Digital Signal Processing 55 (2016) 44–51

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Fast convolutional sparse coding using matrix inversion lemma

Michal Šorel *, Filip Šroubek

Institute of Information Theory and Automation, Czech Academy of Sciences, Pod Vodárenskou věží 4, 182 08 Prague 8, Czech Republic

ARTICLE INFO

Article history:
Available online 3 May 2016

Keywords:
Convolutional sparse coding
Feature learning
Deconvolution networks
Shift-invariant sparse coding

ABSTRACT

Convolutional sparse coding is an interesting alternative to standard sparse coding in modeling shift-invariant signals, giving impressive results for example in unsupervised learning of visual features. In state-of-the-art methods, the most time-consuming parts include inversion of a linear operator related to convolution. In this article we show how these inversions can be computed non-iteratively in the Fourier domain using the matrix inversion lemma. This greatly speeds up computation and makes convolutional sparse coding computationally feasible even for large problems. The algorithm is derived in three variants, one of them especially suitable for parallel implementation. We demonstrate algorithms on two-dimensional image data but all results hold for signals of arbitrary dimension.

1. Introduction

Sparse coding methods learn a dictionary of basis vectors or functions so that observed data could be expressed as a linear combination of only a small number of these atoms [1]. Sparse coding first appeared in neuroscience as a model of visual cortex [2] but found applications in many classification and signal reconstruction tasks. In machine learning sparsity avoids over-fitting and can be thought of as a tool for feature extraction. In signal reconstruction sparse coding can serve as a form of Bayesian prior for image denoising [3], inpainting [4], deblurring [5], super-resolution [6] and audio signal representation [7]. Although finding the dictionary with which the training signals can be represented with optimal sparsity is strongly NP-hard [8], there is a number of effective heuristic algorithms giving an approximate solution in polynomial time [9,10]. Sparse coding is closely related to compressed sensing [11], with results showing that for incoherent dictionaries only a small number of projections is sufficient to exactly reconstruct the original signal [12,13]. Efficient sparse coding algorithms with provable guarantees appeared only recently [14–16].

In image processing applications, both the observed data and dictionary atoms correspond to image patches. A fundamental disadvantage of sparse coding is the assumption that image patches are independent, which typically leads to many atoms being translated versions of one another. The same issue can be expected in audio signals. *Convolutional sparse coding*, also called shift-invariant sparse coding [17–19], is an interesting alternative that found its use in audio classification [20], deconvolutional networks [21] and predictive sparse coding by neural networks [22]. In contrast to standard sparse coding that models a signal as a sparse combination of dictionary vectors, convolutional sparse coding models the signal as a sum of several convolutions of kernels and sparse feature maps.

The goal of this article is to describe a new fast algorithm for convolutional sparse coding. Our solution is based on the fact that the main problem of state-of-the-art algorithms [21,23] is a time consuming inversion of an operator related to convolution. This problem was sidestepped in [24] by using FISTA [25], where the inversion step is essentially replaced by one gradient descent step at the cost of much larger number of iterations necessary to achieve the same precision. In this paper, we adopt an approach close to [23] but show how the most time-consuming step of their algorithm can be computed non-iteratively in the Fourier domain using the matrix inversion lemma, which greatly speeds up computation. Derivation is relatively straightforward for one input signal [26] but more complicated for multiple inputs [27]. As our main contribution, we show three solutions of the multiple-input case, which are all equivalent for the single-input case. One of them is especially suitable for parallel implementation. We also compare efficiency of [21,23] and our algorithm under various conditions and demonstrate the ability of the proposed algorithms to learn kernels at several scales simultaneously. A paper using similar ideas to solve the problem of convolution sparse coding from incomplete data appeared recently in [28].

The rest of the paper is organized as follows. Sec. 2 states the problem of convolutional sparse coding. Sec. 3 shortly explains the main optimization tool we use, which is the alternating direc-

* Corresponding author.
E-mail addresses: sorel@utia.cas.cz (M. Šorel), sroubekf@utia.cas.cz (F. Šroubek).

http://dx.doi.org/10.1016/j.dsp.2016.04.012
1051-2004/

Table 1
Notation.

z_k	feature maps $1 \dots K$
d_k	convolutional kernels $1 \dots K$
K	number of kernels and feature maps
N	number of pixels
L	number of input images
d_k^i	kth convolution kernel for input image i used in the consensus version of the algorithm
Z_k, D_k	operators corresponding to convolution with z_k and d_k
β	parameter of convolutional sparse coding balancing sparsity and accuracy
u_z, u_d	ADMM auxiliary variables for feature maps and convolution kernels
v_z, v_d	ADMM dual variables
λ	ADMM parameter
P	number of ADMM iterations

tion method of multipliers. Algorithms are derived in Sec. 4. Time complexity of the algorithms is summarized in Sec. 5 followed by experiments in Sec. 6 and conclusions in Sec. 7.

2. Convolutional sparse coding

Building on the analogy with compressed sensing, where the sparse representation is provably recoverable using l_1 norm [13], the convolutional sparse coding can be stated as a bi-convex problem

$$\min_{d,z} \frac{1}{2} \left\| y - \sum_{k=1}^{K} d_k * z_k \right\|^2 + \beta \sum_{k=1}^{K} \|z_k\|_1 \quad \text{s.t.} \quad \|d_k\| \le 1, \tag{1}$$

where y is an observed signal, z_k are sparse *feature maps* and d_k corresponding *convolution kernels*. The number of convolution kernels K and positive scalar β are user parameters. Complete list of used variables is provided in Table 1. In this paper, we assume circular boundary conditions, i.e. equivalence of convolution with element-wise multiplication in the Fourier domain, therefore the feature maps are of the same size and dimension as the observed signal. Motivated by applications in learning visual features and modeling image data, we use two-dimensional images in our experiments but convolutional sparse coding can be applied to signals of arbitrary dimension.

Analogously to standard sparse coding and other machine learning and signal modeling approaches, we are interested in two different modes of operation. First we learn convolution kernels from training data by solving the optimization problem (1) as stated above. We call this phase *kernel learning*. Second, in *feature extraction* phase, the kernels are fixed and features are computed only by minimization over feature maps. Even though the terms feature extraction and kernel learning come from machine learning, the same operations are important even for signal modeling and reconstruction. The role of kernel learning is to estimate an *a priori* signal distribution and feature extraction corresponds to Bayesian inference from noisy measurements.

As in sparse coding [10,29], all efficient methods of kernel learning in the literature [21,23,24] alternately minimize over the feature maps z_k while keeping the filters d_k fixed and over the filters while keeping the feature maps fixed, taking the advantage that both sub-problems are convex. In this way the feature extraction is essentially run in each iteration of the kernel learning algorithm.

3. Alternating direction method of multipliers

The main optimization tool we use in both convex sub-problems, which was less efficiently used already in [23], is the alternating direction method of multipliers (ADMM) [30]. Here we present its simplified form, which is equivalent to Douglas–Rachford splitting algorithm [31]. ADMM is a method to minimize the sum of two convex not necessarily differentiable functions

$$\min_x f(x) + g(x). \tag{2}$$

The algorithm is especially useful, if we can efficiently compute a so-called proximal or proximity operator of both functions, defined for f as

$$\operatorname{prox}_{\lambda f}(a) = \arg\min_x \lambda f(x) + \frac{1}{2}\|x-a\|^2 \tag{3}$$

and similarly for g, where scalar $\lambda > 0$ is a parameter. ADMM consists of iteratively executing three update steps

$$x \leftarrow \operatorname{prox}_{\lambda f}(v-u) \tag{4}$$

$$v \leftarrow \operatorname{prox}_{\lambda g}(x+u) \tag{5}$$

$$u \leftarrow u + x - v \tag{6}$$

two of them being computations of proximal operators for f and g, and (6) is a simple update of an auxiliary variable. Stopping criteria for ADMM are discussed in [30], Section 3.3.1.

4. Algorithm

In this section, we show that both convolutional sparse coding sub-problems can be written as a sum of two functions suitable for ADMM and derive how to efficiently compute their proximal operators. The main difference with respect to [23] is much faster computation of one of the proximal operators.

4.1. Minimization over feature maps

We start with the minimization of (1) over feature maps z_i, which can be written in short as

$$\min_z \frac{1}{2}\|y - Dz\|^2 + \beta\|z\|_1 \tag{7}$$

where $D = [D_1, \dots, D_K]$ is an operator composed of convolutions with K kernels d_k and $z = \left[z_1^T, \dots, z_K^T\right]^T$ is a vector of vectorized feature maps. K denotes the number of feature maps and β is a parameter balancing model accuracy and sparsity of the representation. The number of elements in y will be denoted by N. Note that from this place on, we use vector notation, where quantities y, z, etc. are vectors and convolutions with d_k and z_k are expressed as multiplications with circulant (for 2D data block-circulant) matrices D_k and Z_k, respectively.

This is a special case of l_1-regularized linear regression also called Lasso [32]. Authors of [21] solved (7) by a continuation approach, [23,33] used ADMM, decomposing (7) into two functions

$$f_z(z) = \frac{1}{2}\|y - Dz\|^2 \quad \text{and} \tag{8}$$

$$g_z(z) = \beta\|z\|_1. \tag{9}$$

The proximal operator of l_1 norm

$$\arg\min_x \alpha\|x\|_1 + \frac{1}{2}\|x-a\|^2 \tag{10}$$

is a very fast element-wise operation called soft thresholding [30]

$$\operatorname{prox}_{\alpha|x|}(a) = S_\alpha(a) = \begin{cases} a-\alpha & a > \alpha \\ 0 & |a| \le \alpha \\ a+\alpha & a < \alpha, \end{cases} \tag{11}$$

where in our case $\alpha = \lambda\beta$.

The critical and most time-consuming part of the algorithm is the proximal operator of the term f_z

$$\operatorname{prox}_{\lambda f_z}(a) = \arg\min_z \frac{\lambda}{2}\|y - Dz\|^2 + \frac{1}{2}\|z - a\|^2 \tag{12}$$

$$= \left(\lambda D^T D + I\right)^{-1}\left(\lambda D^T y + a\right) \tag{13}$$

where $D^T D + \lambda I$ is a $KN \times KN$ matrix. The same inversion is needed also in the continuation approach of [21]. Both [21] and [23] solve the inversion by the method of conjugate gradients (CG). [23] shows that the inversion can be decomposed into N independent $K \times K$ matrices inverted separately. The inversion is still computationally very demanding (see Sec. 5). Authors of [24] sidestep (13) by the accelerated proximal gradient method, basically replacing (13) by one gradient descent step $z^{k+1} = z^k - \gamma D^T(Dz^k - y)$. This avoids the expensive inversion at the cost of slower convergence.

Next, we present the first contribution of this paper. We show that inversion (13) can be computed very cheaply in the Fourier domain, in a time comparable to one step of gradient descent. Indeed, the Sherman–Morrison–Woodbury inversion lemma gives

$$\left(\lambda D^T D + I\right)^{-1} = I - \lambda D^T\left(I + \lambda D D^T\right)^{-1} D \tag{14}$$

implying

$$\operatorname{prox}_{\lambda f_z}(a) = \left(I - \lambda D^T\left(I + \lambda D D^T\right)^{-1} D\right)\left(\lambda D^T y + a\right) \tag{15}$$

Inversion $\left(I + \lambda D D^T\right)^{-1} = \left(I + \lambda \sum D_k D_k^T\right)^{-1}$ can be computed in the Fourier domain as element-wise multiplication by

$$\frac{1}{1 + \lambda \sum_{k=1}^K \left|\hat{d}_k\right|^2}, \tag{16}$$

where $\left|\hat{d}_k\right|^2$ are squared magnitudes of the coefficients of the Fourier transform of d_k. Moreover, K Fourier transforms in (16) are computed only once at the beginning of the algorithm, since they are either reused (for multiple iterations of feature extraction) or computed in the minimization over kernels. Note that circular boundary conditions needed for computation in the Fourier domain in most cases do not harm the learned kernels as commented in [23] (p. 5).

The algorithm for optimization over feature maps, which also serves as the feature extraction algorithm is summarized in Algorithm 1.

Algorithm 1 Feature extraction algorithm.

1: Initialize $i \leftarrow 0$, $u_z^0 \leftarrow 0$, $v_z^0 \leftarrow 0$
2: Pre-compute Fourier transforms $\hat{d}$
3: **repeat**
4: Solve for $z \leftarrow \operatorname{prox}_{\frac{\lambda}{2}\|y-Dz\|^2}(v_z - u_z)$ using (15) and (16)
5: Update $v_z \leftarrow S_\lambda(z + u_z)$, where thresholding S_λ is defined by (11)
6: Dual variable update $u_z \leftarrow u_z + z - v_z$
7: **until** convergence.

4.2. Minimization over convolution kernels

Minimization of (1) over convolution kernels

$$\min_{\{d_k\}} \frac{1}{2}\left\|y - \sum_{k=1}^K Z_k d_k\right\|^2, \quad \text{s.t. } \|d_k\|_2^2 \le 1, \tag{17}$$

where Z_i are operators of convolution with feature maps z_i, is a convex problem with convex constraints.

In [21] and [24], optimization problem (17) was solved by gradient descent, ignoring the constraint $\|d_k\|_2^2 \le 1$. Similarly to [23], we adopt again the ADMM (4)–(6), with f and g corresponding to $f_d(d) = \frac{1}{2}\left\|y - \sum_i Z_i d_i\right\|^2$ and the indicator function[1] of convex set $\|d_k\|_2^2 \le 1$, respectively. Since kernels are finally computed in the Fourier domain, we work with kernels of the same size as the observation y and enforce their support as an additional constraint. Formally, g could be expressed as

$$g_d(d) = \begin{cases} 0 & \text{if } \forall k\, \|d_k\|_2^2 \le 1 \text{ and } \operatorname{supp}(d) \subset \mathcal{S} \\ +\infty & \text{otherwise} \end{cases}, \tag{18}$$

where $\mathcal{S}$ is kernel support, typically a finite size square. The indicator function g_d is convex and its proximal operator is equivalent to projection on the intersection of the support and a unit ball

$$\operatorname{prox}_{g_d}(a) = \begin{cases} a\mathbf{1}_{\mathcal{S}}/\|a\mathbf{1}_{\mathcal{S}}\| & \text{if } \|a\mathbf{1}_{\mathcal{S}}\| > 1 \\ a\mathbf{1}_{\mathcal{S}} & \text{otherwise,} \end{cases} \tag{19}$$

where $\mathbf{1}_{\mathcal{S}}$ is a mask taking on 1 on $\operatorname{supp}(d)$ and 0 otherwise. Proof is straightforward by the method of Lagrange multipliers.

Similarly to minimization over feature maps, the proximal operator for f_d

$$\operatorname{prox}_{\lambda f_d}(a) = \arg\min_{\{d_i\}} \frac{\lambda}{2}\left\|y - \sum_{k=1}^K Z_k d_k\right\|^2 + \frac{1}{2}\sum_k \|d_k - a_k\|^2$$

$$= \arg\min_{\{d\}} \frac{\lambda}{2}\|y - Zd\|^2 + \frac{1}{2}\|d - a\|^2$$

$$= \left(\lambda Z^T Z + I\right)^{-1}\left(\lambda Z^T y + a\right) \tag{20}$$

was computed in [23] as N separable linear systems of size $K \times K$ by conjugate gradients with the same problem of excessive time complexity as in the first sub-problem. Again, applying the inversion formula (14) on Z gives

$$\operatorname{prox}_{\lambda f_d}(a) = \left(I - \lambda Z^T\left(I + \lambda Z Z^T\right)^{-1} Z\right)\left(\lambda Z^T y + a\right), \tag{21}$$

where $\left(I + \lambda Z Z^T\right)^{-1} = \left(I + \lambda \sum Z_k Z_k^T\right)^{-1}$ can be computed in the Fourier domain by element-wise multiplication with

$$\frac{1}{1 + \lambda \sum_{k=1}^K \left|\hat{z}_k\right|^2}. \tag{22}$$

The resulting algorithm is shown in Algorithm 2. In practice, we iterate several times for each of the optimization sub-problems but in some applications one iteration may be sufficient. To distinguish two levels of iterations, we call the outer iterations *global iterations* and the inner iterations *ADMM iterations*. In the following sections, the number of inner ADMM iterations will be denoted as P and will be common for both sub-problems. Influence of parameters on convergence and learned kernels is discussed in [33].

Algorithm 2 Kernel learning algorithm.

1: Initialize $u_z^0 = 0$, $v_z^0 = z^0$, $u_d^0 = 0$ and $v_d^0 = d^0$ randomly
2: **repeat**
3: Solve for $z \leftarrow \arg\min_z \frac{1}{2}\|y - Dz\|^2 + \beta\|z\|_1$ using Algorithm 1
4: **for** iteration $= 1 \dots P$ **do**
5: Solve for kernels $d \leftarrow \operatorname{prox}_{\frac{\lambda}{2}\|y - Z^{i+1}d\|^2}(v_d - u_d)$ by (21) and (22)
6: Update $v_d \leftarrow \operatorname{prox}_{g_d}(d + u_d)$ by projection (19) or (25)
7: Dual variable update $u_d \leftarrow u_d + d - v_d$
8: **end for**
9: **until** convergence.

[1] Indicator function of a set as defined in convex analysis takes on zero on the set and plus infinity elsewhere.

Table 2
Time complexity of feature extraction and kernel learning per iteration for [21,23] and three versions of our method. N denotes the number of pixels, K the number of convolution kernels, L the number of input images, P the number of ADMM iterations and Q the number of CG iterations in [21]. The time of feature extraction is given for one input image, therefore does not contain the factor L.

	Feature extraction
Zeiler et al. [21]	$O(KN\log N + Q\,KN)$
Bristow et al. [23]	$O\left(KN\log N + \frac{1}{P}K^3N + K^2N\right)$
Proposed	$O(KN\log N + KN)$
	Kernel learning
Zeiler et al. [21]	$O(KLN\log N + Q\,KLN)$
Bristow et al. [23]	$O\left(KLN\log N + \frac{1}{P}K^3N + K^2LN\right)$
Proposed (tiling, 3D)	$O\left(KLN\left(\log N + \log L\right) + KLN\right)$
Proposed (consensus)	$O\left(KLN\log N + KLN\right)$

4.3. Extension to multiple input images

Definition of the convolutional sparse coding problem (1) can be naturally extended to multiple input images as

$$\frac{1}{2}\sum_{l=1}^{L}\left\|y^l - \sum_{k=1}^{K} d_k * z_k^l\right\|^2 + \beta\sum_{l=1}^{L}\sum_{k=1}^{K}\left\|z_k^l\right\|_1, \quad \text{s.t.} \quad \|d_k\| \le 1, \tag{23}$$

where y^l are input images, z_k^l the corresponding feature maps and $L > 1$ the number of input images. Minimization over feature maps works independently for each image and algorithm does not change. Unfortunately, it is not the case for the minimization over convolution kernels (17), since the kernels are shared by all feature maps. As the second main contribution of this paper, we propose three solutions working even in this more complicated situation.

First, we can tile all the images into one large image, possibly padded by zeros, and use the algorithm for one input image described in the previous section. Time complexity is $O\left(LN(\log N + \log L)\right)$, i.e. the algorithm is slightly more than linear in the number of input images. An obvious advantage of this solution is that it is easy to implement and disadvantage possible interference on image boundaries. This algorithm is denoted as "tiling" in Table 2 and Fig. 4.

Second, we can use the original formulation (1) with three-dimensional feature maps and kernels, with images stacked along the third dimension. For signals of other dimensions we analogously lift their dimension by one. The only difference in the algorithm is three-dimensional Fourier transform and kernels constrained to zero not only outside its two-dimensional support but also everywhere else along the third dimension, except the central plane. Asymptotic time complexity per iteration is the same as in the previous case, because three-dimensional convolutions, in addition to two-dimensional FFT for each image, i.e. $O(LN\log N)$, require for each pixel one-dimensional convolution, i.e. $O(NL\log L)$. This variant is denoted as "3D" in Table 2 and Fig. 4.

A difficulty that typically arises in kernel learning is high memory consumption, since the input, auxiliary variables v_d, u_d and result d are all of size KLN. Whereas the feature extraction phase of learning can be easily divided between for example multiple GPU's or computed sequentially, it is difficult for kernels in both approaches described above. This problem can be alleviated using ADMM in a slightly modified way as a special case of so-called global consensus problem [34].

The idea of this third solution is to split computation to work with only one input image at once, by the algorithm described in Sec. 4.2. For this purpose, we consider separate kernels d_k^l for each image y^l and reformulate the problem of optimization (23) over convolution kernels to equivalent

$$\min_{\{d_k^l\}} \frac{1}{2}\sum_{l=1}^{L}\left\|y^l - \sum_{k=1}^{K} d_k^l * z_k^l\right\|^2$$
$$\text{s.t.} \quad \left\|d_k^l\right\|_2^2 \le 1 \text{ and } d_k^1 = d_k^2 = \ldots = d_k^L \tag{24}$$

To apply ADMM, we split (24) very similarly to the case of one input to the sum of two functions f and g, where the function f now corresponds to the first term of (24) and g to the indicator function of the set given by the new constraints. Since f is separable, prox_f can be computed for each image separately using (21) and (22). New constraints form again a convex set and corresponding projection is nothing else than averaging the values updated independently for each input image before projecting by the original equation (19), i.e.

$$\mathrm{prox}_{g_d}(a) = \begin{cases} \bar{a}\mathbf{1}_{\mathcal{S}}/\|\bar{a}\mathbf{1}_{\mathcal{S}}\| & \text{if } \|\bar{a}\mathbf{1}_{\mathcal{S}}\| > 1 \\ \bar{a}\mathbf{1}_{\mathcal{S}} & \text{otherwise,} \end{cases} \tag{25}$$

where $\bar{a} = \frac{1}{K}\sum_k a_k$. Eq. (25) can be again proved by the method of Lagrange multipliers.

As a result, the only part, where the algorithm requires interaction between input images and therefore cannot be simply run in parallel is the element-wise operation (25), i.e. averaging auxiliary variables a_k and computing the projection. Therefore, in theory, parallelization can speed up the algorithm up to L-times. An additional advantage of the third solution is that input images can be of different sizes. In Table 2 and Fig. 4, this variant is denoted as "consensus".

5. Time complexity

Asymptotic time complexity of both algorithms (feature extraction and kernel learning) in $O(.)$ notation is summarized in Table 2. Note that where convenient we use a longer form than necessary. For example, in the last line of the table we have $O\left(KLN\log N + KLN\right)$ instead of equivalent $O(KLN\log N)$ to keep track of additional asymptotically negligible operations that replace more demanding steps used in Bristow's algorithm two lines above.

We start our analysis by feature extraction (Algorithm 1); first three lines in Table 2. The learning phase contains the same computation as a subset (line 3) of Algorithm 2). Time complexity of our algorithm is dominated by the inversion in line 4, since complexity of lines 5 and 6 is linear for each feature map, i.e. in total $O(KN)$.

Operator (15) requires $2K$ Fourier transforms to get to the Fourier domain and back ($O(KN\log N)$) and then negligible $2KN$ multiplications on Fourier coefficients. This takes approximately the same time as one iteration of conjugate gradients used in [21]. Method [23] needs the same $2K$ Fourier transforms per iteration as we do but on top of that requires solution of N linear systems with matrices of size $K \times K$, which requires $O(K^3N)$ operations in the first ADMM iteration. This grows very quickly with K and dominates the computational time even for only a moderate number of kernels (see Fig. 1). For the following ADMM iterations, complexity decreases to $O(K^2N)$, which is still more than $O(KN)$ of the proposed algorithm.

The time complexity of kernel minimization has two components. The time spent in the minimization over feature maps (per input image) is the same as for feature extraction, except that we usually need just one or a small number of ADMM iterations. Time of this phase is always proportional to the number of input images, because all operations are basically repeated L times.

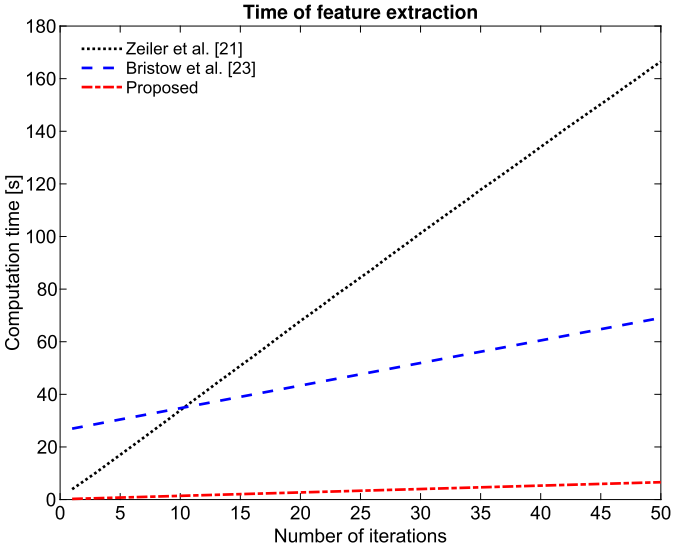


Fig. 1. Computation time of feature extraction for $K = 100$ feature maps for baseline CG based method of Zeiler et al. [21] (dotted line), for Bristow et al. [23] (dashed line) and the proposed (dash–dot line) algorithm. For 50 iterations, our method is 10 and 25 times faster than [23] and [21], respectively.

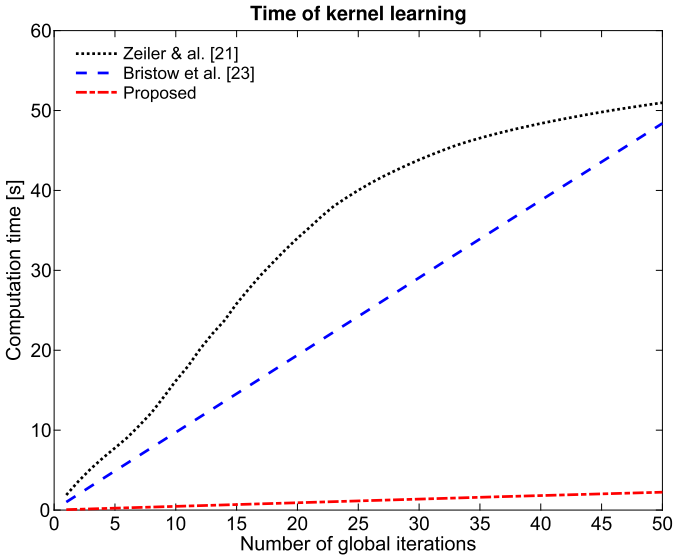


Fig. 2. Computation time of kernel learning algorithm for $K = 10$ kernels and one input image (128×128 pixels). The proposed method (dash–dot line) is about 22 times faster than Bristow et al. [23] (dashed line).

Asymptotic time complexity of kernel learning per iteration is the same $O\left(LN(\log N + \log L)\right)$ for the first two proposed variants, slightly more than $O(LN \log N)$ in the last variant. Similarly to feature extraction, the main speedup is achieved by avoiding slow inversion $O(K^3N)$, instead computing just $O(KN)$ multiplications on Fourier coefficients. Even for a large number of ADMM iterations, where the term $O(K^2N)$ dominates, our algorithm is $O(K)$ times faster.

Care must be taken not to mix time complexity per iteration with convergence. While the first tiling variant converges like [23] (with negligible differences due to boundary conditions) and the second variant does not usually depart much, the third variant may behave differently, see Fig. 4. Exact behavior depends on parameters and input data.

6. Experiments

The main goal of this section is to compare speed of the proposed algorithm with our implementation of methods [21,23]. We also demonstrate how the kernels can be learned simultaneously on several scales. Note that our version of [21] has the same complexity per iteration as the original but as a rule converges faster, because we use ADMM instead of continuation. In our implementation of [23], we reuse the inverted matrices if more than one iteration of ADMM is needed, which is typically the case in feature extraction. All our experiments were implemented in Matlab. In our experiments we work with a set of high-quality images of urban environment (Fig. 5) in ideal sunny-day conditions taken with an SLR camera. RAW data were scaled down four-times in both dimensions to eliminate noise. As a rule we finally cropped images to work with regions of size 128×128 pixels.

Feature extraction, common for all three variants of the proposed algorithm, and [23] give exactly the same results (up to machine precision) and the only difference is speed. Relative tolerance of conjugate gradients in [21] was set to 10^{-9}, which makes the results indistinguishable from [23] and us as well.

In Fig. 1, we show the time of feature extraction as a function of the number of ADMM iterations for $K = 100$ kernels and input image of size 128×128. Notice the slowness of [23] for a small number of iterations (our method is 200 times faster in the first iteration), because of the term $O(K^3N)$. The inverted matrices are reused, which speeds up algorithm in the following ADMM iterations but our method is still asymptotically K times faster. For 50 iterations, we are 10 and 25 times faster than [23] and [21], respectively.

Fig. 2 shows the computation time per iteration for learning of $K = 10$ kernels from the same image. Our algorithm is about 22 times faster than [23]. It is interesting to observe behavior of conjugate gradients, which progressively decreases the number of necessary iterations, which makes the algorithm relatively less inefficient for higher numbers of global iterations. For large K, it can be even faster than [23]. Note that this experiment works with one input image, so there is no difference between the variants of the algorithm.

Conjugate gradients become extremely slow for more than a small number of input images, therefore we do not show them in the next experiment, demonstrating efficiency of learning for different numbers of input images. Fig. 3 shows speedup with respect to Bristow et al. [23]. For $K = 100$ kernels, 5 global iterations and $L = 1,\ 10,\ 100$ input images, the speedup is about 83, 20 and 17 times. In this experiment we used the "3D" variant of the algorithm. Note that the erratic behavior of the solid curve ($L = 1$) depends on computer architecture and probably is connected to memory operations.

Fig. 4 compares convergence of all three variants of our algorithm for $K = 50$ kernels, $L = 10$ input images and $P = 5$ ADMM iterations. In this case the tiling and 3D variant behave almost the same, the consensus converges slower.

Finally, in Fig. 6, we show the kernels estimated from $L = 50$ images. In this experiment, we demonstrate the possibility to learn simultaneously kernels on different scales. Here, we chose 8 kernels of size 8×8, 32 kernels of size 16×16 and 64 kernels of size 32×32. We used $P = 10$ inner ADMM iterations and 1000 global iterations. The number of kernels on different scales was chosen to grow with their dimension but exact number was set arbitrarily. On individual scales, we sorted the kernels according to the average energy contained in the corresponding feature maps. As expected, the highest energy is typically contained in relatively simple edges, followed by kernels we could identify as textures (walls, trees) and the lowest energy is in kernels containing either noise or highly specialized patterns.

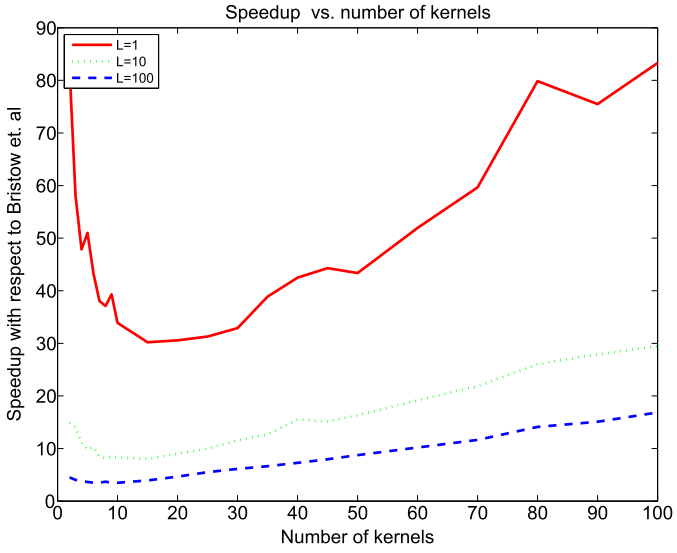


Fig. 3. Speedup of the proposed algorithm with respect to [23] for an increasing number of kernels. For $K = 100$ kernels and $L = 1,\ 10,\ 100$ images, the speedup is about 83, 20 and 17 times.

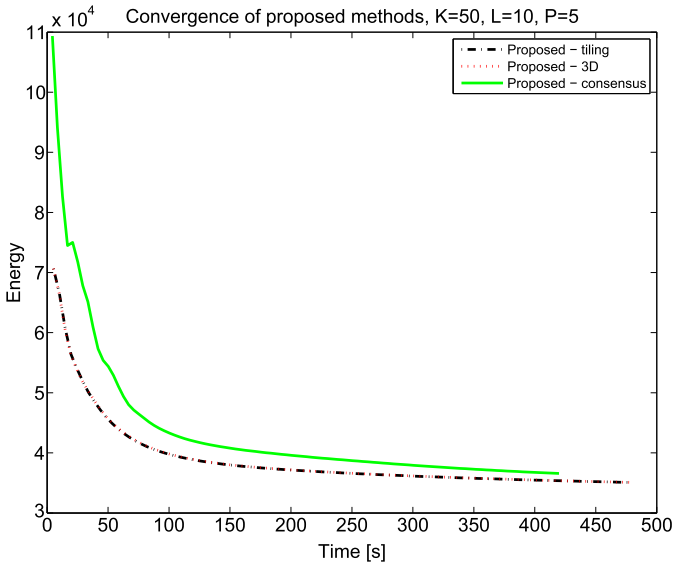


Fig. 4. Energy as a function of time for three variants of the proposed algorithm ($K = 50,\ L = 10,\ P = 5$). In this particular experiment the tiling and 3D variants overlap.

7. Conclusion

In this article we proposed three versions of fast algorithm to solve convolutional sparse coding problem, with the consensus variant especially suitable for parallel implementation. The choice of the algorithm may depend on available memory and convergence in a particular application. Proposed solution makes convolutional sparse coding computationally feasible even for large problems.

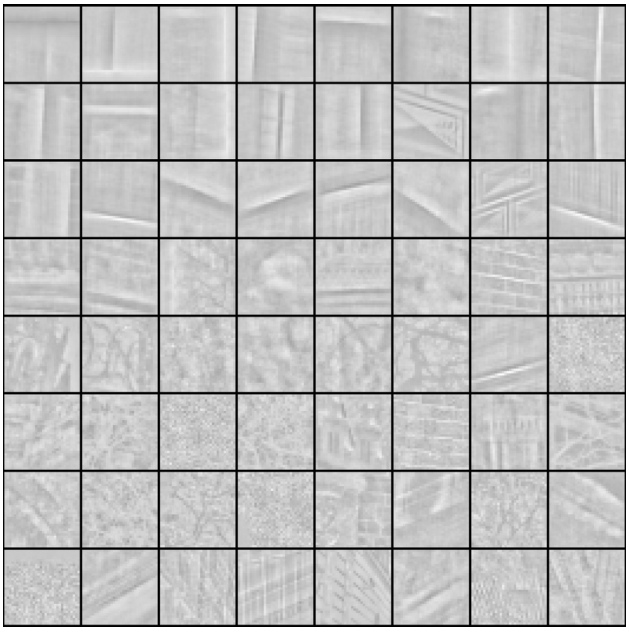
Even though our research was motivated by modeling image priors and computation of image features, and all experiments were performed with image data, the same algorithm can be used for signals of arbitrary dimension.

To help both researchers and practitioners further to investigate potential of convolutional sparse coding, we provide a Matlab implementation of all variants of the proposed algorithm along with our implementation of methods used in [21] and [23]. The code is available at http://zoi.utia.cas.cz/convsparsecoding.

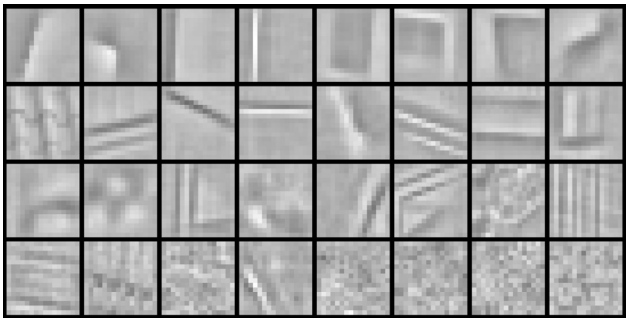
Acknowledgments

The work of Michal Šorel and Filip Šroubek on this research was funded by the Grant Agency of the Czech Republic under project GA13-29225S. The authors would like to thank Brendt Wohlberg from Los Alamos National Laboratory who proposed using the ma-

Fig. 5. 25 of 50 input images used to learn kernels in Fig. 6.



(a) 64 kernels of size 32×32



(b) 32 kernels of size 16×16

(c) 8 kernels of size 8×8

Fig. 6. Kernels learned by the proposed algorithm simultaneously on three different scales ($K = 104$, $L = 50$, $P = 10$). Kernels are sorted by energy of the corresponding feature map in a descending order from top-left to bottom-right. We can observe that the high energy kernels contain meaningful structures whereas the low energy ones resemble noise.

trix inversion formula to speed up convolutional sparse coding for the first time for the helpful discussion regarding this paper.

References

[1] S.S. Chen, D.L. Donoho, M.A. Saunders, Atomic decomposition by basis pursuit, SIAM J. Sci. Comput. 20 (1) (1998) 33–61.
[2] B.A. Olshausen, D.J. Field, Sparse coding with an overcomplete basis set: a strategy employed by v1?, Vis. Res. 37 (23) (1997) 3311–3325.
[3] M. Elad, M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, IEEE Trans. Image Process. 15 (12) (2006) 3736–3745.
[4] M. Aharon, M. Elad, A.M. Bruckstein, On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them, Linear Algebra Appl. 416 (1) (2006) 48–67.
[5] F. Couzinie-Devy, J. Mairal, F. Bach, J. Ponce, Dictionary learning for deblurring and digital zoom, preprint, arXiv:1110.0957.
[6] J. Yang, J. Wright, T. Huang, Y. Ma, Image super-resolution as sparse representation of raw image patches, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, CVPR 2008, IEEE, 2008, pp. 1–8.
[7] M.G. Jafari, M.D. Plumbley, Fast dictionary learning for sparse representations of speech signals, IEEE J. Sel. Top. Signal Process. 5 (5) (2011) 1025–1031.
[8] A.M. Tillmann, On the computational intractability of exact and approximate dictionary learning, IEEE Signal Process. Lett. 22 (1) (2015) 45–49.
[9] K. Engan, S.O. Aase, J. Hakon Husoy, Method of optimal directions for frame design, in: Proceedings of 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 5, IEEE, 1999, pp. 2443–2446.
[10] M. Aharon, M. Elad, A. Bruckstein, K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation, IEEE Trans. Signal Process. 54 (11) (2006) 4311–4322.
[11] E.J. Candès, M.B. Wakin, An introduction to compressive sampling, IEEE Signal Process. Mag. 25 (2) (2008) 21–30.
[12] D.L. Donoho, X. Huo, Uncertainty principles and ideal atomic decomposition, IEEE Trans. Inf. Theory 47 (7) (2001) 2845–2862.
[13] D.L. Donoho, For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution, Commun. Pure Appl. Math. 59 (6) (2006) 797–829.
[14] B. Barak, J.A. Kelner, D. Steurer, Dictionary learning and tensor decomposition via the sum-of-squares method, preprint, arXiv:1407.1543.
[15] S. Arora, A. Bhaskara, R. Ge, T. Ma, More algorithms for provable dictionary learning, preprint, arXiv:1401.0579.
[16] S. Arora, R. Ge, T. Ma, A. Moitra, Simple, efficient, and neural algorithms for sparse coding, preprint, arXiv:1503.00778.
[17] M.S. Lewicki, T.J. Sejnowski, Coding time-varying signals using sparse, shift-invariant representations, Adv. Neural Inf. Process. Syst. (1999) 730–736.
[18] E. Smith, M.S. Lewicki, Efficient coding of time-relative structure using spikes, Neural Comput. 17 (1) (2005) 19–45.
[19] T. Blumensath, M. Davies, Sparse and shift-invariant representations of music, IEEE Trans. Audio Speech Lang. Process. 14 (1) (2006) 50–57.
[20] R.B. Grosse, R. Raina, H. Kwong, A.Y. Ng, Shift-invariant sparse coding for audio classification, in: Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2007, Vancouver, BC, Canada, July 19–22, 2007, 2007, pp. 149–158.
[21] M.D. Zeiler, D. Krishnan, G.W. Taylor, R. Fergus, Deconvolutional networks, in: 2010 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, 2010, pp. 2528–2535.
[22] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, Y.L. Cun, Learning convolutional feature hierarchies for visual recognition, in: Advances in Neural Information Processing Systems, 2010, pp. 1090–1098.
[23] H. Bristow, A. Eriksson, S. Lucey, Fast convolutional sparse coding, in: CVPR, IEEE, 2013, pp. 391–398.
[24] R. Chalasani, J.C. Principe, N. Ramakrishnan, A fast proximal method for convolutional sparse coding, in: IJCNN, IEEE, 2013, pp. 1–5.
[25] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci. 2 (1) (2009) 183–202.
[26] B. Wohlberg, Efficient convolutional sparse coding, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2014, pp. 7173–7177.
[27] B. Wohlberg, Efficient algorithms for convolutional sparse representations, IEEE Trans. Image Process. 25 (1) (2016) 301–315.
[28] F. Heide, W. Heidrich, G. Wetzstein, Fast and flexible convolutional sparse coding, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 5135–5143.
[29] J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online dictionary learning for sparse coding, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 689–696.
[30] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends® Mach. Learn. 3 (1) (2011) 1–122.
[31] J. Douglas, H.H. Rachford, On the numerical solution of heat conduction problems in two and three space variables, Trans. Am. Math. Soc. (1956) 421–439.
[32] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. B 58 (1994) 267–288.
[33] B. Kong, C.C. Fowlkes, Fast convolutional sparse coding (FCSC), Tech. rep., UCI, 2014.
[34] N. Parikh, S. Boyd, Proximal algorithms, Found. Trends Optim. 1 (3) (2014) 127–239, http://dx.doi.org/10.1561/2400000003.

Michal Šorel received the M.Sc. and Ph.D. degrees in computer science from the Charles University in Prague, Czechia, in 1999 and 2007, respectively. In 2012 and 2013 he worked at the Heriot-Watt University in Edinburgh, Scotland and the University of Bern, Switzerland. Currently he is a research fellow in the Institute of Information Theory and Automation, Czech Academy of Sciences. His research interests lie on the interface of computational mathematics, machine learning and image processing. His research work currently concentrates on development of efficient image restoration algorithms. He is a member of the IEEE.

Filip Šroubek received the M.Sc. degree in computer science from the Czech Technical University, Prague, Czech Republic in 1998 and the Ph.D. degree in computer science from Charles University, Prague, Czech Republic in 2003. From 2004 to 2006, he was on a postdoctoral position in the Instituto de Optica, CSIC, Madrid, Spain. In 2010 and 2011, he was the Fulbright Visiting Scholar at the University of California, Santa Cruz. He is currently with the Institute of Information Theory and Automation, the Czech Academy of Sciences. Filip Sroubek is an author of eight book chapters and over 80 journal and conference papers on image fusion, blind deconvolution, super-resolution, and related topics.