Fast and Flexible Convolutional Sparse Coding

Felix Heide
Stanford University, UBC

Wolfgang Heidrich
KAUST, UBC

Gordon Wetzstein
Stanford University

Abstract

Convolutional sparse coding (CSC) has become an increasingly important tool in machine learning and computer vision. Image features can be learned and subsequently used for classification and reconstruction tasks. As opposed to patch-based methods, convolutional sparse coding operates on whole images, thereby seamlessly capturing the correlation between local neighborhoods. In this paper, we propose a new approach to solving CSC problems and show that our method converges significantly faster and also finds better solutions than the state of the art. In addition, the proposed method is the first efficient approach to allow for proper boundary conditions to be imposed and it also supports feature learning from incomplete data as well as general reconstruction problems.

1. Introduction

An increasing number of computer vision tasks rely on natural image statistics. Low-level problems that benefit from good priors include inpainting, denoising, deblurring, and super-resolution, while recognition, classification and other higher-level tasks often use learned features as priors for natural images. In this paper, we revisit one strategy for unsupervised learning of image features: convolutional sparse coding (CSC). CSC was introduced in the context of modeling receptive fields in human vision [18], but it has recently been demonstrated to have important applications in a wide range of computer vision problems such as low/mid-level feature learning, low-level reconstruction [21, 7], as part of more complex hierarchical structures or networks in high-level computer vision challenges [13, 22, 23], and in physically-motivated computational imaging problems [12, 11]. Beyond these applications, CSC could find applications in many other reconstruction tasks and feature-based methods, including deblurring, denoising, inpaiting, classification, localization, and tracking.

CSC is closely related to popular patch-based learning and reconstruction methods [5, 16, 24]. However, features learned with patch-based methods often contain shifted versions of the same features and latent structures of the underlying signal may be lost when dividing it into small patches. A more elegant way to model many of these problems is to use a sum of sparsely-distributed convolutional features. The main drawback of convolutional models, however, is their computational complexity. Not only is it very challenging to find a solution to convolutional sparse coding problems in a reasonable amount of time, but finding a good local minimum is difficult as well. Generally, CSC is a non-convex problem and many existing methods provide little to no guarantees for global convergence. Seminal advances in fast convolutional sparse coding have recently shown that feature learning via CSC can be efficiently solved in the frequency domain. Grosse et al. [9] were the first to propose a frequency domain method for 1D audio signals, while [3, 4, 14] later demonstrate efficient frequency domain approaches for 2D image data. While this is the first step towards making CSC practical, these frequency methods can introduce boundary artifacts for both learning and reconstruction [13] and, as inherently global approaches, make it difficult to work with incomplete data.

Building on recent advances in optimization [6, 2, 19, 1, 20], we propose a new splitting-based approach to convolutional sparse coding. We not only show that our formulation allows us to easily incorporate proper boundary conditions and learn from sparse observations, but we also demonstrate that the proposed method is faster and converges to better solutions than the state of the art. In particular, we make the following contributions:

- We derive a flexible formulation of the convolutional sparse coding problem, and an efficient solution by splitting the objective into a sum of simple, convex functions. This formulation fits into most recent forward-backward optimization frameworks.
- We demonstrate that the proposed method allows for proper boundary conditions to be imposed without sacrificing performance; it converges faster than alternative methods and finds better solutions. We verify the latter using several low-level reconstruction problems.
- We show feature learning from incomplete observations, which has not been demonstrated with existing

CSC solvers.

- We show that the proposed solver is also efficient when working with known features, such as mixtures of Gaussians or physically-motivated convolutional bases.

2. Mathematical Framework

Traditionally, convolutional sparse coding problems are expressed in the form

$$\begin{aligned} &\underset{\mathbf{d},\mathbf{z}}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{x}-\sum_{k=1}^{K}\mathbf{d}_k * \mathbf{z}_k\|_2^2 + \beta\sum_{k=1}^{K}\|\mathbf{z}_k\|_1 \\ &\text{subject to} \quad \|\mathbf{d}_k\|_2^2 \leq 1 \quad \forall k \in \{1,\dots,K\}, \end{aligned} \tag{1}$$

where $\mathbf{z}_k$ are sparse feature maps that approximate the data term $\mathbf{x}$ when convolved with the corresponding filters $\mathbf{d}_k$ of fixed spatial support. Here $\mathbf{x} \in \mathbb{R}^D, \mathbf{z}_k \in \mathbb{R}^D$ are vectorized images, $\mathbf{d}_k \in \mathbb{R}^M$ are the vectorized 2D filters, $k = 1 \dots K$, and $*$ is the 2D convolution operator defined on the vectorized inputs. While the above equation is strictly only valid for a single target image, it can easily be generalized to multiple images $\mathbf{x}$.

Recently Bristow et al. [3, 4] have shown remarkable improvements in efficiency by exploiting Parseval's theorem for solving Eq. (1), which states that the energy of a signal is equivalent — up to a constant — to that of its Fourier transform. We will neglect this constant in the following. Eq. 1 can therefore be reformulated [3, 4, 14] as

$$\begin{aligned} &\underset{\mathbf{d},\mathbf{z}}{\operatorname{argmin}} \ \frac{1}{2}\|\hat{\mathbf{x}}-\sum_{k=1}^{K}\hat{\mathbf{d}}_k \odot \hat{\mathbf{z}}_k\|_2^2 + \beta\sum_{k=1}^{K}\|\mathbf{t}_k\|_1 \\ &\text{subject to} \quad \|\mathbf{s}_k\|_2^2 \leq 1 \quad \forall k \in \{1,\dots,K\} \\ &\qquad\quad \mathbf{s}_k = \mathbf{S}\Phi^T\hat{\mathbf{d}}_k \quad \forall k \in \{1,\dots,K\} \\ &\qquad\quad \mathbf{t}_k = \mathbf{z}_k \quad \forall k \in \{1,\dots,K\}, \end{aligned} \tag{2}$$

which expresses the computationally expensive convolution operations as more efficient multiplications in the frequency domain. Here, $\hat{}$ denotes the frequency representation of a signal, $\odot$ is the component-wise product, Φ is the discrete Fourier transform (DFT) matrix, and $\mathbf{S}$ projects a filter onto its (small) spatial support. The slack variables $\mathbf{s}_k$ and $\mathbf{t}_k$ allow Eq. 2 to be solved by splitting the objective into multiple subproblems that each can be solved efficiently.

Is important to note that Eqs. 1 and 2 are actually only equivalent under the assumption of circular boundary conditions [22]. Kavukcuoglu et al. [13] point out that boundary conditions are one essential hurdle in the convolutional model that affects the optimization even for non-circular boundary conditions, because pixels close to the boundary are, in general, covered by fewer filters than center pixels. While heuristics [3] might be acceptable for learning filters with very small spatial support, this assumption does not necessarily hold for larger filters or for general reconstruction problems. We propose the following, general formulation for convolutional sparse coding:

$$\begin{aligned} &\underset{\mathbf{d},\mathbf{z}}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{x}-\mathbf{M}\sum_{k=1}^{K}\mathbf{d}_k * \mathbf{z}_k\|_2^2 + \beta\sum_{k=1}^{K}\|\mathbf{z}_k\|_1 \\ &\text{subject to} \quad \|\mathbf{d}_k\|_2^2 \leq 1 \quad \forall k \in \{1,\dots,K\}. \end{aligned} \tag{3}$$

Here, $\mathbf{M}$ is a diagonal or block-diagonal matrix, such that it decouples linear systems of the form $(\mathbf{M}^T\mathbf{M}+\mathbb{I})\mathbf{x} = \mathbf{b}$ into many small and independent systems that are efficiently solved. For example, for boundary handling $\mathbf{M}$ can be a binary diagonal matrix that masks out the boundaries of the padded estimation $\sum_{k=1}^{K}\mathbf{d}_k * \mathbf{z}_k$. This allows us to use unmodified filters in boundary regions, thus preserving the convolutional nature of the problem without requiring circular boundaries or other conditions. Furthermore, we show that $\mathbf{M}$ allows for efficient learning and reconstruction from incomplete data.

Unfortunately, Eq. 3 cannot be solved directly with the "Fourier trick" discussed in the literature (Eq. 2). In the following, we derive a formulation that is not only *flexible* in allowing us to solve Eq. 3 efficiently, but we also show that our formulation solves the conventional convolutional sparse coding problem (Eq. 1) *faster* than previous methods and *converges to better solutions*.

2.1. Efficient Splitting of the Objective

To efficiently solve Eq. 3, we reformulate it such that the constraints are included in the objective via an indicator function $\mathrm{ind}_C(\cdot)$, which is defined on the convex set of the constraints $C = \{\mathbf{v} \mid \|\mathbf{S}\mathbf{v}\|_2^2 \leq 1\}$. This yields the following unconstrained objective:

$$\underset{\mathbf{d},\mathbf{z}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}-\mathbf{M}\sum_{k=1}^{K}\mathbf{d}_k * \mathbf{z}_k\|_2^2 + \beta\sum_{k=1}^{K}\|\mathbf{z}_k\|_1 + \sum_{k=1}^{K}\mathrm{ind}_C(\mathbf{d}_k), \tag{4}$$

which can be expressed as the following sum of functions

$$\underset{\mathbf{d},\mathbf{z}}{\operatorname{argmin}} \ f_1(\mathbf{D}\mathbf{z}) + \sum_{k=1}^{K}\left(f_2(\mathbf{z}_k) + f_3(\mathbf{d}_k)\right), \text{ with} \tag{5}$$

$$f_1(\mathbf{v}) = \frac{1}{2}\|\mathbf{x}-\mathbf{M}\mathbf{v}\|_2^2, f_2(\mathbf{v}) = \beta\|\mathbf{v}\|_1, f_3(\mathbf{v}) = \mathrm{ind}_C(\mathbf{v}).$$

Here, $\mathbf{z} = [\mathbf{z}_1^T \dots \mathbf{z}_K^T]^T$ and $\mathbf{D} = [\mathbf{D}_1 \dots \mathbf{D}_K]$ is a concatenation of Toeplitz matrices, each one representing a convolution with the respective filter $\mathbf{d}_k$. Eq. (5) consists of a sum of functions f_i, which are all simple to optimize individually, whereas their sum is challenging. Following [1], we define f_1 with $\mathbf{M}$ included because that splits the data term into two different subproblems involving $\mathbf{M}$ and $\mathbf{D}$ *separately but never jointly*.

2.2. Generalization of the Objective

To derive this result more intuitively, we consider the general objective from (5)

$$\underset{\mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^{I} f_i\left(\mathbf{K}_i \mathbf{z}\right), \tag{6}$$

where $\mathbf{K}_i : \mathbb{R}^{b_i \times a_i}$ are arbitrary matrices, $f_i : \mathbb{R}^{b_i} \to \mathbb{R}$ are closed, proper, convex functions, and $i \in \{1, \ldots, I\}$, such that $f_i(\mathbf{K}_j \cdot) : \mathbb{R}^{a_i} \to \mathbb{R}$; I is the number of functions in the sum.

Eq. 6 is motivated by recent work in image deconvolution [10, 1], which have a similar objective that consists of a sum of simple convex functions. The problem in Eq. 6 can be reformulated as

$$\begin{aligned} \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^{I} f_i\left(\mathbf{K}_i \mathbf{z}\right) = f\left(\mathbf{K}\mathbf{z}\right), \quad \text{with} \\ \mathbf{K} = \begin{bmatrix} \mathbf{K}_1 \\ \vdots \\ \mathbf{K}_I \end{bmatrix} \quad \text{and} \quad f(\mathbf{v}) = \sum_{i=1}^{I} f_i(\mathbf{v}_i), \end{aligned} \tag{7}$$

where $\mathbf{v}_i$ selects the i-th support of $\mathbf{v}$. Using a formulation with the stacked matrix $\mathbf{K}$ allows us to remap Eq. (6) to existing optimization frameworks, as shown for Chambolle and Pock's method in [10], for ADMM in [1], and for stochastic optimization methods in [20].

For clarity, we describe only one possible method to solve Eq. (7): ADMM, which solves problems of the form:

$$\underset{\mathbf{z}}{\operatorname{argmin}}\ h(\mathbf{y}) + g(\mathbf{z}) \quad \text{subject to} \quad \mathbf{A}\mathbf{y} = \mathbf{z}. \tag{8}$$

A detailed discussion of related algorithms can be found in [2]. Using the standard scaled form [2] and setting $h = 0$, $\mathbf{A} = \mathbf{K}$ and $g = f$, which may seem unintuitive before deriving the algorithm, yields Alg. 1, that solves Eq. (7). We observe that the resulting minimization becomes separable in all the f_i.

Algorithm 1 ADMM for a sum of functions in Eq. (7)

1: **for** $k = 1$ to V **do**
2: $\quad \mathbf{y}^{k+1} = \underset{\mathbf{y}}{\operatorname{argmin}}\ \|\mathbf{K}\mathbf{y} - \mathbf{z} + \lambda^k\|_2^2$
3: $\quad \mathbf{z}_i^{k+1} = \mathbf{prox}_{\frac{f_i}{\rho}}(\mathbf{K}_i \mathbf{y}_i^{k+1} + \lambda_i^k) \quad \forall i \in \{1, \ldots, I\}$
4: $\quad \lambda^{k+1} = \lambda^k + (\mathbf{K}\mathbf{y}^{k+1} - \mathbf{z}^{k+1})$
5: **end for**

Since in this CSC formulation, $\mathbf{M}$ is included in f_1, we can solve the update in line 2 of Alg. 1 efficiently in the Fourier domain. Note that the combination of splitting $\mathbf{M}$ with the filter generation $\mathbf{D}\mathbf{z}$ via $f_1(\mathbf{D}\mathbf{z})$ leads to a non-standard application of ADMM; the standard approach [2] would combine $\mathbf{MD}$ as a single operator. The notational shortcut $\mathbf{prox}_{\cdot}$ is the proximal operator as defined in [19]. Many different proximal operators are known in the literature [19]; formulating our method using these operators allow us to easily apply the known derivations. Although we derived a solution to Eq. 7 using ADMM, this is equally possible for Chambolle and Pock's method or stochastic optimization [20].

2.3. ADMM for CSC-specific Subproblems

We continue to discuss how the general ADMM-based algorithm described in the previous subsection is applied to convolutional sparse coding. Solving the first quadratic subproblem from Alg. 1 (line 2) gives

$$\mathbf{y}_{\text{opt}} = \underset{\mathbf{y}}{\operatorname{argmin}}\ \|\mathbf{K}\mathbf{y} - \tau\|_2^2 = (\mathbf{K}^T\mathbf{K})^{-1}(\mathbf{K}^T\tau) \tag{9}$$

Here, we have set $\tau = \mathbf{z} - \lambda^k$ as a notational shortcut. Depending on whether we solve for the filters (i.e. $\mathbf{y} = \mathbf{d}$) or for the feature maps (i.e. $\mathbf{y} = \mathbf{z}$), we get:

$$\begin{aligned} \mathbf{d}_{\text{opt}} &= (\mathbf{Z}^\dagger\mathbf{Z} + 2\mathbb{I})^{-1}(\mathbf{Z}^\dagger\tau_1 + \tau_2 + \tau_3) \quad \text{for } \mathbf{y} = \mathbf{d} \\ \mathbf{z}_{\text{opt}} &= (\mathbf{D}^\dagger\mathbf{D} + \mathbb{I})^{-1}(\mathbf{D}^\dagger\tau_1 + \tau_2) \quad \text{for } \mathbf{y} = \mathbf{z} \end{aligned} \tag{10}$$

Here $\mathbf{Z}$ is a concatenation of Toeplitz matrices for the respective sparse codes $\mathbf{z}_k$ and τ_i selects again the i-th support of τ as defined for Eq. (7). The operator $\cdot^\dagger$ defines here the conjugate transpose, with notation borrowed from [14]. In both cases, one can find a variable reordering for the equations systems in Eq. (10), that makes $(\mathbf{Z}^\dagger\mathbf{Z} + 2\mathbb{I})$ and $(\mathbf{D}^\dagger\mathbf{D} + \mathbb{I})$ block-diagonal [3, 14], which makes the inversion efficient by parallelization over the $j \in \{1 \ldots D\}$ different blocks. The inverse can be efficiently computed for each block j using the Woodbury formula, giving

$$\begin{aligned} (\mathbf{Z}_j^\dagger\mathbf{Z}_j + 2\mathbb{I})^{-1} &= \frac{1}{2}\mathbb{I} - \frac{1}{2}\mathbf{Z}_j^\dagger(2\mathbb{I} + \mathbf{Z}_j\mathbf{Z}_j^\dagger)^{-1}\mathbf{Z}_j \\ (\mathbf{D}_j^\dagger\mathbf{D}_j + \mathbb{I})^{-1} &= \mathbb{I} - \frac{\mathbf{D}_j^\dagger\mathbf{D}_j}{1 + \mathbf{D}_j\mathbf{D}_j^\dagger}, \end{aligned} \tag{11}$$

where the second equation holds, since a block in $\mathbf{D}_j$ is just a row-vector. We compute the first inverse $(2\mathbb{I} + \mathbf{Z}_j\mathbf{Z}_j^\dagger)^{-1}$ by computing its Cholesky factorization. In contrast to the direct inversion in [4] (due to the code update, this has to be done in each iteration of their method), caching this factorization leads to a significantly decreased running time as described below.

The proximal operators for Alg. 1 (line 3) are simple to derive and well known in literature [19]:

$$\mathbf{prox}_{\theta f_1}(\mathbf{v}) = (\mathbb{I} + \theta\mathbf{M}^T\mathbf{M})^{-1}(\mathbf{v} + \theta\mathbf{M}^T\mathbf{x}) \quad \text{Quadratic}$$

$$\mathbf{prox}_{\theta f_2}(\mathbf{v}) = \max\left(1 - \frac{\theta\beta}{|\mathbf{v}|}, 0\right) \odot \mathbf{v} \quad \text{Shrinkage}$$

$$\mathbf{prox}_{\theta f_3}(\mathbf{v}) = \begin{cases} \frac{\mathbf{S}\mathbf{v}}{\|\mathbf{S}\mathbf{v}\|_2} & : \|\mathbf{S}\mathbf{v}\|_2^2 \geq 1 \\ \mathbf{S}\mathbf{v} & : \text{else} \end{cases} \quad \text{Projection}$$

where the inverse in $\mathbf{prox}_{\theta f_1}$ is straightforward to evaluate as $\mathbf{M}$ is usually a diagonal matrix.

2.4. Alternating for the biconvex problem

Above, we have described Alg. 1, which can solve the bi-convex problem (3) for $\mathbf{z}$ or $\mathbf{d}$ when the respective other variable is fixed. To jointly solve for both, we follow the standard approach of alternating between them, yielding Alg. 2.

Algorithm 2 CSC learning using coordinate descent

1: Algorithm penalty parameters: $\rho_{\mathbf{d}} \in \mathbb{R}^+, \rho_{\mathbf{z}} \in \mathbb{R}^+$
2: Initialize variables: $\mathbf{d}^0, \mathbf{z}^0, \lambda_{\mathbf{d}}^0, \lambda_{\mathbf{z}}^0$
3: **repeat** {Outer iterations}
4: **Kernel update:** Solve Eq. (5) w.r.t. $\mathbf{d}$: $\mathbf{d}^i, \lambda_{\mathbf{d}}^i \leftarrow \operatorname{argmin}_{\mathbf{d}} f_1(\mathbf{Z}\mathbf{d}) + \sum_{k=1}^{K} f_3(\mathbf{d}_k)$ using Alg. 1 with $\rho = \rho_{\mathbf{d}},\ \lambda = \lambda_{\mathbf{d}}^{i-1}$
5: **Code update:** Solve Eq. (5) w.r.t. $\mathbf{z}$: $\mathbf{z}^i, \lambda_{\mathbf{z}}^i \leftarrow \operatorname{argmin}_{\mathbf{z}} f_1(\mathbf{D}\mathbf{z}) + \sum_{k=1}^{K} f_2(\mathbf{z}_k)$ using Alg. 1 with $\rho = \rho_{\mathbf{z}},\ \lambda = \lambda_{\mathbf{z}}^{i-1}$
6: **until** No more progress in both directions.

With this alternating approach, we have constructed a coordinate descent for $\mathbf{z}$ and $\mathbf{d}$. The individual Lagrange multipliers are initialized with the ones from the previous iteration. In practice, we run each of the two sub-routines until sufficient progress has been made. The step-size of the coordinate descent is defined by the progress each local optimization makes. Using a constant number of P iterations for each substep gave us a sufficiently good performance. We stop the algorithm when none of the two optimizations can further decrease the objective; a local minimum is found. It also follows that our algorithm monotonically decreases the objective for its iteration sequence $\mathbf{d}^i, \mathbf{z}^i$.

2.5. Implementation details

For the objective in Eq. (3), we found that the parameter $\beta = 1$ delivers a good tradeoff between sparsity and data fit. All results in this paper are computed with this setting. We have verified that other settings of β lead to quantitatively similar results.

For Algorithm 2, we have chosen the heuristic values $\rho_{\mathbf{d}} = 1/(100 \cdot \max(\mathbf{x}))$ and $\rho_{\mathbf{z}} = 1/(10 \cdot \max(\mathbf{x}))$. This choice dampens variations in the optimization path of the filters more than for the codes.

3. Analysis

3.1. Complexity Analysis

This section analyzes the complexity of the proposed optimization approach and compares the theoretical runtime with alternative methods. For D being the number of pixels for a single image in $\mathbf{x}$, N being the number of images, K being the number of kernels to learn, M being the size of the filter support, and P inner iterations (of the substeps in Alg. 2), the computation cost is shown in Table 1.

We observe immediately that Bristow's method has significantly better performance than Zeiler et al. when $K < DM$. Its dominating cost is the inversion of the D linear systems. Note that in contrast to spatial methods, Bristow's method, as well as ours, is independent of the filter size M.

For the proposed method, we consider two cases: $K > N$ and $K \leq N$. In the first case ($K > N$), we exploit the inversion trick as explained in Eq. (11). Here, each of the D matrices $\mathbf{Z}_j$ is an $N \times K$ matrix. Thus, by using Eq. (11), we reduce the cost of the inverse from K^3 to KN^2. Since we cache the factorizations, this cost is only for one of the P local iterations. For the other $(P-1)$ iterations, the back-solves cost only KN (instead of K^2 for the naive inversion).

In the second case, when $K \leq N$, we have the full cost of the Cholesky factorization K^3 of the D matrices $\mathbf{Z}_j$ once per P iterations, but again for all other $(P-1)$ iterations, only the back-solve cost K^2. Thus, by caching the factorizations, we are able to achieve a speedup of the linear systems by $\frac{P}{1+(P-1)/K}$ in this case. For our setting of $P = 10$, even a moderate number of $K = 100$ filters leads to a speedup of $9\times$.

In the following, we show that not only the complexity per iteration decreases, but the convergence improves as well.

Method	Cost (in flops)
Zeiler et al. [22]	$PN \cdot (\underbrace{KD}_{\text{Conjugate gradient}} \cdot \underbrace{KDM}_{\text{Spatial convolutions}} + \underbrace{KD}_{\text{Shrinkage}})$
Bristow et al. [3, 4]	$PN \cdot (\underbrace{K^3D}_{\text{Linear systems}} + \underbrace{KD\log(D)}_{\text{FFTs}} + \underbrace{KD}_{\text{Shrinkage}})$
Ours ($K > N$)	$\underbrace{KN^2D + (P-1)KND}_{\text{Linear systems}} + PN \cdot (\underbrace{KD\log(D)}_{\text{FFTs}} + \underbrace{KD}_{\text{Shrinkage}})$
Ours ($K \leq N$)	$\underbrace{K^3D + (P-1)K^2D}_{\text{Linear systems}} + PN \cdot (\underbrace{KD\log(D)}_{\text{FFTs}} + \underbrace{KD}_{\text{Shrinkage}})$

Table 1: Cost of our approach and other recent methods.

3.2. Convergence

For two datasets of different size, we plot the empirical convergence of the proposed algorithm and compare it to the state of the art in Fig. 1. In both cases we learn $K = 100$ filters.

The first dataset is the fruit datasets [22] with $N = 10$ images. In this case, we have $K > N$. The proposed algorithm converges in 13 iterations whereas [3, 4] has a slowly decreasing objective and was fully converged after about 300 iterations. To be able to compare objective values, all compared methods here are implemented using circular convolution, with edge-tapering applied to make the

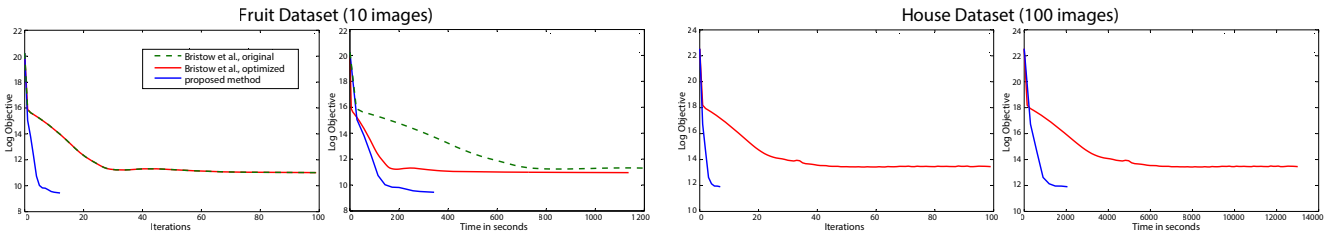


Figure 1: Convergence for two datasets (left $N = 10$ images, right $N = 100$). The proposed algorithm converges to a better solution in less time than competing methods.

convolution circular. Only the central part of the convolution without a padding of kernel size M is included in the objective values. Note that the solution of our algorithm is not only found in fewer iterations, but it also converges to a solution that has a lower objective. The objective combines the data fitting term as well as the sparsity term. We used the same sparsity weighting (from [3, 4]) for the comparison, although the same qualitative convergence was measured for different weights.

We also plot the convergence in an absolute time scale demonstrating the achieved speedup. We have further compared our method to [3, 4] with our factorization strategy from the first line in Eq (11). While this strategy improves the speed of their method since the inverses in each iteration are done more efficiently, the performance does not reach that of our method.

For the second dataset, we have randomly sampled $N = 100$ images of size 100×100 from a city scene (dataset and original image in the supplement). In this case $K \leq N$ and N, K are large. Thus, solving the linear systems becomes the dominating cost (see Table 1) and the benefit of caching (which cannot be done in [3, 4]) becomes even more apparent. Hence, especially for large datasets, our method converges faster and usually finds a better optimum. Note that we are already comparing to [3, 4] with our improved factorization strategy from the first line in Eq (11).

We also compare convolutional coding to patch-based sparse coding. One of the main challenges are large datasets, for which most patch-based methods become infeasible. Consider learning from 10 images with 1000×1000 pixels each. Incorporating all patches into the learning requires 10 million training patches. K-SVD, for example, could not handle that much data on our computer, so we ran a comparison for 10 100×100 pixel images. Using all patches in the training set and 100 iterations, K-SVD took 13.1 *hours* to converge whereas our method only took about 4.5 *minutes* (both on an Intel Xeon E5/Core i7 machine with 132 GB RAM).

In addition to the convergence analysis above, we also show the evolution of the filters throughout the learning process in Fig. 2 for the fruit dataset. Initially, the filters seem random and then turn into Gaussian blob-like structures after a few iterations. After about 8 iterations, the observed structures are very similar to those frequently occurring in patch-based dictionary learning methods, whereas the filters eventually converge to Gabor-like shapes.

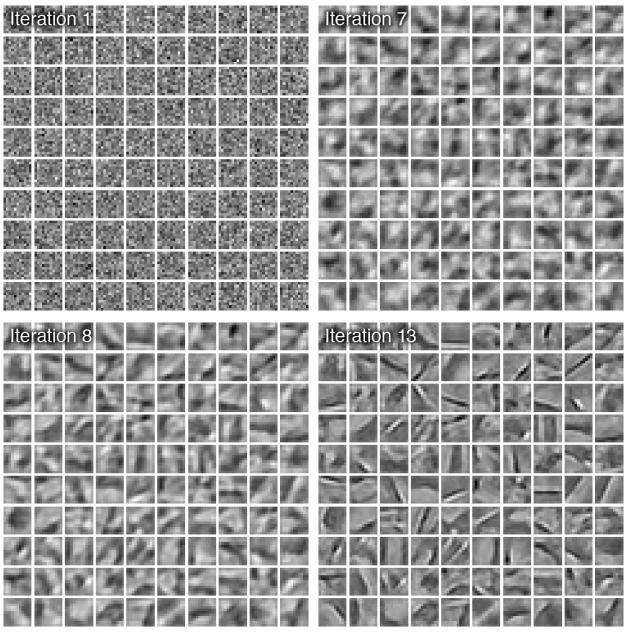


Figure 2: Visualization of filters learned from fruit dataset after 1, 7, 8, and 13 iterations. The evolution goes from random patterns over Gaussian blob-like structures and eventually converges to filters that bear similarity to Gabor patches.

4. Learning Features

4.1. Filter Learning

We trained our filters on the fruit and city datasets [22] with local contrast normalization applied. Figure 3 shows the resulting filters after convergence (ours after 13 iterations, Bristow after 300 iterations).

Although the filters look similar at first glance, our re-

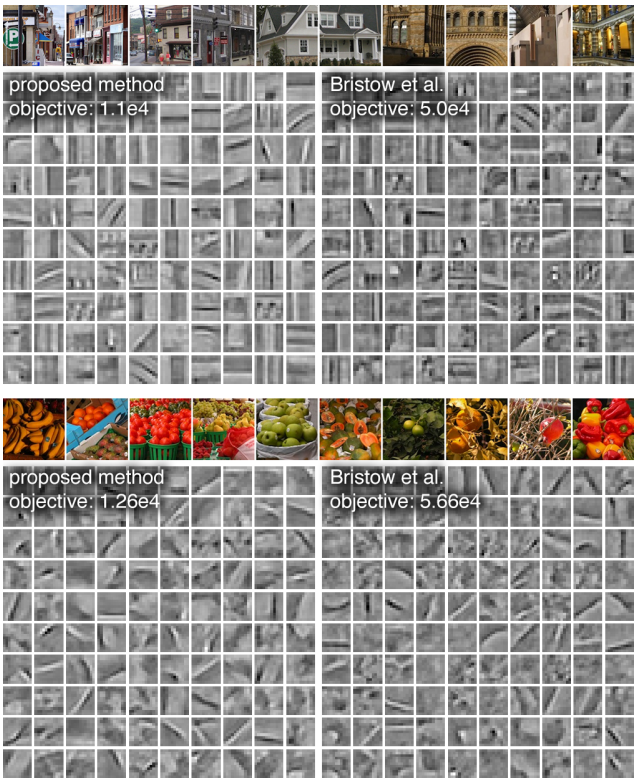


Figure 3: Filters learned on the city and fruit datasets [22]. We show thumbnails of the datasets along with filters learned with the proposed method (left) and with that described in [3, 4]. In both cases, our method finds a local optimum with an objective that is $3-4\times$ lower than comparable methods.

sults contain fewer dataset-specific features, which makes them more general as we demonstrate for reconstructions of other types of images in Sec. 5.1.

4.2. Boundary Conditions

Eq. 3 is an elegant formulation that allows for general boundary conditions to be integrated into the learning and also the reconstruction steps. Usually, we mask out the boundary so that it does not contribute to the objective function. As seen in Fig. 4, the boundary is still reconstructed via extrapolated data fitting — lines and other high-frequency structures continue across the image boundary but quickly fall off thereafter.

4.3. Learning from Sparse Data

The mixing matrix $\mathbf{M}$ in Eq. 3 not only allows for general boundary constraints but for any type of linear operator to be applied. In Fig. 5, we demonstrate that this can be used for learning filters from incomplete data. We subsequently use the filters learned from incomplete measurements of an

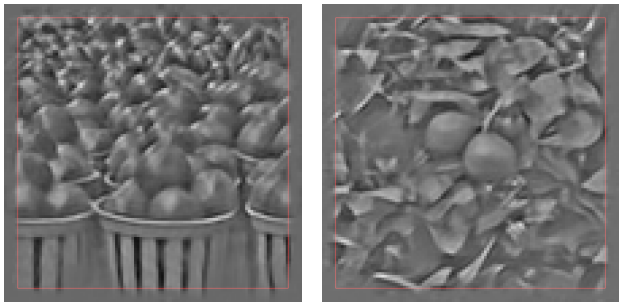
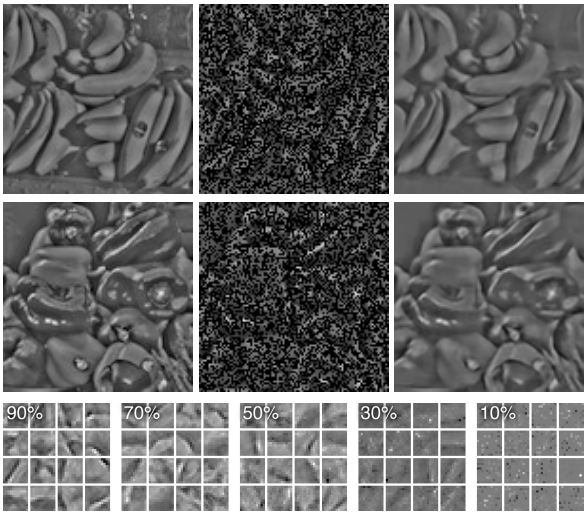


Figure 4: The proposed formulation allows us to use non-circular boundary conditions, as demonstrated for two examples. In practice, the regions outside the image boundary (red) are extrapolated but do not affect the objective.



Subsampling	90 %	70 %	50 %	30 %	10 %
PSNR	23.2 dB	21.3 dB	19.2 dB	16.5 dB	14.8 dB

Figure 5: Learning from sparse observations. The top two rows show examples for 50% sampling. The original images are shown in the left column, randomly subsampled observations in the center, and reconstructions of the entire image using filters learned from these sparse observations on the right. Bottom table and filters: evaluation of the learned filters for different subsampling rate of the data. *We show 16 out of the total 100 learned filters* for each sampling rate above the table. Please see the supplement for the full set of filters. One can see that for less than 50% sampling, the reconstruction quality significantly drops due decreasing filter quality.

image to inpaint the missing parts of that image and evaluate the achieved peak signal-to-noise ratio for varying levels of incompleteness. As is expected, the quality of the reconstructions drops with a decreasing amount of observations. Nevertheless, learning filters from incomplete measurements may be interesting for many applications (e.g.

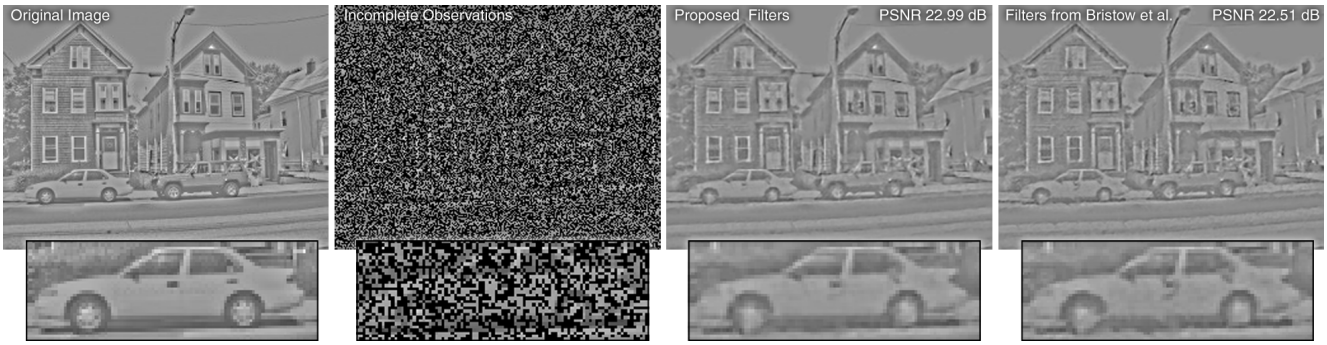


Figure 6: Inpainting example showing: original image (left), randomly sampled incomplete observations (center left), reconstruction with filters learned with the proposed algorithm (center right), and filters from [3, 4] (right). In addition to this example, we evaluate the reconstruction quality for a larger dataset in Fig. 7.

Image	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
PSNR ours	**23.46**	**25.60**	**24.64**	24.66	**30.13**	**28.10**	**24.42**	**24.97**	**22.07**	**26.15**	**25.90**	**20.76**	**24.20**	**24.06**	**23.60**	**24.28**	**22.26**	**26.03**	**21.26**	**28.36**	**22.89**	**21.52**
PSNR [3, 4]	23.06	24.58	24.28	**24.71**	29.40	27.27	23.99	24.62	21.79	25.13	25.22	20.50	23.92	23.57	23.37	23.91	21.77	25.74	21.10	27.80	22.74	21.42

Image	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
PSNR ours	**23.21**	**25.35**	**24.69**	24.70	**29.88**	**27.82**	**24.24**	**25.44**	**21.88**	**26.29**	**26.05**	**20.55**	**24.26**	**23.80**	**23.46**	**24.58**	**21.90**	**25.86**	**21.16**	**28.11**	**22.96**	**21.43**
PSNR [3, 4]	22.91	24.68	24.44	**24.72**	29.33	27.28	23.81	25.21	21.60	25.72	25.58	20.33	23.92	23.44	23.03	24.32	21.55	25.70	21.00	27.72	22.62	21.35

Figure 7: Reconstruction quality for filters learned with the proposed algorithm (tables, center row) and the filters proposed in [3, 4] (tables, bottom row). All reconstructions are performed for 50% subsampling. The upper table shows the reconstruction results with the filters learned from the fruit dataset from [22], the lower one shows the reconstructions with the filters from the city dataset. The dataset consists of 22 images, *none of which are part of the training set for learning the filters*. With the exception of image 4, our algorithm results in higher-quality reconstructions for both filter sets.

adaptive filter learning for demosaicking), but is currently not supported by any existing (efficient) method for convolutional sparse coding. Fig. 5 shows that even for subsampling rates of up to 50%, the learned filters and quality of the inpainted reconstructions are reasonably good.

5. Reconstruction

In this section, we evaluate the proposed algorithm for reconstructing signals when the filters are either already learned or known a priori. Reconstruction problems are solved for all filters with the code update step from Alg. 2.

5.1. Validation of Reconstruction

Fig. 6 shows an example image reconstructed from incomplete measurements. This is an inpainting problem, which we test with filters learned from the fruit database. We compare reconstruction quality using filters learned with the proposed method and filters learned with the method described in [3, 4]. Not only do our filters lead to a better reconstruction around edges and sharp features, but our framework allows us to solve this inpainting problem without sacrificing performance in the CSC solver, which was not possible in previous work due to the "Fourier trick". The experiment in Fig. 7 evaluates reconstruction quality for a dataset consisting of 22 images and, with a single exception, shows improved quality for both filter sets learned on the city and fruit dataset compared to previous work.

5.2. Non-normalized Data

In most results, we show contrast-normalized examples. However, non-normalized data can be intuitively handled in two ways. Either the filters are directly learned from non-normalized training data or a low-frequency term for the DC offset is added to the set of filter after learning. Typical Gabor-like filters with different DC offsets are observed for the former approach. The latter approach can be interpreted as adding a smoothness prior in the form of the low-frequency term rather than rigorously enforcing sparsity. A reconstruction then has to jointly solve for the filter coefficients as well as the low-frequency term, which is shown in Fig. 8. We have also compared this approach with a state-

Figure 8: Inpainting non-normalized data: randomly-subsampled, incomplete observations (left), reconstruction with the proposed filters (center and right).

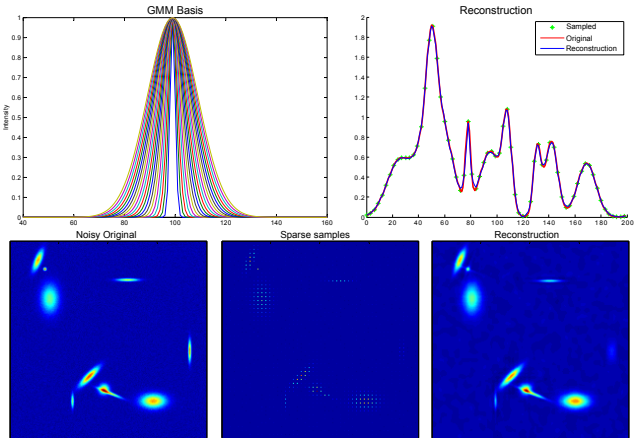


Figure 9: Reconstructions for known convolutional basis. The filters in this example are sampled from 1D Gaussians (top left) and used to fit a convolution model to a sparse set of samples (top right). The same experiment is performed in 2D, where a target signal is corrupted by noise (bottom left), subsampled (bottom center), and then reconstructed from *only 6.25% of the noisy measurements* (bottom right).

of-the-art compressive sensing method from Dong et al. [8]. Using the same dataset from Fig. 7 (50% random sampling with non-normalized data) their method achieves a mean PSNR of 23.5 dB while ours achieves 29.0 dB. This preliminary results suggests that further investigation of sparse convolutional coding might lead to many fruitful applications even outside of the low-level feature learning.

5.3. Reconstruction with Known Basis

We also evaluate the proposed method for fitting convolutional models to sparse and noisy measurements when the filters are known a priori. Recent examples of these types of reconstructions have appeared in the computational imaging domain where the employed basis is motivated by physical models (e.g., [11, 12]). In general, physically-motivated convolutional sparse coding may have a wide range of applications in radar, sonar, ultrasound and seismic imaging, but we leave a detailed evaluation of these applications to future work. Figure 9 demonstrates reconstructions of sparsely-sampled data in 1D and 2D. Filters are sampled from a Gaussian distribution and the measurements of the 2D example are further corrupted by iid Gaussian noise with a standard deviation of $\sigma = 0.01$. This experiment demonstrates the utility of CSC to non-feature-learning-type applications, such as general Gaussian mixture models. The proposed method is capable of recovering the latent signal with a high quality from *only 6.25% of the samples*.

6. Discussion

In summary, we propose a new method for learning and reconstruction problems using convolutional sparse coding. Our formulation is flexible in allowing for proper boundary conditions, it allows for feature learning from incomplete observations, or any type of linear operator applied to the estimation. We demonstrate that our framework is faster than the state of the art and converges to better solutions.

Future Work Although already faster than existing methods, our formulation is inherently parallel and the runtime could be significantly improved by an efficient GPU implementation. It would be interesting to evaluate learning and reconstructing features in higher-dimensional problems, such as 3D hyperspectral image data [15] or 4D light fields [17]. Finally, it would be interesting to apply the proposed framework to more complex hierarchical convolutional structures and networks [13, 22] that could be particularly useful for high-level computer vision applications, such as recognition.

Conclusion Convolutional sparse coding is a powerful framework that has the potential to replace or supplement popular patch-based learning and reconstruction methods. These are applicable to a wide range of computer vision problems, such as feature learning, denoising, inpainting, and demosaicking. With the proposed method, we hope to contribute a practical approach to solving general CSC problems efficiently and in the most flexible manner.

Acknowledgements

Felix Heide was supported by a Four-year Fellowship from the University of British Columbia. Gordon Wetzstein was supported by the Intel Strategic Research Alliance on Compressive Sensing.

References

[1] M. S. Almeida and M. A. Figueiredo. Frame-based image deblurring with unknown boundary conditions using the alternating direction method of multipliers. In *Proc. ICIP*, pages 582–585, 2013. 1, 2, 3

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

[3] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding. In *Proc. CVPR*, pages 391–398, 2013. 1, 2, 3, 4, 5, 6, 7

[4] H. Bristow and S. Lucey. Optimization methods for convolutional sparse coding. *arXiv:1406.2407*, 2014. 1, 2, 3, 4, 5, 6, 7

[5] A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009. 1

[6] V. Cevher, S. Becker, and M. Schmidt. Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. *IEEE Signal Processing Magazine*, 31(5):32–43, 2014. 1

[7] B. Chen, G. Polatkan, G. Sapiro, D. Blei, D. Dunson, and L. Carin. Deep learning with hierarchical convolutional factor analysis. *IEEE Trans. PAMI*, 35(8):1887–1901, 2013. 1

[8] W. Dong, G. Shi, X. Li, Y. Ma, and F. Huang. Compressive sensing via nonlocal low-rank regularization. *IEEE Transactions on Image Processing*, 23(8):3618 – 3632, 2014. 8

[9] R. B. Grosse, R. Raina, H. Kwong, and A. Y. Ng. Shift-invariance sparse coding for audio classification. In *Proc. UAI*, pages 149–158, 2007. 1

[10] F. Heide, M. Rouf, M. B. Hullin, B. Labitzke, W. Heidrich, and A. Kolb. High-quality computational imaging through simple lenses. *ACM Trans. Graph.*, 32(5):149, 2013. 3

[11] F. Heide, L. Xiao, A. Kolb, M. B. Hullin, and W. Heidrich. Imaging in scattering media using correlation image sensors and sparse convolutional coding. *OSA Opt. Exp.*, 22(21):26338–26350, Oct 2014. 1, 8

[12] X. Hu, Y. Deng, X. Lin, J. Suo, Q. Dai, C. Barsi, and R. Raskar. Robust and accurate transient light transport decomposition via convolutional sparse coding. *OSA Opt. Lett.*, 39(11):3177–3180, 2014. 1, 8

[13] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierachies for visual recognition. In *Proc. NIPS*, 2010. 1, 2, 8

[14] B. Kong and C. C. Fowlkes. Fast Convolutional Sparse Coding (FCSC). Technical report, UCI, May 2014. 1, 2, 3

[15] X. Lin, G. Wetzstein, Y. Liu, and Q. Dai. Dual-coded compressive hyperspectral imaging. *OSA Opt. Lett.*, 39(7):2044–2047, 2014. 8

[16] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proc. ICML*, pages 689–696. ACM, 2009. 1

[17] K. Marwah, G. Wetzstein, Y. Bando, and R. Raskar. Compressive light field photography using overcomplete dictionaries and optimized projections. *ACM Trans. Graph. (SIGGRAPH)*, 32(4):46:1–46:12, 2013. 8

[18] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311 – 3325, 1997. 1

[19] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013. 1, 3

[20] M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv:1309.2388*, 2013. 1, 3

[21] A. Szlam, K. Kavukcuoglu, and Y. LeCun. Convolutional matching pursuit and dictionary training. *arXiv:1010.0422*, 2010. 1

[22] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Proc. CVPR*, pages 2528–2535, 2010. 1, 2, 4, 5, 6, 7, 8

[23] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Proc. ICCV*, pages 2018–2025, 2011. 1

[24] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Proc. ICCV*, pages 479–486, 2011. 1