Online Convolutional Sparse Coding with Sample-Dependent Dictionary

Yaqing Wang [1] **Quanming Yao** [1 2] **James T. Kwok** [1] **Lionel M. Ni** [3]

Abstract

Convolutional sparse coding (CSC) has been popularly used for the learning of shift-invariant dictionaries in image and signal processing. However, existing methods have limited scalability. In this paper, instead of convolving with a dictionary shared by all samples, we propose the use of a sample-dependent dictionary in which each filter is a linear combination of a small set of base filters learned from data. This added flexibility allows a large number of sample-dependent patterns to be captured, which is especially useful in the handling of large or high-dimensional data sets. Computationally, the resultant model can be efficiently learned by online learning. Extensive experimental results on a number of data sets show that the proposed method outperforms existing CSC algorithms with significantly reduced time and space complexities.

1. Introduction

Convolutional sparse coding (CSC) (Zeiler et al., 2010) has been successfully used in image processing (Gu et al., 2015; Heide et al., 2015), signal processing (Cogliati et al., 2016), and biomedical applications (Pachitariu et al., 2013; Andilla & Hamprecht, 2014; Chang et al., 2017; Jas et al., 2017; Peter et al., 2017). It is closely related to sparse coding (Aharon et al., 2006), but CSC is advantageous in that its shift-invariant dictionary can capture shifted local patterns common in signals and images. Each data sample is then represented by the sum of a set of filters from the dictionary convolved with the corresponding codes.

Traditional CSC algorithms operate in the batch mode (Kavukcuoglu et al., 2010; Zeiler et al., 2010; Bristow et al., 2013; Heide et al., 2015; Šorel & Šroubek, 2016; Wohlberg, 2016; Papyan et al., 2017), which take $O(NK^2P + NKP\log P)$ time and $O(NKP)$ space (where N is the number of samples, K is the number of filters, and P is the dimensionality. Recently, a number of online CSC algorithms have been proposed for better scalability (Degraux et al., 2017; Liu et al., 2017; Wang et al., 2018). As data samples arrive, relevant information is compressed into small history statistics, and the model is incrementally updated. In particular, the state-of-the-art OCSC algorithm (Wang et al., 2018) has the much smaller time and space complexities of $O(K^2P + KP\log P)$ and $O(K^2P)$, respectively.

However, the complexities of OCSC still depend quadratically on K, and cannot be used with a large number of filters. The number of local patterns that can be captured is thus limited, and may lead to inferior performance especially on higher-dimensional data sets. Besides, the use of more filters also leads to a larger number of expensive convolution operations. Rigamonti et al. (2013) and Sironi et al. (2015) proposed to post-process the learned filters by approximating them with separable filters, making the convolutions less expensive. However, as learning and post-processing are then two independent procedures, the resultant filters may not be optimal. Moreover, these separate filters cannot be updated online with the arrival of new samples.

Another direction to scale up CSC is via distributed computation (Bertsekas & Tsitsiklis, 1997). By distributing the data and workload onto multiple machines, the recent consensus CSC algorithm (Choudhury et al., 2017) can handle large, higher-dimensional data sets such as videos, multispectral images and light field images. However, the heavy computational demands of the CSC problem are only shared over the computing platform, but not fundamentally reduced.

In this paper, we propose to approximate the possibly large number of filters by a sample-dependent combination of a small set of base filters learned from the data. While the standard CSC dictionary is shared by all samples, we propose each sample to have its own "personal" dictionary to compensate for the reduced flexibility of using these

[1]Department of Computer Science and Engineering, Hong Kong University of Science and Technology University, Hong Kong [2]4Paradigm Inc, Beijing, China. [3]Department of Computer and Information Science, University of Macau, Macau. Correspondence to: jamesk@cse.ust.hk <James T. Kwok>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

base filters. In this way, the representation power can remain the same but with a reduced number of parameters. Computationally, this structure also allows efficient online learning algorithms to be developed. Specifically, the base filter can be updated by the alternating direction method of multipliers (ADMM) (Boyd et al., 2011), while the codes and combination weights can be learned by accelerated proximal algorithms (Yao et al., 2017). Extensive experimental results on a variety of data sets show that the proposed algorithm is more efficient in both time and space, and outperforms existing batch, online and distributed CSC algorithms.

The rest of the paper is organized as follows. Section 2 briefly reviews convolutional sparse coding. Section 3 describes the proposed algorithm. Experimental results are presented in Section 4, and the last section gives some concluding remarks.

Notations: For a vector a, its ith element is $a(i)$, ℓ_1-norm is $\|a\|_1 = \sum_i |a(i)|$ and ℓ_2-norm is $\|a\|_2 = \sqrt{\sum_i a^2(i)}$. The convolution of two vectors a and b is denoted $a * b$. For a matrix A, $A^\star$ is its complex conjugate, and $A^\dagger = (A^\top)^\star$ its conjugate transpose. The Hadamard product of two matrices A and B is $(A \odot B)(i,j) = A(i,j)B(i,j)$. The identity matrix is denoted I. $\mathcal{F}(x)$ is the Fourier transform that maps x from the spatial domain to the frequency domain, while $\mathcal{F}^{-1}(x)$ is the inverse operator which maps $\mathcal{F}(x)$ back to x.

2. Review: Convolutional Sparse Coding

Given samples $\{x_1, \ldots, x_N\}$ in $\mathbb{R}^P$, CSC learns a shift-invariant dictionary $D \in \mathbb{R}^{M\times K}$, with the columns $\{D(:,k)\}$ representing the K filters. Each sample x_i is encoded as $Z_i \in \mathbb{R}^{P\times K}$, with the kth column being the code convolved with filter $D(:,k)$. The dictionary and codes are learned together by solving the optimization problem:

$$\min_{D\in\mathcal{D},\{Z_i\}} \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left\|x_i - \sum_{k=1}^{K} D(:,k) * Z_i(:,k)\right\|_2^2 + \beta\|Z_i\|_1, \tag{1}$$

where the first term measures the signal reconstruction error, $\mathcal{D} = \{D : \|D(:,k)\|_2 \le 1, \forall k = 1,\ldots,K\}$ ensures that the filters are normalized, and $\beta > 0$ is a regularization parameter controlling the sparsity of Z_i's.

Convolution in (1) is performed in the spatial domain. This takes $O(KPM)$ time, and is expensive. In contrast, recent CSC methods perform convolution in the frequency domain, which takes $O(KP\log P)$ time (Mallat, 1999) and is faster for typical choices of M and P. Let $\tilde{x}_i \equiv \mathcal{F}(x_i)$, $\tilde{D}(:,k) \equiv \mathcal{F}(D(:,k))$, and $\tilde{Z}_i(:,k) \equiv \mathcal{F}(Z_i(:,k))$ be the Fourier-transformed counterparts of $x_i, D(:,k)$ and $Z_i(:,k)$. The codes and filters are updated in an alternating manner by block coordinate descent, as:

Update Codes: Given $\tilde{D}$, each $\tilde{Z}_i$ is independently obtained as

$$\begin{aligned}\min_{\tilde{Z}_i,U_i}\ & \frac{1}{2P}\left\|\tilde{x}_i - \sum_{k=1}^{K}\tilde{D}(:,k)\odot\tilde{Z}_i(:,k)\right\|_2^2 + \beta\|U_i\|_1 \\ \text{s.t. }\ & \mathcal{F}(U_i(:,k)) = \tilde{Z}_i(:,k),\ k = 1,\ldots,K,\end{aligned} \tag{2}$$

where U_i is an auxiliary variable.

Update Dictionary: $\tilde{D}$ is updated by solving

$$\begin{aligned}\min_{\tilde{D},V}\ & \frac{1}{2NP}\sum_{i=1}^{N}\left\|\tilde{x}_i - \sum_{k=1}^{K}\tilde{D}(:,k)\odot\tilde{Z}_i(:,k)\right\|_2^2 \\ \text{s.t. }\ & \mathcal{F}(V(:,k)) = \tilde{D}(:,k),\ k = 1,\ldots,K, \\ & \|\mathcal{C}(V(:,k))\|_2^2 \le 1,\ k = 1,\ldots,K,\end{aligned} \tag{3}$$

where V is an auxiliary variable, and $\mathcal{C}(\cdot)$ crops the extra $P - M$ dimensions in $V(:,k)$.

Both (2) and (3) can be solved by the alternating direction method of multipliers (ADMM) (Boyd et al., 2011). Subsequently, $\{\tilde{Z}_i\}$ and $\tilde{D}$ can be transformed back to the spatial domain as $Z_i(:,k) = \mathcal{F}^{-1}(\tilde{Z}_i(:,k))$ and $D(:,k) = \mathcal{C}(\mathcal{F}^{-1}(\tilde{D}(:,k)))$. Note that while Z_i's (in the spatial domain) are sparse, the FFT-transformed $\tilde{Z}_i$'s are not.

On inference, given the learned dictionary D, the testing sample x_j is reconstructed as $\sum_{k=1}^{K} D(:,k) * Z_j(:,k)$, where Z_j is the obtained code.

2.1. Post-Processing for Separable Filters

Filters obtained by CSC are non-separable and subsequent convolutions may be slow. To speed this up, they can be post-processed and approximated by separable filters (Rigamonti et al., 2013; Sironi et al., 2015). Specifically, the learned D is approximated by SW, where $S \in \mathbb{R}^{M\times R}$ contains R rank-1 base filters $\{S(:,1),\ldots,S(:,R)\}$, and $W \in \mathbb{R}^{R\times K}$ contains the combination weights. However, this often leads to performance deterioration.

2.2. Online CSC

An online CSC algorithm (OCSC) is recently proposed in (Wang et al., 2018). Given the Fourier-transformed sample $\tilde{x}_t$ and dictionary $\tilde{D}_{t-1}$ from the last iteration, the corresponding $\{\tilde{Z}_t, U_t\}$ are obtained as in (2). The following Proposition updates $\tilde{D}_t$ and V_t by reformulating (3) for use with smaller history statistics.

Proposition 1 ((Wang et al., 2018)). *$\tilde{D}_t, V_t$ can be obtained*

by solving the optimization problem:

$$\begin{aligned} \min_{\tilde{D},V} \quad & \frac{1}{2P}\sum_{p=1}^{P}\tilde{D}(p,:)H_t(:,:,p)\tilde{D}^{\dagger}(:,p) \\ & -2\tilde{D}(p,:)G_t(:,p) \qquad (4) \\ \textit{s.t.} \quad & \mathcal{F}(V(:,k)) = \tilde{D}(:,k),\ k=1,\dots,K, \\ & \|\mathcal{C}(V(:,k))\|_2^2 \le 1,\ k=1,\dots,K, \end{aligned}$$

where $H_t(:,:,p) = \frac{1}{t}\sum_{i=1}^{t}\tilde{Z}_i^{\top}(:,p)\tilde{Z}_i^{\star}(p,:) \in \mathbb{R}^{K\times K}$, *and* $G_t(:,p) = \frac{1}{t}\sum_{i=1}^{t}\tilde{x}_i^{\star}(p)\tilde{Z}_i^{\top}(:,p) \in \mathbb{R}^{K}$.

Problem (4) can be solved by ADMM. The total space complexity is only $O(K^2P)$, which is independent of N. Moreover, H_t and G_t can be updated incrementally.

Two other online CSC reformulations have also been proposed recently. Degraux et al. (2017) performs convolution in the spatial domain, and is slow. Liu et al. (2017) performs convolution in the frequency domain, but requires expensive huge sparse matrix operations.

3. Online CSC with Sample-Dependent Dictionary

Though OCSC scales well with sample size N, its space complexity still depends on K quadratically. This limits the number of filters that can be used and can impact performance. Motivated by the ideas of separable filters in Section 2.1, we enable learning with more filters by approximating the K filters with R base filters, where $R \ll K$. In contrast to the separable filters, which are obtained by post-processing and may not be optimal, we propose to learn the dictionary directly during signal reconstruction. Moreover, filters in the dictionary are combined from the base filters in a sample-dependent manner.

3.1. Problem Formulation

Recall that each x_i in (1) is represented by $\sum_{k=1}^{K} D(:,k) * Z_i(:,k)$. Let $B \in \mathbb{R}^{M\times R}$, with columns $\{B(:,r)\}$ being the base filters. We propose to represent x_i as:

$$x_i = \sum_{k=1}^{K}\left(\sum_{r=1}^{R} W_i(r,k)B(:,r)\right) * Z_i(:,k), \qquad (5)$$

where $W_i \in R^{R\times K}$ is the matrix for the filter combination weights. In other words, each $D(:,k)$ in (1) is replaced by $\sum_{r=1}^{R} W_i(r,k)B(:,r)$, or equivalently,

$$D_i = BW_i, \qquad (6)$$

which is sample-dependent. As will be seen, this allows the W_i's to be learned independently (Section 3.3). This also leads to more sample-dependent patterns being captured and thus better performance (Section 4.4).

Sample-dependent filters have been recently studied in convolutional neural networks (CNN) (Jia et al., 2016). Empirically, this outperforms standard CNNs in one-shot learning (Bertinetto et al., 2016), video prediction (Jia et al., 2016) and image deblurring (Kang et al., 2017). Jia et al. (2016) uses a specially designed neural network to learn the filters, and does not consider the CSC model. In contrast, the sample-dependent filters here are integrated into CSC.

The dictionary can also be adapted to individual samples by fine-tuning (Donahue et al., 2014). However, learning the initial shared dictionary is still expensive when K is large. Besides, as will be shown in Section 4.2, the proposed method outperforms fine-tuning empirically.

3.2. Learning

Plugging (6) into the CSC formulation in (1), we obtain

$$\min_{B,\{W_i,Z_i\}} \ \frac{1}{N}\sum_{i=1}^{N} f_i(B,W_i,Z_i) + \beta\|Z_i\|_1 \quad (7)$$

$$\text{s.t.} \quad BW_i \in \mathcal{D},\ i=1,\dots,N, \qquad (8)$$

$$B \in \mathcal{B}, \qquad (9)$$

where

$$f_i(B,W_i,Z_i) \equiv \frac{1}{2}\left\| x_i - \sum_{k=1}^{K}\left(\sum_{r=1}^{R} W_i(r,k)B(:,r)\right) * Z_i(:,k)\right\|_2^2,$$

and $\mathcal{B} \equiv \{B : \|B(:,r)\|_2 \le 1, \forall\, r=1,\dots,R\}$. As B and W_i are coupled together in (8), this makes the optimization problem difficult. The following Proposition decouples B and W_i. All the proofs are in the Appendix.

Proposition 2. *For* $B \in \mathcal{B}$, *we have* $BW_i \in \mathcal{D}$ *if (i)* $W_i \in \mathcal{W}_{\ell 1} \equiv \{W : \|W_i(:,k)\|_1 \le 1,\ k=1,\dots,K\}$, *or (ii)* $W_i \in \mathcal{W}_{\ell 2} \equiv \{W : \|W_i(:,k)\|_2 \le 1/\sqrt{R},\ k=1,\dots,K\}$.

To simplify notations, we use $\mathcal{W}$ to denote $\mathcal{W}_{\ell 1}$ or $\mathcal{W}_{\ell 2}$. By imposing either one of the above structures on $\{W_i\}$, we have the following optimization problem:

$$\begin{aligned} \min_{B,\{W_i,Z_i\}} \quad & \frac{1}{N}\sum_{i=1}^{N} f_i(B,W_i,Z_i) + \beta\|Z_i\|_1 \qquad (10) \\ \text{s.t.} \quad & B \in \mathcal{B},\ \text{ and }\ W_i \in \mathcal{W}, i=1,\dots,N. \end{aligned}$$

On inference with sample x_j, the corresponding (W_j, Z_j) can be obtained by solving (10) with the learned B fixed.

3.3. Online Learning Algorithm for (10)

As in Section 2.2, we propose an online algorithm for better scalability. At the tth iteration, consider

$$\begin{aligned} \min_{B,\{W_i,Z_i\}} \quad & \frac{1}{t}\sum_{i=1}^{t} f_i(B,W_i,Z_i) + \beta\|Z_i\|_1 \qquad (11) \\ \text{s.t.} \quad & B\in\mathcal{B}, \text{ and } W_i\in\mathcal{W},\ i=1,\dots,N. \end{aligned}$$

Let $\tilde{B}(:,r) \equiv \mathcal{F}(B(:,r))$, where $B(:,r)\in\mathbb{R}^M$ is zero-padded to be P-dimensional. Note that the number of convolutions can be reduced from K to R by rewriting the summation above as $\sum_{r=1}^{R} B(:,r) * (\sum_{k=1}^{K} W_i(r,k)Z_i(:,k))$. The following Proposition rewrites (11) and performs convolutions in the frequency domain.

Proposition 3. *Problem* (11) *can be rewritten as*

$$\begin{aligned} \min_{\tilde{B},\{W_i,Z_i\}} \quad & \frac{1}{t}\sum_{i=1}^{t} \tilde{f}_i(\tilde{B},W_i,Z_i)+\beta\|Z_i\|_1 \qquad (12) \\ \textit{s.t.} \quad & \|\mathcal{C}(\mathcal{F}^{-1}(\tilde{B}(:,r)))\|_2\le 1,\ r=1,\dots,R, \\ & W_i\in\mathcal{W},\ i=1,\dots,N, \end{aligned}$$

where $\tilde{f}_i(\tilde{B},W_i,Z_i) \equiv \frac{1}{2P}\|\tilde{x}_i - \sum_{r=1}^{R}\tilde{B}(:,r)\odot\tilde{Y}_i(:,r)\|_2^2$, *and* $\tilde{Y}_i(:,r)\equiv\mathcal{F}(Z_iW_i^\top(:,r))$.

The spatial-domain base filters can be recovered from $\tilde{B}$ as $B(:,r)=\mathcal{C}(\mathcal{F}^{-1}(\tilde{B}(:,r)))$.

3.3.1. Obtaining $\tilde{B}_t$

From (12), $\tilde{B}_t$ can be obtained by solving the subproblem:

$$\begin{aligned} \min_{\tilde{B},\bar{V}} \quad & \frac{1}{2tP}\sum_{i=1}^{t}\left\|\tilde{x}_i - \sum_{r=1}^{R}\tilde{B}(:,r)\odot\tilde{Y}_i(:,r)\right\|_2^2 \\ \text{s.t.} \quad & \mathcal{F}(\bar{V}(:,r)) = \tilde{B}(:,r), r=1,\dots,R, \\ & \|\mathcal{C}(\bar{V}(:,r))\|_2\le 1, r=1,\dots,R, \end{aligned}$$

where $\bar{V}$ is an auxiliary variable. This is of the same form as (3). Hence, analogous to (4), $\tilde{B}_t$ can be obtained as:

$$\begin{aligned} \min_{\tilde{B},\bar{V}} \quad & \frac{1}{2tP}\sum_{i=1}^{t}\sum_{p=1}^{P}\tilde{B}(p,:)\bar{H}_t(:,:,p)\tilde{B}^\dagger(:,p) \\ & -2\tilde{B}(p,:)\bar{G}_t(:,p) \qquad (13) \\ \text{s.t.} \quad & \mathcal{F}(\bar{V}(:,r)) = \tilde{B}(:,r), r=1,\dots,R, \\ & \|\mathcal{C}(\bar{V}(:,r))\|_2\le 1, r=1,\dots,R, \end{aligned}$$

where $\bar{H}_t(:,:,p)=\frac{1}{t}\sum_{i=1}^{t}\tilde{Y}_i^\top(:,p)\tilde{Y}_i^\star(p,:)\in\mathbb{R}^{R\times R}$, and $\bar{G}_t(:,p) = \frac{1}{t}\sum_{i=1}^{t}\tilde{x}_i^\star(p)\tilde{Y}_i^\top(:,p)\in\mathbb{R}^R$. They can be incrementally updated as

$$\bar{H}_t(:,:,p) = \frac{t-1}{t}\bar{H}_{t-1}(:,:,p)+\frac{1}{t}\tilde{Y}_t^\top(:,p)\tilde{Y}_t^\star(p,:), \quad (14)$$

$$\bar{G}_t(:,p) = \frac{t-1}{t}\bar{G}_{t-1}(:,p) + \frac{1}{t}\tilde{x}_t^\star(p)\tilde{Y}_t^\top(:,p). \quad (15)$$

Problem (13) can then be solved using ADMM as in (4).

3.3.2. Obtaining W_t and Z_t

With the arrival of x_t, we fix the base filters to $\tilde{B}_{t-1}$ learned at the last iteration, and obtain (W_t, Z_t) from (12) as:

$$\min_{W,Z} F(W,Z)\equiv\tilde{f}_t(\tilde{B}_{t-1},W,Z)+I_{\mathcal{W}}(W)+\beta\|Z\|_1, \quad (16)$$

where $I_{\mathcal{W}}(W)$ is the indicator function on $\mathcal{W}$ (i.e., $I_{\mathcal{W}}(W)=0$ if $W\in\mathcal{W}$ and ∞ otherwise).

As in the CSC literature, it can be shown that ADMM can also be used to solve (16). While CSC's code update subproblem in (2) is convex, problem (16) is nonconvex and existing convergence results for ADMM (Wang et al., 2015) do not apply.

In this paper, we will instead use the nonconvex and inexact accelerated proximal gradient (niAPG) algorithm (Yao et al., 2017). This is a recent proximal algorithm for nonconvex problems. As the regularizers on W and Z in (16) are independent, the proximal step w.r.t. the two blocks can be performed separately as: $(\text{prox}_{I_{\mathcal{W}}(\cdot)}(W), \text{prox}_{\beta\|\cdot\|_1}(Z))$ (Parikh & Boyd, 2014). As shown in (Parikh & Boyd, 2014), these individual proximal steps can be easily computed (for $\mathcal{W}=\mathcal{W}_{\ell 1}$ or $\mathcal{W}_{\ell 2}$).

3.3.3. Complete Algorithm

The whole procedure, which will be called "Sample-dependent Convolutional Sparse Coding (SCSC)", is shown in Algorithm 1. Its space complexity, which is dominated by $\bar{H}_t$ and $\bar{G}_t$, is $O(R^2P)$. Its per-iteration time complexity is $O(RKP + RP\log P)$, where the $O(RKP)$ term is due to gradient computation, and $O(RP\log P)$ is due to FFT/inverse FFT. Table 1 compares its complexities with those of the other online and distributed CSC algorithms. As can be seen, SCSC has much lower time and space complexities as $R\ll K$.

4. Experiments

Experiments are performed on a number of data sets (Table 2). *Fruit* and *City* are two small image data sets that have been commonly used in the CSC literature (Zeiler et al., 2010; Bristow et al., 2013; Heide et al., 2015; Papyan et al., 2017). We use the default training and testing splits provided in (Bristow et al., 2013). The images are pre-processed as in (Zeiler et al., 2010; Heide et al., 2015; Wang et al., 2018), which includes conversion to grayscale, feature standardization, local contrast normalization and edge tapering. These two data sets are small. In some experiments, we will also use two larger data sets, *CIFAR-10* (Krizhevsky & Hinton, 2009) and *Flower* (Nilsback & Zisserman, 2008). Following (Heide et al., 2015; Choudhury et al., 2017; Papyan et al., 2017; Wang et al., 2018), we set the filter size M as 11×11, and the regularization parameter

Table 1. Comparing the proposed SCSC algorithm with other scalable CSC algorithms on per iteration cost. For CCSC, the cost is measured per machine, and L is the number of machines in the distributed system. Usually, $N \gg K \gg R$, and $P \gg M$.

	space	code update time	filter update time
OCSC (Wang et al., 2018)	$O(K^2P)$	$O(KP + KP\log P)$	$O(K^2P + KP\log P)$
OCDL-Degraux (Degraux et al., 2017)	$O(K^2M^2 + KPM)$	$O(K^2P^3)$	$O(K^2PM^2 + KPM)$
OCDL-Liu (Liu et al., 2017)	$O(K^2P)$	$O(KP + KP\log P)$	$O(K^2P + KP\log P)$
CCSC (Choudhury et al., 2017)	$O(\frac{NKP}{L} + KP)$	$O(\frac{NKP}{L} + NKP\log(\frac{P}{L}))$	$O(\frac{NK^2P}{L} + \frac{NKP}{L}\log(\frac{P}{L}))$
SCSC	$O(R^2P)$	$O(RKP + RP\log P)$	$O(R^2P + RP\log P)$

Algorithm 1 Sample-dependent CSC (SCSC).

Initialize $W_0 \in \mathcal{W}$, $B_0 \in \mathcal{B}$, $\bar{H}_0 = \mathbf{0}$, $\bar{G}_0 = \mathbf{0}$;
for $t = 1, 2, \ldots, T$ **do**
  draw x_t from $\{x_i\}$;
  $\tilde{x}_t = \mathcal{F}(x_t)$;
  obtain W_t, Z_t using niAPG;
  for $r = 1, 2, \ldots, R$ **do**
    $\tilde{Y}_t(:, r) = \mathcal{F}(Z_t W_t^\top(:, r))$;
  end for
  update $\{\bar{H}_t(:, :, 1), \ldots, \bar{H}_t(:, :, P)\}$ using (14);
  update $\{\bar{G}_t(:, 1), \ldots, \bar{G}_t(:, P)\}$ using (15);
  update $\bar{B}_t$ by (13) using ADMM;
end for
for $r = 1, 2, \ldots, R$ **do**
  $B_T(:, r) = \mathcal{C}(\mathcal{F}^{-1}(\tilde{B}_T(:, r)))$;
end for
output B_T.

β in (1) as 1.

Table 2. Summary of the image data sets used.

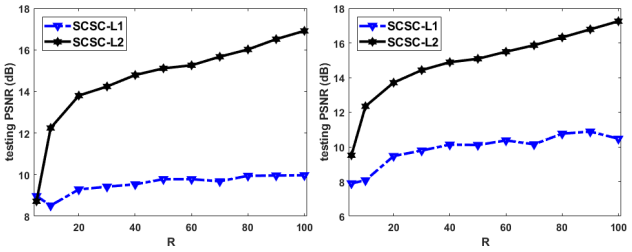
	size	#training	#testing
Fruit	100×100	10	4
City	100×100	10	4
CIFAR-10	32×32	50,000	10,000
Flower	500×500	2,040	6,149

To evaluate efficacy of the learned dictionary, we will mainly consider the task of image reconstruction as in (Aharon et al., 2006; Heide et al., 2015; Sironi et al., 2015). The reconstructed image quality is evaluated by the testing peak signal-to-noise ratio (Papyan et al., 2017): PSNR $= \frac{1}{|\Omega|}\sum_{x_j \in \Omega} 20\log_{10}\left(\frac{\sqrt{P}}{\|\hat{x}_j - x_j\|_2}\right)$, where $\hat{x}_j$ is the reconstruction of x_j from test set Ω. The experiment is repeated five times with different dictionary initializations.

4.1. Choice of $\mathcal{W}$: $\mathcal{W}_{\ell 1}$ versus $\mathcal{W}_{\ell 2}$

First, we study the choice of $\mathcal{W}$ in Proposition 2. We compare SCSC-L1, which uses $\mathcal{W} = \mathcal{W}_{\ell 1}$, with SCSC-L2, which uses $\mathcal{W} = \mathcal{W}_{\ell 2}$. Experiments are performed on *Fruit* and *City*. As in (Heide et al., 2015; Papyan et al., 2017; Wang et al., 2018), the number of filters K is set to 100. Recall the space complexity results in Table 1, we define the compression ratio of SCSC relative to OCSC (using the same K) as CR $= (K/R)^2$. We vary R in $\{K/20, K/10, K/9, \ldots, K/2, K\}$. The corresponding CR is $\{400, 100, 81, \ldots, 1\}$.

Results are shown in Figure 1. As can be seen, SCSC-L1 is much inferior. Figure 2(a) shows the weight W_j obtained with $K = 100$ and $R = 10$ by SCSC-L1 on a test sample x_j from *City* (results on the other data sets are similar). As can be seen, most of its entries are zero because of the sparsity induced by the ℓ_1 norm. The expressive power is severely limited as typically only one base filter is used to approximate the original filter. On the other hand, the W_j learned by SCSC-L2 is dense and has more nonzero entries (Figure 2(b)). In the sequel, we will only focus on SCSC-L2, which will be simply denoted as SCSC.



(a) *Fruit*. (b) *City*.

Figure 1. Testing PSNR's of SCSC-L1 and SCSC-L2 at different R's on the *Fruit* and *City* data sets.

(a) SCSC-L1. (b) SCSC-L2.

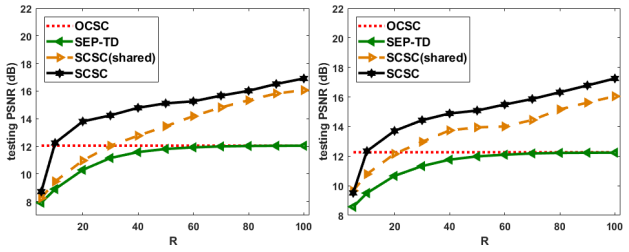
Figure 2. Weight matrices obtained on a test sample from *City*. Each column corresponds to an original filter.

4.2. Sample-Dependent Dictionary

In this experiment, we set $K = 100$, and compare SCSC with the following algorithms that use sample-independent dictionaries: (i) SCSC (shared): This is a SCSC variant in which all W_i's in (5) are the same. Its optimization is based on alternating minimization. (ii) Separable filters learned by tensor decomposition (SEP-TD) (Sironi et al., 2015), which

is based on post-processing the (shared) dictionary learned by OCSC as reviewed in Section 2.1; (iii) OCSC (Wang et al., 2018): the state-of-the-art online CSC algorithm.

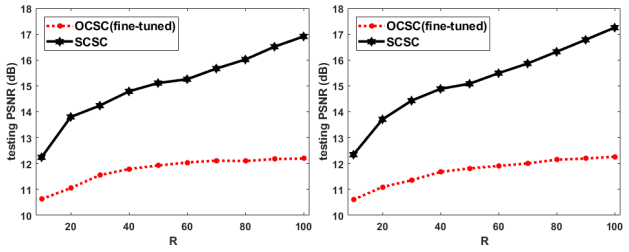
Results are shown in Figure 3. As can be seen, SCSC always outperforms SCSC(shared) and SEP-TD, and outperforms OCSC when $R = 10$ (corresponding to CR $= 100$) or above. This demonstrates the advantage of using a sample-dependent dictionary.



(a) *Fruit*. (b) *City*.

Figure 3. Testing PSNR vs R for CSC algorithms using shared and sample-dependent dictionaries.

Next, we compare against OCSC with fine-tuned filters, which are also sample-dependent. Specifically, given test sample x_j, we first obtain its code Z_j from (2) with the learned dictionary D, and then fine-tune D by solving (3) using the newly computed Z_j. As in (Donahue et al., 2014), this is repeated for a few iterations.[1] We set OCSC's K to be equal to SCSC's R, so that the two methods take the same space (Table 1). The K used in SCSC is still 100. Results are shown in Figure 4. As can be seen, though fine-tuning improves the performance of OCSC slightly, this approach of generating sample-dependent filters is still much worse than SCSC.



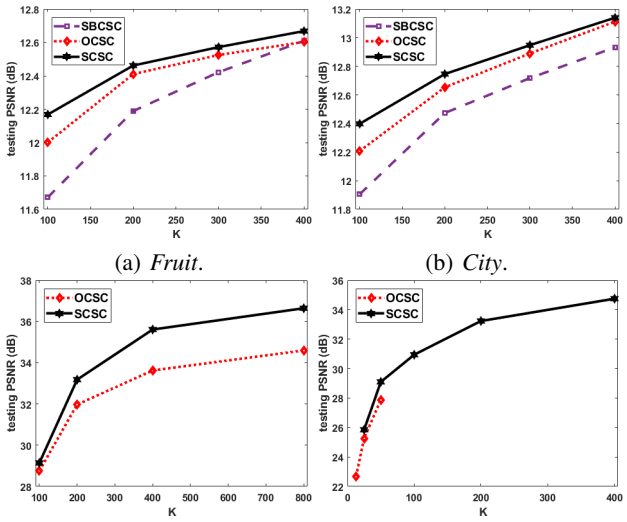
(a) *Fruit*. (b) *City*.

Figure 4. Comparison of SCSC and OCSC with fine-tuning.

4.3. Learning with More Filters

Recall that SCSC allows the use of more filters (i.e., a larger K) because of its lower time and space complexities. In this Section, we demonstrate that this can lead to better performance. We compare SCSC with two most recent batch and online CSC methods, namely, slice-based CSC (SBCSC) (Papyan et al., 2017) and OCSC. For SCSC, we set $R = 10$ for *Fruit* and *City*, and $R = 30$ for *CIFAR-10* and *Flower*.

Figure 5 shows the testing PSNR's at different K's. As can be seen, a larger K consistently leads to better performance for all methods. SCSC allows the use of a larger K because of its much smaller memory footprint. For example, on *CIFAR-10*, CR $= 1024$ at $K = 800$; on *Flower*, CR $= 1600$ at $K = 400$.



(a) *Fruit*. (b) *City*.

(c) *CIFAR-10*. (d) *Flower*.

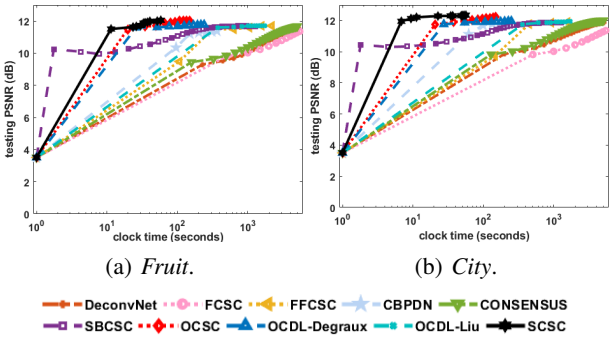
Figure 5. Effect of K on the testing PSNR. Note that SBCSC cannot be run on *CIFAR-10* and *Flower*, which are large. For OCSC, it can only run up to $K = 50$ on *Flower*.

4.4. Comparison with the State-of-the-Art

First, we perform experiments on the two smaller data sets of *Fruit* and *City*, with $K = 100$. We set $R = 10$ (i.e., CR $= 100$) for SCSC. This is compared with the batch CSC algorithms, including (i) deconvolution network (DeconvNet) (Zeiler et al., 2010), (ii) fast CSC (FCSC) (Bristow et al., 2013), (iii) fast and flexible CSC (FFCSC) (Heide et al., 2015), (iv) convolutional basis pursuit denoising (CBPDN) (Wohlberg, 2016), (v) the CONSENSUS algorithm (Šorel & Šroubek, 2016), and (vi) slice-based CSC (SBCSC) (Papyan et al., 2017). We also compare with the online CSC algorithms, including (vii) OCSC (Wang et al., 2018), (viii) OCDL-Degraux (Degraux et al., 2017), and (ix) OCDL-Liu (Liu et al., 2017).

Figure 6 shows convergence of the testing PSNR with clock time. As also demonstrated in (Degraux et al., 2017; Liu et al., 2017; Wang et al., 2018), online CSC methods converge faster and have better PSNR than batch CSC methods. Among the online methods, SCSC has comparable PSNR as OCSC, but is faster and requires much less storage (CR $= 100$).

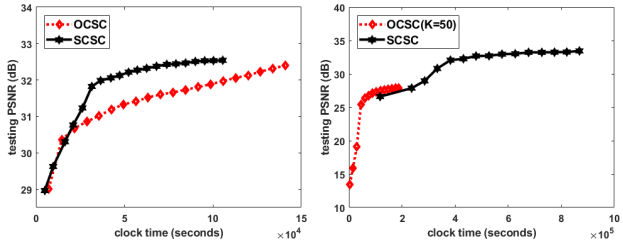
[1] In the experiments, we stop after five iterations.



(a) *Fruit*. (b) *City*.

Figure 6. Testing PSNR on the small data sets.

Next, we perform experiments on the two large data sets, *CIFAR-10* and *Flower*. All the batch CSC algorithms and two online CSC algorithms, OCDL-Degraux and OCDL-Liu, cannot handle such large data sets. Hence, we will only compare SCSC with OCSC. On *CIFAR-10*, we set $K = 300$, and the corresponding CR for SCSC is 100. On *Flower*, K is still 300 for SCSC. However, OCSC can only use $K = 50$ because of its much larger memory footprint. Figure 7 shows convergence of the testing PSNR. In both cases, SCSC significantly outperforms OCSC.



(a) *CIFAR-10*. (b) *Flower*.

Figure 7. Testing PSNR on the large data sets.

4.5. Higher-Dimensional Data

In this section, we perform experiments on data sets with dimensionalities larger than two. To alleviate the large memory problem, Choudhury *et al.* (2017) proposed the use of distributed algorithms. Here, we show that SCSC can effectively handle these data sets using one single machine.

Experiments are performed on three data sets (Table 3) in (Choudhury et al., 2017). The *Video* data set contains image subsequences recorded in an airport (Li et al., 2004). The length of each video is 7, and each image frame is of size 100×100. The *Multispectral* data contains 60×60 patches from multispectral images (covering 31 wavelengths) of real-world objects and materials (Yasuma et al., 2010). The *Light field* data contains 60×60 patches of light field images on objects and scenes (Kalantari et al., 2016). For each pixel, the light rays are from 8×8 different directions. Following (Choudhury et al., 2017), we set the filter size M to $11 \times 11 \times 11$ for *Video*, $11 \times 11 \times 31$ for *Multispectral*, and $11 \times 11 \times 8 \times 8$ for *Light field*.

Table 3. Summary of the higher-dimensional data sets used.

	size	#training	#testing
Video	100×100×7	573	143
Multispectral	60×60×31	2,200	1,000
Light field	60×60×8×8	7,700	385

We compare SCSC with OCSC and the concensus CSC (CCSC) (Choudhury et al., 2017) algorithms, with $K = 50$. For fair comparison, only one machine is used for all methods. We do not compare with the batch methods and the two online methods (OCDL-Degraux and OCDL-Liu) as they are not scalable (as already shown in Section 4.4).

Because of the small memory footprint of SCSC, we run it on a GTX 1080 Ti GPU in this experiment. OCSC is also run on GPU for *Video*. However, OCSC can only run on CPU for *Multispectral* and *Light field*. CCSC, which needs to access all the samples and codes during processing, can only be on CPU.[2]

Results are shown in Table 4. Note that SCSC is the only method that can handle the whole of *Video*, *Multispectral* and *Light field* data sets on a single machine. In comparison, CCSC can only handle a maximum of 30 *Video* samples, 40 *Multispectral* samples, and 35 *Light field* samples. OCSC can handle the whole of *Video* and *Multispectral*, but cannot converge in 2 days when the whole *Light field* data set is used. Again, SCSC outperforms OCSC and CCSC.

As for speed, SCSC is the fastest. However, note that this is for reference only as SCSC is run on GPU while the others (except for OCSC on *Video*) are run on CPU. Nevertheless, this still demonstrates an important advantage of SCSC, namely that its small memory footprint can benefit from the use of GPU, while the others cannot.

4.6. Image Denoising and Inpainting

In previous experiments, superiority of the learned dictionary is demonstrated by reconstruction of clean images. In this section, we further examine the learned dictionary on two applications: image denoising and inpainting. Ten test images provided by (Choudhury et al., 2017) are used. In denoising, we add Gaussian noise with zero mean and variance 0.01 to the test images (the average input PSNR is 10dB). In inpainting, we random sub-sample 50% of the pixels as 0 (the average input PSNR is 9.12dB). Following (Heide et al., 2015; Choudhury et al., 2017; Papyan et al.,

[2]For *Video*, the memory used (in GB) by CCSC, OCSC, SCSC (with $R = 5$) and SCSC (with $R = 10$) are 28.73, 7.58, 2.66, and 2.87, respectively. On *Multispectral*, they are 28.26, 11.09, 0.73 and 0.76; on *Light field*, they are 29.79, 15.94, 7.26 and 8.88, respectively.

Table 4. Results on the higher-dimensional data sets. PSNR is in dB and clock time is in hours. Timing results based on GPU are marked with asterisks.

		Video		*Multispectral*		*Light field*	
		PSNR	time	PSNR	time	PSNR	time
CCSC		20.43±0.11	11.91±0.07	17.67±0.14	27.88±0.07	13.70±0.09	**8.99±0.11**
OCSC		33.17±0.01	1.41±0.04*	30.12±0.02	31.19±0.02	-	-
SCSC	$R = 5$	35.30±0.02	**0.73±0.02***	30.51±0.02	**1.21±0.03***	29.30±0.03	11.12±0.07*
	$R = 10$	**38.02±0.03**	0.81±0.01*	**31.71±0.01**	1.40±0.01*	**31.70±0.02**	17.97±0.05*

Table 5. Testing PSNR (dB) on image denoising and inpainting.

	denoising			inpainting		
	SBCSC	OCSC	SCSC	SBCSC	OCSC	SCSC
Wind Mill	14.88±0.03	16.20±0.03	**17.27±0.02**	**29.76±0.13**	29.40±0.14	**29.76±0.08**
Sea Rock	14.80±0.02	16.01±0.02	**17.10±0.02**	24.92±0.06	25.04±0.04	**25.17±0.04**
Parthenon	14.97±0.02	16.33±0.01	**17.44±0.03**	27.06±0.06	26.79±0.04	**28.04±0.04**
Rolls Royce	15.23±0.01	16.27±0.01	**17.63±0.02**	24.96±0.13	24.66±0.10	**25.06±0.05**
Fence	15.21±0.04	16.53±0.02	**17.56±0.03**	26.81±0.05	26.71±0.08	**26.85±0.05**
Car	16.90±0.01	18.05±0.03	**20.06±0.05**	29.60±0.07	29.40±0.09	**30.44±0.04**
Kid	14.90±0.01	16.21±0.02	**17.22±0.03**	25.36±0.01	25.42±0.07	**25.67±0.07**
Tower	14.89±0.02	16.19±0.01	**18.36±0.05**	26.64±0.04	26.48±0.06	**26.96±0.03**
Fish	16.40±0.01	17.40±0.01	**18.61±0.02**	27.49±0.03	26.98±0.08	**27.23±0.07**
Food	16.38±0.01	17.68±0.02	**18.56±0.03**	29.96±0.05	29.62±0.08	**31.49±0.02**

2017), we use a binary weight matrix to mask out positions of the missing pixels. We use the filters learned from *Fruit* in Section 4.4. SCSC is compared with (batch) SBCSC and (online) OCSC.

Results are shown in Table 5. As can be seen, the PSNRs obtained by SCSC are consistently higher than those by the other methods. This shows that the dictionary, which yields high PSNR on image reconstruction, also leads to better performance in other image processing applications.

4.7. Solving (16): niAPG vs ADMM

Finally, we compare the performance of ADMM and niAPG in solving subproblem (16). We use a training sample x_i from *City*. The experiment is repeated five times with different (W_i, Z_i) initializations. Figure 8 shows convergence of the objective in (16) with time. As can be seen, niAPG has fast convergence while ADMM fails to converge. Figure 9 shows $\|\tilde{Y}_i - \mathcal{F}(Z_i W_i^\top)\|_F^2$, which measures violation of the ADMM constraints, with the number of iterations. As can be seen, the violation does not go to zero, which indicates that ADMM does not converge.

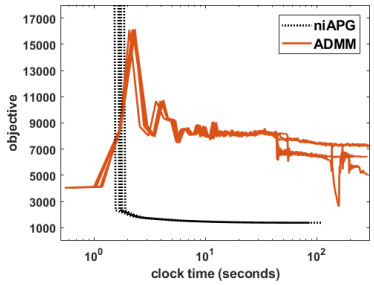


Figure 8. Convergence of niAPG and ADMM on solving (16).

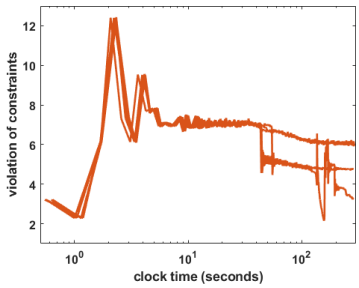


Figure 9. Constraint violation in ADMM.

5. Conclusion

In this paper, we proposed a novel CSC extension, in which each sample has its own sample-dependent dictionary constructed from a small set of shared base filters. Using online learning, the model can be efficiently updated with low time and space complexities. Extensive experiments on a variety of data sets including large image data sets and higher-dimensional data sets all demonstrate its efficiency and scalability.

Acknowledgements

The second author especially thanks Weiwei Tu and Yuqiang Chen from 4Paradigm Inc. This research was supported in part by the Research Grants Council, Hong Kong, under Grant 614513, and by the University of Macau Grant SRG2015-00050-FST.

References

Aharon, M., Elad, M., and Bruckstein, A. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

Andilla, F. and Hamprecht, F. Sparse space-time deconvolution for calcium image analysis. In *Advances in Neural Information Processing Systems*, pp. 64–72, 2014.

Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P., and Vedaldi, A. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, pp. 523–531, 2016.

Bertsekas, D. and Tsitsiklis, J. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Bristow, H., Eriksson, A., and Lucey, S. Fast convolutional sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 391–398, 2013.

Chang, H., Han, J., Zhong, C., Snijders, A., and Mao, J. Unsupervised transfer learning via multi-scale convolutional sparse coding for biomedical applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Choudhury, B., Swanson, R., Heide, F., Wetzstein, G., and Heidrich, W. Consensus convolutional sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4280–4288, 2017.

Cogliati, A., Duan, Z., and Wohlberg, B. Context-dependent piano music transcription with convolutional sparse coding. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 24(12):2218–2230, 2016.

Degraux, K., Kamilov, U. S., Boufounos, P. T., and Liu, D. Online convolutional dictionary learning for multimodal imaging. In *IEEE International Conference on Image Processing*, pp. 1617–1621, 2017.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, pp. 647–655, 2014.

Gu, S., Zuo, W., Xie, Q., Meng, D., Feng, X., and Zhang, L. Convolutional sparse coding for image super-resolution. In *International Conference on Computer Vision*, pp. 1823–1831, 2015.

Heide, F., Heidrich, W., and Wetzstein, G. Fast and flexible convolutional sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5135–5143, 2015.

Jas, M., La Tour, T. D., Simsekli, U., and Gramfort, A. Learning the morphology of brain signals using alpha-stable convolutional sparse coding. In *Advances in Neural Information Processing Systems*, pp. 1099–1108, 2017.

Jia, X., De Brabandere, B., Tuytelaars, T., and Gool, L. V. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pp. 667–675, 2016.

Kalantari, N. K., Wang, T., and Ramamoorthi, R. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics*, 35(6):193, 2016.

Kang, D., Dhar, D., and Chan, A. Incorporating side information by adaptive convolution. In *Advances in Neural Information Processing Systems*, pp. 3870–3880, 2017.

Kavukcuoglu, K., Sermanet, P., Boureau, Y., Gregor, K., Mathieu, M., and LeCun, Y. Learning convolutional feature hierarchies for visual recognition. In *Advances in Neural Information Processing Systems*, pp. 1090–1098, 2010.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Li, L., Huang, W., Gu, I. Y., and Tian, Q. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13 (11):1459–1472, 2004.

Liu, J., Garcia-Cardona, C., Wohlberg, B., and Yin, W. Online convolutional dictionary learning. In *IEEE International Conference on Image Processing*, pp. 1707–1711, 2017.

Mallat, S. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

Nilsback, M. and Zisserman, A. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729, 2008.

Pachitariu, M., Packer, A., Pettit, N., Dalgleish, H., Hausser, M., and Sahani, M. Extracting regions of interest from

biological images with convolutional sparse block coding. In *Advances in Neural Information Processing Systems*, pp. 1745–1753, 2013.

Papyan, V., Romano, Y., Sulam, J., and Elad, M. Convolutional dictionary learning via local processing. In *International Conference on Computer Vision*, pp. 5296–5304, 2017.

Parikh, N. and Boyd, S. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

Peter, S., Kirschbaum, E., Both, M., Campbell, L., Harvey, B., Heins, C., Durstewitz, D., Diego, F., and Hamprecht, F. A. Sparse convolutional coding for neuronal assembly detection. In *Advances in Neural Information Processing Systems*, pp. 3678–3688, 2017.

Rigamonti, R., Sironi, A., Lepetit, V., and Fua, P. Learning separable filters. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2754–2761, 2013.

Sironi, A., Tekin, B., Rigamonti, R., Lepetit, V., and Fua, P. Learning separable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):94–106, 2015.

Šorel, M. and Šroubek, F. Fast convolutional sparse coding using matrix inversion lemma. *Digital Signal Processing*, 55:44–51, 2016.

Wang, Y., Yin, W., and Zeng, J. Global convergence of ADMM in nonconvex nonsmooth optimization. *arXiv preprint arXiv:1511.06324*, 2015.

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. Scalable online convolutional sparse coding. *IEEE Transactions on Image Processing*, 2018.

Wohlberg, B. Efficient algorithms for convolutional sparse representations. *IEEE Transactions on Image Processing*, 25(1):301–315, 2016.

Yao, Q., Kwok, J., Gao, F., Chen, W., and Liu, T.-Y. Efficient inexact proximal gradient algorithm for nonconvex problems. In *International Joint Conferences on Artifical Intelligence*, pp. 3308–3314, 2017.

Yasuma, F., Mitsunaga, T., Iso, D., and Nayar, S. K. Generalized assorted pixel camera: postcapture control of resolution, dynamic range, and spectrum. *IEEE Transactions on Image Processing*, 19(9):2241–2253, 2010.

Zeiler, M., Krishnan, D., Taylor, G., and Fergus, R. Deconvolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2528–2535, 2010.