ONLINE CONVOLUTIONAL DICTIONARY LEARNING

Jialin Liu[a]   *Cristina Garcia-Cardona*[b]   *Brendt Wohlberg*[c]   *Wotao Yin*[a]

[a]Department of Mathematics, UCLA, Los Angeles, CA
[b]CCS Division, Los Alamos National Laboratory, Los Alamos, NM
[c]Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM

ABSTRACT

While a number of different algorithms have recently been proposed for convolutional dictionary learning, this remains an expensive problem. The single biggest impediment to learning from large training sets is the memory requirements, which grow at least linearly with the size of the training set since all existing methods are batch algorithms. The work reported here addresses this limitation by extending online dictionary learning ideas to the convolutional context.

Index Terms— Convolutional Sparse Representation, Convolutional Dictionary Learning, ADMM

1. INTRODUCTION

Sparse representations and dictionary learning have become ubiquitous techniques in signal and image processing, computer vision, and machine learning [1]. Most dictionary learning algorithms (e.g. [2, 3]) are *batch* methods in that they require the access to all training data when they start, so the training data size is limited by the amount of available memory. *Online* methods [4, 5, 6, 7], in contrast, are designed to operate on small subsets of the training data at a time, making it possible to process a large training data set with limited memory. These methods continuously aggregate past training data, updating the current learned dictionary to incorporate the sparse codes obtained for the new training data. The updates depend on accumulating the sparse codes computed while training, and do not require accessing previous sparse codes. These methods can therefore run in constant memory and at a computation cost linear in the total training data size.

Consider the linear representation, $D\mathbf{x} \approx \mathbf{s}$, where $\mathbf{s}$ is a signal of size N to represent, D is a dictionary, and $\mathbf{x}$ is the representation. A *convolutional representation* [8] implements $D\mathbf{x}$ as a sum of convolutions, $\sum_{m=1}^{M} \mathbf{d}_m * \mathbf{x}_m \approx \mathbf{s}$, where $\mathbf{d}_m$ are dictionary 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 $\mathbf{s}$. Given $\{\mathbf{d}_m\}$ and $\mathbf{s}$, a *sparse* convolutional representation can be obtained by solving the Convolutional Basis Pursuit DeNoising (CBPDN)

ℓ_1-minimization problem

$$\arg\min_{\{\mathbf{x}_m\}_{m=1}^{M}} \frac{1}{2}\left\|\sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\right\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1 \ . \quad (1)$$

Given a set of K training signals $\{\mathbf{s}_k\}_{k=1}^{K}$, the dictionary learning problem is

$$\arg\min_{\{\mathbf{d}_m\},\{\mathbf{x}_{k,m}\}} \frac{1}{2}\sum_k\left\|\sum_m \mathbf{d}_m * \mathbf{x}_{k,m} - \mathbf{s}_k\right\|_2^2 + \lambda \sum_k \sum_m \|\mathbf{x}_{k,m}\|_1$$

$$\text{subject to } \|\mathbf{d}_m\|_2 = 1, \ \forall m \in \{1, \dots, M\} \ , \quad (2)$$

where the coefficient maps $\mathbf{x}_{k,m}, m \in \{1, \dots, M\}$, represent $\mathbf{s}_k, k \in \{1, \dots, K\}$. The norm constraint avoids the scaling ambiguity between $\mathbf{d}_m$ and $\mathbf{x}_{k,m}$. As the notation is cumbersome, it is convenient to define the linear operator D_m such that $D_m\mathbf{x}_{k,m} = \mathbf{d}_m * \mathbf{x}_{k,m}$ and let $D \triangleq (D_1 \ D_2 \cdots D_M)$ and $\mathbf{x}_k \triangleq (\mathbf{x}_{k,1}^T \ \mathbf{x}_{k,2}^T \cdots \mathbf{x}_{k,M}^T)^T$. Then, we can write $D\mathbf{x}_k = \sum_{m=1}^{M} \mathbf{d}_m * \mathbf{x}_{k,m} \approx \mathbf{s}_k$.

The most recent approaches for solving (2), all of which are *batch* methods, use alternating minimization over $\{\mathbf{x}_{k,m}\}$ and $\{\mathbf{d}_m\}$, where each minimization subproblem is approximated by performing a few iterations of ADMM [9, 10, 11, 12, 13, 14]. When K is large, the $\mathbf{d}_m$ update subproblem is computationally expensive since it depends on all the KM maps $\mathbf{x}_{k,m}$ of size N each, thus preventing the use of a large training set. The purpose of the present work is to develop online dictionary learning methods for training data sets that are much larger than those that are presently feasible.

2. ONLINE DICTIONARY LEARNING

In the online setting, training signals are processed in a streaming fashion, $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \cdots, \mathbf{s}^{(t)}, \cdots$. The coefficient maps $\{\mathbf{x}_m^{(t)}\}_{m=1}^{M}$ of the tth training signal $\mathbf{s}^{(t)}$ are computed via CBPDN (1) using the latest dictionary $\{\mathbf{d}_m^{(t-1)}\}_{m=1}^{M}$,

$$\{\mathbf{x}_m^{(t)}\}_{m=1}^{M} \leftarrow \text{CBPDN}\left(\mathbf{s}^{(t)}, \{\mathbf{d}_m^{(t-1)}\}_{m=1}^{M}\right) , \quad (3)$$

which is a relatively cheap step since only the current signal $\mathbf{s}^{(t)}$ is involved. Define the loss function for $\mathbf{s}^{(t)}$

$$f^{(t)}(D) \triangleq \frac{1}{2}\left\|\sum_m \mathbf{x}_m^{(t)} * \mathbf{d}_m - \mathbf{s}^{(t)}\right\|_2^2 . \quad (4)$$

The simplest way to update the dictionary is to minimize the loss function at the current iterate,

$$D^{(t)} \leftarrow \arg\min_D f^{(t)}(D) + \iota_{\text{CPN}}(D) \,, \quad (5)$$

where ι_{CPN} is the indicator function of the constraint set for filter support and normalisation [11]. However, this approach may overfit each $\mathbf{s}^{(t)}$ and never converge to a dictionary that represents the features of the entire training sequence. A better approach, inspired by [5], introduces the *surrogate function*

$$\mathcal{F}^{(t)}(D) = f^{(1)}(D) + \cdots + f^{(t)}(D) \,, \quad (6)$$

based on which, the updated dictionary is computed as,

$$D^{(t)} \leftarrow \arg\min_D \mathcal{F}^{(t)}(D) + \iota_{\text{CPN}}(D) \,. \quad (7)$$

Modified variants of (7) are proposed in Section 2.1 below, with its fast algorithm presented in Section 2.2.

2.1. Acceleration via Modified Surrogate Function

At the current 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 information provided by the new training samples is to compute a weighted combination of these contributions, as routinely done in other online schemes [4, 5]. This combination considers more strongly the more recent updates, since those are the result of a more extensively trained dictionary.

Consider the surrogate function (6) written recursively as

$$\mathcal{F}^{(t)}(D) = \mathcal{F}^{(t-1)}(D) + f^{(t)}(D) \,, \quad (8)$$

and instead of a direct combination, use a factor to weight past (i.e. outdated) contributions

$$\mathcal{F}_{\text{mod}}^{(t)}(D) = \alpha^{(t)} \mathcal{F}_{\text{mod}}^{(t-1)}(D) + f^{(t)}(D) \,. \quad (9)$$

Here $\alpha^{(t)} \in (0, 1)$ is a *forgetting factor*, which has its own time evolution [5]

$$\alpha^{(t)} = (1 - 1/t)^p \quad (10)$$

regulated by the *forgetting exponent* $p > 0$. This is a reasonable choice since, 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.

A large forgetting factor α (small p) can be expected to lead to a stable algorithm since all the training signals are given nearly equal weights as their information is accumulated in D. However, it also leads to slow convergence. An extreme case is as $p \to 0$, $\alpha^{(t)} \to 1$, which recovers (6). A small forgetting factor, conversely, leads to faster convergence since it gives past, less accurate information lower weights. But if the factor is too small (p is too large), the surrogate function is overwhelmingly influenced by the current training signal $\mathbf{s}^{(t)}$, causing the convergence to be unstable. As

---

Algorithm 1: Main Algorithm: Online Convolutional Dictionary Learning

Initiate: Initialize $D^{(0)}$ with random dictionary.
$\hat{A}_{\text{mod}}^{(0)} \leftarrow 0, \hat{\mathbf{b}}_{\text{mod}}^{(0)} \leftarrow 0$.

1 **for** $t = 0, 1, \cdots, T$ **do**
2   Sample a signal $\mathbf{s}^{(t)}$.
3   Solve sparse coding problem (3).
4   Compute FFT: $\hat{\mathbf{x}}^{(t)} = \text{FFT2}(\mathbf{x}^{(t)})$.
5   Accumulate with forgetting factor $\alpha^{(t)}$,

$$\begin{aligned} \hat{A}_{\text{mod}}^{(t)} &\leftarrow \alpha^{(t)} \hat{A}_{\text{mod}}^{(t-1)} + (\hat{\mathbf{x}}^{(t)})^H (\hat{\mathbf{x}}^{(t)}) \\ \hat{\mathbf{b}}_{\text{mod}}^{(t)} &\leftarrow \alpha^{(t)} \hat{\mathbf{b}}_{\text{mod}}^{(t-1)} + (\hat{\mathbf{x}}^{(t)})^H (\hat{\mathbf{s}}^{(t)}) \end{aligned} \quad (12)$$

6   D-update: Solve (11) via Algorithm 2.
7 **end**

Output: $D^{(T)}$

$p \to \infty$, we have $\alpha^{(t)} \to 0$, so only the loss function $f^{(t)}(D)$ of the current $\mathbf{s}^{(t)}$ is considered, which recovers (5). Based on the modified surrogate function $\mathcal{F}_{\text{mod}}^{(t)}(D)$ in (9), the dictionary update (7) is modified correspondingly to

$$D^{(t)} \leftarrow \arg\min_D \mathcal{F}_{\text{mod}}^{(t)}(D) + \iota_{\text{CPN}}(D) \,. \quad (11)$$

2.2. Minimizing Modified Surrogate Function

A popular approach for solving quadratic minimization problems like (7) and (11) is Fast Iterative Shrinkage-Thresholding (FISTA) [15], which computes a gradient at each step. According to the notation in Section 1, each loss function can be written as $f^{(t)}(D) = \frac{1}{2}\|D\mathbf{x}^{(t)} - \mathbf{s}^{(t)}\|_2^2$. Thus, the gradient for the surrogate function can be computed as

$$\nabla \mathcal{F}^{(t)}(D) = \left(\sum_{\tau=1}^{t} (\mathbf{x}^{(\tau)})^T \mathbf{x}^{(\tau)} \right) D - \left(\sum_{\tau=1}^{t} (\mathbf{x}^{(\tau)})^T \mathbf{s}^{(\tau)} \right) . \quad (13)$$

We cannot follow this formula directly since the cost would increase linearly in t. Instead we perform the iterative updates [5]

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

at a constant cost. So, at a constant cost, we can also compute $\nabla \mathcal{F}^{(t)}(D) = A^{(t)} D - \mathbf{b}^{(t)}$.

A further significant improvement is to take advantage of the convolutional property of the linear operator D. The convolution is implemented in the frequency domain, using the Fast Fourier Transform (FFT). Inspired by the frequency domain FISTA variant for the CBPDN problem [11], we propose a frequency domain FISTA to solve (11), which is described in Algorithm 2. Each loss function in the frequency domain has the form

$$f^{(t)}(\hat{D}) \triangleq \frac{1}{2}\left\|\hat{D}\hat{\mathbf{x}}^{(t)} - \hat{\mathbf{s}}^{(t)}\right\|_2^2 = \frac{1}{2}\left\|\sum_m \hat{\mathbf{d}}_m \odot \hat{\mathbf{x}}_m^{(t)} - \hat{\mathbf{s}}^{(t)}\right\|_2^2 , \quad (14)$$

Algorithm 2: D-update: Frequency domain FISTA for solving (11)

Input: Information matrix $\hat{A}^{(t)}_{\text{mod}}$ and $\hat{\mathbf{b}}^{(t)}_{\text{mod}}$.
Initiate: Let $G^0 = D^{(t-1)}, \hat{G}^0 = \text{FFT2}(G^0)$.
  Auxiliary variable $G^0_{\text{aux}} = G^0$.
  Let $\gamma^0 = 0$ for acceleration.

1 **for** $j = 0, 1, 2, \ldots$ *until convergence* **do**

2 Compute FFT, $\hat{G}^j_{\text{aux}} = \text{FFT2}(G^j_{\text{aux}})$.

3 Compute gradient in frequency domain,

$$\nabla \mathcal{F}^{(t)}_{\text{mod}}(\hat{G}^j_{\text{aux}}) = \hat{A}^{(t)}_{\text{mod}} \hat{G}^j_{\text{aux}} - \hat{\mathbf{b}}^{(t)}_{\text{mod}} . \qquad (15)$$

4 Compute dictionary,

$$G^{j+1} \leftarrow \text{proj}_{C_{PN}}\left(\text{IFFT2}\left(\hat{G}^j_{\text{aux}} - \eta \nabla \hat{f}^{(t)}(\hat{G}^j_{\text{aux}}) \right) \right) . \qquad (16)$$

  Compute auxiliary dictionary (Nesterov acceleration) for next step,

$$\gamma^{j+1} = \left(1 + \sqrt{1 + 4(\gamma^j)^2} \right)/2 ,$$

$$G^{j+1}_{\text{aux}} = G^{j+1} + \frac{\gamma^j - 1}{\gamma^{j+1}}(G^{j+1} - G^j) . \qquad (17)$$

5 **end**

Output: $D^{(t)} \leftarrow G^J$, where J is the last iterate.

---

where $\hat{\cdot}$ denote frequency-domain values, $\hat{\mathbf{x}}^{(t)}$ is obtained by applying the Fourier transforms to each $\mathbf{x}^{(t)}_m$ in $\mathbf{x}^{(t)}$, and "$\odot$" is point-wise multiplication. Then, the gradient can be computed as $\nabla \mathcal{F}^{(t)}(\hat{D}) = \hat{A}^{(t)}\hat{D} - \hat{\mathbf{b}}^{(t)}$, where $\hat{A}^{(t)}$ and $\hat{\mathbf{b}}^{(t)}$ are iteratively updated through

$$\hat{A}^{(t)} = \hat{A}^{(t-1)} + (\hat{\mathbf{x}}^{(t)})^H \hat{\mathbf{x}}^{(t)} , \quad \hat{\mathbf{b}}^{(t)} = \hat{\mathbf{b}}^{(t-1)} + (\hat{\mathbf{x}}^{(t)})^H \hat{\mathbf{s}}^{(t)}.$$

This accumulation is weighted by the forgetting factor, as derived from the modified surrogate function, to yield expression (12). The main algorithm is given in Algorithm 1. Its D-update step calls *frequency domain FISTA* which is described in Algorithm 2.

Direct extension of the online approach for regular dictionary learning leads to matrices $A^{(t)}$ and $\hat{A}^{(t)}$ of size $\mathcal{O}(M^2N^2)$, which is prohibitive except for very small N (M is usually much smaller than N, so the quadratic order is less problematic). However, since the frequency-domain product $(\hat{\mathbf{x}}^{(t)})^H \hat{\mathbf{x}}^{(t)}$ has only $\mathcal{O}(M^2N)$ non-zero values, this structure can be exploited to obtain a corresponding reduction in storage requirements for $\hat{A}^{(t)}$.

2.3. Region Sampling (Limited Memory Version)

In Section 2.2, the information matrix $\hat{A}^{(t)}_{\text{mod}}$ of size $\mathcal{O}(M^2N)$ 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, $\hat{A}^{(t)}_{\text{mod}}$ requires a large amount of memory. To reduce memory size, we sample small regions of the whole signal. Specifically, given a current signal $\mathbf{s}^{(t)} \in N$, we sample small regions $\mathbf{s}^{(t)}_{\text{samp},1}, \mathbf{s}^{(t)}_{\text{samp},2}, \ldots \in \tilde{N}$, with $\tilde{N} < N$, and treat them as if they were different signals.

In this way, the training signal sequence becomes:

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

In our experiments, we sample each 256×256 image to obtain small 64×64 regions before Algorithm 1 is called. For the experiments reported here we use circular boundary conditions rather than the more careful boundary handling [10, 14] that would be necessary for smaller regions. We call this approach "Online-Samp.".

3. RESULTS

All the experiments are conducted using MATLAB R2016a running on a workstation with 2 Intel Xeon(R) X5650 CPUs clocked at 2.67GHz. The dictionary size is $8 \times 8 \times 32$ and the training and testing image size is 256×256. As in [5], dictionaries are evaluated by comparing the functional values obtained by computing CBPDN (1) on the test set. The training set consists of 50 images and the test set consists of 5 separate images. Four of the training set images were standard images (Lena, Barbara, Kiel, Mandrill), and the remainder of the training images, and the testing images, were cropped and rescaled from a set of images, of a variety of scenes, obtained from Flickr.

3.1. Effect of Forgetting Exponent p

An efficient algorithm requires a good choice of the forgetting factor $\alpha^{(t)}$. In this paper, we used the forgetting factor evolution defined in (10), where p is the crucial exponent parameter to tune, as we discussed in Section 2.1. The experiments reported in Fig 1 compare the convergence resulting from difference choices of p for the full training set of 50 images. In Fig 1(a), when $1 \leq p \leq 10$, the algorithm is faster when p increases ($p = 5$ is almost the same with $p = 10$). In Fig 1(b), if p increases continuously, the algorithm gets more and more unstable (for $p = \infty$, the algorithm reduces to the "naive" update scheme (5)). Thus, $p = 5$ is a reasonable choice.

3.2. Comparison with Batch Learning

In this section, online convolutional dictionary learning with a set of 50 training images and with parameter $p = 5$ is compared with batch learning [11, 16] with training sizes of 10, 20, and 50. The result is shown in Fig 2. According to this figure, online convolutional dictionary learning is the best after 200 seconds. For batch learning, a small training set leads to inaccurate results but a large training set leads to a large computational cost. Online learning handles only one image at a time, and accumulates the previous information in a compact way, making it efficient for a large training set.
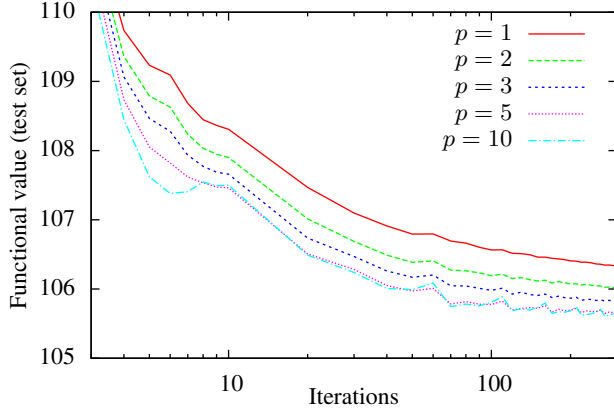
(a) $1 \le p \le 10$: larger p leads to faster convergence.
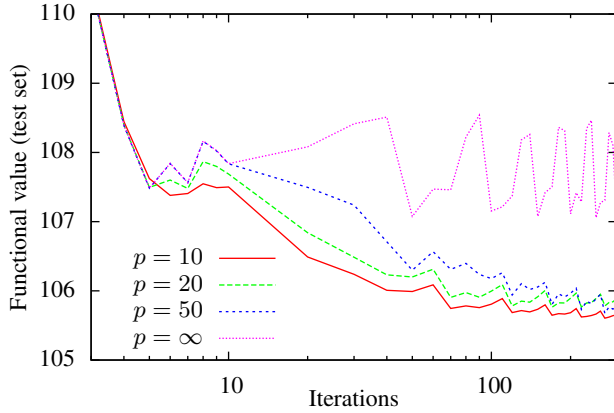
(b) $10 \le p \le \infty$: too large p leads to instability.

Fig. 1. A comparison of the convergence behaviour of the online dictionary learning algorithm for different forgetting exponents p. Note that the functional value is evaluated on the testing set, not the training set.
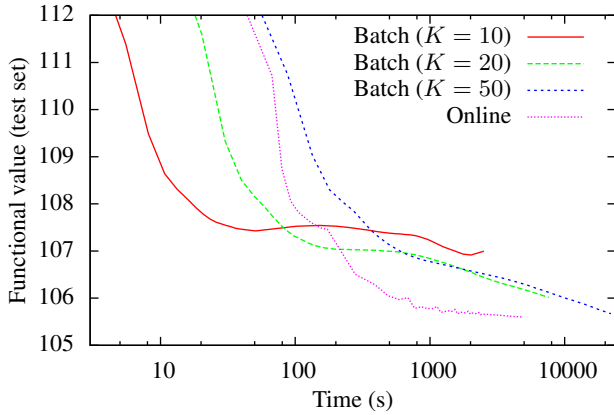

Fig. 2. Computation time comparison of batch and online learning. Note that the functional value is evaluated on the testing set, not the training set.
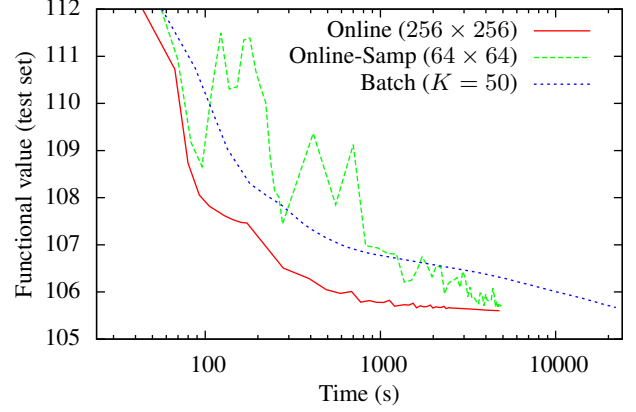

Fig. 3. A comparison of Online, Online-samp, and Batch algorithms with a training set of 50 images. The functional value is evaluated on the testing set, not the training set.

Schemes	Memory (MB)
Batch ($K = 10$)	618
Batch ($K = 20$)	1197
Batch ($K = 50$)	2902
Online (256×256)	1213
Online-Samp. (64×64)	133

Table 1. Memory Usage Comparison in Megabytes.

3.3. Region Sampling (Limited Memory Version)

To avoid high memory usage, we use the technique proposed in Section 2.3, and consider a sample region with size 64×64. We found experimentally that $p = 40$ is a good choice for size 64×64. As shown in Fig. 3, the Online-Samp. scheme is not as stable as online scheme without sampling, but it still shows good performance after enough iterations, and requires substantially less memory, as shown in Table 1.

4. CONCLUSIONS

We have proposed the first online convolutional dictionary learning algorithms capable of learning from a training image set of arbitrary size. Our approaches are based on an extension of ideas from online dictionary learning for standard sparse representations. The first of these processes an entire training image at a time; while the $\mathcal{O}(NM^2)$ memory cost is vastly better than the $\mathcal{O}(N^2M^2)$ cost that would correspond to a direct extension of prior methods for standard sparse representations, it can still be high when M is large. The second approach further reduces memory usage by sampling regions from each training image, at the expense of somewhat worse convergence behaviour.

5. REFERENCES

[1] J. Mairal, F. Bach, and J. Ponce, "Sparse modeling for image and vision processing," *Foundations and Trends in Computer Graphics and Vision*, vol. 8, no. 2-3, pp. 85–283, 2014. doi:10.1561/0600000058

[2] M. Aharon, M. Elad, and A. M. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006. doi:10.1109/tsp.2006.881199

[3] K. Engan, S. O. Aase, and J. H. Husøy, "Method of optimal directions for frame design," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, vol. 5, 1999, pp. 2443–2446. doi:10.1109/icassp.1999.760624

[4] K. Skretting and K. Engan, "Recursive least squares dictionary learning algorithm," *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2121–2130, Apr. 2010. doi:10.1109/tsp.2010.2040671

[5] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[6] C. Lu, J. Shi, and J. Jia, "Online robust dictionary learning," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, June 2013, pp. 415–422. doi:10.1109/CVPR.2013.60

[7] Y. Naderahmadian, M. A. Tinati, and S. Beheshti, "Generalized adaptive weighted recursive least squares dictionary learning," *Signal Processing*, vol. 118, pp. 89–96, 2016. doi:10.1016/j.sigpro.2015.06.013

[8] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, Jun. 2010, pp. 2528–2535. doi:10.1109/cvpr.2010.5539957

[9] H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, Jun. 2013, pp. 391–398. doi:10.1109/CVPR.2013.57

[10] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, 2015, pp. 5135–5143. doi:10.1109/CVPR.2015.7299149

[11] B. Wohlberg, "Efficient algorithms for convolutional sparse representations," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 301–315, Jan. 2016. doi:10.1109/TIP.2015.2495260

[12] M. Šorel and F. Šroubek, "Fast convolutional sparse coding using matrix inversion lemma," *Digital Signal Processing*, vol. 55, pp. 44–51, 2016. doi:10.1016/j.dsp.2016.04.012

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

[14] ——, "Boundary handling for convolutional sparse representations," in *Proc. IEEE Conf. Image Process. (ICIP)*, Phoenix, AZ, USA, Sep. 2016, pp. 1833–1837. doi:10.1109/ICIP.2016.7532675

[15] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009. doi:10.1137/080716542

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