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 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, and 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, stochastic gradient descent, recursive least squares

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 typically overcomplete basis. For example, we can approximate an an N-dimensional signal $\mathbf{s} \in \mathbb{R}^N$ as

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

where $D = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_M] \in \mathbb{R}^{N \times M}$ is the overcomplete basis or *dictionary* with M *atoms* and $\mathbf{x} = [x_1, x_2, \cdots, x_M]^T \in \mathbb{R}^M$ 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]

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

Sparse representations have been widely used in, for example, denoising [15, 35], super-resolution [62, 68], classification [60], and face recognition [59]. 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 [41] such as wavelets [38] or Discrete Cosine Transform (DCT) [25], but learned dictionaries can provide better performance [2, 15].

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:

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

* **Funding:** This research was supported by the U.S. Department of Energy through the LANL/LDRD Program.

†Department of Mathematics, UCLA, Los Angeles, CA 90095

‡CCS Division, Los Alamos, NM 87545, USA

§Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA. Email: brendt@lanl.gov, Tel: +1 505 667 6886, Fax: +1 505 665 5757

¶Department of Mathematics, UCLA, Los Angeles, CA 90095

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

Batch dictionary learning methods sample a batch of training signals $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_K\}$ before training, and minimize the objective function:

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

In the training process, batch learning methods such as FOCUSS/MOD [17, 16], K-SVD [2], and the work for global samples [61] require simultaneous access to all the training samples.

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} step of training. The framework of online dictionary learning is:

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

where BPDN refers to model (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, 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. Different kinds of D-update methods have been proposed in the literature. Online dictionary learning algorithms can be classified into two classes:

Class I: first-order algorithms [50, 34, 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) [43, 14], Iterative Reweighted Least Squares (IRLS) [33, 51], Kernel RLS [20], second-order Stochastic Approximation (SA) [36, 46, 64, 44, 67, 29], 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 the surrogate function. These surrogate functions involve both first-order and second-order information, 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 Sec. 3 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)* [30, 63] [55, Sec. II], a highly structured sparse representation model, has recently attracted increasing attention for a variety of imaging inverse problems [23, 32, 65, 39, 54, 66]. CSC aims to represent a given signal $\mathbf{s} \in \mathbb{R}^N$ as a sum of convolutions,

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

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

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

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 following function:

$$\begin{aligned} \arg\min_{\{\mathbf{d}_m\},\{\mathbf{x}_{k,m}\}} & \frac{1}{2}\sum_{k=1}^{K}\Big\| \sum_{m=1}^{M} \mathbf{d}_m * \mathbf{x}_{k,m} - \mathbf{s}_k \Big\|_2^2 + \lambda \sum_{k=1}^{K}\sum_{m=1}^{M} \|\mathbf{x}_{k,m}\|_1 \\ & \text{subject to } \|\mathbf{d}_m\|_2 \leq 1, \ \forall m \in \{1,\ldots,M\} \ , \end{aligned} \tag{8}$$

where the coefficient maps $\mathbf{x}_{k,m}$, $k \in \{1,\ldots,K\}$, $m \in \{1,\ldots,M\}$, represent $\mathbf{s}_k$, and the norm constraint avoids the scaling ambiguity between $\mathbf{d}_m$ and $\mathbf{x}_{k,m}$. Most current CDL algorithms [8, 24, 23, 55, 49, 53, 21] 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.

Complexity of batch CDL. When K is large, the $\mathbf{d}_m$ update subproblem is computationally expensive. Based on the state-of-the-art results [49, 21], the single step complexity and memory usage are both $\mathcal{O}(KNM)$. With the typical values $K = 40, N = 256 \times 256, M = 64$, $KNM \approx 1.7 \times 10^8$, 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:

1. An efficient first-order online CDL method (Algorithm 5).
2. An efficient second-order online CDL method (Algorithm 4) that improves the algorithm proposed in [31], and its convergence proof.
3. An online CDL method that is able to learn dictionaries from partially masked training set (Algorithm 6).
4. An analysis of the forgetting factor[1] used in Algorithm 4.
5. An analysis of the stopping condition in the D-update.
6. An analysis of circular boundary conditions on dictionary learning.

Relationship with other works. Recently, two other works on online CDL [12, 52] have appeared. Both of them study second-order SA methods. They use the same framework from [36] but different methods to update D. [12] uses projected coordinate descent and [52] uses iterated Sherman-Morrison. Our previous work [31] uses frequency-domain FISTA to update D, uses a *forgetting factor* technique, inspired by [43, 37], to correct the surrogate function, and uses "region-sampling" to reduce the memory cost. In this paper, the second-order SA algorithm 4 improves over the

[1] This technique is used in previous works [43, 37, 46, 44], but not theoretically analyzed.

similar algorithm in [31] by applying two techniques: an improved stopping condition of FISTA and image-splitting. The former technique largely reduces the inner-loop iterations of D-update; the latter, compared with "region-sampling", fully utilizes all the information in the training set. With these techniques, Algorithm 4 converges faster.

The method in [12] works on a variant of problem (8), and is therefore not comparable with our methods, and [52], which appeared while we were completing our work, studies the same problem (8) with ours. The algorithm in it is expected to be comparable with our previous work [31] because they share the same framework. Our Algorithm 5, however, uses a different framework. Compared to [31], it runs faster while also using less memory.

2. Preliminaries. Here we introduce our notation. The signal is denoted by $\mathbf{s} \in \mathbb{R}^N$, and the dictionaries by $\mathbf{d} = (\mathbf{d}_1\ \mathbf{d}_2\ \dots\ \mathbf{d}_M)^T \in \mathbb{R}^{MD}$, where the dictionary kernels (or filters) $\mathbf{d}_m \in \mathbb{R}^D$. Coefficient maps are denoted by $\mathbf{x} = (\mathbf{x}_1\ \mathbf{x}_2\ \dots\ \mathbf{x}_M)^T \in \mathbb{R}^{MN}$, where $\mathbf{x} \in \mathbb{R}^N$ 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 $X \triangleq \left(X_1\ X_2 \cdots X_M\right)$. Then, we have

$$X\mathbf{d} \triangleq \sum_{m=1}^{M} X_m\mathbf{d}_m = \sum_{m=1}^{M} \mathbf{d}_m * \mathbf{x}_m \approx \mathbf{s}\ . \tag{9}$$

Hence, $X \in \mathbb{R}^{MD} \to \mathbb{R}^N$, 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. 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

$$l(\mathbf{d}, \mathbf{x}; \mathbf{s}) = (1/2)\left\|X\mathbf{d} - \mathbf{s}\right\|_2^2\ . \tag{10}$$

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

$$f(\mathbf{d}; \mathbf{s}) \triangleq \min_{\mathbf{x}} \left\{ l(\mathbf{d}, \mathbf{x}; \mathbf{s}) + \lambda \left\|\mathbf{x}\right\|_1 \right\}, \tag{11}$$

$$\mathbf{x}^*(\mathbf{d}; \mathbf{s}) \triangleq \arg\min_{\mathbf{x}} \left\{ l(\mathbf{d}, \mathbf{x}; \mathbf{s}) + \lambda \left\|\mathbf{x}\right\|_1 \right\}. \tag{12}$$

The online CDL problem can be formulated as

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

where C is the constraint set of $\mathrm{C} = \{\mathbf{d} \mid \left\|\mathbf{d}_m\right\|^2 \le 1, \forall m\}$ and $\iota_{\mathrm{C}}(\cdot)$ is the indicator function[2] of set C.

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

2.2. Mathematical derivation: computing in the frequency domain. Before introducing our algorithms for (13), we consider a basic problem and a frequency domain gradient descent in this section, which are also used in Sections 3 and 4.

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

$$\min_{\mathbf{d}\in\mathbb{R}^{MD}} l(\mathbf{d},\mathbf{x};\mathbf{s}) + \iota_{\mathrm{C}}(\mathbf{d}) \;, \tag{14}$$

for which we can apply projected gradient descent (GD) [5]:

$$\mathbf{d}^{(t)} = \mathrm{Proj}_{\mathrm{C}}\Big(\mathbf{d}^{(t-1)} - \eta^{(t)}\nabla l(\mathbf{d}^{(t-1)},\mathbf{x};\mathbf{s})\Big) \;, \tag{15}$$

where (t) is the iteration index and the gradient of l is given by

$$\nabla l(\mathbf{d},\mathbf{x};\mathbf{s}) = X^T\big(X\mathbf{d}-\mathbf{s}\big) \;. \tag{16}$$

Since X is a linear operator from $\mathbb{R}^{MD}$ to $\mathbb{R}^N$, the cost of directly computing (16) is $\mathcal{O}(NMD)$.

It is well known that convolving two signals of the same size corresponds to the pointwise multiplication of their frequency representations. Our method below for solving (14) takes advantage of this property. First, we zero-pad each $\mathbf{d}_m$ from $\mathbb{R}^D$ to $\mathbb{R}^N$ to match the size of $\mathbf{s}$. Then the basic problem can be written as

$$\min_{\mathbf{d}\in\mathbb{R}^{MN}} l(\mathbf{d},\mathbf{x};\mathbf{s}) + \iota_{C_{\mathrm{PN}}}(\mathbf{d}) \;, \tag{17}$$

where the set C_{PN} is defined as

$$C_{\mathrm{PN}} \triangleq \{\mathbf{d}_m \in \mathbb{R}^N : (I-P)\mathbf{d}_m = 0, \|\mathbf{d}_m\|^2 \le 1\} \;. \tag{18}$$

Operator P preserves part of $\mathbf{d}_m$ and masks the remaining part to zeros. Projected GD for (17) is

$$\mathbf{d}^{(t)} = \mathrm{Proj}_{C_{\mathrm{PN}}}\Big(\mathbf{d}^{(t-1)} - \eta^{(t)}\nabla l(\mathbf{d}^{(t-1)},\mathbf{x};\mathbf{s})\Big) \;. \tag{19}$$

Then, using the Plancheral formula, we can write the loss function l as

$$l(\mathbf{d},\mathbf{x};\mathbf{s}) = \underbrace{\Big\|\sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\Big\|_2^2}_{\|X\mathbf{d}-\mathbf{s}\|^2} = \underbrace{\Big\|\sum_m \hat{\mathbf{d}}_m \odot \hat{\mathbf{x}}_m - \hat{\mathbf{s}}\Big\|_2^2}_{\|\hat{X}\hat{\mathbf{d}}-\hat{\mathbf{s}}\|^2} \tag{20}$$

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 $\hat{X} = \big(\hat{X}_1\ \hat{X}_2 \cdots \hat{X}_M\big)$ is a linear operator. Since each $\hat{X}_m$ is diagonal, $\hat{X}$ is much cheaper to compute than X. Define the loss function in the frequency domain

$$\hat{l}(\hat{\mathbf{d}},\hat{\mathbf{x}};\hat{\mathbf{s}}) = (1/2)\big\|\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}\big\|^2 \;, \tag{21}$$

which is a real valued function defined in the complex domain. The Cauchy-Riemann condition [3] implies that (21) is not differentiable unless it is constant. However, the *conjugate cogradient* below [45] exists and can be used for optimization:

$$\nabla\hat{l}(\hat{\mathbf{d}},\hat{\mathbf{x}};\hat{\mathbf{s}}) \triangleq \hat{X}^H(\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}) \;. \tag{22}$$

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 $\mathcal{O}(NMD)$. Based on (22), we have the following modified gradient descent:

$$\mathbf{d}^{(t)} = \mathrm{Proj}_{C_{\mathrm{PN}}}\left(\mathrm{IFFT}\Big(\hat{\mathbf{d}}^{(t-1)} - \eta^{(t)}\nabla\hat{l}(\hat{\mathbf{d}}^{(t-1)}, \hat{\mathbf{x}}; \hat{\mathbf{s}})\Big)\right) . \tag{23}$$

To compute (23), 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 (23), the iterate $\mathbf{d}$ 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 (19) and (23). We can prove:

$$\hat{X}^H(\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}) = \mathrm{FFT}\big(X^T(X\mathbf{d} - \mathbf{s})\big) , \quad \forall \mathbf{x}, \mathbf{d}, \mathbf{s} , \tag{24}$$

which means that the conjugate cogradient of $\hat{l}$ is equivalent with the gradient of l. Thus, modified GD (23) coincides with standard GD (19) using conjugate cogradient.

A proof of (24) given in Appendix B. A similar result is also given in [40] under the assumption of "conjugate symmetry".

3. Second-order SA method: surrogate function approaches. Now we consider the CDL problem (13) when the training signals $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \cdots, \mathbf{s}^{(t)}, \cdots$ arrive in a streaming fashion.

3.1. Algorithm 1: basic surrogate function. Second-order stochastic approximation based algorithms are popular in dictionary learning [36, 43, 46, 64, 44, 67, 29]. In this section, we apply the method in [36] (the method in [43] is similar, but without the normalization constraint on the dictionaries) to the CDL problem (13). In the next subsection, we discuss issues and our resolutions.

Given all the past training samples $\{s^{(\tau)}\}_{\tau=1}^{t}$, the objective function in (13) could be approximated by $F^{(t)}$ as

$$F^{(t)}(\mathbf{d}) = \frac{1}{t}\Big(\sum_{\tau=1}^{t} f(\mathbf{d}; \mathbf{s}^{(\tau)})\Big) \approx F(\mathbf{d}) = \mathbb{E}_{\mathbf{s}}[f(\mathbf{d}; \mathbf{s})] . \tag{25}$$

Central limit theorem tells us $F^{(t)} \to F$ as $t \to \infty$. However, $F^{(t)}$ is not computationally friendly. Each item in $F^{(t)}$ is, by (11) and (12),

$$f(\mathbf{d}; \mathbf{s}^{(\tau)}) = l\Big(\mathbf{d}, \mathbf{x}^*(\mathbf{d}; \mathbf{s}^{(\tau)}); \mathbf{s}^{(\tau)}\Big) + \lambda \left\|\mathbf{x}^*(\mathbf{d}; \mathbf{s}^{(\tau)})\right\|_1 .$$

Thus, when we update the dictionary from $\mathbf{d}^{(t-1)}$ to $\mathbf{d}^{(t)}$, we have to re-compute all the past $\mathbf{x}^*$ again because it depends on $\mathbf{d}$.

To approximate F 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 $\mathbf{d}^{(t-1)}$:

$$\mathbf{x}^{(t)} = \mathbf{x}^*\Big(\mathbf{d}^{(t-1)}, \mathbf{s}^{(t)}\Big) . \tag{26}$$

Then the surrogate of $f(\mathbf{d}; \mathbf{s}^{(t)})$ is defined as

$$f^{(t)}(\mathbf{d}) \triangleq l\Big(\mathbf{d}, \mathbf{x}^{(t)}; \mathbf{s}^{(t)}\Big) + \lambda\|\mathbf{x}^{(t)}\|_1 = \frac{1}{2}\left\|X^{(t)}\mathbf{d} - \mathbf{s}^{(t)}\right\|^2 + \lambda\|\mathbf{x}^{(t)}\|_1 , \tag{27}$$

Algorithm 1: Online Convolutional Dictionary Learning (Basic Surrogate Function)

Initialize: Initialize $\mathbf{d}^{(0)}$, let $A^{(0)} \leftarrow 0, \mathbf{b}^{(0)} \leftarrow 0$.
for $t = 1, \cdots, T$ **do**
  Sample a signal $\mathbf{s}^{(t)}$.
  Solve convolutional sparse coding problem (26).
  Compute Hessian matrix $A^{(t)}$ and vector $\mathbf{b}^{(t)}$ (30).
  Update dictionary by solving (29) with FISTA.
end
Output: $\mathbf{d}^{(T)}$

where $X^{(t)}$ is the operator corresponding to $\mathbf{x}^{(t)}$. Given the past information $\mathbf{s}^{(1)}, \cdots, \mathbf{s}^{(t)}$ and $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(t)}$, the surrogate function of $F^{(t)}$ is defined as

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

Then, at the t^{th} step, the dictionary is updated as

$$\mathbf{d}^{(t)} = \operatorname*{arg\,min}_{\mathbf{d} \in \mathbb{R}^{MD}} \mathcal{F}^{(t)}(\mathbf{d}) + \iota_{\mathrm{C}}(\mathbf{d}) \,. \tag{29}$$

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

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

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

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

which has a constant cost per step and yields $\nabla \mathcal{F}^{(t)}(\mathbf{d}) = (A^{(t)}\mathbf{d} - \mathbf{b}^{(t)})/t$. Here $(X^{(t)})^T X^{(t)}$ is the Hessian matrix of $f^{(t)}$. 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. The full algorithm is summarized in Algorithm 1.

3.2. Algorithm 3: improved surrogate function. Algorithm 1 has three practical issues:

- Inaccurate loss function: The surrogate function $\mathcal{F}^{(t)}$ involves old loss functions $f^{(1)}, f^{(2)}, \cdots$, which contain old information $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots$. For example, $\mathbf{x}^{(1)}$ is computed using $\mathbf{d}^{(0)}$ (cf. (26)).
- Expense of computing $A^{(t)}$: Since X is an operator $X : \mathbb{R}^{MD} \to \mathbb{R}^N$, the number of flops of computing $(X^{(t)})^T(X^{(t)})$ is $\mathcal{O}(D^2M^2N)$, which is quite large.
- FISTA is slow at solving subproblem (29): FISTA takes many steps to reach a sufficient accuracy.

To address these points, three modifications of Algorithm 1 are given in this section.[3]

3.2.1. 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 [43, 37, 46, 44]. 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. To incorporate the forgetting factor, we modify our gradient computation (30) as

$$\begin{aligned} A_{\text{mod}}^{(t)} &= \alpha^{(t)} A_{\text{mod}}^{(t-1)} + (X^{(t)})^T X^{(t)} \;, \\ \mathbf{b}_{\text{mod}}^{(t)} &= \alpha^{(t)} \mathbf{b}_{\text{mod}}^{(t-1)} + (X^{(t)})^T \mathbf{s}^{(t)} \;, \\ \nabla \mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}) &= (1/\Lambda^{(t)})(A_{\text{mod}}^{(t)} \mathbf{d} - \mathbf{b}_{\text{mod}}^{(t)}) \;. \end{aligned}$$

Based on the modified surrogate function $\mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d})$, the dictionary update (29) is modified correspondingly to

$$\mathbf{d}^{(t)} = \arg\min_{\mathbf{d} \in \mathbb{R}^{MD}} \mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}) + \iota_C(\mathbf{d}) \;. \tag{35}$$

This technique is used in some previous dictionary learning works, as we mentioned before, but 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:

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

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: $\mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}^{(t)}) - F_{\text{mod}}^{(t)}(\mathbf{d}^{(t)}) \to 0$.

[3] Improvements I and II have been addressed in our previous work [31]. In the present article, we include their theoretical analysis and introduce the new enhancement of the stopping criterion (Improvement III).

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 $F^{(t)}_{\text{mod}}$ is a statistic 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 $\mathcal{F}^{(t)}_{\text{mod}}$ since it gives large weights to old information. An extreme case is, as $p \to 0$, the modified surrogate function (31) reduces to the standard one (28). Section 7.1.1 reports the related numerical results.

3.2.2. Improvement II: solving subproblem (35) with frequency-domain FISTA. Based on the discussion in Section 2.2, (35) can be written in an equivalent form:

$$\mathbf{d}^{(t)} = \arg\min_{\mathbf{d}\in\mathbb{R}^{MN}} \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}) + \iota_{C_{\text{PN}}}(\mathbf{d}) \ . \tag{37}$$

Then, after we apply the acceleration techniques in Section 2.2, the gradient of $\mathcal{F}^{(t)}_{\text{mod}}$ in the frequency domain can be computed as

$$\begin{aligned} \hat{A}^{(t)}_{\text{mod}} &= \alpha^{(t)} \hat{A}^{(t-1)}_{\text{mod}} + (\hat{X}^{(t)})^H \hat{X}^{(t)} \ , \\ \hat{\mathbf{b}}^{(t)}_{\text{mod}} &= \alpha^{(t)} \hat{\mathbf{b}}^{(t-1)}_{\text{mod}} + (\hat{X}^{(t)})^H \hat{\mathbf{s}}^{(t)} \ , \end{aligned} \tag{38}$$

$$\nabla \hat{\mathcal{F}}^{(t)}_{\text{mod}}(\hat{\mathbf{d}}) = (1/\Lambda^{(t)})(\hat{A}^{(t)}_{\text{mod}} \hat{\mathbf{d}} - \hat{\mathbf{b}}^{(t)}_{\text{mod}}) \ . \tag{39}$$

With the gradient in the frequency domain (39), we can apply FISTA in the frequency domain on problem (37), as in Algorithm 2.

Complexity analysis.[4] If we solve (35) directly, the operator $X^{(t)}$ is a linear operator that $\mathbb{R}^{DM} \to \mathbb{R}^N$. Thus, the complexity of computing the Hessian matrix of $f^{(t)}$, $(X^{(t)})^T X^{(t)}$, is $\mathcal{O}(D^2M^2N)$. Moreover, the memory cost is $\mathcal{O}(D^2M^2)$. In comparison, if we solve (37) in the frequency domain, the frequency-domain operator $\hat{X}^{(t)} = (\hat{X}_1, \hat{X}_2, \cdots, \hat{X}_M)$ is a linear operator that $\mathbb{C}^{MN} \to \mathbb{C}^N$, which seems to lead to a larger complexity to compute Hessian: $\mathcal{O}(M^2N^3)$ flops and $\mathcal{O}(M^2N^2)$ memory cost. However, each component $\hat{X}_m$ is diagonal. Hence, the frequency-domain product $(\hat{X}^{(t)})^H \hat{X}^{(t)}$ has only $\mathcal{O}(M^2N)$ non-zero values. Both the number of flops and memory cost are $\mathcal{O}(M^2N)$. Take typical values $M = 64, D = 16 \times 16, N = 256 \times 256$; then, the flops $\mathcal{O}(M^2N)$ is much less than $\mathcal{O}(D^2M^2N)$, and the memory cost $\mathcal{O}(M^2N)$ is comparable to $\mathcal{O}(D^2M^2)$.

3.2.3. Improvement III: the stopping condition of FISTA. Another issue in Algorithm 1 is the stopping of FISTA. A fixed small tolerance will cause too many inner-loop iterations at initial steps. Another strategy, as used in SPAMS [36] and [27], 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 (37) is solved *inexactly*, but the online learning algorithm is still theoretically guaranteed to converge. The stopping accuracy is increasing as t increases. Moreover, with warm start (using $\mathbf{d}^{(t-1)}$ as the initial point of the t^{th} step), the number of inner-loop iterations stays moderate as t increases.

[4] In our previous paper [31], we compared the complexity of computing (37) in the spatial and frequency domains. But a more meaningful comparison should be between solving (35) in the spatial domain and solving (37) in the frequency domain, as we do here.

Algorithm 2: D-update: Modified FISTA for solving subproblem (37)

Input: Hessian matrix $\hat{A}_{\text{mod}}^{(t)}$ and vector $\hat{\mathbf{b}}_{\text{mod}}^{(t)}$.
Dictionary of last iterate: $\mathbf{d}^{(t-1)}$.
Initialize: Let $\mathbf{g}^0 = \mathbf{d}^{(t-1)}$ (warm start), $\mathbf{g}_{\text{aux}}^0 = \mathbf{g}^0$, $\gamma^0 = 1$.
for $j = 0, 1, 2, \ldots$ *until convergence* **do**
  Compute DFT: $\hat{\mathbf{g}}_{\text{aux}}^j = \text{FFT}(\mathbf{g}_{\text{aux}}^j)$.
  Compute conjugate cogradient $\nabla \hat{\mathcal{F}}_{\text{mod}}^{(t)}(\hat{\mathbf{g}}_{\text{aux}}^j)$ by (39).
  Compute the next iterate:

$$\mathbf{g}^{j+1} = \text{Proj}_{C_{\text{PN}}}\left(\text{IFFT}\left(\hat{\mathbf{g}}_{\text{aux}}^j - \eta \nabla \hat{\mathcal{F}}_{\text{mod}}^{(t)}(\hat{\mathbf{g}}_{\text{aux}}^j)\right)\right) . \tag{40}$$

  Let $\gamma^{j+1} = \left(1 + \sqrt{1 + 4(\gamma^j)^2}\right)/2$, then compute the auxiliary variable:

$$\mathbf{g}_{\text{aux}}^{j+1} = \mathbf{g}^{j+1} + \frac{\gamma^j - 1}{\gamma^{j+1}}(\mathbf{g}^{j+1} - \mathbf{g}^j) . \tag{41}$$

end
Output: $\mathbf{d}^{(t)} \leftarrow \mathbf{g}^J$, where J is the last iterate.

Stopping metric. We use the Fixed Point Residual (FPR) [11] in this paper:

$$\mathbf{R}^{(t)}(\mathbf{g}) \triangleq \left\| \mathbf{g} - \text{Proj}_{C_{\text{PN}}}\left(\mathbf{g} - \eta \nabla \mathcal{F}_{\text{mod}}^{(t)}(\mathbf{g})\right) \right\| . \tag{42}$$

There are two reasons to use it. One is its simplicity; if FISTA is used to solve (37), this metric can be computed directly as $\mathbf{R}^{(t)}(\mathbf{g}_{\text{aux}}^j) = \left\|\mathbf{g}^{j+1} - \mathbf{g}_{\text{aux}}^j\right\|$. 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:

$$\mathbf{R}^{(t)}(\mathbf{g}_{\text{aux}}^j) \le \tau^{(t)} \triangleq \tau_0/(1 + \alpha t) , \tag{43}$$

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 (43) is satisfied, we stop the D-update (Algorithm 2) and continue to the next step. The full algorithm is summarized in Algorithm 3. The effect of this stopping condition is theoretically analyzed in Propositions 3 and 4, and numerically demonstrated in Sec. 7.1.3 below.

3.3. Algorithm 4 (Surrogate-Splitting): a limited memory version of Algorithm 3. In Algorithm 3, the Hessian matrix of the surrogate function $\hat{A}_{\text{mod}}^{(t)}$ with the size of $\mathcal{O}(M^2N)$ is maintained and updated. (Recall that M is the total number of dictionary filters and N is the signal dimension.) When M and N are large, say $M = 64$ and $N = 256 \times 256$, we have $M^2N \approx 2.7 \times 10^8$. Thus $\hat{A}_{\text{mod}}^{(t)}$ still requires a large amount of memory. To reduce memory usage, we use small regions instead of the whole signal. Specifically, as illustrated in Fig. 1, we split a signal $\mathbf{s}^{(t)} \in N$ into small regions $\mathbf{s}_{\text{split},1}^{(t)}, \mathbf{s}_{\text{split},2}^{(t)}, \ldots \in \tilde{N}$, with $\tilde{N} < N$, and treat them as if they were distinct signals. In this way, the training signal sequence becomes

Algorithm 3: Online Convolutional Dictionary Learning (Improved Surrogate Function)

Initialize: Initialize $\mathbf{d}^{(0)}$, let $\hat{A}_{\text{mod}}^{(0)} \leftarrow 0, \hat{\mathbf{b}}_{\text{mod}}^{(0)} \leftarrow 0$.
for $t = 1, \cdots, T$ **do**
  Sample a signal $\mathbf{s}^{(t)}$.
  Solve convolutional sparse coding problem (26).
  Compute the Hessian matrix $\hat{A}_{\text{mod}}^{(t)}$ and vector $\hat{\mathbf{b}}_{\text{mod}}^{(t)}$ (32, 38).
  Update dictionary by solving (37) with Algorithm 2 *until stopping condition (43) is satisfied.*
end
Output: $\mathbf{d}^{(T)}$

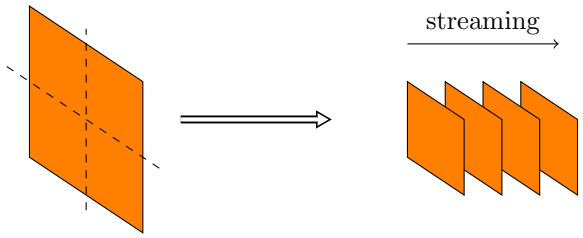


FIG. 1. *An example of image splitting:* $N = 256 \times 256 \rightarrow \tilde{N} = 128 \times 128$.

Algorithm 4: Online Convolutional Dictionary Learning (Surrogate-Splitting)

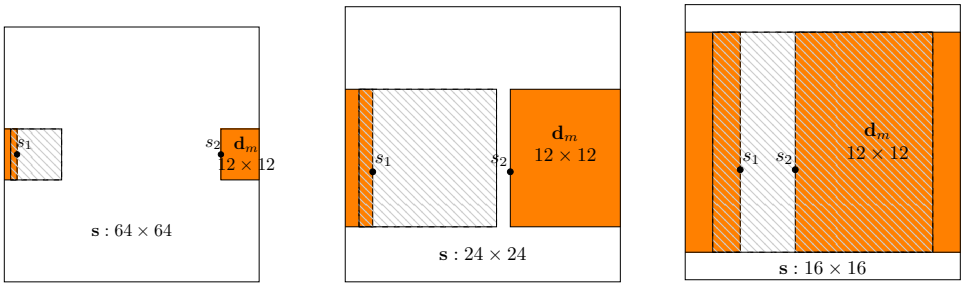
Initialize: Initialize $\mathbf{d}^{(0)}$, let $\hat{A}_{\text{mod}}^{(0)} \leftarrow 0, \hat{\mathbf{b}}_{\text{mod}}^{(0)} \leftarrow 0$.
for $t = 1, \cdots, T$ **do**
  Sample a signal $\mathbf{s}^{(t)}$.
  Split $\mathbf{s}^{(t)}$ into several regions[5]: $\{\mathbf{s}_{\text{split},1}^{(t)}, \cdots, \mathbf{s}_{\text{split},n}^{(t)}\}$.
  Shuffle the n small regions into a random order.
  for $i = 1, \cdots, n$ **do**
    Compute line 3 – line 5 in Algorithm 3 for $\mathbf{s}_{\text{split},i}^{(t)}$.
  end
end
Output: $\mathbf{d}^{(T)}$

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

We call this approach "Surrogate-Splitting", which is listed in Algorithm 4.

Boundary issues. The use of *circular boundary conditions* for signals that are

[5] In our previous work [31], 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 claim 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, which avoids this issue. The "Surrogate-Splitting" algorithm 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.

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

not periodic has the potential to introduce boundary artifacts in the representation, and therefore also in the learned dictionary [63]. When the size of the training images is much larger than the filters, 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 filter size is illustrated in Fig. 2. In Sec. 7.1.2, we study this effect and show that using a splitting size that is twice of the kernel size in each dimension is sufficient to avoid artifacts, as expected from the argument illustrated in Fig. 2.

4. First-order SA method: modified SGD (Algorithm 5) . Now we consider first-order algorithms for the CDL problem (13). Projected SGD can be applied directly as

$$\mathbf{d}^{(t)} = \mathrm{Proj}_C\Big(\mathbf{d}^{(t-1)} - \eta^{(t)} \nabla f(\mathbf{d}^{(t-1)}; \mathbf{s}^{(t)})\Big) \,,$$

where parameter $\eta^{(t)}$ is the *step size*[6]. Given the definition of f in (11), $\nabla f(\mathbf{d}^{(t-1)}; \mathbf{s}^{(t)})$ is the partial derivative on $\mathbf{d}$ at the optimal $\mathbf{x}$ [37, 10], 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 $\mathbf{s}^{(t)}$, which is given by (26). Then we can compute the gradient as

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

Based on equation (24), we can compute the conjugate cogradient of the frequency-domain loss function $\hat{f}$:

$$\nabla \hat{f}(\hat{\mathbf{d}}^{(t-1)}; \hat{\mathbf{s}}^{(t)}) = \big(\hat{X}^{(t)}\big)^H \Big(\hat{X}^{(t)} \hat{\mathbf{d}}^{(t-1)} - \hat{\mathbf{s}}^{(t)}\Big) \,. \tag{44}$$

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

Algorithm 5: Online Convolutional Dictionary Learning (Modified SGD)

Initialize: Initialize $\mathbf{d}^{(0)}$ with a random dictionary.

for $t = 1, \cdots, T$ **do**
  Sample a signal $\mathbf{s}^{(t)}$.
  Solve convolutional sparse coding problem (26).
  Compute the conjugate cogradient of $\hat{f}$ (44).
  Update dictionary via modified SGD (45).
end

Output: $\mathbf{d}^{(T)}$

Algorithm	Single step complexity	Memory usage
Batch learning algorithms [49, 21]	$\mathcal{O}(KMN)$	$\mathcal{O}(KMN)$
Algorithm 1 + FISTA	$\mathcal{O}(D^2M^2N)$	$\mathcal{O}(D^2M^2)$
Algorithm 3 + Algorithm 2	$\mathcal{O}(M^2N)$	$\mathcal{O}(M^2N)$
Algorithm 4 + Algorithm 2	$\mathcal{O}(M^2N)$	$\mathcal{O}(M^2\tilde{N})$
Algorithm 5	$\mathcal{O}(MN)$	$\mathcal{O}(MN)$

TABLE 1
Single step complexity and memory usage of the algorithms in Sections 3 and 4

Similar to Section 2.2, projected SGD can be modified to the following equation:

$$\mathbf{d}^{(t)} = \mathrm{Proj}_{C_{\mathrm{PN}}}\left(\mathrm{IFFT}\left(\hat{\mathbf{d}}^{(t-1)} - \eta^{(t)}\nabla\hat{f}(\hat{\mathbf{d}}^{(t-1)}; \hat{\mathbf{s}}^{(t)})\right)\right) \tag{45}$$

The full algorithm is summarized in Algorithm 5.

Summary of the complexities of the algorithms. We summarize the single-step and memory usage complexities in Table 1. Algorithm 3 is improved based on Algorithm 1; Algorithm 4 is the limited memory version of Algorithm 3. Thus, our most effective algorithms are Algorithms 4 and 5. In Section 7, we will numerically compare these two methods with batch methods.

5. Learning from masked images. In this section, we focus on the masked CDL problem [24, 53] for which there are no existing online algorithms:

$$\min_{\mathbf{d}} \mathbb{E}_{\mathbf{s}}[f_{\mathrm{mask}}(\mathbf{d}; \mathbf{s})] + \iota_{\mathrm{C}}(\mathbf{d}) \;, \tag{46}$$

where f_{mask} is defined as

$$f_{\mathrm{mask}}(\mathbf{d}; \mathbf{s}) \triangleq \min_{\{\mathbf{x}_m\}} \frac{1}{2}\left\| W \odot \Big(\sum_{m=1}^{M} \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \Big) \right\|_2^2 + \lambda \sum_{m=1}^{M} \|\mathbf{x}_m\|_1 \;. \tag{47}$$

The *masking matrix* W is usually a $\{0, 1\}$-valued matrix that masks unknown or unreliable pixels. Similarly to (44), we can derive the conjugate cogradient of $\hat{f}_{\mathrm{mask}}$:

$$\nabla\hat{f}_{\mathrm{mask}}(\hat{\mathbf{d}}^{(t-1)}; \hat{\mathbf{s}}^{(t)}) = \left(\hat{X}^{(t)}\right)^H \mathrm{FFT}\left\{ W \odot \mathrm{IFFT}\left(\hat{X}^{(t)}\hat{\mathbf{d}}^{(t-1)} - \hat{\mathbf{s}}^{(t)}\right) \right\} . \tag{48}$$

With (48) in hand, we propose an online method to solve masked CDL problem (46):

$$\mathbf{d}^{(t)} = \mathrm{Proj}_{C_{\mathrm{PN}}}\left(\mathrm{IFFT}\left(\hat{\mathbf{d}}^{(t-1)} - \eta^{(t)}\nabla\hat{f}_{\mathrm{mask}}(\hat{\mathbf{d}}^{(t-1)}; \hat{\mathbf{s}}^{(t)})\right)\right) \tag{49}$$

Algorithm 6: Online Masked Convolutional Dictionary Learning (Solving (46))

Initialize: Initialize $\mathbf{d}^{(0)}$ with a random dictionary.

for $t = 1, \cdots, T$ **do**

  Sample a signal $\mathbf{s}^{(t)}$.

  Solve masked convolutional sparse coding problem (47):

$$\mathbf{x}^{(t)} = \text{CBPDN-mask}(\mathbf{d}^{(t-1)}; \mathbf{s}^{(t)})$$

  Compute gradient (48).

  Update dictionary via modified SGD (49).

end

Output: $\mathbf{d}^{(T)}$

The full algorithm is summarized in Algorithm 6.

Equivalence of the frequency domain gradient and spatial domain gradient. For the masked loss function,

$$\nabla \hat{f}_{\text{mask}}(\hat{\mathbf{d}}^{(t-1)}; \hat{\mathbf{s}}^{(t)}) = \text{FFT}\Big\{ \nabla f_{\text{mask}}(\mathbf{d}^{(t-1)}; \mathbf{s}^{(t)}) \Big\} . \tag{50}$$

Our proposed algorithm (49) is mathematically equivalent with standard projected stochastic gradient descent on masked CDL problem (46), while our method is computationally cheaper because operator $\hat{X}$ is easier to compute.

6. Convergence. In this section, we study the convergence of Algorithms 3. Algorithm 4, compared with Algorithm 3, only changes the method of splitting the training signals. Thus, the convergence proof of Algorithm 4 follows exactly with that of Algorithm 3. Algorithms 5 and 6, by (24) and (50) respectively, are equivalent to the standard projected SGD. Thus, by properly choosing step sizes $\eta^{(t)}$, Algorithm 5 converges to a stationary point [22]. A diminishing step size rule $\eta^{(t)} = a/(b+t)$ is used in other dictionary learning works [1, 36].

First, we start with some assumptions[7]:

Assumption 1. *All the signals follow 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 be easily guaranteed by normalizing each training signal. Assumption 2 is a common assumption in dictionary learning and other linear regression papers [36, 13]. 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 C for details. Assumption 3 is a common assumption in RLS (see Definition (3.1) in [28]) and dictionary learning (see Assumption B in [37]).

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*

[7] The specific formulas for Assumptions 2 and 3 are shown in Appendix C.

of Z: $\hat{Z}^n_{mod} \triangleq \frac{1}{\sum_{i=1}^n (i/n)^p} \sum_{i=1}^n (i/n)^p Z_i$. *Then, we have*

$$\sqrt{n}(\hat{Z}^n_{mod} - \mu) \xrightarrow{d} N\Big(0, \frac{p+1}{\sqrt{2p+1}}\sigma\Big) . \tag{51}$$

$$\mathbb{E}\Big[\sqrt{n}\big|\hat{Z}^n_{mod} - \mu\big|\Big] = \mathcal{O}(1) . \tag{52}$$

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 D.3.

Proposition 2 (Convergence of functions). *With Assumptions 1-3, we have*

$$\mathbb{E}\Big[\sqrt{t}\big\|F - F^{(t)}\big\|_\infty\Big] \leq M , \tag{53}$$

$$\mathbb{E}\Big[\sqrt{t}\big\|F - F^{(t)}_{mod}\big\|_\infty\Big] \leq \frac{p+1}{\sqrt{2p+1}} M , \tag{54}$$

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

This proposition is an extension of Donsker's theorem (see Lemma 7 in [37] and Chapter 19 in [48]). The proof is given in Appendix D.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:*

$$(\mathbf{d}^*)^{(t)} = \arg\min_{\mathbf{d}} \mathcal{F}^{(t)}_{mod}(\mathbf{d}) + \iota_C(\mathbf{d}) . \tag{55}$$

Let $\mathbf{d}^{(t)}$ *be the solution obtained by the frequency-domain FISTA (Algorithm 2) with our proposed stopping condition (43). Then, we have*

$$\big\|\mathbf{d}^{(t)} - (\mathbf{d}^*)^{(t)}\big\| \leq \mathcal{O}\left(t^{-1}\right) . \tag{56}$$

The proof is given in Appendix D.1.

Proposition 4 (The convergence rate of Algorithm 3). *Let* $\mathbf{d}^{(t)}$ *be the sequence generated by Algorithm 3. Then, we have*

$$\big\|\mathbf{d}^{(t+1)} - \mathbf{d}^{(t)}\big\| = \mathcal{O}\left(t^{-1}\right) . \tag{57}$$

Compared with Lemma 1 in [37], which shows the convergence rate of the exact D-update algorithm (Algorithm 1), our Proposition 4 shows that the inexact D-update (43) shares the same rate. Since our inexact version stops FISTA earlier, it is faster. The proof of this proposition is given in Appendix D.2.

Theorem 1 (Almost sure convergence of Algorithm 3). *Let* $\mathcal{F}^{(t)}_{mod}$ *be the surrogate function sequence,* $\mathbf{d}^{(t)}$ *the iterate sequence, both generated by Algorithm 3. Then we have, with probability* 1,

1. $\mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}^{(t)})$ *converges.*
2. $\mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}^{(t)}) - F(\mathbf{d}^{(t)}) \to 0$.
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 D.5.

7. Numerical results: learning from clean data set. All the experiments are conducted in MATLAB R2016a running on a workstation with 2 Intel Xeon(R) X5650 CPUs clocked at 2.67GHz. The dictionary size is $12\times12\times64$, and the signal size is 256×256. Dictionaries are evaluated by comparing the functional values obtained by computing CBPDN (11) on the test set. A smaller function value indicates a better dictionary. Similar methods to evaluate dictionary are also used in other dictionary learning works: [37, 47]. The training set consists of 40 images selected from the MIRFLICKR-1M dataset [26][8], and the test set consists of 20 different images from the same source. The penalty parameter is chosen as $\lambda = 0.1$.

Originally, all the images used are of size 512×512. To accelerate the experiments, we crop the borders of both the training images and testing images and preserve the middle part to yield 256×256. The training and testing images are pre-processed by dividing by 255 to rescale the pixel values to the range $[0, 1]$ and highpass filtering[9].

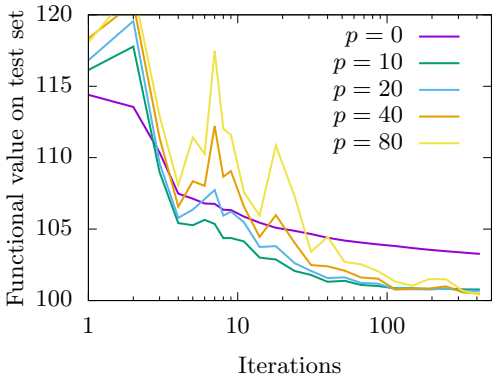
7.1. Tuning parameters for Surrogate-Splitting (Algorithm 4). There are 3 parameters to tune in Algorithm 4: the forgetting exponent p, splitting size $\tilde{N}$, and stopping tolerance of FISTA $\tau^{(t)}$. We study the effect of these parameters. The experiment results are shown in Fig. 3.

7.1.1. Forgetting exponent p. In this section, we fix $\tilde{N} = 64 \times 64$, which is illustrated as a good choice in the next paragraphs, and $\tau^{(t)} = 10^{-3}$, which is small enough. Fig. 3(a) shows the results. It shows that, when $p = 0$, the curve is monotonic and with small oscillation, but it converges slowly, and the dictionaries obtained are not accurate. (Higher function values indicate more inaccurate dictionaries.) When p is larger, the algorithm converges to lower function values. When p is too large, for instance, $p \in \{40, 80\}$, the curve oscillates 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.

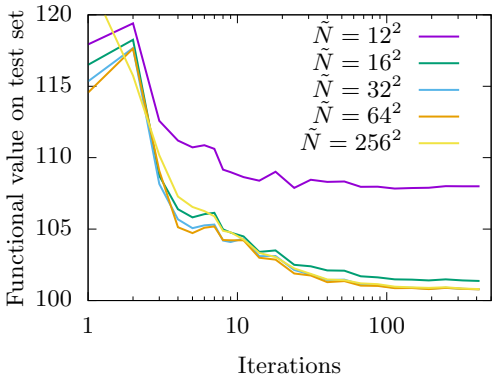
7.1.2. Region splitting size $\tilde{N}$ and boundary artifacts. In this section, we fix $\tau^{(t)} = 10^{-4}$, which is small enough. Fig. 3(b) shows the results. Moreover, we report the dictionaries obtained with different $\tilde{N}$ in Fig. 4. In our experiments, we only consider square signals ($\tilde{N} = 12 \times 12, 16 \times 16, 32 \times 32, 64 \times 64, 256 \times 256$) and square dictionary kernels ($D = 12 \times 12$). When $\tilde{N} \geq 2^2 D$, say $\tilde{N} = 32 \times 32$ or $\tilde{N} = 64 \times 64$, the algorithm converges to a good function value, which is the same with that of Algorithm 3 without image-splitting. However, when $\tilde{N}$ is smaller than the threshold $2^2 D$, say $\tilde{N} = 16 \times 16$ or 12×12, the algorithm converges to a higher function value, which implies worse dictionaries. Thus, we can conclude that *the*

[8]The actual image data contained in this dataset is of very low resolution since the dataset 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.

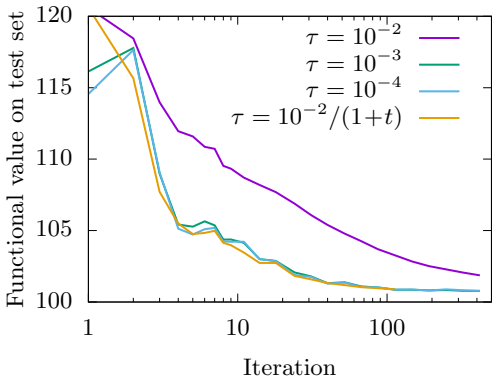
[9]The highpass component is the difference between the input signal and a lowpass component computed by Tikhonov regularization with a gradient term [58, pg. 3], with regularization parameter 5.0.



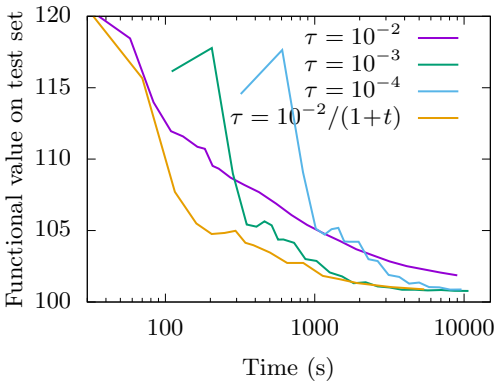
(a) Effect of forgetting exponent p. A small p leads to a higher function value (inaccurate approximate); a large p leads to instability. $p = 10$ is a good choice.



(b) Effect of the region splitting size $\tilde{N}$. Boundary artifacts become significant when splitting region size is smaller than twice of the dictionary kernel size. Here the kernel size is 12×12, thus 24×24 is the threshold.



(c) Effect of subproblem stopping tolerance $\tau^{(t)}$. A large tolerance leads to inaccuracy. Our diminishing tolerance starting with a large tolerance and increasing the accuracy as t increases, avoids this issue.



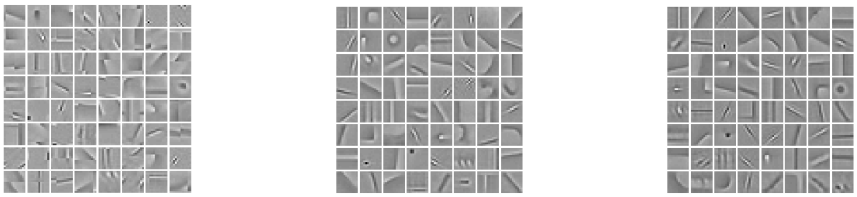
(d) Effect of subproblem stopping tolerance $\tau^{(t)}$. A small enough tolerance is accurate enough but slow. Our diminishing tolerance scheme is faster since we stop the inner loop earlier.

FIG. 3. *Tuning parameters of Surrogate-Splitting (Algorithm 4)*

splitting size should be at least twice of the dictionary kernel size in each dimension. Otherwise, it will lead to boundary artifacts. This phenomenon is consistent with our statements in Section 3.3. The artifacts are specifically displayed in Fig. 4. When $\tilde{N}$ is smaller than the threshold, say 12×12, the features learned are incomplete.

In the following sections, we will uniformly use 64×64, since it provides a good balance between memory cost and functional convergence.

7.1.3. Stopping tolerance of FISTA $\tau^{(t)}$. Fig. 3(c) and 3(d) show the effect of using different $\tau^{(t)}$. If we use a fixed tolerance as our stopping condition, then when the tolerance is large, say $\tau^{(t)} = 10^{-2}$, the function value decreases fast at first but goes to a higher value in the end; when the tolerance is small, say $\tau^{(t)} = 10^{-3}$ or



(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 $\tilde{N} = 256 \times 256$.

FIG. 4. *Effect of splitting regions size $\tilde{N}$ (Algorithm 4).*

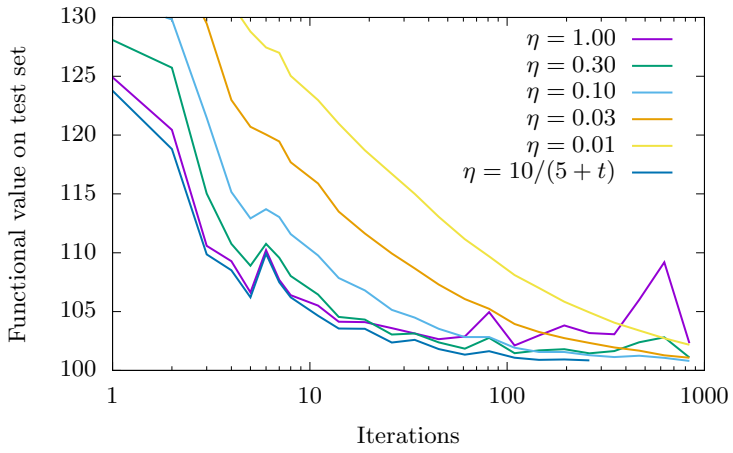


FIG. 5. *Tuning the step size of modified SGD (Algorithm 5). With a large fixed η, the algorithm converges fast at first but unstable in the end; with a small fixed η, the algorithm converges slowly. A diminishing step size is a good balance.*

10^{-4}, the algorithm converges slowly. Consider our proposed diminishing tolerance rule (43) $\tau^{(t)} = 0.01/(1+t)$. When $t = 0$, we have $\tau^{(0)} = 10^{-2}$; at the end, $\tau^{(100)} \approx 10^{-4}$. Thus, the diminishing tolerance fixes this issue and balances the convergence rate and accuracy. Based on the results shown in Sections 7.1.1, 7.1.2, and 7.1.3, $p = 10, \tilde{N} = 64 \times 64, \tau^{(t)} = 10^{-2}/(1+t)$ is a good choice for Algorithm 4.

7.2. Tuning parameters for modified SGD (Algorithm 5). The only parameter to tune in Algorithm 5 is the step size $\eta^{(t)}$. 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 Fig. 5. We test the convergence performance of fixed step size scheme with values: $\eta_0 \in \{1,\ 0.3,\ 0.1,\ 0.03,\ 0.01\}$. We also test the convergence performance of 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 Fig. 5. When a large fixed step size is used, the function value decreases fast initially but becomes unstable in the end. A smaller step size causes the opposite. A diminishing step size balances accuracy and convergence rate.

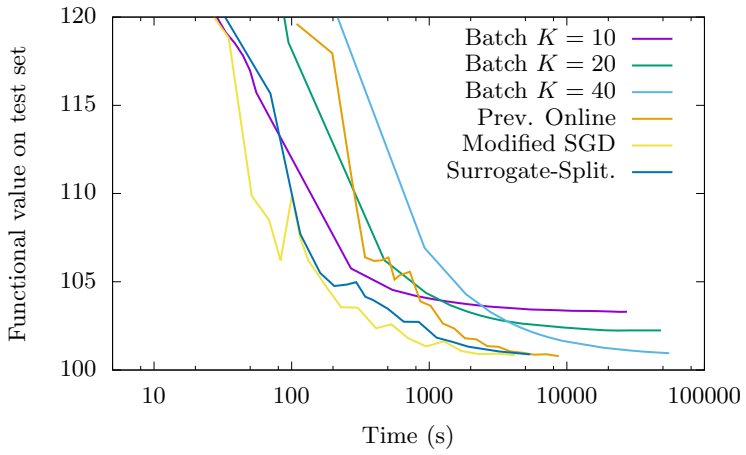


FIG. 6. *Main Result I: convergence speed comparison on the clean data set. Notice that the time axis is logarithmically scaled. In this plot, "Prev. Online" refers to our algorithm "Online-Samp" proposed in [31], "Modified SGD" refers to Algorithm 5 and "Surrogate-Split." refers to Algorithm 4.*

7.3. 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. A performance comparison of batch and online methods is presented in Fig. 6. The batch results are computed using the ADMM consensus dictionary update [49, 21], which is currently one of the leading methods. For batch learning algorithms, we test on a subset of the whole training set with $10, 20, 40$ images. 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 (40 images in total). All the parameters are tuned as follows. For batch learning algorithms, we use the "adaptive penalty parameter" scheme in [57]. For modified SGD (Algorithm 5), we use the step size of $10/(5+t)$. For Surrogate-Splitting (Algorithm 4), we use $p = 10, \tau^{(t)} = 0.01/(1+t), \tilde{N} = 64 \times 64$. For our algorithm proposed in [31], we use $p = 10, \tau^{(t)} = 10^{-3}, \tilde{N} = 64 \times 64$.

The time axis is logarithmically scaled. Thus, the advantage of online learning is significant. To get to the same function value 101 on the test set, batch learning takes 15 hours, our previous method [31] takes around 1.5 hours, Algorithm 4 takes around 1 hour, and Algorithm 5 takes less than 1 hour. We can conclude that, both modified SGD (Algorithm 5) and Surrogate-Splitting (Algorithm 4) converge faster than the batch learning algorithms and our previous method.

7.4. Main result II: memory usage. As Table 2 shows, both Algorithm 5 and 4 save a large amount of memory.

7.5. Main result III: dictionaries obtained by different algorithms. We print out the dictionaries obtained by algorithms in Section 7.3 in Fig. 7. A small training set, say 10 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 Fig. 7(c), 7(e), and 7(f)) within much less time (see Fig. 6) and much less memory usage (see Table 2).

8. 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

Scheme	Memory (MB)
Batch ($K = 10$)	1959.58
Batch ($K = 20$)	3887.08
Batch ($K = 40$)	7742.08
Our algorithm "Online-Samp" in [31]	158.11
Algorithm 4: Surrogate-Splitting	158.11
Algorithm 5: modified SGD	154.84

TABLE 2
Main Result II: Memory Usage Comparison in Megabytes.

set with *salt-and-pepper* impulse noise at known pixel locations. We apply the noise to $10\%, 20\%$, and 30% of the positions, as Fig. 8 shows. We pre-process the data set by applying a highpass filter computed as the difference between the input and a nonlinear lowpass filter[10]. When the number of noisy pixels is low, say 10%, SGD without masking (Algorithm 5) still can learn some features, as Fig. 8(b) demonstrates. When the number of noisy pixels is significant, say 30%, SGD without masking "learns" nothing valid, as Fig. 8(h) demonstrates. However, SGD with masking technique (Algorithm 6) works much better because it "ignores" the noisy positions.

8.1. Masked CDL: online algorithm (Algorithm 6) vs batch algorithm. We compare our online masked SGD and a batch dictionary learning algorithm[11] [53] incorporating the mask via the mask decoupling technique [24]. The results are shown in Fig. 9. We use the function value on the *clean* test set as the criterion. Our online algorithm converges much faster and more stably. Our Algorithm 6 takes $1 \sim 2$ hours to converge while the mask-decoupling scheme needs more than 10 hours.

9. Conclusions. We have proposed two efficient online convolutional dictionary learning methods. Both of them have theoretical convergence guarantee and show good performance on both time and memory usage. Compared to recent online CDL works [12, 52], which use the same framework but different D-update algorithms, our second order SA method improves the framework by several practical techniques. Our first order SA 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 the first-order SA method, we have also proposed an online dictionary learning method, which is able to learn meaningful dictionaries from a partially masked training set.

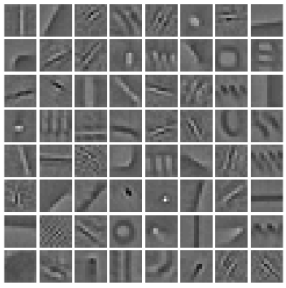
Appendix A. Derivation of conjugate cogradient (22) of the frequency-domain loss function $\hat{l}$.

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: $f(x) = f(Rx + iIx)$, where $Rx, Ix \in \mathbb{R}^n$ are the real part and imaginary part, respectively. By [45], "conjugate cogradient" is defined as

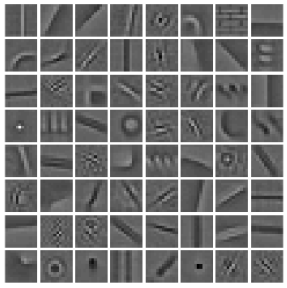
$$\nabla f(x) \triangleq \frac{\partial f}{\partial Rx} + i\frac{\partial f}{\partial Ix} \,. \tag{58}$$

[10]The lowpass component was computed by ℓ_2 total-variation denoising [42] of the input image with a spatial mask informed by the known locations of corrupted pixels in the data fidelity term.

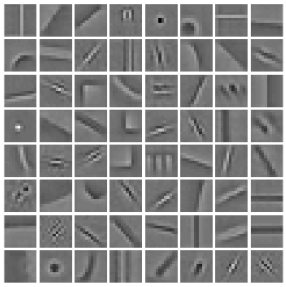
[11]We used the implementation `cbpdndlmd.m` from the Matlab version of the SPORCO library [56], with the Iterated Sherman-Morrison dictionary update solver option [55].



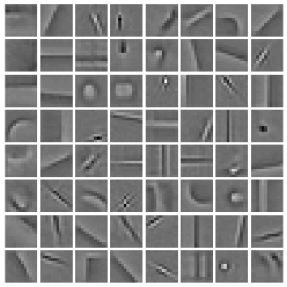
(a) Dictionaries learned by batch learning algorithm (10 training images): many "random" kernels.



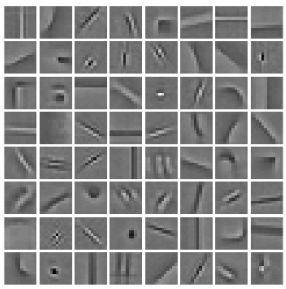
(b) Dictionaries learned by batch learning algorithm (20 training images): less "random" kernels, more valid features



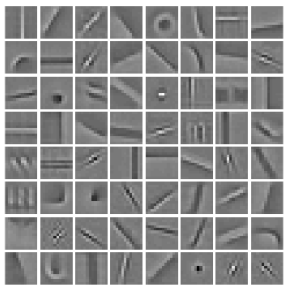
(c) Dictionaries learned by batch learning algorithm (40 training images): almost all kernels are valid.



(d) Dictionaries learned by [31], which is similar to 7(c).



(e) Dictionaries learned by Surrogate-Splitting (Algorithm 4), which is similar to 7(c).



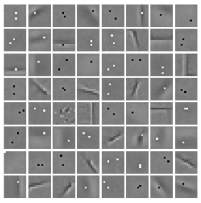
(f) Dictionaries learned by modified SGD (Algorithm 5), which is similar to 7(c).

FIG. 7. *Main result III: Dictionaries learned from the clean data set.*

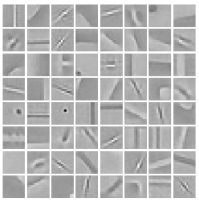
Based on (58), we give a derivation of (22).

Recall the definition $\hat{l}(\hat{\mathbf{d}}, \hat{\mathbf{x}}; \hat{\mathbf{s}}) = 1/2\left\|\hat{X}\hat{\mathbf{d}} - \hat{\mathbf{s}}\right\|^2$. Plugging $\hat{X} = R\hat{X} + iI\hat{X}$,

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

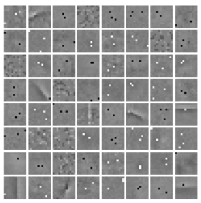


(b) Learning with SGD (Algorithm 5): some features learned.



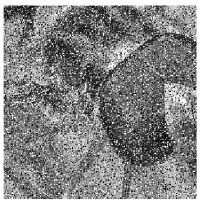
(c) Learning with masked SGD (Algorithm 6): clean features learned.

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

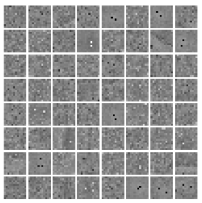


(e) Learning with SGD (Algorithm 5): few features learned.

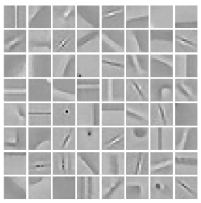
(f) Learning with masked SGD (Algorithm 6): clean features learned.



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



(h) Learning with SGD (Algorithm 5): almost no valid features.



(i) Learning with masked SGD (Algorithm 6): clean features learned.

FIG. 8. *Learning from the noisy training set.*

$\hat{\mathbf{d}} = R\hat{\mathbf{d}} + iI\hat{\mathbf{d}}$, and $\hat{\mathbf{s}} = R\hat{\mathbf{s}} + iI\hat{\mathbf{s}}$ into $\hat{l}$, we have

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

The partial derivatives on $R\hat{\mathbf{d}}$ and $I\hat{\mathbf{d}}$ are, respectively,

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

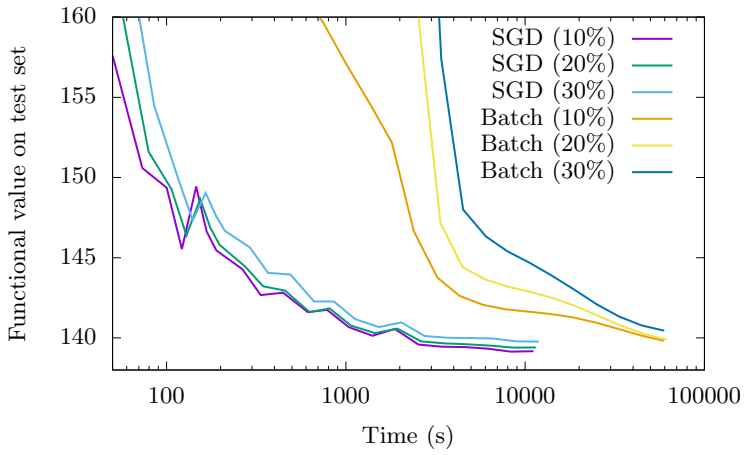


FIG. 9. *For masked CDL problem (46), Algorithm 6 performs better than batch learning algorithms.*

Therefore,

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

By the definition of conjugate cogradient (58), 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: (24) and (50).

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

$$\hat{\mathbf{x}}=\mathcal{F}\mathbf{x}\,,\quad \hat{X}=\mathcal{F}X\mathcal{F}^H\,. \tag{59}$$

The first one is by definition. To prove the second one, notice that

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

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

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

which is exactly (24). Applying the same technique, we can prove (50). ∎

Appendix C. Details of the assumptions.

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

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

where $\mathbf{x} \in \mathbb{R}^{MN}$, $\mathbf{s} \in \mathbb{R}^N$, $D : \mathbb{R}^{MN} \to \mathbb{R}^N$. The coefficient map $\mathbf{x}$ is usually sparse, and Λ is the non-zero index of $\mathbf{x}$. Then, we have

$$D\mathbf{x} = D_\Lambda \mathbf{x}_\Lambda \ .$$

By results in [19], the sparse coding problem (12) has the unique solution if $D_\Lambda^T D_\Lambda$ is invertible, and its unique solution is

$$\mathbf{x}_\Lambda^* = (D_\Lambda^T D_\Lambda)^{-1}(D_\Lambda^T \mathbf{s} - \lambda \text{sign}(\mathbf{x}_\Lambda^*)) \ . \tag{60}$$

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

$$\sigma_{\min}(D_\Lambda^T D_\Lambda) \geq \kappa \ . \tag{61}$$

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

$$\langle \nabla \mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}) - \nabla \mathcal{F}_{\text{mod}}^{(t)}(\tilde{\mathbf{d}}), \mathbf{d} - \tilde{\mathbf{d}} \rangle \geq \mu \left\| \mathbf{d} - \tilde{\mathbf{d}} \right\|^2 \ , \tag{62}$$

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

Appendix D. 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* $f^{(t)}$ *(27) and* $\mathcal{F}_{mod}^{(t)}$*(36) are uniformly L-smooth, i.e.*

$$\begin{aligned} \left\| \nabla f^{(t)}(\mathbf{d}) - \nabla f^{(t)}(\tilde{\mathbf{d}}) \right\| &\leq L_f \left\| \mathbf{d} - \tilde{\mathbf{d}} \right\| \\ \left\| \nabla \mathcal{F}_{mod}^{(t)}(\mathbf{d}) - \nabla \mathcal{F}_{mod}^{(t)}(\tilde{\mathbf{d}}) \right\| &\leq L_{\mathcal{F}} \left\| \mathbf{d} - \tilde{\mathbf{d}} \right\| \ , \end{aligned} \tag{63}$$

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

Proof. First, we consider a single surrogate function:

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

By $\mathbf{d} \in \mathrm{C}$ (the compact support of $\mathbf{d}$), Assumption 1 (the compact support of $\mathbf{s}$), and equation (60) (regularity of convolutional sparse coding), we have $\mathbf{x}^{(t)}$ is uniformly bounded. Therefore, $X^{(t)}$, the operator form of $\mathbf{x}^{(t)}$, is also uniformly bounded:

$$\left\| X^{(t)} \right\| \leq M, \tag{64}$$

for all t, for some $M > 0$, which is independent of t.
By (31), we have

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

which, together with (64), implies (63). ∎

D.1. Proof of Proposition 3. Given the strong-convexity (62) and smoothness (63) of the surrogate function, we start to prove Proposition 3.

Proof. To prove (56), 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 $\mathbf{g}^j_{\mathrm{aux}}$ are the iterates generated in Algorithm 2, and j is the loop index. Then, we want to show that

$$\left\|\mathbf{g}^{j+1} - \mathbf{g}^*\right\| \leq C\mathbf{R}^{(t)}(\mathbf{g}^j_{\mathrm{aux}}) \ , \quad \forall j \geq 0 \ . \tag{65}$$

By (24), it is enough to prove for the spatial-domain FISTA. By strong convexity and smoothness of $\mathcal{F}$, we obtain

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

Combining the above inequality and the definition of FPR (42), we have

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

Let the step size be small enough $\eta \leq \min(\mu/L^2, 1/\mu)$, we have $0 \leq 1-2\mu\eta+\eta^2L^2 \leq 1$, which implies (65). Combining (65) and (43), we get (56). ∎

D.2. Proof of Proposition 4.

Proof. Recall $(\mathbf{d}^*)^{(t)}$ (55) 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 (43)). Then, by the strong convexity of $\mathcal{F}_{\text{mod}}^{(t)}$, we have

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

Let $r^{(t)} = \|\mathbf{d}^{(t+1)} - \mathbf{d}^{(t)}\|$. If $r^{(t)} \leq C/t$, Proposition 4 is directly proved. Otherwise, Proposition 3 (56) implies $r^{(t)} - \|\mathbf{d}^{(t)} - (\mathbf{d}^*)^{(t)}\| \geq r^{(t)} - C/t \geq 0$ and

$$\mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}^{(t+1)}) - \mathcal{F}_{\text{mod}}^{(t)}(\mathbf{d}^{(t)}) \geq \mu\Big(r^{(t)} - \frac{C}{t}\Big)^2 - \frac{LC^2}{t^2} \,. \tag{66}$$

On the other hand,

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

Now we will give the upper bounds of $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3$. Given the smoothness of $\mathcal{F}_{\text{mod}}^{(t)}$(63) and $(\mathbf{d}^*)^{(t+1)}$ being the minimizer of $\mathcal{F}_{\text{mod}}^{(t)}$, we have an upper bound of $\mathbf{T}_2$:

$$\begin{aligned}
\mathbf{T}_2 =&\mathcal{F}_{\text{mod}}^{(t+1)}(\mathbf{d}^{(t+1)}) - \mathcal{F}_{\text{mod}}^{(t+1)}(\mathbf{d}^{(t)}) \\
=&\mathcal{F}_{\text{mod}}^{(t+1)}(\mathbf{d}^{(t+1)}) - \mathcal{F}_{\text{mod}}^{(t+1)}((\mathbf{d}^*)^{(t+1)}) - \Big(\mathcal{F}_{\text{mod}}^{(t+1)}(\mathbf{d}^{(t)}) - \mathcal{F}_{\text{mod}}^{(t+1)}((\mathbf{d}^*)^{(t+1)})\Big) \\
\leq&L\|\mathbf{d}^{(t+1)} - (\mathbf{d}^*)^{(t+1)}\|^2 - 0 \leq \frac{LC^2}{t^2} \,.
\end{aligned} \tag{67}$$

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

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

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 equation (64) (boundedness of X). The last inequality is derived by the follows:

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

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

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

Therefore,

$$\mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t+1)}) - \mathcal{F}^{(t)}_{\text{mod}}(\mathbf{d}^{(t)}) \leq \frac{C_1}{t} r^{(t)} + \frac{LC^2}{t^2} \ . \tag{68}$$

Combining (66) and (68), we have

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

which implies

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

We write it in a neat way,

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

Finally, $r^{(t)}$ is bounded by $r^{(t)} \leq (C_2 + \sqrt{C_2^2 + C_3})/t$. (57) is proved. ∎

D.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 $\{Y_i\}$. Firstly, 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

$$\frac{1}{s_n^{2+\delta}} \sum_{i=1}^n \mathbb{E}\Big[|Y_i - \mu_i|^{2+\delta}\Big] \leq \frac{1}{s_n^{2+\delta}} \sum_{i=1}^n (i^p\sigma)^{2+\delta} = \mathcal{O}\Big(\frac{n^{2p+1+\delta p}}{n^{2p+1+\delta p+\delta/2}}\Big) = \mathcal{O}(n^{-\delta/2}) \ . \tag{69}$$

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

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

Given the following 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

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

Then (51) is obtained by $\frac{1}{s_n}\sum_{i=1}^{n}(Y_i-\mu_i)\xrightarrow{\mathrm{d}}N(0,1)$ and $(1+1/n)\to 1$.

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

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

By the independence between different Z_i, we have

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

where B is the upper bound of Z_i as Z_i is compact supported. (52) is proved. ∎

D.4. Proof of Proposition 2.

Proof. Firstly, 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

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

for some $B>0$, for fixed $\mathbf{d}$. Since F and $F^{(t)}_{\mathrm{mod}}$ are continuously differentiable and have uniformly bounded derivatives (64), we have $\mathbb{E}\Big[\sqrt{t}\big|F(\mathbf{d})-F^{(t)}_{\mathrm{mod}}(\mathbf{d})\big|\Big]$ is uniformly continuous w.r.t $\mathbf{d}$ on a compact set C. Thus, the boundedness of $\mathbb{E}\Big[\sqrt{t}\big|F(\mathbf{d})-F^{(t)}_{\mathrm{mod}}(\mathbf{d})\big|\Big]$ on each $\mathbf{d}$ implies the boundedness for all $\mathbf{d}\in\mathrm{C}$. Inequality (54) is proved. Taking $p\to 0$, we have (53). ∎

D.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 [37], we will show that $u^{(t)}$ is a "quasi-martingale" [18].

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

The bound of $\mathbf{T}_2$ is given by (67). 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 (11) 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

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

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

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

where the second inequality follows from (54). Given the bound of $\mathbf{T}_2$ (67) and $\mathbf{T}_4$, we have

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

which implies that $u^{(t+1)}$ generated by Algorithm 3 is a quasi-martingale. Thus, by results in [7, Sec. 4.4] or [37, Theorem 6], we have $u^{(t)}$ converges almost surely.
For the proofs of 2, 3 and 4, using the results in Proposition 4, 2 in this paper, following the same proof line of Proposition 3 and 4 in [37], we can obtain the results in 2, 3 and 4. ∎

REFERENCES

[1] M. Aharon and M. Elad, *Sparse and redundant modeling of image content using an image-signature-dictionary*, SIAM Journal on Imaging Sciences, 1 (2008), pp. 228–247.

[2] M. Aharon, M. Elad, and A. Bruckstein, *k-SVD: An algorithm for designing overcomplete dictionaries for sparse representation*, IEEE Transactions on Signal Processing, 54 (2006), pp. 4311–4322.

[3] L. V. Ahlfors, *Complex analysis: an introduction to the theory of analytic functions of one complex variable*, McGraw-Hill, 1979.

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

[5] D. P. Bertsekas, *Nonlinear programming*, Athena scientific Belmont, 1999.

[6] P. Billingsley, *Probability and measure*, John Wiley & Sons, 2008.

[7] L. Bottou, *Online learning and stochastic approximations*, Online learning in neural networks, 17 (1998), p. 142.

[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.

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

[10] J. M. Danskin, *The theory of max-min, with applications*, SIAM Journal on Applied Mathematics, 14 (1966), pp. 641–664.

[11] D. Davis and W. Yin, *Convergence rate analysis of several splitting schemes*, in Splitting Methods in Communication, Imaging, Science, and Engineering, Springer, 2016, pp. 115–163.

[12] K. Degraux, U. S. Kamilov, P. T. Boufounos, and D. Liu, *Online convolutional dictionary learning for multimodal imaging*, Preprint arXiv:1706.04256, (2017).

[13] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al., *Least angle regression*, The Annals of Statistics, 32 (2004), pp. 407–499.

[14] E. M. Eksioglu, *Online dictionary learning algorithm with periodic updates and its application to image denoising*, Expert Systems with Applications, 41 (2014), pp. 3682–3690.

[15] M. Elad and M. Aharon, *Image denoising via sparse and redundant representations over learned dictionaries*, IEEE Transactions on Image Processing, 15 (2006), pp. 3736–3745.

[16] K. Engan, S. O. Aase, and J. H. Husoy, *Method of optimal directions for frame design*, in Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 5, 1999, pp. 2443–2446.

[17] K. Engan, B. D. Rao, and K. Kreutz-Delgado, *Frame design using focuss with method of optimal directions (mod)*, in Proceedings of Norwegian Signal Processing Symposium (NORSIG), 1999, pp. 65–69.

[18] D. L. Fisk, *Quasi-martingales*, Transactions of the American Mathematical Society, 120 (1965), pp. 369–389.

[19] J.-J. Fuchs, *Recovery of exact sparse representations in the presence of bounded noise*, IEEE Transactions on Information Theory, 51 (2005), pp. 3601–3608.
[20] W. Gao, J. Chen, C. Richard, and J. Huang, *Online dictionary learning for kernel LMS*, IEEE Transactions on Signal Processing, 62 (2014), pp. 2765–2777.
[21] C. Garcia-Cardona and B. Wohlberg, *Subproblem coupling in convolutional dictionary learning*, in Proceedings of IEEE International Conference on Image Processing (ICIP), Sept. 2017. Accepted.
[22] S. Ghadimi and G. Lan, *Stochastic first-and zeroth-order methods for nonconvex stochastic programming*, SIAM Journal on Optimization, 23 (2013), pp. 2341–2368.
[23] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, *Convolutional sparse coding for image super-resolution*, in Proceedings of International Conference on Computer Vision (ICCV), Dec. 2015.
[24] 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.
[25] K. Huang and S. Aviyente, *Sparse representation for signal classification*, in Advances in neural information processing systems, 2007, pp. 609–616.
[26] 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, 2010, pp. 527–536, https://doi.org/10.1145/1743384.1743475.
[27] 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.
[28] R. M. Johnstone, C. R. Johnson, R. R. Bitmead, and B. D. Anderson, *Exponential convergence of recursive least squares with exponential forgetting factor*, Systems & Control Letters, 2 (1982), pp. 77–82.
[29] 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.
[30] M. S. Lewicki and T. J. Sejnowski, *Coding time-varying signals using sparse, shift-invariant representations*, in Advances in neural information processing systems, M. J. Kearns, S. A. Solla, and D. A. Cohn, eds., vol. 11, 1999, pp. 730–736.
[31] J. Liu, C. Garcia-Cardona, B. Wohlberg, and W. Yin, *Online convolutional dictionary learning*, in Proceedings of IEEE International Conference on Image Processing (ICIP), Sept. 2017. Accepted.
[32] Y. Liu, X. Chen, R. K. Ward, and Z. J. Wang, *Image fusion with convolutional sparse representation*, IEEE Signal Processing Letters, (2016), https://doi.org/10.1109/lsp.2016.2618776.
[33] 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.
[34] J. Mairal, F. Bach, and J. Ponce, *Task-driven dictionary learning*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 34 (2012), pp. 791–804.
[35] J. Mairal, F. Bach, and J. Ponce, *Sparse modeling for image and vision processing*, Foundations and Trends in Computer Graphics and Vision, 8 (2014), pp. 85–283, https://doi.org/10.1561/0600000058.
[36] 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, 2009, pp. 689–696.
[37] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, *Online learning for matrix factorization and sparse coding*, Journal of Machine Learning Research, 11 (2010), pp. 19–60.
[38] S. Mallat, *A wavelet tour of signal processing*, Academic press, 1999.
[39] 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), Apr. 2016, pp. 518–521, https://doi.org/10.1109/ISBI.2016.7493321.
[40] 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.
[41] R. Rubinstein, A. M. Bruckstein, and M. Elad, *Dictionaries for sparse representation modeling*, Proceedings of the IEEE, 98 (2010), pp. 1045–1057.
[42] L. Rudin, S. J. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms.*, Physica D. Nonlin. Phenomena, 60 (1992), pp. 259–268, https://doi.org/10.1016/

0167-2789(92)90242-f.

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

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

[45] L. Sorber, M. V. Barel, and L. D. Lathauwer, *Unconstrained optimization of real functions in complex variables*, SIAM Journal on Optimization, 22 (2012), pp. 879–898.

[46] 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.

[47] 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.

[48] A. W. Van der Vaart, *Asymptotic statistics*, vol. 3, Cambridge University Press, 2000.

[49] M. Šorel and F. Šroubek, *Fast convolutional sparse coding using matrix inversion lemma*, Digital Signal Processing, (2016), https://doi.org/10.1016/j.dsp.2016.04.012.

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

[51] 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.

[52] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, *Online convolutional sparse coding*, Preprint arXiv:1706.06972, (2017).

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

[54] 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, July 2016, https://doi.org/10.1109/IVMSPW.2016.7528229.

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

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

[57] B. Wohlberg, *ADMM penalty parameter selection by residual balancing*, Preprint arXiv:1704.06209, (2017).

[58] B. Wohlberg, *SPORCO: A Python package for standard and convolutional sparse representations*, in Proceedings of the 15th Python in Science Conference, Austin, TX, USA, July 2017, pp. 1–8.

[59] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, *Sparse representation for computer vision and pattern recognition*, Proceedings of the IEEE, 98 (2010), pp. 1031–1044.

[60] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, *Robust face recognition via sparse representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 31 (2009), pp. 210–227.

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

[62] J. Yang, J. Wright, T. S. Huang, and Y. Ma, *Image super-resolution via sparse representation*, IEEE Transactions on Image Processing, 19 (2010), pp. 2861–2873.

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

[64] G. Zhang, Z. Jiang, and L. S. Davis, *Online semi-supervised discriminative dictionary learning for sparse representation*, in Proceedings of Asian Conference on Computer Vision (ACCV), 2012, pp. 259–273.

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

[66] H. Zhang and V. M. Patel, *Convolutional sparse and low-rank coding-based rain streak removal*, in Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV), March 2017.

[67] 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.
[68] 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.