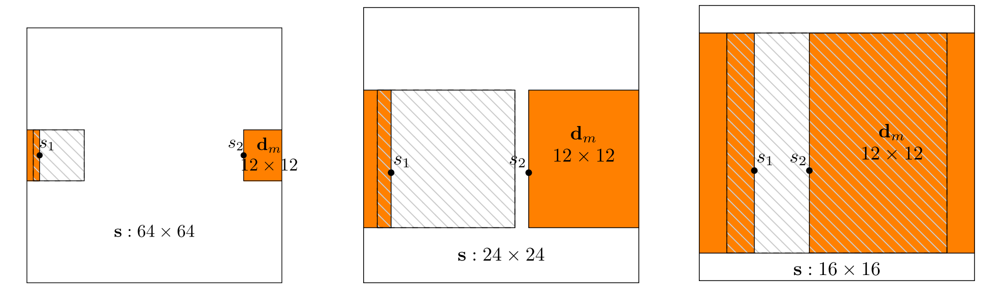
$$\{\mathbf{s}_{\text{split}}\} \triangleq \{\mathbf{s}^{(1)}_{\text{split},1}, \dots, \mathbf{s}^{(1)}_{\text{split},n}, \mathbf{s}^{(2)}_{\text{split},1}, \dots, \mathbf{s}^{(2)}_{\text{split},n}, \dots\} .$$

Boundary issues. The use of *circular boundary conditions* for signals that are not periodic has the potential to introduce boundary artifacts in the representation and therefore also in the learned dictionary [70]. When the size of the training images is much larger than the kernels, there is some evidence that the effect on the learned dictionary is negligible [8], but it is reasonable to expect that these effects will become more pronounced for smaller training images, such as the regions we obtain when using a small splitting size $\tilde{N}$. The possibility of severe artifacts when the image size approaches the kernel size is illustrated in Figure 2. In section 6.2.2, we study this effect and show that using a splitting size that is twice the kernel size in each dimension is sufficient to avoid artifacts, as expected from the argument illustrated in Figure 2.

4.3. Improvement III: Stopping FISTA early. Another issue in surrogate function method is the stopping condition of FISTA. A small fixed tolerance will result in too many inner-loop iterations for the initial steps. Another strategy, as used in SPAMS [37, 28], is a fixed number of inner-loop iterations, but it does not have any theoretical convergence guarantee.

In this article, we propose a "diminishing tolerance" scheme in which subproblem (35) is solved *inexactly*, but the online learning algorithm is still theoretically guaranteed to converge. The stopping accuracy is increasing as t increases. Specifically, the stopping tolerance is decreased as t increases. Moreover, with a warm start (using $\mathbf{d}^{(t-1)}$ as the initial solution

[7]In our previous work [32], we sample some small regions from the whole signals in the limited memory algorithm, which performs worse than the algorithm training with the whole signals. We claimed that the performance sacrifice is caused by the circular boundary condition. In fact, this is caused by the sampling. In that paper, we sample small regions with random center position and fixed size. If we sample small regions in this way, some parts of the image are not sampled, but some are sampled several times. Consequently, in the present paper, we propose the "image-splitting" technique in Algorithm 2, which avoids this issue. It only shows worse performance when the splitting size is smaller than a threshold, which is actually caused by the boundary condition.



(a) When the signal size 64×64 is much larger than the kernel size 12×12, pixels s_1, s_2 in the same filter are far from each other. Thus, they do not interact with each other.

(b) When the signal size 24×24 is twice the kernel size 12×12, s_1, s_2 still do not interact. It is the smallest signal size to avoid boundary artifacts.

(c) When the signal size 16×16 is less than twice the kernel size 12×12, s_1, s_2 interact with one another. This leads to artifacts in practice.

Figure 2. *An illustration of the boundary artifacts with two-dimensional square signals and dictionary kernels.*

for the t^{th} step), the number of inner-loop iterations stays moderate as t increases, which is validated by the results in Figure 8.

Stopping metric. We use the fixed point residual (FPR) [12]

$$\mathbf{R}^{(t)}(\mathbf{g}) \triangleq \left\| \mathbf{g} - \mathrm{Proj}_{\mathrm{C}}\left(\mathbf{g} - \eta\nabla\mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{g})\right)\right\| \tag{37}$$

for two reasons. One is its simplicity; if FISTA is used to solve (35), this metric can be computed directly as $\mathbf{R}^{(t)}(\mathbf{g}^j_{\mathrm{aux}}) = \|\mathbf{g}^{j+1} - \mathbf{g}^j_{\mathrm{aux}}\|$. The other is that a small FPR implies a small distance to the exact solution of the subproblem, as shown in Proposition 3 below.

Stopping condition. In this paper, we consider the following stopping condition:

$$\mathbf{R}^{(t)}(\mathbf{g}^j_{\mathrm{aux}}) \le \tau^{(t)} \triangleq \tau_0/(1+\alpha t)\ , \tag{38}$$

where the tolerance $\tau^{(t)}$ is large during the first several steps and reduces to zeros at the rate of $\mathcal{O}(1/t)$ as t increases. In the t^{th} step, once (38) is satisfied, we stop the D-update (FISTA) and continue to the next step. The effect of this stopping condition is theoretically analyzed in Propositions 3 and 4 and numerically demonstrated in section 6.2.3 below.

4.4. Improvement IV: Computational techniques in solving subproblem (35). Based on the discussion in section 2.3, we have two options to solve subproblem (35). One is to solve in the spatial domain utilizing sparsity. The gradient of $\mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{d})$ is

$$\nabla\mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{d}) = \frac{1}{\Lambda^{(t)}}\sum_{\tau=1}^{t}(\tau/t)^p\left((X^{(t)})^TX^{(t)}\mathbf{d} - (X^{(\tau)})^T\mathbf{s}^{(\tau)}\right) = \frac{1}{\Lambda^{(t)}}\left(A^{(t)}_{\mathrm{mod}}\mathbf{d} - \mathbf{b}^{(t)}_{\mathrm{mod}}\right),$$

where $A^{(t)}_{\mathrm{mod}}$ and $\mathbf{b}^{(t)}_{\mathrm{mod}}$ are calculated in a recursive form in line 5 of Algorithm 2. The other option is to update in the frequency domain. The conjugate cogradient of $\hat{\mathcal{F}}^{(t)}_{\mathrm{mod}}(\hat{\mathbf{d}})$ is

$$\nabla\hat{\mathcal{F}}^{(t)}_{\mathrm{mod}}(\hat{\mathbf{d}}) = \frac{1}{\Lambda^{(t)}}\sum_{\tau=1}^{t}(\tau/t)^p\left((\hat{X}^{(t)})^T\hat{X}^{(t)}\hat{\mathbf{d}} - (\hat{X}^{(\tau)})^T\hat{\mathbf{s}}^{(\tau)}\right) = \frac{1}{\Lambda^{(t)}}\left(\hat{A}^{(t)}_{\mathrm{mod}}\hat{\mathbf{d}} - \hat{\mathbf{b}}^{(t)}_{\mathrm{mod}}\right),$$

Algorithm 2. Online Convolutional Dictionary Learning (Surrogate Splitting).

```
   Initialize: Initialize $\mathbf{d}^{(0)}$, let $A^{(0)}_{\text{mod}} \leftarrow 0, \mathbf{b}^{(0)}_{\text{mod}} \leftarrow 0$ or $\hat{A}^{(0)}_{\text{mod}} \leftarrow 0, \hat{\mathbf{b}}^{(0)}_{\text{mod}} \leftarrow 0$.
1  for $t = 1, \ldots, T$ do
2      Sample a signal $\mathbf{s}^{(t)}$ from $\{\mathbf{s}_{\text{split}}\}$.
3      Solve convolutional sparse coding problem (15a) to obtain $\mathbf{x}^{(t)}$.
4      if Option I then
5          Update $A^{(t)}_{\text{mod}}, \mathbf{b}^{(t)}_{\text{mod}}$ in the spatial domain with sparse matrix $X^{(t)}$:
               $A^{(t)}_{\text{mod}} = \alpha^{(t)} A^{(t-1)}_{\text{mod}} + (X^{(t)})^T X^{(t)},\ \mathbf{b}^{(t)}_{\text{mod}} = \alpha^{(t)} \mathbf{b}^{(t-1)}_{\text{mod}} + (X^{(t)})^T \mathbf{s}^{(t)}$
6          Solve the following subproblem with FISTA (stopping condition (38)):
               $\mathbf{d}^{(t)} = \operatorname*{arg\,min}_{\mathbf{d} \in \mathbb{R}^{ML}} \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}) + \iota_{\text{C}}(\mathbf{d})$ .
7      else if Option II then
8          Update $\hat{A}^{(t)}_{\text{mod}}, \hat{\mathbf{b}}^{(t)}_{\text{mod}}$ in the frequency domain:
               $\hat{A}^{(t)}_{\text{mod}} = \alpha^{(t)} \hat{A}^{(t-1)}_{\text{mod}} + (\hat{X}^{(t)})^H \hat{X}^{(t)},\ \hat{\mathbf{b}}^{(t)}_{\text{mod}} = \alpha^{(t)} \hat{\mathbf{b}}^{(t-1)}_{\text{mod}} + (\hat{X}^{(t)})^H \hat{\mathbf{s}}^{(t)}$
9          Solve the following subproblem with frequency domain FISTA (stopping
            condition (38), see Appendix C):
               $\mathbf{d}^{(t)} = \operatorname*{arg\,min}_{\mathbf{d} \in \mathbb{R}^{MN}} \hat{\mathcal{F}}^{(t)}_{\text{mod}}(\hat{\mathbf{d}}) + \iota_{\text{C}_{\text{PN}}}(\mathbf{d})$ .
10     end
11 end
   Output: $\mathbf{d}^{(T)}$
```

where $\hat{A}^{(t)}_{\text{mod}}$ and $\hat{\mathbf{b}}^{(t)}_{\text{mod}}$ are calculated in a recursive form in line 7 of Algorithm 2. With the gradients, we can apply FISTA or frequency-domain FISTA on the problem (35), as in Algorithm 2.

Complexity analysis of Algorithm 2. If we solve (35) directly, the operator $X^{(t)}$ is a linear operator from $\mathbb{R}^{LM}$ to $\mathbb{R}^N$. Thus, the complexity of computing the Hessian matrix of $f^{(t)}$, $(X^{(t)})^T X^{(t)}$, is $\mathcal{O}(L^2M^2N)$, and the memory cost is $\mathcal{O}(L^2M^2)$. Otherwise, if we solve (35) utilizing the sparsity of X, the computational cost of computing $(X^{(t)})^T X^{(t)}$ can be reduced to $\mathcal{O}(L^2M^2N\rho)$, where ρ is the density of sparse matrix $X^{(t)}$, but the memory cost is still $\mathcal{O}(L^2M^2)$ because $(X^{(t)})^T X^{(t)}$ is not sparse, although $X^{(t)}$ is. In comparison, if we solve (35) in the frequency domain, the frequency-domain operator $\hat{X}^{(t)} = (\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_M)$ is a linear operator from $\mathbb{C}^{MN}$ to $\mathbb{C}^N$, which seems to lead to a larger complexity to compute the Hessian: $\mathcal{O}(M^2N^3)$ flops and $\mathcal{O}(M^2N^2)$ memory cost. However, since each component

Table 2

Single-step complexity and memory usage of Algorithm 2. N: signal dimension; M: number of dictionary kernels; L: size of each kernel; ρ: average density of the coefficient maps; J: average loops of FISTA in each step. This is numerically tested in Table 4.

Scheme	Single-step complexity	Memory usage
Spatial (dense matrix)	$T_{\text{CBPDN}} + \mathcal{O}(L^2M^2N) + J \times \mathcal{O}(L^2M^2)$	$\mathcal{O}(L^2M^2) + \mathcal{O}(LMN)$
Spatial (sparse matrix)	$T_{\text{CBPDN}} + \mathcal{O}(L^2M^2N\rho) + J \times \mathcal{O}(L^2M^2)$	$\mathcal{O}(L^2M^2) + \mathcal{O}(LMN\rho)$
Frequency update	$T_{\text{CBPDN}} + (\mathcal{O}(M^2N) + \mathcal{O}(MN\log(N)))$ $+ J \times (\mathcal{O}(M^2N) + \mathcal{O}(MN\log(N)))$	$\mathcal{O}(M^2N)$

$\hat{X}_m$ is diagonal, the frequency-domain product $(\hat{X}^{(t)})^H\hat{X}^{(t)}$ has only $\mathcal{O}(M^2N)$ nonzero values. Both the number of flops and the memory cost are $\mathcal{O}(M^2N)$. The complexities are listed in Table 2.

4.5. Convergence of Algorithm 2.

First, we start with some assumptions:[8]

Assumption 1. *All the signals are drawn from a distribution with a compact support.*

Assumption 2. *Each sparse coding step (12) has a unique solution.*

Assumption 3. *The surrogate functions are strongly convex.*

Assumption 1 can easily be guaranteed by normalizing each training signal. Assumption 2 is a common assumption in dictionary learning and other linear regression papers [37, 14]. Practically, it must be guaranteed by choosing a sufficiently large penalty parameter λ in (12) because a larger penalty parameter leads to a sparser $\mathbf{x}$. See Appendix D for details. Assumption 3 is a common assumption in RLS (see Definition 3.1 in [29]) and dictionary learning (see Assumption B in [38]).

Proposition 1 (weighted central limit theorem). *Suppose $Z_i \overset{i.i.d}{\sim} P_Z(z)$ with a compact support, expectation μ, and variance σ^2. Define the weighted approximation of Z: $\hat{Z}^n_{\text{mod}} \triangleq \frac{1}{\sum_{i=1}^n (i/n)^p} \sum_{i=1}^n (i/n)^p Z_i$. Then we have*

$$\sqrt{n}(\hat{Z}^n_{\text{mod}} - \mu) \xrightarrow{d} N\left(0, \frac{p+1}{\sqrt{2p+1}}\sigma\right) \tag{39}$$

$$\mathbb{E}\left[\sqrt{n}|\hat{Z}^n_{\text{mod}} - \mu|\right] = \mathcal{O}(1)\ . \tag{40}$$

This proposition is an extension of the central limit theorem (CLT). As $p \to 0$, it reduces to the standard CLT. The proof is given in Appendix E.3.

Proposition 2 (convergence of functions). *With Assumptions 1–3, we have*

$$\mathbb{E}\left[\sqrt{t}\|F - F^{(t)}\|_\infty\right] \leq M \tag{41}$$

$$\mathbb{E}\left[\sqrt{t}\|F - F^{(t)}_{\text{mod}}\|_\infty\right] \leq \frac{p+1}{\sqrt{2p+1}}M \tag{42}$$

where $M > 0$ is some constant unrelated with t and $\|f\|_\infty = \sup_{\mathbf{d}\in C}\|f(\mathbf{d})\|$.

[8]The specific formulas for Assumptions 2 and 3 are shown in Appendix D.

This proposition is an extension of Donsker's theorem (see Lemma 7 in [38] and Chapter 19 in [53]). The proof is given in Appendix E.4.

Moreover, it shows that weighted approximation $F^{(t)}_{\text{mod}}$ and standard approximation $F^{(t)}$ have the same asymptotic convergence rate $\mathcal{O}(1/\sqrt{t})$. However, the error bound factor $(p+1)/\sqrt{2p+1}$ is a monotone increasing function in $p \geq 0$. Thus, a larger p leads to a larger variance and slower convergence of $F^{(t)}_{\text{mod}}$. This explains why we cannot choose p to be too large.

Proposition 3 (convergence of FPR implies convergence of iterates). *Let $(\mathbf{d}^*)^{(t)}$ be the exact minimizer of the t^{th} subproblem:*

$$(\mathbf{d}^*)^{(t)} = \arg\min_{\mathbf{d}} \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}) + \iota_C(\mathbf{d}) \ . \tag{43}$$

Let $\mathbf{d}^{(t)}$ be the solution obtained by the frequency-domain FISTA (Algorithm 3) with our proposed stopping condition (38). Then we have

$$\left\|\mathbf{d}^{(t)} - (\mathbf{d}^*)^{(t)}\right\| \leq \mathcal{O}\left(t^{-1}\right) \ . \tag{44}$$

The proof is given in Appendix E.1.

Proposition 4 (the convergence rate of Algorithm 2). *Let $\mathbf{d}^{(t)}$ be the sequence generated by Algorithm 2. Then we have*

$$\left\|\mathbf{d}^{(t+1)} - \mathbf{d}^{(t)}\right\| = \mathcal{O}\left(t^{-1}\right). \tag{45}$$

Compared with Lemma 1 in [38], which shows the convergence rate of the surrogate function method with exact D-update, our Proposition 4 shows that the inexact D-update (38) shares the same rate. Since our inexact version stops FISTA earlier, it is faster. The proof of this proposition is given in Appendix E.2.

Theorem 1 (almost sure convergence of Algorithm 2). *Let $\mathcal{F}^{(t)}_{\text{mod}}$ be the surrogate function sequence and $\mathbf{d}^{(t)}$ the iterate sequence, both generated by Algorithm 2. Then we have, with probability 1, the following:*

1. $\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t)})$ *converges.*
2. $\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t)}) - F(\mathbf{d}^{(t)}) \to 0$.
3. $F(\mathbf{d}^{(t)})$ *converges.*
4. $\text{dist}(\mathbf{d}^{(t)}, V) \to 0$, *where V is the set of stationary points of the CDL problem (13).*

The proof is given in Appendix E.5.

5. Learning from masked images.

In this section, we focus on the masked CDL problem (9), for which there are no existing online algorithms. Let

$$l_{\text{mask}}(\mathbf{d}, \mathbf{x}; \mathbf{s}) \triangleq \frac{1}{2}\left\|W \odot \left(\sum_{m=1}^{M} \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\right)\right\|_2^2 + \lambda \sum_{m=1}^{M} \|\mathbf{x}_m\|_1 \ .$$

The objective function is defined as

$$f_{\text{mask}}(\mathbf{d}; \mathbf{s}) \triangleq \min_{\mathbf{x}} l_{\text{mask}}(\mathbf{d}, \mathbf{x}; \mathbf{s}) \ , \tag{46}$$

allowing (9) to be written as a stochastic minimization problem:

$$\min_{\mathbf{d}} \mathbb{E}_{\mathbf{s}}[f_{\text{mask}}(\mathbf{d};\mathbf{s})] + \iota_{\text{C}}(\mathbf{d}) \; . \tag{47}$$

Both Algorithms 1 and 2 can be applied to the masked CDL problem (47). First, we write l_{mask} in a concise form:

$$l_{\text{mask}}(\mathbf{d},\mathbf{x};\mathbf{s}) = 1/2 \left\| W \odot X\mathbf{d} - W \odot \mathbf{s} \right\|_2^2 + \lambda \left\| \mathbf{x} \right\|_1 .$$

Thus, if we substitute operator X with $W \odot X$ and $\mathbf{s}$ with $W \odot \mathbf{s}$ and substitute standard CSC with masked CSC [25, 59], everything is the same as CDL without W, and we can apply Algorithms 1 and 2, with Option I, on (47). The numerical results for masked CDL are reported in section 7.

A variant of Algorithm 1 with Option II is also able to solve (47). First, l_{mask} on the frequency domain is

$$\hat{l}_{\text{mask}}(\hat{\mathbf{d}},\hat{\mathbf{x}};\hat{\mathbf{s}}) = 1/2 \left\| W \odot \text{IFFT}\left(\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}\right) \right\|_2^2 + \lambda \left\| \mathbf{x} \right\|_1 .$$

Similarly to the derivation in section 3, we derive the conjugate cogradient of $\hat{f}_{\text{mask}}$:

$$\begin{aligned} \nabla \hat{f}_{\text{mask}}(\hat{\mathbf{d}}^{(t-1)};\hat{\mathbf{s}}^{(t)}) &= \frac{\partial \hat{l}_{\text{mask}}}{\partial \hat{\mathbf{d}}}\left(\hat{\mathbf{d}}^{(t-1)},\hat{\mathbf{x}}^{(t)};\hat{\mathbf{s}}^{(t)}\right) \\ &= \left(\hat{X}^{(t)}\right)^H \text{FFT}\left\{W \odot \text{IFFT}\left(\hat{X}^{(t)}\hat{\mathbf{d}}^{(t-1)} - \hat{\mathbf{s}}^{(t)}\right)\right\} \; . \end{aligned}$$

The masked variant of Algorithm 1, Option II, can be derived from $\nabla \hat{f}_{\text{mask}}$ as

$$\mathbf{d}^{(t)} = \text{Proj}_{\text{C}_{\text{PN}}}\left(\text{IFFT}\left(\hat{\mathbf{d}}^{(t-1)} - \eta^{(t)} \nabla \hat{f}_{\text{mask}}(\hat{\mathbf{d}}^{(t-1)};\hat{\mathbf{s}}^{(t)})\right)\right) \; .$$

Algorithm 2, Option II, however, is not easily applied to (47). Computing the Hessian matrix in the frequency domain

$$\begin{aligned} \hat{A}^{(t)}_{\text{mask}} &= \alpha^{(t)} \hat{A}^{(t-1)}_{\text{mask}} + (\hat{X}^{(t)})^H \text{FFT}\left(W \odot \text{IFFT}(\hat{X}^{(t)})\right) \\ &= \alpha^{(t)} \hat{A}^{(t-1)}_{\text{mask}} + (\hat{X}^{(t)})^H \text{FFT}\left(W \odot X^{(t)}\right) \end{aligned}$$

requires an FFT on each column of a matrix $W \odot X^{(t)} \in \mathbb{R}^{MN \times N}$, which is very computationally expensive.

6. Numerical results: Learning from a clean data set.

All the experiments are computed using MATLAB R2016a running on a workstation with 2 Intel Xeon(R) X5650 CPUs clocked at 2.67 GHz. Implementations of these algorithms are available in the MATLAB version of the SPORCO software library [63] and will be included in a future release of the Python version of this library. The dictionary size is $12 \times 12 \times 64$, and the signal size is 256×256. Dictionaries are evaluated by comparing the functional values obtained by computing CBPDN (12) on the test set. A smaller functional value indicates a better dictionary. Similar methods to evaluate

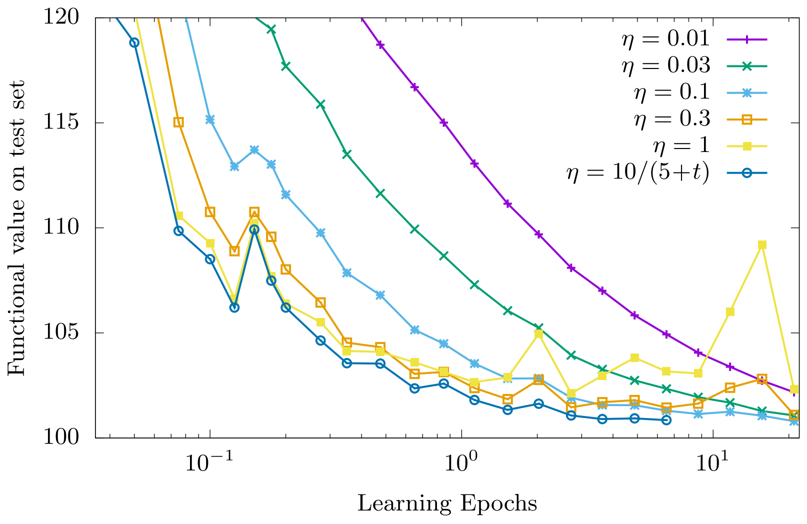


Figure 3. *Tuning the step size of modified SGD (Algorithm 1). A learning epoch is one pass through the whole training set. With a large fixed η, the algorithm converges fast at first but becomes unstable later on; with a small fixed η, the algorithm converges slowly. A diminishing step size provides a good balance.*

the dictionary are also used in other dictionary learning works [38, 52]. The regularization parameter is chosen as $\lambda = 0.1$.

The training set consists of 40 images selected from the MIRFLICKR-1M data set [27],[9] and the test set consists of 20 different images from the same source. All of the images used were originally of size 512×512. To accelerate the experiments, we crop the borders of both the training images and testing images and preserve the central part to yield 256×256. The training and testing images are preprocessed by dividing by 255 to rescale the pixel values to the range $[0, 1]$ and high-pass filtering.[10]

In this work, we solve the convolutional sparse coding step using an ADMM algorithm [58] with an adaptive penalty parameter scheme [64]. The stopping condition is that both primal and dual normalized residuals [64] be less than 10^{-3}, and the relaxation parameter is set to 1.8 [62].

6.1. Validation of Algorithm 1. First, we test the effect of step size $\eta^{(t)}$ in Algorithm 1. We can choose either a fixed step size or a diminishing step size:

$$\eta^{(t)} = \eta_0 \quad \text{or} \quad \eta^{(t)} = a/(t+b).$$

The results of experiments to determine the best choice of η are reported in Figure 3. We test the convergence performance of fixed step size scheme with values $\eta_0 \in \{1,\ 0.3,\ 0.1,\ 0.03,\ 0.01\}$.

[9]The actual image data contained in this data set are of very low resolution since the data set is primarily targeted at image classification tasks. The original images from which those used here were derived were obtained by downloading the original images from Flickr that were used to derive the MIRFLICKR-1M images.

[10]The preprocessing is applied due to the inability of the standard CSC model to effectively represent low-frequency/large-scale image components [61, section 3]. In this case, the high-pass component is computed as the difference between the input signal and a low-pass component computed by Tikhonov regularization with a gradient term [65, p. 3], with regularization parameter 5.0.

Table 3

Comparison between different options of Algorithm 1. $\lambda = 0.1$, average density of X: 0.0037. This is the validation of Table 1.

Schemes	Average single-step complexity (seconds)				Memory usage (MB)
	CBPDN	FFT/IFFT	Update $\mathbf{d}^{(t)}$	Total	
Spatial (dense matrix)	14.8	0	1.978	16.8	2346.44
Spatial (sparse matrix)	14.8	0	0.241	15.1	111.38
Frequency domain	14.8	0.047	0.025	14.9	154.84

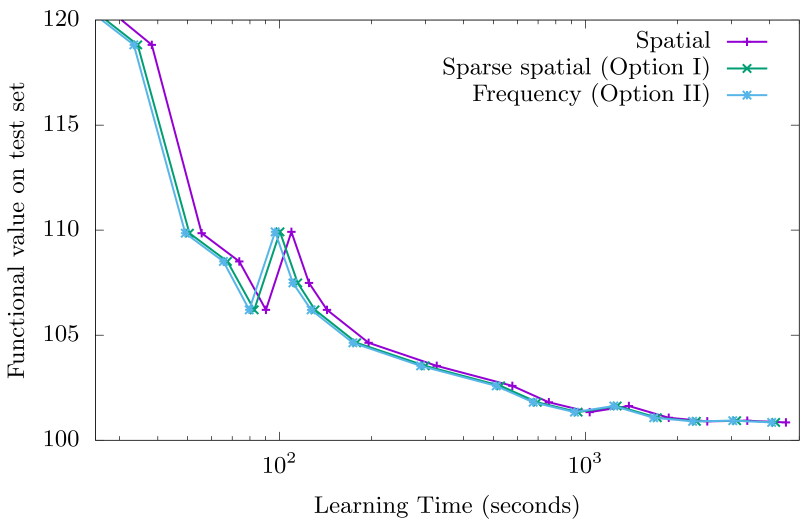


Figure 4. *Different options of Algorithm 1. Frequency-domain update (Option II) performs the best.*

We also test the convergence performance of the diminishing step size scheme with values $a \in \{5, 10, 20\}; b \in \{5, 10, 20\}$ and report the best ($a = 10, b = 5$) in Figure 3. When a large fixed step size is used, the functional value decreases fast initially but becomes unstable later on. A smaller step size causes the opposite. A diminishing step size balances accuracy and convergence rate.

Second, we test the computational techniques (computing with sparsity/computing in the frequency domain), as Table 3 shows. Both techniques reduce the complexity of updating $\mathbf{d}^{(t)}$. Option I has better memory cost, while Option II has better calculation time. Figure 4 shows the objective values versus training time.

6.2. Validation of Algorithm 2. For Algorithm 2, we test the four techniques separately: the forgetting exponent p, image splitting with size $\tilde{N}$, stopping tolerance of FISTA $\tau^{(t)}$, and computational techniques (sparsity or frequency-domain update).

6.2.1. Validation of Improvement I: Forgetting exponent p. In this section, we fix $\tilde{N} = 256 \times 256$ (no splitting) and $\tau^{(t)} = 10^{-4}$, which is small enough to give an accurate solution. Figure 5 shows that when $p = 0$, the curve is monotonic and with small oscillation, but it converges to a higher functional value. When p is larger, the algorithm converges to lower functional values. When p is too large, for instance, $p \in \{40, 80\}$, the curve oscillates

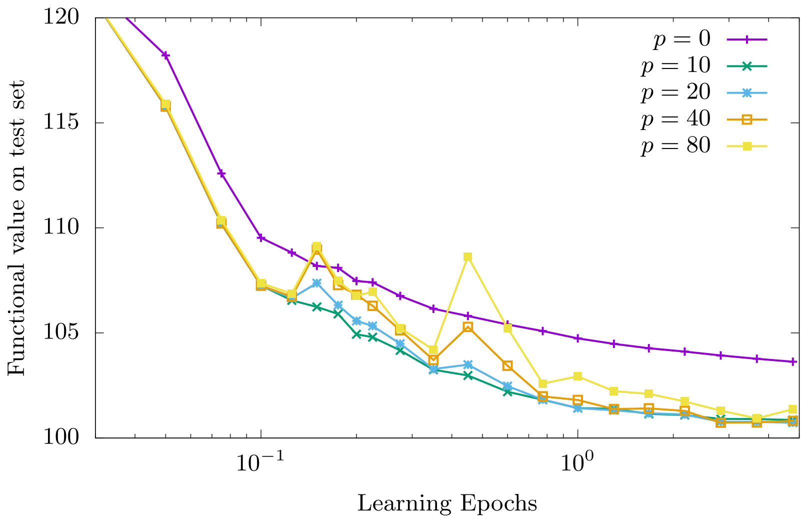


Figure 5. *Effect of Technique* I *(forgetting exponent p) in Algorithm* 2*. A small p leads to a higher functional value, while a large p leads to instability. $p = 10$ is a good choice.*

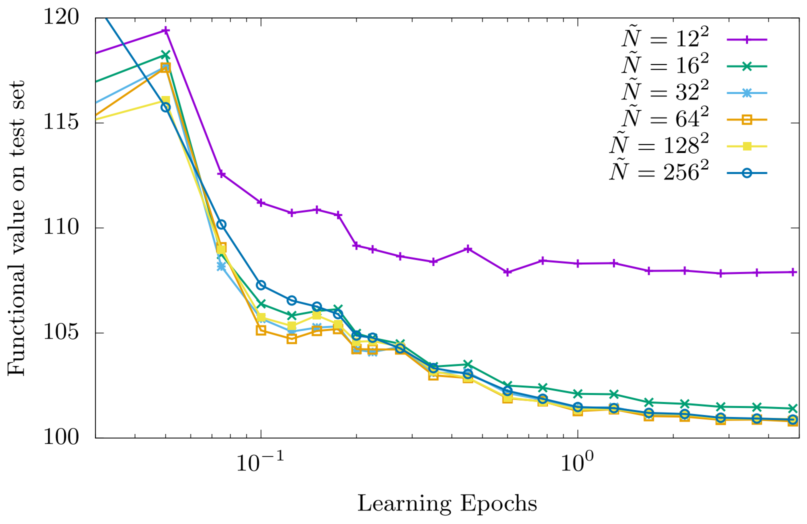
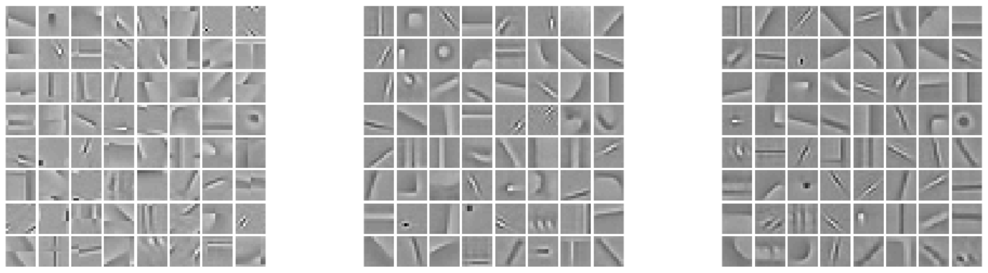


Figure 6. *Effect of the Technique* II *(image-splitting with size $\tilde{N}$) in Algorithm* 2*. A learning epoch is one pass through the whole training set. Boundary artifacts become significant when the splitting region size is smaller than twice the dictionary kernel size. Here the kernel size is 12×12 so the threshold is 24×24.*

severely, which indicates large variance. These results are consistent with Propositions 1 and 2. In the remaining sections, we fix $p = 10$ since it is shown to be a good choice.

6.2.2. Validation of Improvement II: Image splitting with size $\tilde{N}$ and boundary artifacts. In this section, we again fix $\tau^{(t)} = 10^{-4}$. Convergence comparisons are shown in Figure 6, and the dictionaries obtained with different $\tilde{N}$ are displayed Figure 7. In our experiments, we only consider square signals ($\tilde{N} = 12 \times 12, 16 \times 16, 32 \times 32, 64 \times 64, 256 \times 256$) and square dictionary kernels ($L = 12 \times 12$). When $\tilde{N} \geq 2^2 L$, say, $\tilde{N} = 32 \times 32$ or $\tilde{N} = 64 \times 64$,



(a) Dictionaries learned by $\tilde{N} = 12\times 12$: some incomplete features.

(b) Dictionaries learned by $\tilde{N} = 64 \times 64$.

(c) Dictionaries learned by $\tilde{N} = 256\times 256$ (no splitting).

Figure 7. *Effect of Technique* II *(image splitting with size $\tilde{N}$) in Algorithm* 2*: visualization of boundary artifacts.*

the algorithm converges to a good functional value, which is the same as that without image splitting. However, when $\tilde{N}$ is smaller than the threshold 2^2L, say, $\tilde{N} = 16 \times 16$ or 12×12, the algorithm converges to a higher functional value, which implies worse dictionaries. Thus, we can conclude that *the splitting size should be at least twice the dictionary kernel size in each dimension. Otherwise, it will lead to boundary artifacts.* This phenomenon is consistent with the discussion in section 4.2. The artifacts are specifically displayed in Figure 7. When $\tilde{N}$ is smaller than the threshold, say, 12×12, the features learned are incomplete.

This section only studies the effect, due to boundary artifacts, of image splitting on objective functional values. As Table 2 shows, it also helps reducing computing time and memory cost, which is numerically validated in section 6.2.4.

6.2.3. Validation of Improvement III: Stopping tolerance of FISTA $\tau^{(t)}$. In this section, we fix $p = 10, \tilde{N} = 256 \times 256$ (no splitting). Figure 8 shows the effect of using different $\tau^{(t)}$. Using a small stopping tolerance $\tau^{(t)} = 10^{-4}$ leads to a good functional value 101.1 but a large number of FISTA iterations, while a large tolerance 10^{-2} leads to a large functional value 104.4 and a small number of FISTA iterations. Consider our proposed diminishing tolerance rule (38) $\tau^{(t)} = 0.01/t$. When the algorithm starts, $t = 1$, we have $\tau^{(1)} = 10^{-2}$. At the end of the algorithm, $t = 100$, $\tau^{(100)} = 10^{-4}$. Based on the results in Figure 8, our diminishing tolerance avoids a large number of FISTA loops, especially at the initial steps, while losing little accuracy, as the final objective, 101.3, is close to 101.1.

6.2.4. Validation of Improvement IV: Computational techniques. In this section, we fix $p = 10, \tau^{(t)} = 0.01/t$, and compare the calculation time and memory usage of the spatial-domain update and the frequency-domain update. Table 4 illustrates that image splitting helps reduce the single-step complexity and memory usage for both Option I (spatial-domain update) and Option II (frequency-domain update). For Option II, the advantage of smaller splitting size $\tilde{N}$ is more significant than that of option I. When $\tilde{N} = 256 \times 256$, Option I is much better than Option II; but when $\tilde{N} = 64 \times 64$, the single-step time of Option II is comparable with that of Option I. The reason for this is that, for Option I, reducing $\tilde{N}$ only helps reduce the single-step time cost of CBPDN, updating Hessian matrix $A^{(t)}$ and the loops of FISTA, but does not help reduce the time cost of single-step time cost in FISTA. However,

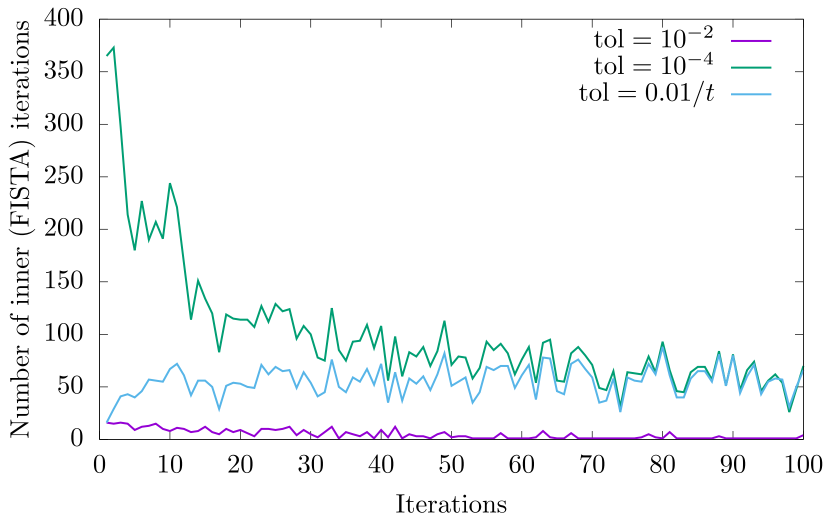


Figure 8. *Effect of Technique* III *(stopping FISTA early) in Algorithm* 2*. Final objective with* $\tau^{(t)} = 10^{-4}$*:* 101.1*. Final objective with* $\tau^{(t)} = 10^{-2}$*:* 104.4*. Final objective with* $\tau^{(t)} = 0.01/t$*:* 101.3*, where* t *is the iteration index with range* $1 \leq t \leq 100$*. Our diminishing tolerance* $\tau^{(t)} = 0.01/t$ *provides a useful balance in that it reduces the number of FISTA iterations while losing little accuracy on the final functional value.*

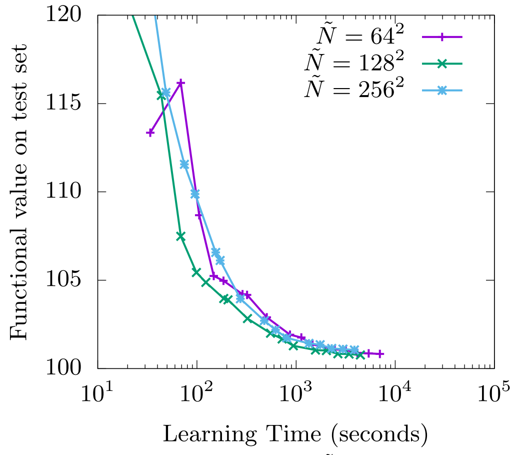
Table 4

Comparison of two options in Algorithm 2 *with different splitting size* $\tilde{N}$*.* $\lambda = 0.1$*, average density of* X*:* 0.0037*. This is the validation of Table* 2*. When* $\tilde{N} = 64 \times 64$*, the two options share similar performance: Option* I *is better on time cost, and Option* II *is better on memory cost. Image splitting is necessary for Option* II *but not necessary for Option* I*.*

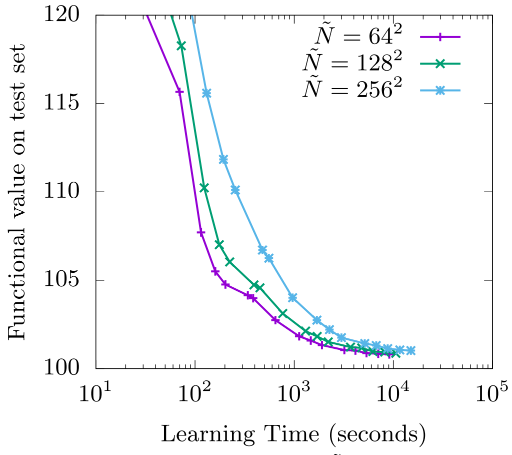
$\tilde{N}$	Average single-step complexity (seconds)				Memory usage (MB)
	CBPDN	Update $A^{(t)}$	FISTA (loops $\times$ single step)	Total	
Update in the spatial domain with dense matrix					
256×256	14.8	25.1	57×0.017	40.9	3058.56
128×128	3.42	6.80	37×0.017	10.8	1258.37
64×64	1.05	2.25	24×0.017	3.71	808.32
(Option I) Update in the spatial domain with sparse matrix					
256×256	14.8	4.47	57×0.017	20.3	486.91
128×128	3.42	1.77	37×0.017	5.82	366.51
64×64	1.05	0.84	24×0.017	2.30	342.90
(Option II) Update in the frequency domain (including extra time caused by FFT)					
256×256	14.8	0.89	57×1.068	76.6	2458.84
128×128	3.42	0.22	37×0.244	12.7	622.28
64×64	1.05	0.06	24×0.072	2.84	158.11

for Option II, image splitting not only reduces those three complexities but also reduces the single-step complexity of FISTA. Furthermore, Option II uses much less memory than Option I when $\tilde{N} = 64 \times 64$.

Figure 9(a) and Figure 9(b) compare the objective functional value versus time. Figure 9(a) indicates that reducing $\tilde{N}$ does *not* help a lot for Option I. Table 4 shows that smaller $\tilde{N}$



(a) Algorithm 2, Option I. $\tilde{N} = 128 \times 128$ is a good choice.



(b) Algorithm 2, Option II. $\tilde{N} = 64 \times 64$ converges fast.

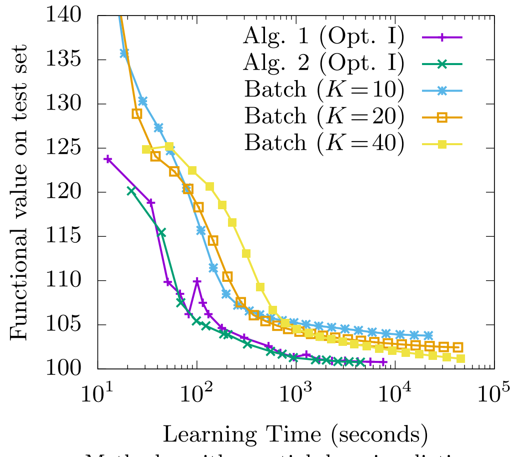
Figure 9. *Effect of splitting region size $\tilde{N}$ on different options.*

reduces the single-step complexity, but it also reduces the gain in each step because a smaller splitting size leads to less information used for training. This is a trade-off. By Figure 9(a), $\tilde{N} = 128 \times 128$ is a good choice.

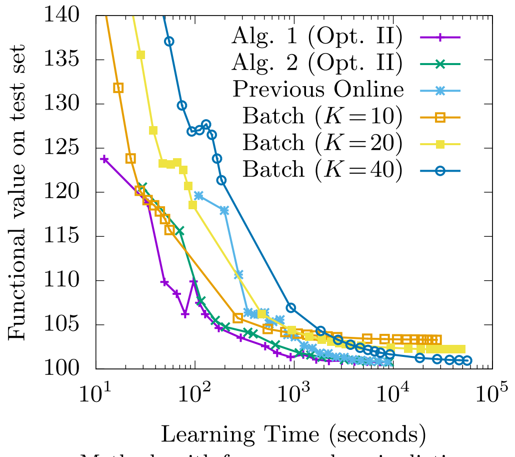
Option II, in contrast, benefits more from smaller $\tilde{N}$, as can be seen from Figure 9(b) and Table 4. Although splitting a training image reduces the gain in each step, the benefit overwhelms the loss. Thus, for Option II, the smaller the splitting size, the better, as long as $\tilde{N}$ is larger than the threshold for boundary artifacts.

6.3. Main result I: Convergence speed. In this section, we study the convergence speeds of all the methods on the clean data set, without a masking operator. We compare our methods with two leading batch learning algorithms: the method of Papyan et al. [41], which uses K-SVD and updates the dictionary in the spatial domain, and an algorithm [22] that uses the ADMM consensus dictionary update [54], which is computed in the frequency domain. For batch learning algorithms, we test on subsets of 10, 20, and 40 images selected from the training set. For online learning algorithms, since they are scalable in the size of the training set, we just test our methods on the whole training set of 40 images. All the parameters are tuned as follows. For batch learning algorithms (Papyan et al.), we use the software they released, and for batch learning algorithms (ADMM consensus update), we use the "adaptive penalty parameter" scheme in [64]. For modified SGD (Algorithm 1), we use the step size of $10/(5+t)$. For surrogate splitting (Algorithm 2), we use $p = 10, \tau^{(t)} = 0.01/t, \tilde{N} = 128 \times 128$ for the spatial-domain update and $\tilde{N} = 64 \times 64$ for the frequency-domain update, as we tuned in the previous sections. For our algorithm proposed in [32], we use $p = 10, \tau^{(t)} = 10^{-3}, \tilde{N} = 64 \times 64$.

The performance comparison of batch and online methods is presented in Figure 10. The advantage of online learning is significant (note that the time axis is logarithmically scaled). To obtain the same functional value 101 on the test set, batch learning takes 15 hours, our previous method [32] takes around 1.5 hours, Algorithm 2 with Option II takes around 1 hour, and Algorithms 1 and 2 with Option I take less than 1 hour. We can conclude that, both



(a) Methods with spatial-domain dictionary update scheme. Online algorithms, both Algorithm 1 and 2, outperform batch method (Papyan et al. [41]), Algorithm 2 performs the best.



(b) Methods with frequency-domain dictionary update scheme. In this plot, "Prev. Online" refers to our algorithm "Online-Samp" proposed in [32]. Both the online algorithms converge faster than batch methods (ADMM consensus dictionary update [54, 22]).

Figure 10. *Main result* I*: convergence speed comparison between online algorithms and batch algorithms.*

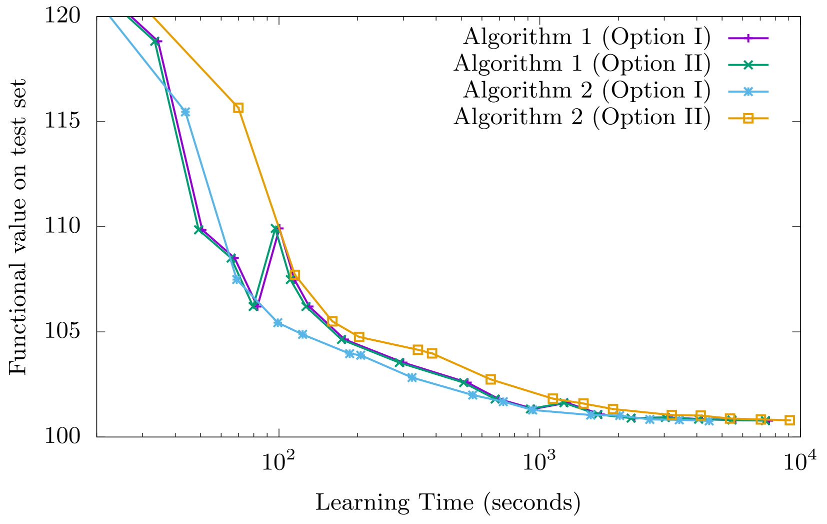


Figure 11. *Main result* I*: convergence speed comparison between online methods in this article.*

modified SGD (Algorithm 1) and surrogate splitting (Algorithm 2) converge faster than the batch learning algorithms and our previous method.

6.4. Main result II: Memory usage. As Table 5 shows, both Algorithms 1 and 2 save a large amount of memory.

6.5. Main result III: Dictionaries obtained by different algorithms. In Figure 12, we display the dictionaries obtained by the algorithms in section 6.3. A small training set, say, 10

Table 5
Main result II*: Memory usage comparison in megabytes.*

Scheme	Memory (MB)
Batch learning (consensus update, batch $K = 10$)	1959.58
Batch learning (consensus update, batch $K = 20$)	3887.08
Batch learning (consensus update, batch $K = 40$)	7742.08
Batch learning (Papyan et al. [41], batch $K = 10$)	1802.29
Batch learning (Papyan et al. [41], batch $K = 20$)	3390.24
Batch learning (Papyan et al. [41], batch $K = 40$)	6566.15
Our algorithm "Online-Samp" in [32]	158.11
Algorithm 1, Option I (sgd-spatial)	111.38
Algorithm 1, Option II (sgd-frequency)	154.84
Algorithm 2, Option I (surro-spatial)	342.90
Algorithm 2, Option II (surro-frequency)	158.11

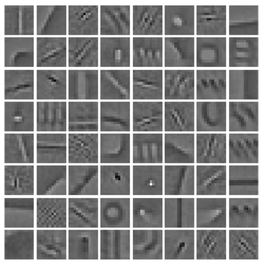
images, leads to some random kernels in the dictionaries. A training set containing 40 images works much better. Our algorithms can learn comparable dictionaries (see Figure 12(c), 12(i), and 12(h)) within much less time (see Figure 10) and much less memory usage (see Table 5).

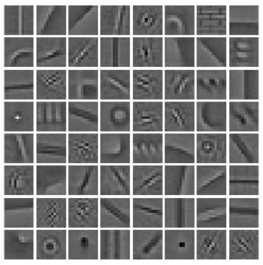
7. Numerical results: Learning from the noisy data set. In this section, we try to learn dictionaries from noisy images. We test the algorithms on the training set with *salt-and-pepper* impulse noise at known pixel locations. We apply the noise to $10\%, 20\%$, and 30% of the pixels, as Figure 13 shows. We use the data set with 40 training images and 20 testing images with uniform size 256×256, which are the same with those in section 6. All the images are preprocessed by applying a high-pass filter computed as the difference between the input and a nonlinear low-pass filter.[11] When the number of noisy pixels is low, say, 10%, SGD without masking (Algorithm 1) still can learn some features, as Figure 13(b) demonstrates. When the number of noisy pixels is significant, say, 30%, SGD without masking "learns" nothing valid, as Figure 13(h) demonstrates. However, SGD with masking technique works much better because it "ignores" the noisy pixels.

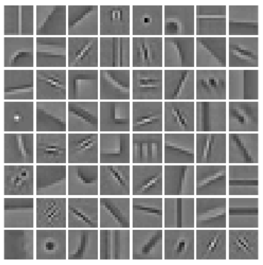
7.1. Masked CDL: Online algorithms vs, batch algorithm. We compare our algorithms with masked loss function and a batch dictionary learning algorithm[12] [59] incorporating the mask via the mask decoupling technique [25]. We use additive mask simulation (AMS) [59] to solve sparse coding step with the masked objective function. The parameters for Algorithms 1 and 2 are chosen similarly as those in section 6. For Algorithm 1, we choose $\eta^{(t)} = 10/(5+t)$ and Option I. For Algorithm 2, we choose $p = 10, \tilde{N} = 128^2, \tau^{(t)} = 0.01/(10+t)$, and Option I. A comparison of functional values on the *noise-free* test set is shown in Figure 14. Our online algorithms converge much faster and more stably. Algorithms 1 and 2 take around 1 hour to converge, while the mask-decoupling scheme requires more than 10 hours.

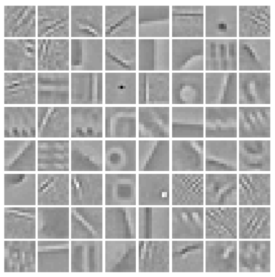
[11]The low-pass component was computed by ℓ_2 total-variation denoising [46] of the input image with a spatial mask informed by the known locations of corrupted pixels in the data fidelity term.

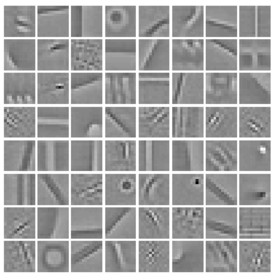
[12]We used the implementation `cbpdndlmd.m` from the MATLAB version of the SPORCO library [63], with the iterated Sherman–Morrison dictionary update solver option [62].

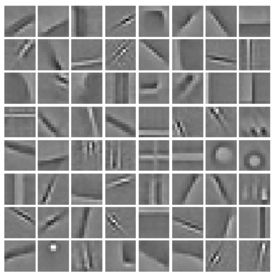

(a) Dictionaries learned by batch learning algorithm (consensus update, 10 training images): many "random" kernels.

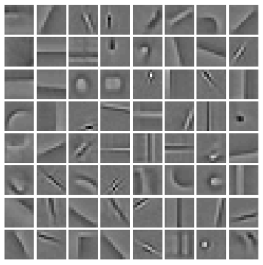

(b) Dictionaries learned by batch learning algorithm (consensus update, 20 training images): less "random" kernels, more valid features.

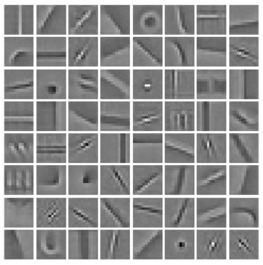

(c) Dictionaries learned by batch learning algorithm (consensus update, 40 training images): almost all kernels are valid.


(d) Dictionaries learned by batch learning algorithm (Papyan et al. [41], 10 training images): many "random" kernels.


(e) Dictionaries learned by batch learning algorithm (Papyan et al. [41], 20 training images): less "random" kernels, more valid features.


(f) Dictionaries learned by batch learning algorithm (Papyan et al. [41], 40 training images): almost all kernels are valid.


(g) Dictionaries learned by [32], almost all kernels are valid.


(h) Dictionaries learned by Algorithm 1, almost all kernels are valid.

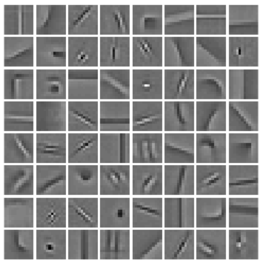
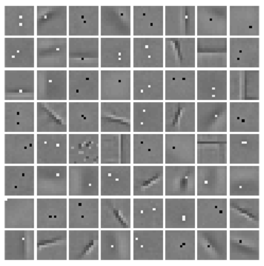

(i) Dictionaries learned by Algorithm 2, almost all kernels are valid.

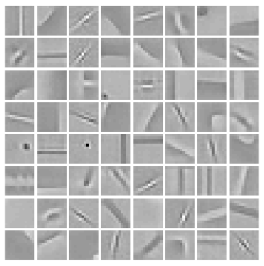
Figure 12. *Main result* III*: A comparison of dictionaries learned using different algorithms.*

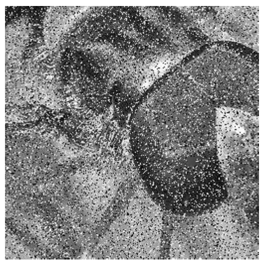
8. Numerical results: Learning from a large data set.

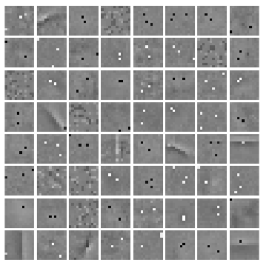
In this section, we test the feasibility of our methods on a large data set, which is not tractable for batch methods. This training set consists of 1000 images of size 256×256 selected from the MIRFLICKR-1M data

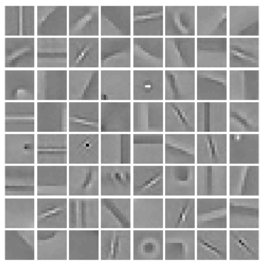
(a) One of the training images. (10% pixels corrupted (salt-and-pepper noise))

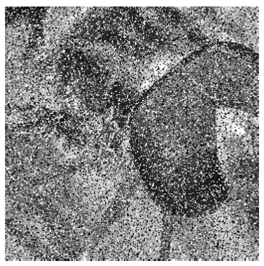

(b) Learning with normal Algorithm 1: some features learned.

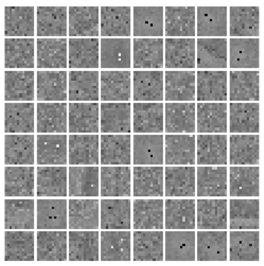

(c) Learning with Algorithm 1 on masked loss function: clean features learned.


(d) One of the training images. (20% pixels corrupted (salt-and-pepper noise))


(e) Learning with normal Algorithm 1: few features learned.


(f) Learning with Algorithm 1 on masked loss function: clean features learned.


(g) One of the training images. (30% pixels corrupted (salt-and-pepper noise))


(h) Learning with normal Algorithm 1: almost no valid features.

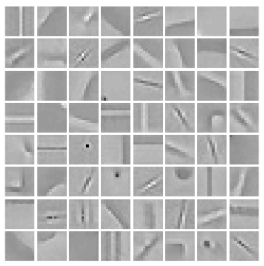

(i) Learning with Algorithm 1 on masked loss function: clean features learned.

Figure 13. *Learning from the noisy training set.*

set, and the testing set consists of 50 distinct images with the same size from the same source. A dictionary of 100 kernels with size 12×12 is trained, and the related experiment results are reported in Figure 15.

The parameters for Algorithms 1 and 2 are chosen the same as those in section 6. For Algorithm 1, we choose $\eta^{(t)} = 10/(5+t)$ and Option I. For Algorithm 2, we choose $p = 10$, $\tilde{N} = 128^2, \tau^{(t)} = 0.01/t$, and Option I.

Unlike the experiments in section 6, we run our algorithms with only one epoch, i.e., true online learning. Results in Figure 15 demonstrate that our Algorithms 1 and 2 are both

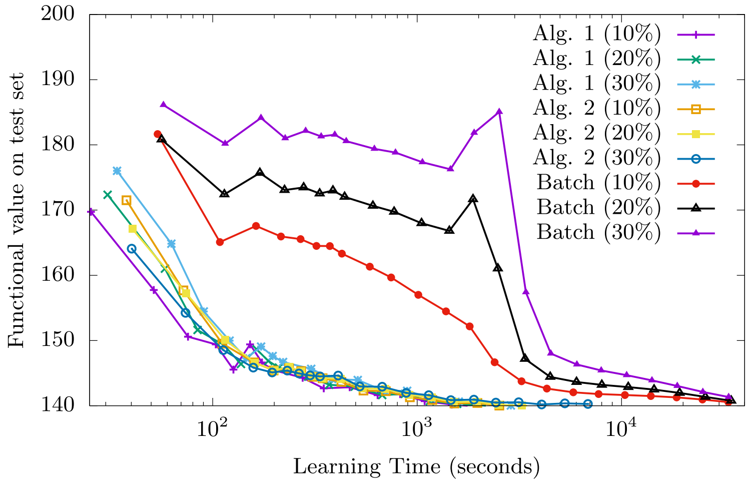
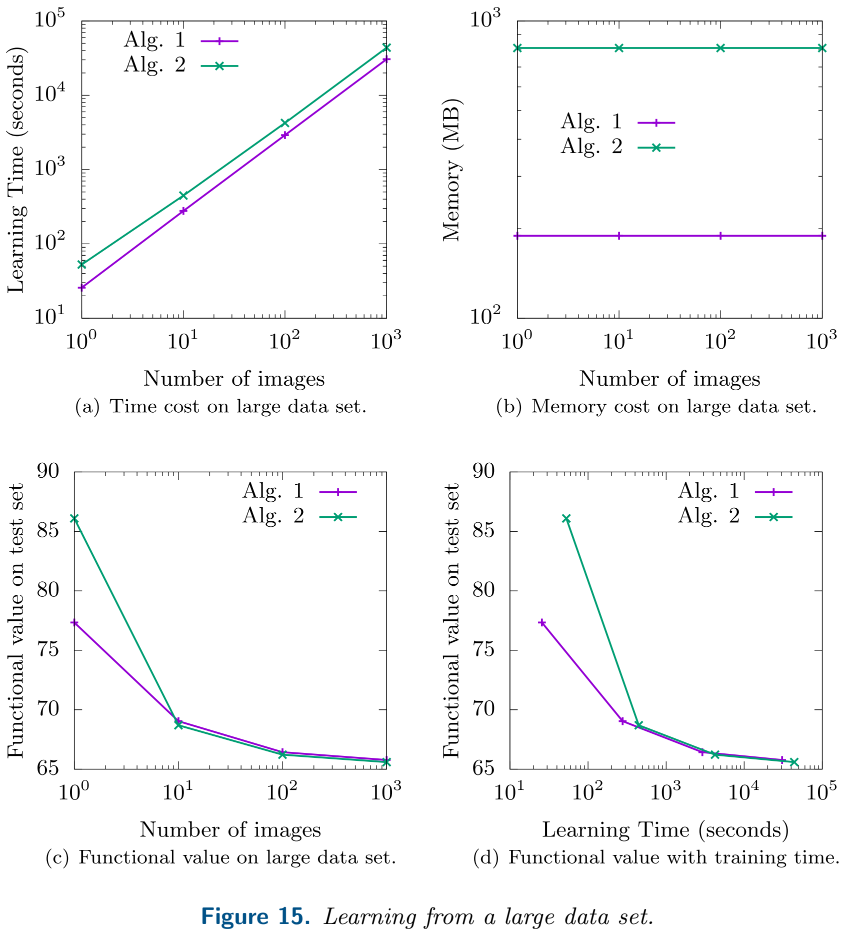


Figure 14. *Learning from the noisy training set: speed test. Both Algorithms* 1 *and* 2 *use Option* I. *"Batch" refers to iterated Sherman–Morrison dictionary update* [62] *with mask decoupling technique* [25], *as in* [59].



(a) Time cost on large data set.

(b) Memory cost on large data set.

(c) Functional value on large data set.

(d) Functional value with training time.

Figure 15. *Learning from a large data set.*

feasible on large data set. The first-order method, Algorithm 1, has a cheaper single step, but it learns less with the same number of iterations. The second-order method, Algorithm 2, has the converse behavior, achieveing a slightly smaller functional value with the same number of iterations. Finally, Figure 15(d) shows that the two algorithms have similar performance in terms of functional value on the test set with respect to training time. This result is consistent with those in section 6.

9. Conclusions.

We have proposed two efficient online convolutional dictionary learning methods. Both of them have a theoretical convergence guarantee and show good performance on both time and memory usage. Compared to recent online CDL works [13, 57], which use the same framework but different D-update algorithms, our second-order method improves the framework by several practical techniques. Our first-order method, to the best of our knowledge, is the first attempt to use first-order methods in online CDL. It shows better performance in time and memory usage and requires fewer parameters to tune. Moreover, based on these two methods, we have also proposed an online dictionary learning method, which is able to learn meaningful dictionaries from a partially masked training set. Although only single-channel images are considered in this article, our online methods can easily be extended to the multichannel case [60].

Appendix A. Derivation of conjugate cogradient (23) of the frequency-domain loss function $\hat{l}$.

Consider a real-valued function defined on the complex domain $f : \mathbb{C}^n \to \mathbb{R}$, which can be viewed as a function defined on the $2n$-dimensional real domain: $f(x) = f(\Re(x) + i\Im(x))$, where $\Re(x), \Im(x) \in \mathbb{R}^n$ are the real part and the imaginary part, respectively. By [49], "conjugate cogradient" is defined as

$$\nabla f(x) \triangleq \frac{\partial f}{\partial \Re(x)} + i \frac{\partial f}{\partial \Im(x)} . \tag{48}$$

Based on (48), we give a derivation of (23).

Recall the definition $\hat{l}(\hat{\mathbf{d}}, \hat{\mathbf{x}}; \hat{\mathbf{s}}) = 1/2\|\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}\|^2$. Substituting $\hat{X} = \Re(\hat{X}) + i\Im(\hat{X})$, $\hat{\mathbf{d}} = \Re(\hat{\mathbf{d}}) + i\Im(\hat{\mathbf{d}})$, and $\hat{\mathbf{s}} = \Re(\hat{\mathbf{s}}) + i\Im(\hat{\mathbf{s}})$ into $\hat{l}$, we have

$$\begin{aligned}
&\hat{l}(\hat{\mathbf{d}}, \hat{\mathbf{x}}; \hat{\mathbf{s}}) \\
&= \frac{1}{2}\left\|\Re(\hat{X})\Re(\hat{\mathbf{d}}) - \Im(\hat{X})\Im(\hat{\mathbf{d}}) - \Re(\hat{\mathbf{s}}) + i\left(\Im(\hat{X})\Re(\hat{\mathbf{d}}) + \Re(\hat{X})\Im(\hat{\mathbf{d}}) - \Im(\hat{\mathbf{s}})\right)\right\|^2 \\
&= \frac{1}{2}\left\|\Re(\hat{X})\Re(\hat{\mathbf{d}}) - \Im(\hat{X})\Im(\hat{\mathbf{d}}) - \Re(\hat{\mathbf{s}})\right\|^2 + \frac{1}{2}\left\|\Im(\hat{X})\Re(\hat{\mathbf{d}}) + \Re(\hat{X})\Im(\hat{\mathbf{d}}) - \Im(\hat{\mathbf{s}})\right\|^2 .
\end{aligned}$$

The partial derivatives on $\Re(\hat{\mathbf{d}})$ and $\Im(\hat{\mathbf{d}})$ are, respectively,

$$\begin{aligned}
\frac{\partial \hat{l}}{\partial \Re(\hat{\mathbf{d}})} &= \Re(\hat{X})^T\left(\Re(\hat{X})\Re(\hat{\mathbf{d}}) - \Im(\hat{X})\Im(\hat{\mathbf{d}}) - \Re(\hat{\mathbf{s}})\right) + \Im(\hat{X})^T\left(\Im(\hat{X})\Re(\hat{\mathbf{d}}) + \Re(\hat{X})\Im(\hat{\mathbf{d}}) - \Im(\hat{\mathbf{s}})\right) \\
\frac{\partial \hat{l}}{\partial \Im(\hat{\mathbf{d}})} &= \Im(\hat{X})^T\left(-\Re(\hat{X})\Re(\hat{\mathbf{d}}) + \Im(\hat{X})\Im(\hat{\mathbf{d}}) + \Re(\hat{\mathbf{s}})\right) + \Re(\hat{X})^T\left(\Im(\hat{X})\Re(\hat{\mathbf{d}}) + \Re(\hat{X})\Im(\hat{\mathbf{d}}) - \Im(\hat{\mathbf{s}})\right).
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\hat{X}^H(\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}) \\
&\quad = (\Re(\hat{X}) - i\Im(\hat{X}))^T\big((\Re(\hat{X})\Re(\hat{\mathbf{d}}) - \Im(\hat{X})\Im(\hat{\mathbf{d}}) - \Re(\hat{\mathbf{s}})) \\
&\qquad\qquad\qquad\qquad + i\big(\Im(\hat{X})\Re(\hat{\mathbf{d}}) + \Re(\hat{X})\Im(\hat{\mathbf{d}}) - \Im(\hat{\mathbf{s}})\big)\big) \\
&\quad = \frac{\partial \hat{l}}{\partial R\hat{\mathbf{d}}} + i\frac{\partial \hat{l}}{\partial I\hat{\mathbf{d}}} \,.
\end{aligned}$$

By the definition of conjugate cogradient (48), the right side of the above equation is the conjugate cogradient of $\hat{l}$, i.e.,

$$\nabla\hat{l}(\hat{\mathbf{d}}, \hat{\mathbf{x}}; \hat{\mathbf{s}}) = \hat{X}^H(\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}) \,.$$

Appendix B. Proof of the equivalence between the gradients in the frequency domain and spatial domain (25).

Proof. Let $\mathcal{F}$ be the Fourier operator from $\mathbb{C}^N$ to $\mathbb{C}^N$, so that $\mathcal{F}^{-1} = \mathcal{F}^H$ is the inverse Fourier operator. $\mathbf{x}$ and X are the vector form and the operator form of the coefficient map, respectively. $\hat{\mathbf{x}}$ and $\hat{X}$ are the corresponding vector and operator in the frequency domain. By definition, we have that $\hat{\mathbf{x}} = \mathcal{F}\mathbf{x}$. We claim that

$$\hat{X} = \mathcal{F}X\mathcal{F}^H \,. \tag{49}$$

To prove this, notice that

$$\hat{X}\hat{\mathbf{d}} = \mathcal{F}\left(\mathbf{x} * \mathbf{d}\right) = \mathcal{F}\left(X\mathbf{d}\right) = \mathcal{F}X\mathcal{F}^H\mathcal{F}\mathbf{d} = \mathcal{F}X\mathcal{F}^H\hat{\mathbf{d}} \quad \forall \mathbf{d} \in \mathbb{R}^N \,.$$

Thus, we have $\hat{X} = \mathcal{F}X\mathcal{F}^H$. With this equation, we have

$$\begin{aligned}
\hat{X}^H(\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}) &= \left(\mathcal{F}X\mathcal{F}^H\right)^H\left(\mathcal{F}X\mathcal{F}^H\mathcal{F}\mathbf{d} - \mathcal{F}\mathbf{s}\right) = \left(\mathcal{F}X^T\mathcal{F}^H\right)(\mathcal{F}X\mathbf{d} - \mathcal{F}\mathbf{s}) \\
&= \mathcal{F}(X^T(X\mathbf{d} - \mathbf{s})) \,,
\end{aligned}$$

which is exactly (25). ■

Appendix C. Frequency-domain FISTA. To solve (35), we propose frequency-domain FISTA, Algorithm 3. It calculates the gradient in the frequency domain and does projection and extrapolation in the spatial domain. Mathematically speaking, (25) illustrates that frequency-domain FISTA is actually equivalent with standard FISTA. However, calculating the convolutional operator in the frequency domain reduces computing time. Thus, our algorithm is faster.

Appendix D. Details of the assumptions.

D.1. Description of Assumption 2. To represent Assumption 2 in a concise way, we use the notation

$$D\mathbf{x} = \sum_{m=1}^{M} \mathbf{d}_m * \mathbf{x}_m \approx \mathbf{s} \,,$$

Algorithm 3. Frequency-domain FISTA for solving subproblem (35).

```
  Input: Hessian matrix $\hat{A}^{(t)}_{\mathrm{mod}}$ and vector $\hat{\mathbf{b}}^{(t)}_{\mathrm{mod}}$.
  Dictionary of last iterate: $\mathbf{d}^{(t-1)}$.
  Initialize: Let $\mathbf{g}^0 = \mathbf{d}^{(t-1)}$ (warm start), $\mathbf{g}^0_{\mathrm{aux}} = \mathbf{g}^0$, $\gamma^0 = 1$.
1 for $j = 0, 1, 2, \dots$ until condition (38) is satisfied do
2     Compute DFT: $\hat{\mathbf{g}}^j_{\mathrm{aux}} = \mathrm{FFT}(\mathbf{g}^j_{\mathrm{aux}})$.
3     Compute conjugate cogradient: $\nabla\hat{\mathcal{F}}^{(t)}_{\mathrm{mod}}(\hat{\mathbf{g}}^j_{\mathrm{aux}}) = \frac{1}{\Lambda^{(t)}}\left(\hat{A}^{(t)}_{\mathrm{mod}}\hat{\mathbf{g}}^j_{\mathrm{aux}} - \hat{\mathbf{b}}^{(t)}_{\mathrm{mod}}\right)$ .
4     Compute the next iterate:
      (50)  $\mathbf{g}^{j+1} = \mathrm{Proj}_{\mathrm{C_{PN}}}\left(\mathrm{IFFT}\left(\hat{\mathbf{g}}^j_{\mathrm{aux}} - \eta\nabla\hat{\mathcal{F}}^{(t)}_{\mathrm{mod}}(\hat{\mathbf{g}}^j_{\mathrm{aux}})\right)\right)$ .
      Let $\gamma^{j+1} = \left(1 + \sqrt{1 + 4(\gamma^j)^2}\right)/2$, then compute the auxiliary variable:
      (51)  $\mathbf{g}^{j+1}_{\mathrm{aux}} = \mathbf{g}^{j+1} + \frac{\gamma^j - 1}{\gamma^{j+1}}(\mathbf{g}^{j+1} - \mathbf{g}^j)$ .
5 end
  Output: $\mathbf{d}^{(t)} \leftarrow \mathbf{g}^J$, where $J$ is the last iterate.
```

where $\mathbf{x} \in \mathbb{R}^{MN}$, $\mathbf{s} \in \mathbb{R}^N$, and $D : \mathbb{R}^{MN} \to \mathbb{R}^N$ is the convolutional dictionary. Then CBPDN problem (7) could be written as

$$\min_{\mathbf{x}\in\mathbb{R}^{MN}} (1/2)\left\|D\mathbf{x} - \mathbf{s}\right\|_2^2 + \lambda\left\|\mathbf{x}\right\|_1 \ . \tag{52}$$

The coefficient map $\mathbf{x}$ is usually sparse, and Λ is the set of indices of nonzero elements in $\mathbf{x}$. Then we have $D\mathbf{x} = D_\Lambda \mathbf{x}_\Lambda$. By the results in [20], problem (52) has the unique solution if $D_\Lambda^T D_\Lambda$ is invertible,[13] and its unique solution satisfies

$$\mathbf{x}^*_\Lambda = (D_\Lambda^T D_\Lambda)^{-1}(D_\Lambda^T \mathbf{s} - \lambda \mathrm{sign}(\mathbf{x}^*_\Lambda)) \ . \tag{53}$$

Specifically, Assumption 2 is *for all signals* $\mathbf{s}$ *and dictionaries* $\mathbf{d}$*, the smallest singular value of* $D_\Lambda^T D_\Lambda$ *is lower bounded by a positive number, i.e.,*

$$\sigma_{\min}(D_\Lambda^T D_\Lambda) \geq \kappa \ . \tag{54}$$

Except for condition (54), other types of uniqueness conditions of CSC are studied in recent works [40, 42, 50].

[13]Although [20] only studies standard sparse coding, the uniqueness condition can be applied to the convolutional case because the only condition in their proof is "for a convex function $f(x)$ on $\mathbb{R}^n$, x a minimum if and only if $0 \in \partial f(x)$." The only assumption is the convexity of the function, with no assumptions on the signals and dictionaries. Thus, large signals and convolutional dictionaries as in our case are consistent with the condition in [20].

D.2. Description of Assumption 3. Specifically, Assumption 3 is *the surrogate functions* $\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d})$ *are uniformly strongly convex, i.e.,*

$$\left\langle \nabla \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}) - \nabla \mathcal{F}^{(t)}_{\text{mod}}(\tilde{\mathbf{d}}), \mathbf{d} - \tilde{\mathbf{d}} \right\rangle \geq \mu \big\|\mathbf{d} - \tilde{\mathbf{d}}\big\|^2 \tag{55}$$

for all $t, \mathbf{d}, \tilde{\mathbf{d}}$ *for some* $\mu > 0$.

Appendix E. Proofs of propositions and the theorem. Before proving propositions, we introduce a useful lemma.

Lemma 2 (uniform smoothness of surrogate functions). *Under Assumptions 1 and 2, we have that* $f^{(t)}$ (27) *and* $\mathcal{F}^{(t)}_{\text{mod}}$(36) *are uniformly L-smooth, i.e.,*

$$\begin{aligned} \big\|\nabla f^{(t)}(\mathbf{d}) - \nabla f^{(t)}(\tilde{\mathbf{d}})\big\| &\leq L_f \big\|\mathbf{d} - \tilde{\mathbf{d}}\big\| \\ \big\|\nabla \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}) - \nabla \mathcal{F}^{(t)}_{\text{mod}}(\tilde{\mathbf{d}})\big\| &\leq L_{\mathcal{F}} \big\|\mathbf{d} - \tilde{\mathbf{d}}\big\| \end{aligned} \tag{56}$$

for all $t, \mathbf{d}, \tilde{\mathbf{d}}$ *for some constants* $L_f > 0, L_{\mathcal{F}} > 0$.

Proof. First, we consider a single surrogate function:

$$\big\|\nabla f^{(t)}(\mathbf{d}) - \nabla f^{(t)}(\tilde{\mathbf{d}})\big\| = \big\|(X^{(t)})^T(X^{(t)})(\mathbf{d} - \tilde{\mathbf{d}})\big\| \,.$$

By $\mathbf{d} \in \mathrm{C}$ (the compact support of $\mathbf{d}$), Assumption 1 (the compact support of $\mathbf{s}$), and (53) (regularity of convolutional sparse coding), we have that $\mathbf{x}^{(t)}$ is uniformly bounded. Therefore, $X^{(t)}$, the operator form of $\mathbf{x}^{(t)}$, is also uniformly bounded,

$$\big\|X^{(t)}\big\| \leq M \tag{57}$$

for all t for some $M > 0$, which is independent of t.

By (31), we have

$$\begin{aligned} \Big\|\nabla \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}) - \nabla \mathcal{F}^{(t)}_{\text{mod}}(\tilde{\mathbf{d}})\Big\| =& \left\| \frac{1}{\Lambda^{(t)}} \sum_{\tau=1}^{t} (\tau/t)^p (X^{(\tau)})^T (X^{(\tau)})(\mathbf{d} - \tilde{\mathbf{d}}) \right\| \\ \leq& \frac{1}{\Lambda^{(t)}} \sum_{\tau=1}^{t} (\tau/t)^p \Big\|(X^{(\tau)})^T (X^{(\tau)})(\mathbf{d} - \tilde{\mathbf{d}})\Big\| \,, \end{aligned}$$

which, together with (57), implies (56). ■

E.1. Proof of Proposition 3. Given the strong convexity (55) and smoothness (56) of the surrogate function, we start to prove Proposition 3.

Proof. To prove (44), we consider a more general case. Let $\mathbf{g}^*$ be the minimizer of the following subproblem:

$$\mathbf{g}^* = \arg\min_{\mathbf{d}} \mathcal{F}(\mathbf{d}) + \iota_{\mathrm{C}}(\mathbf{d}) \,,$$

where $\mathcal{F}$ is μ-strongly convex and L-smooth. Moreover, $\mathbf{g}^j$ and $\mathbf{g}^j_{\text{aux}}$ are the iterates generated in Algorithm 3, and j is the loop index. Then we want to show that

$$\left\|\mathbf{g}^{j+1} - \mathbf{g}^*\right\| \leq C\mathbf{R}^{(t)}(\mathbf{g}^j_{\text{aux}}) \quad \forall j \geq 0\,. \tag{58}$$

By (25), it is enough to prove the above for the spatial-domain FISTA. By strong convexity and smoothness of $\mathcal{F}$, we obtain

$$\begin{aligned}
&\|\mathbf{g}^{j+1} - \mathbf{g}^*\|^2 \\
&\quad = \left\|\text{Proj}(\mathbf{g}^j_{\text{aux}} - \eta\nabla\mathcal{F}(\mathbf{g}^j_{\text{aux}})) - \text{Proj}(\mathbf{g}^* - \eta\nabla\mathcal{F}(\mathbf{g}^*))\right\|^2 \\
&\quad \leq \left\|\mathbf{g}^j_{\text{aux}} - \eta\nabla\mathcal{F}(\mathbf{g}^j_{\text{aux}}) - \mathbf{g}^* - \eta\nabla\mathcal{F}(\mathbf{g}^*)\right\|^2 \\
&\quad = \left\|\mathbf{g}^j_{\text{aux}} - \mathbf{g}^* - \eta\left(\nabla\mathcal{F}(\mathbf{g}^j_{\text{aux}}) - \nabla\mathcal{F}(\mathbf{g}^*)\right)\right\|^2 \\
&\quad = \|\mathbf{g}^j_{\text{aux}} - \mathbf{g}^*\|^2 - 2\eta\langle\mathbf{g}^j_{\text{aux}} - \mathbf{g}^*, \nabla\mathcal{F}(\mathbf{g}^j_{\text{aux}}) - \nabla\mathcal{F}(\mathbf{g}^*)\rangle + \eta^2\left\|\nabla\mathcal{F}(\mathbf{g}^j_{\text{aux}}) - \nabla\mathcal{F}(\mathbf{g}^*)\right\|^2 \\
&\quad \leq (1 - 2\mu\eta + \eta^2L^2)\|\mathbf{g}^j_{\text{aux}} - \mathbf{g}^*\|^2\,.
\end{aligned}$$

Combining the above inequality and the definition of FPR (37), we have

$$\begin{aligned}
\mathbf{R}(\mathbf{g}^j_{\text{aux}}) &= \left\|\mathbf{g}^j_{\text{aux}} - \text{Proj}\left(\mathbf{g}^j_{\text{aux}} - \eta\nabla\mathcal{F}(\mathbf{g}^j_{\text{aux}})\right)\right\| \\
&= \|\mathbf{g}^j_{\text{aux}} - \mathbf{g}^{(j+1)}\| \\
&= \|\mathbf{g}^j_{\text{aux}} - \mathbf{g}^* - (\mathbf{g}^{(j+1)} - \mathbf{g}^*)\| \\
&\geq \|\mathbf{g}^j_{\text{aux}} - \mathbf{g}^*\| - \|\mathbf{g}^{(j+1)} - \mathbf{g}^*\| \\
&\geq \left(1 - \sqrt{1 - 2\mu\eta + \eta^2L^2}\right)\|\mathbf{g}^j_{\text{aux}} - \mathbf{g}^*\| \\
&\geq \frac{1 - \sqrt{1 - 2\mu\eta + \eta^2L^2}}{\sqrt{1 - 2\mu\eta + \eta^2L^2}}\|\mathbf{g}^{j+1} - \mathbf{g}^*\|\,.
\end{aligned}$$

Let the step size be small enough $\eta \leq \min(\mu/L^2, 1/\mu)$; then we have $0 \leq 1 - 2\mu\eta + \eta^2L^2 \leq 1$, which implies (58). Combining (58) and (38), we get (44). ■

E.2. Proof of Proposition 4.

Proof. Recall $(\mathbf{d}^*)^{(t)}$ (43) is the "exact solution" of the t^{th} iterate and $\mathbf{d}^{(t)}$ is the "inexact solution" of the t^{th} iterate (i.e., the approximated solution obtained by stopping condition (38)). Then, by the strong convexity of $\mathcal{F}^{(t)}_{\text{mod}}$, we have

$$\begin{aligned}
&\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t)}) \\
&\quad = \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t)}_{\text{mod}}((\mathbf{d}^*)^{(t)}) - \left(\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t)}) - \mathcal{F}^{(t)}_{\text{mod}}((\mathbf{d}^*)^{(t)})\right) \\
&\quad \geq \mu\|\mathbf{d}^{(t+1)} - (\mathbf{d}^*)^{(t)}\|^2 - L\|\mathbf{d}^{(t)} - (\mathbf{d}^*)^{(t)}\|^2 \\
&\quad \geq \mu\left(\|\mathbf{d}^{(t+1)} - \mathbf{d}^{(t)}\| - \|\mathbf{d}^{(t)} - (\mathbf{d}^*)^{(t)}\|\right)^2 - L\|\mathbf{d}^{(t)} - (\mathbf{d}^*)^{(t)}\|^2\,.
\end{aligned}$$

Let $r^{(t)} = \|\mathbf{d}^{(t+1)} - \mathbf{d}^{(t)}\|$. If $r^{(t)} \leq C/t$, Proposition 4 is directly proved. Otherwise, Proposition 3 (44) implies $r^{(t)} - \|\mathbf{d}^{(t)} - (\mathbf{d}^*)^{(t)}\| \geq r^{(t)} - C/t \geq 0$ and

$$\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t)}) \geq \mu\left(r^{(t)} - \frac{C}{t}\right)^2 - \frac{LC^2}{t^2}\,. \tag{59}$$

On the other hand,

$$\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t)}) = \underbrace{\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t+1)}_{\text{mod}}(\mathbf{d}^{(t+1)})}_{\mathbf{T}_1}$$
$$+ \underbrace{\mathcal{F}^{(t+1)}_{\text{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t+1)}_{\text{mod}}(\mathbf{d}^{(t)})}_{\mathbf{T}_2} + \underbrace{\mathcal{F}^{(t+1)}_{\text{mod}}(\mathbf{d}^{(t)}) - \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t)})}_{\mathbf{T}_3} .$$

Now we will give the upper bounds of $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3$. Given the smoothness of $\mathcal{F}^{(t)}_{\text{mod}}$(56) and $(\mathbf{d}^*)^{(t+1)}$ being the minimizer of $\mathcal{F}^{(t)}_{\text{mod}}$, we have an upper bound of $\mathbf{T}_2$:

$$\begin{aligned} \mathbf{T}_2 &= \mathcal{F}^{(t+1)}_{\text{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t+1)}_{\text{mod}}(\mathbf{d}^{(t)}) \\ &= \mathcal{F}^{(t+1)}_{\text{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t+1)}_{\text{mod}}((\mathbf{d}^*)^{(t+1)}) - \left(\mathcal{F}^{(t+1)}_{\text{mod}}(\mathbf{d}^{(t)}) - \mathcal{F}^{(t+1)}_{\text{mod}}((\mathbf{d}^*)^{(t+1)})\right) \\ &\le L\|\mathbf{d}^{(t+1)} - (\mathbf{d}^*)^{(t+1)}\|^2 - 0 \le \frac{LC^2}{t^2} . \end{aligned} \tag{60}$$

Based on (31), the gradient of $\mathcal{F}^{(t)}_{\text{mod}} - \mathcal{F}^{(t+1)}_{\text{mod}}$ is bounded by

$$\begin{aligned} &\left\|\nabla\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}) - \nabla\mathcal{F}^{(t+1)}_{\text{mod}}(\mathbf{d})\right\| \\ &\quad = \left\|\nabla\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}) - \frac{\alpha^{(t+1)}\Lambda^{(t)}}{\Lambda^{(t+1)}}\nabla\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}) - \frac{1}{\Lambda^{(t+1)}}\nabla f^{(t+1)}(\mathbf{d})\right\| \\ &\quad \le \frac{1}{\Lambda^{(t+1)}}\left\|\nabla\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d})\right\| + \frac{1}{\Lambda^{(t+1)}}\left\|\nabla f^{(t+1)}(\mathbf{d})\right\| \le C_0/(\Lambda^{(t+1)}) \le C_1/t \end{aligned}$$

for some constant $C_1 > 0$. The second inequality follows from $\mathbf{d} \in \mathrm{C}$ (the compact support of $\mathbf{d}$), Assumption 1 (the compact support of $\mathbf{s}$), and (57) (boundedness of X). The last inequality is derived by the follows:

$$\frac{1}{\Lambda^{(t+1)}} = \frac{(t+1)^p}{\sum_{\tau=1}^{(t+1)}\tau^p} \le \frac{(t+1)^p}{\int_0^{(t+1)}\tau^p \mathrm{d}\tau} = \frac{p}{t+1} .$$

Then $\mathcal{F}^{(t)}_{\text{mod}} - \mathcal{F}^{(t+1)}_{\text{mod}}$ is a Lipschitz continuous function with $L = C_1/t$, which implies

$$\mathbf{T}_1 + \mathbf{T}_3 \le \frac{C_1}{t}r^{(t)} .$$

Therefore,

$$\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t)}) \le \frac{C_1}{t}r^{(t)} + \frac{LC^2}{t^2} . \tag{61}$$

Combining (59) and (61), we have

$$\mu\left(r^{(t)} - \frac{C}{t}\right)^2 - \frac{LC^2}{t^2} \le \frac{C_1}{t}r^{(t)} + \frac{LC^2}{t^2} ,$$

which implies

$$(r^{(t)})^2 - \frac{2C+C_1}{t} r^{(t)} \leq \frac{2LC^2}{\mu t^2} \,.$$

This can be written more neatly as

$$(r^{(t)})^2 - 2\frac{C_2}{t} r^{(t)} \leq \frac{C_3}{t^2} \quad \text{for some } C_2 > 0, C_3 > 0 \,.$$

Finally, $r^{(t)}$ is bounded by $r^{(t)} \leq (C_2 + \sqrt{C_2^2 + C_3})/t$, and (45) is proved. ■

E.3. Proof of Proposition 1.

Proof. Define a sequence of random variables $Y_i = i^p Z_i$. Their expectations and variances are $\mu_i = i^p \mu$ and $\sigma_i^2 = i^{2p}\sigma^2$, respectively. Now we apply the Lyapunov central limit theorem on the stochastic sequence $\{Y_i\}$. First, we check the Lyapunov condition [6]. Let

$$s_n^2 = \sum_{i=1}^n \sigma_i^2 = \sum_{i=1}^n i^{2p}\sigma^2 = \Theta(n^{2p+1});$$

then we have

$$\frac{1}{s_n^{2+\delta}} \sum_{i=1}^n \mathbb{E}\left[|Y_i - \mu_i|^{2+\delta}\right] \leq \frac{1}{s_n^{2+\delta}} \sum_{i=1}^n (i^p\sigma)^{2+\delta} = \mathcal{O}\left(\frac{n^{2p+1+\delta p}}{n^{2p+1+\delta p+\delta/2}}\right) = \mathcal{O}(n^{-\delta/2}) \,. \tag{62}$$

The Lyapunov condition is satisfied, so, by the Lyapunov central limit theorem, we have $\frac{1}{s_n}\sum_{i=1}^n (Y_i - \mu_i) \xrightarrow{\text{d}} N(0,1)$. Furthermore, the definition of $\hat{Z}^n_{\text{mod}}$ indicates

$$\begin{aligned}\frac{1}{s_n}\sum_{i=1}^n (Y_i - \mu_i) &= \frac{1}{s_n}\sum_{i=1}^n i^p(Z_i - \mu) = \frac{\sum_{i=1}^n i^p}{\sqrt{\sum_{i=1}^n i^{2p}}\sigma}\left(\frac{1}{\sum_{i=1}^n i^p}\sum_{i=1}^n i^p(Z_i-\mu)\right) \\ &= \frac{\sum_{i=1}^n i^p}{\sqrt{\sum_{i=1}^n i^{2p}}\sigma}(\hat{Z}^n_{\text{mod}} - \mu) \,.\end{aligned}$$

Given the inequalities

$$\sum_{i=1}^n i^p < \int_1^{n+1} s^p \mathrm{d}s < \frac{1}{p+1}(n+1)^{p+1} \,, \quad \sum_{i=1}^n i^p > \int_0^n s^p \mathrm{d}s = \frac{1}{p+1}(n)^{p+1} \,,$$

we have

$$\begin{aligned}\frac{1}{s_n}\sum_{i=1}^n (Y_i - \mu_i) &\leq \left(1 + \frac{1}{n}\right)^{p+1} \frac{1}{\sigma}\frac{\sqrt{2p+1}}{p+1}\sqrt{n}(\hat{Z}^n_{\text{mod}} - \mu) \,, \\ \frac{1}{s_n}\sum_{i=1}^n (Y_i - \mu_i) &\geq \left(1 + \frac{1}{n}\right)^{-(p+1)} \frac{1}{\sigma}\frac{\sqrt{2p+1}}{p+1}\sqrt{n}(\hat{Z}^n_{\text{mod}} - \mu) \,.\end{aligned}$$

Then (39) is obtained by $\frac{1}{s_n}\sum_{i=1}^n (Y_i - \mu_i) \xrightarrow{\text{d}} N(0,1)$ and $(1 + 1/n) \to 1$.

The formula $\mathrm{Var}(X) = \mathbb{E}X^2 - (\mathbb{E}X)^2 \geq 0$ implies

$$\left(\mathbb{E}\left[\sqrt{n}|\hat{Z}^n_{\mathrm{mod}} - \mu|\right]\right)^2 \leq \mathbb{E}\left[n|\hat{Z}^n_{\mathrm{mod}} - \mu|^2\right] .$$

By the independence of different Z_i, we have

$$\mathbb{E}\left[n|\hat{Z}^n_{\mathrm{mod}} - \mu|^2\right] = \frac{n}{(\sum_{i=1}^n i^p)^2}\sum_{i=1}^n \mathbb{E}\left[i^{2p}|Z_i - \mu|^2\right] \leq \frac{(p+1)^2}{2p+1}B^2 ,$$

where B is the upper bound of Z_i as Z_i is compact supported, and (40) is proved. ■

E.4. Proof of Proposition 2.

Proof. First, we fix $\mathbf{d} \in \mathrm{C}$. Let $i \to \tau, n \to t, Z_i \to f(\mathbf{d};\mathbf{s}^{(\tau)})$; then, by Proposition 1, we have

$$\mathbb{E}\left[\sqrt{t}|F(\mathbf{d}) - F^{(t)}_{\mathrm{mod}}(\mathbf{d})|\right] \leq \frac{p+1}{\sqrt{2p+1}}B \quad \forall t \in \{1,2,\ldots\}$$

for some $B > 0$ for fixed $\mathbf{d}$. Since F and $F^{(t)}_{\mathrm{mod}}$ are continuously differentiable and have uniformly bounded derivatives (57), we have that $\mathbb{E}[\sqrt{t}|F(\mathbf{d}) - F^{(t)}_{\mathrm{mod}}(\mathbf{d})|]$ is uniformly continuous w.r.t $\mathbf{d}$ on a compact set C. Thus, the boundedness of $\mathbb{E}[\sqrt{t}|F(\mathbf{d}) - F^{(t)}_{\mathrm{mod}}(\mathbf{d})|]$ on each $\mathbf{d}$ implies the boundedness for all $\mathbf{d} \in \mathrm{C}$. Inequality (42) is proved. Taking $p \to 0$, we have (41). ■

E.5. Proof of Theorem 1.

Proof. Let $u^{(t)} = \mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)})$. Inspired by the proof of Proposition 3 in [38], we will show that $u^{(t)}$ is a "quasi-martingale" [19]:

$$\begin{aligned} &u^{(t+1)} - u^{(t)} \\ &\quad = \mathcal{F}^{(t+1)}_{\mathrm{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)}) \\ &\quad = \underbrace{\mathcal{F}^{(t+1)}_{\mathrm{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t+1)}_{\mathrm{mod}}(\mathbf{d}^{(t)})}_{\mathbf{T}_2} + \underbrace{\mathcal{F}^{(t+1)}_{\mathrm{mod}}(\mathbf{d}^{(t)}) - \mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)})}_{\mathbf{T}_4} . \end{aligned}$$

The bound of $\mathbf{T}_2$ is given by (60). Furthermore, definition (27) tells us $f^{(t+1)}(\mathbf{d}^{(t)}) = f(\mathbf{d}^{(t)};\mathbf{s}^{(t+1)})$, which implies

$$\begin{aligned} \mathbf{T}_4 &= \mathcal{F}^{(t+1)}_{\mathrm{mod}}(\mathbf{d}^{(t)}) - \mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)}) \\ &= \left(\frac{1}{\Lambda^{(t+1)}}f(\mathbf{d}^{(t)};\mathbf{s}^{(t+1)}) + \frac{\alpha^{(t+1)}\Lambda^{(t)}}{\Lambda^{(t+1)}}\mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)})\right) - \mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)}) \\ &= \frac{f(\mathbf{d}^{(t)};\mathbf{s}^{(t+1)}) - F^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)})}{\Lambda^{(t+1)}} + \frac{F^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)}) - \mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)})}{\Lambda^{(t+1)}} . \end{aligned}$$

By the definitions of f (12) and F (36), we have $F^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)}) \leq \mathcal{F}^{(t)}_{\mathrm{mod}}(\mathbf{d}^{(t)})$. Define $\mathcal{G}^t$ as all the previous information: $\mathcal{G}^t \triangleq \{\mathbf{x}^{(\tau)},\mathbf{s}^{(\tau)},\mathbf{d}^{(\tau)}\}_{\tau=1}^t$. Thus, taking conditional expectation, we obtain

$$\mathbb{E}[\mathbf{T}_4|\mathcal{G}^t] \leq \frac{1}{\Lambda^{(t+1)}} \left(\mathbb{E}[f(\mathbf{d}^{(t)}; \mathbf{s}^{(t+1)})|\mathcal{G}^t] - F_{\text{mod}}^{(t)}(\mathbf{d}^{(t)})\right) = \frac{1}{\Lambda^{(t+1)}} \left(F(\mathbf{d}^{(t)}) - F_{\text{mod}}^{(t)}(\mathbf{d}^{(t)})\right) .$$

Therefore, the positive part of $\mathbb{E}[\mathbf{T}_4|\mathcal{G}^t]$ is bounded by

$$\mathbb{E}[\mathbf{T}_4|\mathcal{G}^t]^+ \leq \frac{1}{\Lambda^{(t+1)}} \|F - F_{\text{mod}}^{(t)}\|_\infty = \mathcal{O}\left(\frac{1}{t^{3/2}}\right) ,$$

where the second inequality follows from (42). Given the bound of $\mathbf{T}_2$ (60) and $\mathbf{T}_4$, we have

$$\sum_{t=1}^{\infty} \mathbb{E}[\mathbb{E}[u^{(t+1)} - u^{(t)}|\mathcal{G}^t]^+] \leq \sum_{t=1}^{\infty} \left(\mathcal{O}\left(\frac{1}{t^{3/2}}\right) + \mathcal{O}\left(\frac{1}{t^2}\right)\right) < +\infty ,$$

which implies that $u^{(t+1)}$ generated by Algorithm 2 is a quasi-martingale. Thus, by results in [7, section 4.4] or [38, Theorem 6], we have that $u^{(t)}$ converges almost surely.

For the proofs of 2, 3, and 4, using the results in Propositions 42 and in this paper, following the same proof line of Propositions 3 and 4 in [38], we can obtain the results in 2, 3, and 4. ∎

Acknowledgment. The authors thank the two anonymous reviewers for their careful reading and valuable comments that helped improve the final version of this manuscript.

REFERENCES

[1] M. Aharon and M. Elad, *Sparse and redundant modeling of image content using an image-signature-dictionary*, SIAM J. Imaging Sci., 1 (2008), pp. 228–247, https://doi.org/10.1137/07070156X.

[2] M. Aharon, M. Elad, and A. Bruckstein, *K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation*, IEEE Trans. Signal Process., 54 (2006), pp. 4311–4322, https://doi.org/10.1109/TSP.2006.881199.

[3] L. V. Ahlfors, *Complex Analysis: An Introduction to the Theory of Analytic Functions of One Complex Variable*, McGraw-Hill, New York, 1979.

[4] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202, https://doi.org/10.1137/080716542.

[5] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Lexington, MA, 1999.

[6] P. Billingsley, *Probability and Measure*, John Wiley & Sons, New York, 2008.

[7] L. Bottou, *On-line learning and stochastic approximations*, in On-Line Learning in Neural Networks, D. Saad, ed., Cambridge University Press, Cambridge, 1999, pp. 9–42, https://doi.org/10.1017/CBO9780511569920.003.

[8] H. Bristow, A. Eriksson, and S. Lucey, *Fast convolutional sparse coding*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 391–398, https://doi.org/10.1109/CVPR.2013.57.

[9] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., 20 (1998), pp. 33–61, https://doi.org/10.1137/S1064827596304010.

[10] I. Y. Chun and J. A. Fessler, *Convolutional dictionary learning: Acceleration and convergence*, IEEE Trans. Image Process., (2017), https://doi.org/10.1109/TIP.2017.2761545.

[11] J. M. Danskin, *The theory of max-min, with applications*, SIAM J. Appl. Math., 14 (1966), pp. 641–664.

[12] D. Davis and W. Yin, *Convergence rate analysis of several splitting schemes*, in Splitting Methods in Communication, Imaging, Science, and Engineering, Springer, New York, 2016, pp. 115–163, https://doi.org/10.1007/978-3-319-41589-5_4.

[13] K. Degraux, U. S. Kamilov, P. T. Boufounos, and D. Liu, *Online convolutional dictionary learning for multimodal imaging*, in Proceedings of the IEEE International Conference on Image Processing (ICIP), 2017, arXiv:1706.04256.

[14] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al., *Least angle regression*, Anna. Statist., 32 (2004), pp. 407–499.
[15] E. M. Eksioglu, *Online dictionary learning algorithm with periodic updates and its application to image denoising*, Expert Syste. Appl., 41 (2014), pp. 3682–3690, https://doi.org/10.1016/j.eswa.2013.11.036.
[16] M. Elad and M. Aharon, *Image denoising via sparse and redundant representations over learned dictionaries*, IEEE Trans. Image Process., 15 (2006), pp. 3736–3745, https://doi.org/10.1109/TIP.2006.881969.
[17] K. Engan, S. O. Aase, and J. H. Husøy, *Method of optimal directions for frame design*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 5 (1999), pp. 2443–2446, https://doi.org/10.1109/ICASSP.1999.760624.
[18] K. Engan, B. D. Rao, and K. Kreutz-Delgado, *Frame design using FOCUSS with method of optimal directions (MOD)*, in Proceedings of the Norwegian Signal Processing Symposium (NORSIG), 1999, pp. 65–69.
[19] D. L. Fisk, *Quasi-martingales*, Trans. Ameri. Math. Soc., 120 (1965), pp. 369–389.
[20] J.-J. Fuchs, *Recovery of exact sparse representations in the presence of bounded noise*, IEEE Trans. Inform. Theory, 51 (2005), pp. 3601–3608, https://doi.org/10.1109/ICASSP.2004.1326312.
[21] W. Gao, J. Chen, C. Richard, and J. Huang, *Online dictionary learning for kernel LMS*, IEEE Trans. Signal Process., 62 (2014), pp. 2765–2777, https://doi.org/10.1109/TSP.2014.2318132.
[22] C. Garcia-Cardona and B. Wohlberg, *Subproblem coupling in convolutional dictionary learning*, in Proceedings of the IEEE International Conference on Image Processing (ICIP), 2017.
[23] S. Ghadimi and G. Lan, *Stochastic first- and zeroth-order methods for nonconvex stochastic programming*, SIAM J. Optim., 23 (2013), pp. 2341–2368, https://doi.org/10.1137/120880811.
[24] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, *Convolutional sparse coding for image super-resolution*, in Proceedings of the International Conference on Computer Vision (ICCV), 2015, https://doi.org/10.1109/ICCV.2015.212.
[25] F. Heide, W. Heidrich, and G. Wetzstein, *Fast and flexible convolutional sparse coding*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 5135–5143, https://doi.org/10.1109/CVPR.2015.7299149.
[26] K. Huang and S. Aviyente, *Sparse representation for signal classification*, in Advances in Neural Information Processing Systems, 2007, pp. 609–616.
[27] M. J. Huiskes, B. Thomee, and M. S. Lew, *New trends and ideas in visual concept detection: The MIR Flickr retrieval evaluation initiative*, in Proceedings of the International Conference on Multimedia Information Retrieval (MIR), ACM, New York, 2010, pp. 527–536, https://doi.org/10.1145/1743384.1743475.
[28] R. Jenatton, J. Mairal, F. R. Bach, and G. R. Obozinski, *Proximal methods for sparse hierarchical dictionary learning*, in Proceedings of the 27th International Conference on Machine Learning (ICML), 2010, pp. 487–494.
[29] R. M. Johnstone, C. R. Johnson, R. R. Bitmead, and B. D. O. Anderson, *Exponential convergence of recursive least squares with exponential forgetting factor*, Systems Control Lett., 2 (1982), pp. 77–82.
[30] S. P. Kasiviswanathan, H. Wang, A. Banerjee, and P. Melville, *Online l_1-dictionary learning with application to novel document detection*, in Advances in Neural Information Processing Systems, 2012, pp. 2258–2266.
[31] M. S. Lewicki and T. J. Sejnowski, *Coding time-varying signals using sparse, shift-invariant representations*, in Advances in Neural Information Processing Systems, Vol. 11, M. J. Kearns, S. A. Solla, and D. A. Cohn, eds., 1999, pp. 730–736.
[32] J. Liu, C. Garcia-Cardona, B. Wohlberg, and W. Yin, *Online convolutional dictionary learning*, in Proceedings of the IEEE International Conference on Image Processing (ICIP), 2017, arXiv:1706.09563.
[33] Y. Liu, X. Chen, R. K. Ward, and Z. J. Wang, *Image fusion with convolutional sparse representation*, IEEE Signal Proc. Lett., 2016, https://doi.org/10.1109/lsp.2016.2618776.
[34] C. Lu, J. Shi, and J. Jia, *Online robust dictionary learning*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 415–422, https://doi.org/10.1109/CVPR.2013.60.

[35] J. Mairal, F. Bach, and J. Ponce, *Task-driven dictionary learning*, IEEE Trans. Pattern Anal. Mach. Intel., 34 (2012), pp. 791–804, https://doi.org/10.1109/TPAMI.2011.156.

[36] J. Mairal, F. Bach, and J. Ponce, *Sparse modeling for image and vision processing*, Found. Trends Comput. Graphics Vis., 8 (2014), pp. 85–283, https://doi.org/10.1561/0600000058.

[37] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, *Online dictionary learning for sparse coding*, in Proceedings of the 26th Annual International Conference on Machine Learning (ICML), ACM, New York, 2009, pp. 689–696, https://doi.org/10.1145/1553374.1553463.

[38] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, *Online learning for matrix factorization and sparse coding*, J. Mach. Learn. Res., 11 (2010), pp. 19–60.

[39] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, New York, 1999.

[40] V. Papyan, Y. Romano, and M. Elad, *Convolutional neural networks analyzed via convolutional sparse coding*, J. Mach. Learn. Res., 18 (2017), pp. 1–52.

[41] V. Papyan, Y. Romano, J. Sulam, and M. Elad, *Convolutional Dictionary Learning via Local Processing*, preprint, arXiv:1705.03239, 2017.

[42] V. Papyan, J. Sulam, and M. Elad, *Working locally thinking globally: Theoretical guarantees for convolutional sparse coding*, IEEE Trans. Signal Process., 65 (2017), pp. 5687–5701, https://doi.org/10.1109/TSP.2017.2733447.

[43] T. M. Quan and W.-K. Jeong, *Compressed sensing reconstruction of dynamic contrast enhanced MRI using GPU-accelerated convolutional sparse coding*, in IEEE International Symposium on Biomedical Imaging (ISBI), 2016, pp. 518–521, https://doi.org/10.1109/ISBI.2016.7493321.

[44] O. Rippel, J. Snoek, and R. P. Adams, *Spectral representations for convolutional neural networks*, in Advances in Neural Information Processing Systems, 2015, pp. 2449–2457.

[45] R. Rubinstein, A. M. Bruckstein, and M. Elad, *Dictionaries for sparse representation modeling*, Proc. IEEE, 98 (2010), pp. 1045–1057, https://doi.org/10.1109/JPROC.2010.2040551.

[46] L. Rudin, S. J. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D Nonlin. Phenom., 60 (1992), pp. 259–268, https://doi.org/10.1016/0167-2789(92)90242-f.

[47] K. Skretting and K. Engan, *Recursive least squares dictionary learning algorithm*, IEEE Trans. Signal Process., 58 (2010), pp. 2121–2130, https://doi.org/10.1109/tsp.2010.2040671.

[48] K. Slavakis and G. B. Giannakis, *Online dictionary learning from big data using accelerated stochastic approximation algorithms*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2014, pp. 16–20, https://doi.org/10.1109/ICASSP.2014.6853549.

[49] L. Sorber, M. V. Barel, and L. D. Lathauwer, *Unconstrained optimization of real functions in complex variables*, SIAM J. Optim., 22 (2012), pp. 879–898, https://doi.org/10.1137/110832124.

[50] J. Sulam, V. Papyan, Y. Romano, and M. Elad, *Multi-Layer Convolutional Sparse Modeling: Pursuit and Dictionary Learning*, preprint, arXiv:1708.08705, 2017.

[51] Z. Szabó, B. Póczos, and A. Lőrincz, *Online group-structured dictionary learning*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 2865–2872, https://doi.org/10.1109/CVPR.2011.5995712.

[52] Y. Tang, Y.-B. Yang, and Y. Gao, *Self-paced dictionary learning for image classification*, in Proceedings of the 20th ACM International Conference on Multimedia, 2012, pp. 833–836, https://doi.org/10.1145/2393347.2396324.

[53] A. W. Van der Vaart, *Asymptotic Statistics*, Vol. 3, Cambridge University Press, Cambridge, 2000.

[54] M. Šorel and F. Šroubek, *Fast convolutional sparse coding using matrix inversion lemma*, Digi. Signal Process., 2016, https://doi.org/10.1016/j.dsp.2016.04.012.

[55] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, *Locality-constrained linear coding for image classification*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3360–3367, https://doi.org/10.1109/CVPR.2010.5540018.

[56] N. Wang, J. Wang, and D.-Y. Yeung, *Online robust non-negative dictionary learning for visual tracking*, in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2013, pp. 657–664, https://doi.org/10.1109/ICCV.2013.87.

[57] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, *Online Convolutional Sparse Coding*, preprint, arXiv:1706.06972, 2017.

[58] B. Wohlberg, *Efficient convolutional sparse coding*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2014, pp. 7173–7177, https://doi.org/10.1109/ICASSP.2014.6854992.

[59] B. Wohlberg, *Boundary handling for convolutional sparse representations*, in Proceedings of the IEEE Conference Image Processing (ICIP), Phoenix, AZ, 2016, pp. 1833–1837, https://doi.org/10.1109/ICIP.2016.7532675.

[60] B. Wohlberg, *Convolutional sparse representation of color images*, in Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), Santa Fe, NM, 2016, pp. 57–60, https://doi.org/10.1109/SSIAI.2016.7459174.

[61] B. Wohlberg, *Convolutional sparse representations as an image model for impulse noise restoration*, in Proceedings of the IEEE Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), Bordeaux, France, 2016, https://doi.org/10.1109/IVMSPW.2016.7528229.

[62] B. Wohlberg, *Efficient algorithms for convolutional sparse representations*, IEEE Trans. Image Process., 25 (2016), pp. 301–315, https://doi.org/10.1109/TIP.2015.2495260.

[63] B. Wohlberg, *SParse Optimization Research COde (SPORCO)*, software library available from http://purl.org/brendt/software/sporco, 2016.

[64] B. Wohlberg, *ADMM Penalty Parameter Selection by Residual Balancing*, preprint, arXiv:1704.06209, 2017.

[65] B. Wohlberg, *SPORCO: A Python package for standard and convolutional sparse representations*, in Proceedings of the 15th Python in Science Conference, Austin, TX, 2017, pp. 1–8, https://doi.org/10.25080/shinma-7f4c6e7-001.

[66] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, *Sparse representation for computer vision and pattern recognition*, Proc. IEEE, 98 (2010), pp. 1031–1044, https://doi.org/10.1109/JPROC.2010.2044470.

[67] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, *Robust face recognition via sparse representation*, IEEE Trans. Pattern Anal. Mach. Intell., 31 (2009), pp. 210–227, https://doi.org/10.1109/TPAMI.2008.79.

[68] Y. Xu and W. Yin, *A fast patch-dictionary method for whole image recovery*, Inverse Probl. Imaging, 10 (2016), pp. 563–583, https://doi.org/10.3934/ipi.2016012.

[69] J. Yang, J. Wright, T. S. Huang, and Y. Ma, *Image super-resolution via sparse representation*, IEEE Trans. Image Process., 19 (2010), pp. 2861–2873, https://doi.org/10.1109/TIP.2010.2050625.

[70] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, *Deconvolutional networks*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 2528–2535, https://doi.org/10.1109/cvpr.2010.5539957.

[71] G. Zhang, Z. Jiang, and L. S. Davis, *Online semi-supervised discriminative dictionary learning for sparse representation*, in Proceedings of the Asian Conference on Computer Vision (ACCV), 2012, pp. 259–273, https://doi.org/10.1007/978-3-642-37331-2_20.

[72] H. Zhang and V. M. Patel, *Convolutional sparse coding-based image decomposition*, in Proceedings of the British Machine Vision Conference (BMVC), York, UK, 2016.

[73] H. Zhang and V. M. Patel, *Convolutional sparse and low-rank coding-based rain streak removal*, in Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), 2017, https://doi.org/10.1109/WACV.2017.145.

[74] S. Zhang, S. Kasiviswanathan, P. C. Yuen, and M. Harandi, *Online dictionary learning on symmetric positive definite manifolds with vision applications*, in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2015, pp. 3165–3173.

[75] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, *A survey of sparse representation: Algorithms and applications*, IEEE Access, 3 (2015), pp. 490–530, https://doi.org/10.1109/ACCESS.2015.2430359.