Efficient Algorithms for Convolutional Sparse Representations

Brendt Wohlberg

EDICS: SMR-REP

***Abstract*—When applying sparse representation techniques to images, the standard approach is to independently compute the representations for a set of overlapping image patches. This method performs very well in a variety of applications, but results in a representation that is multi-valued and not optimised with respect to the entire image. An alternative representation structure is provided by a *convolutional sparse representation*, in which a sparse representation of an entire image is computed by replacing the linear combination of a set of dictionary vectors by the sum of a set of convolutions with dictionary filters. The resulting representation is both single-valued and jointly optimised over the entire image. While this form of sparse representation has been applied to a variety of problems in signal and image processing and computer vision, the computational expense of the corresponding optimisation problems has restricted application to relatively small signals and images. This paper presents new, efficient algorithms that substantially improve on the performance of other recent methods, contributing to the development of this type of representation as a practical tool for a wider range of problems.**

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

I. Introduction

Sparse representation [1], [2] is a widely used technique for a very broad range of signal and image processing applications. Given a signal $\mathbf{s}$ and a *dictionary* matrix D, *sparse coding* is the inverse problem of finding the sparse representation $\mathbf{x}$ with only a few non-zero entries such that $D\mathbf{x} \approx \mathbf{s}$. Most sparse coding algorithms optimize a functional consisting of a data fidelity term and a sparsity inducing penalty

$$\arg\min_{\mathbf{x}} \frac{1}{2} \|D\mathbf{x} - \mathbf{s}\|_2^2 + \lambda R(\mathbf{x}) \;, \tag{1}$$

or constrained forms such as

$$\arg\min_{\mathbf{x}} R(\mathbf{x}) \;\text{ such that }\; \|D\mathbf{x} - \mathbf{s}\|_2 \leq \epsilon \;, \tag{2}$$

or

$$\arg\min_{\mathbf{x}} \|D\mathbf{x} - \mathbf{s}\|_2^2 \;\text{ such that }\; R(\mathbf{x}) \leq \tau \;, \tag{3}$$

where $R(\cdot)$ denotes a sparsity inducing function such as the ℓ^1 norm or the ℓ^0 "norm"[1]. The two leading families of sparse coding methods are a wide variety of convex optimization algorithms (e.g. Alternating Direction Method of Multipliers (ADMM) [3]) solving Eq. (1) when $R(\mathbf{x}) = \|\mathbf{x}\|_1$ and a family of greedy algorithms (e.g. Matching Pursuit (MP) [4] and Orthogonal Matching Pursuit (OMP) [5]) for approximate solution of Eq. (2) or Eq. (3) when $R(\mathbf{x}) = \|\mathbf{x}\|_0$.

If the dictionary D is analytically defined and corresponds to a linear operator with a fast transform (e.g. the Discrete Wavelet Transform), a representation for an entire signal or image can easily be computed. More recently, however, it has been realised that improved performance can be obtained by learning the dictionary from a set of training data relevant to a specific problem [6]; this inverse problem is known as *dictionary learning*. In this case computing a sparse representation for an entire signal is not feasible, the usual approach being to apply the decomposition independently to a set of overlapping blocks covering the signal.

An alternative representation structure is provided by a *convolutional* sparse representation, which models an entire signal or image as a sum over a set of convolutions of coefficient maps, of the same size as the signal or image, with their corresponding dictionary filters. While convolutional forms can be constructed for each of Eq. (1)–(3), the focus here will be on Eq. (1) with $R(\mathbf{x}) = \|\mathbf{x}\|_1$, i.e. the Basis Pursuit DeNoising (BPDN) [7] problem, the corresponding convolutional form being Convolutional BPDN (CBPDN), defined as

$$\arg\min_{\{\mathbf{x}_m\}} \frac{1}{2} \Big\| \sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \Big\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1 \;, \tag{4}$$

where $\{\mathbf{d}_m\}$ is a set of M dictionary *filters*, $*$ denotes convolution, and $\{\mathbf{x}_m\}$ is a set of coefficient maps. (For notational simplicity $\mathbf{s}$ and each of the $\{\mathbf{x}_m\}$ are considered to be N dimensional vectors, where N is the number of pixels in an image.) This type of representation has already been considered for a variety of signal and image processing and computer vision applications, but its use is constrained by the considerable computational expense of the associated sparse coding and dictionary learning problems, which has restricted its application to relatively small signals and images.

The present paper addresses the development of a more efficient algorithm for minimising Eq. (4), and extends the approach to the corresponding dictionary learning problem. The main contributions are:

- A thorough literature survey that reveals the existence of two equivalent but almost entirely independent streams of research related to these representations (Sec. II).
- An efficient ADMM algorithm for minimising Eq. (4) (Sec. III). The main component of this algorithm, initially proposed in an earlier work [8], is an efficient method

The author is with Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA. Email: brendt@lanl.gov, Tel: +1 505 667 6886

This research was supported by the U.S. Department of Energy through the LANL/LDRD Program and by UC Lab Fees Research grant 12-LR-236660.

[1]The quotes represent a reminder that this is not a true norm.

for solving a large linear system with a specific structure (Sec. III-B) in a way that reduces the computational cost from the $\mathcal{O}(M^3)$ of the previous approach to $\mathcal{O}(M)$. Extensions to [8] include the use of over-relaxation to improve the rate of convergence, and a more thorough comparison of the options for solving the main linear system and for selecting the ADMM penalty parameter.
- A performance comparison with leading alternative algorithms for the CBPDN problem, including the development of a new, substantially faster variant of one of these methods (Sec. IV).
- Extension of the efficient linear system solution method for CBPDN to the more difficult linear system encountered in dictionary learning based on CBPDN (Sec. V).
- Proposing the learning and use of multi-scale dictionaries, both of which are far simpler in the context of convolutional sparse representations than in the standard patch-based context (Sec. V-D).

While these techniques are also applicable to 1-d signals (see e.g. [9], [10]), discussion here focuses on greyscale images. In all computational experiments the 8 bit images are divided by 255 so that pixel values are within the interval [0,1], and, following common practice (see e.g. [11], [12]), convolutional sparse coding/dictionary learning algorithms are applied to test/training images after a highpass filtering preprocessing step[2].

II. Literature Survey

There are two independent streams of research, with almost no cross-citation, related to the convolutional form of sparse representations.

A. Convolutional Sparse Representations

The more recent of these streams is a product of the deep learning community; the convolutional form of the representation is primary, inspired by the desire to modify convolutional neural networks [13] to provide a generative model. The idea appears to have been proposed independently and almost simultaneously by Zeiler et al. [14] and Yang et al. [15], although the latter, while also originating within the machine learning community, does not seem to have influenced any later work in this direction. Subsequent work within this stream has also tended to consider the convolutional form of the representation as an integral part of a multi-layer neural network with application to image classification tasks [11], [12], [16], [17], although some more recent work has specifically addressed the concept from a more traditional sparse representations perspective [18]–[21].

Most of these works [11], [12], [14], [16], [20], [21] have posed the sparse coding and dictionary learning problems in the form of CBPDN, the exceptions being probabilistic/Bayesian models [17], [19] and convolutional extensions [18] of MP and K-SVD [22].

[2]The sparse representation is therefore a representation of the highpass component, and a complete image representation includes both the lowpass component and the sparse representation.

The most frequent application area within this stream has been image classification [11], [12], [14], [15], [17] and detection [11], [16] tasks, but segmentation in biological imagery has also been considered [19], and image denoising has received very brief attention [14].

B. Translation-Invariant Sparse Representations

The other stream of research has origins in the neuroscience and signal processing communities, the unifying concept being the construction of *translation-invariant* representations. The resulting representation is equivalent to the convolutional representation, but the motivation is translation-invariance and the convolutional form is derived rather than given. The earliest work to construct representations directly equivalent to Eq. (4) is that of Lewicki and Sejnowski [23]. Work within this stream [23]–[44] has concentrated primarily on properties of the representation and signal processing applications. A related idea that is not equivalent to convolutional representations is the construction of representations that are invariant to continuous translations via Taylor series or other interpolation methods [45], [46]. Some of these works have considered constructing representations that are invariant to a more general set of transformations [25], [40]–[42], most commonly being image translations and rotations.

A range of sparse coding and dictionary learning forms have been considered within this stream, including penalized form Eq. (1) with a log function based sparsity inducing regularization terms [24] or with ℓ^1 regularization [25], [35], [39]–[41]; probabilistic/Bayesian models [23], [26]–[30], [37], which generally correspond to various forms of Eq. (1) with either log [26], [29] or ℓ^1 [37] regularization; and constrained forms Eq. (2) [38], [42] or Eq. (3) [32], [34], [36], [43], [44].

Many of these works focus on properties of the representation rather than considering any specific application. The majority of applications that have been addressed within this stream have been to signal decomposition [27], [29], [30], [47], classification [35], and blind source separation [28], [30], [37] problems in audio [27]–[30], [35], [37], [47] or medical diagnostic signals [28], [32], [44]. Image denoising [38], image inpainting [39], [43], and video decomposition [26] have received some attention.

C. Sparse Coding Algorithms

Solutions for the convolutional constrained forms of sparse coding have employed convolutional extensions of MP [18], [19], [31], [36], [38] or variants of OMP [32], [34], [42]–[44]. In most of these cases the primary focus of the work is not on sparse coding algorithm development, and only [18], [19], [36], [38], [42] discuss efficient convolutional extensions of these methods in any detail.

A wide variety of algorithms have been proposed for CBPDN and related unconstrained forms. A few different approaches combine heuristic dictionary subset selection and optimization within that subset; these include a greedy subset search via fast convolution followed by a conjugate gradient method to compute the optimal coefficients on the selected subset [23], a similar greedy subset selection followed by an

IRLS algorithm to solve the BPDN problem on the subset [29], [30], [47], and the feature-sign search algorithm [35]. Other methods include block coordinate relaxation [39], [40] (for a restricted variant of CBPDN in which at most one filter is permitted to be active at each spatial location), coordinate descent [11], an alternating minimization method (not ADMM) [14], ISTA [12] or equivalent to ISTA but not identified as such [37], [41], FISTA [16], [20], and ADMM [21].

Most of these methods solve the problem in the spatial domain, but [21], [35], [37], [38], [43], [44] exploit the convolution theorem to solve at least part of the problem in the Discrete Fourier Transform (DFT) domain.

D. Dictionary Learning Algorithms

A number of different authors have considered the development of convolutional extensions of the K-SVD dictionary update [18], [19], [33], [36], [38], [43]. A block coordinate relaxation approach [44] shares the property of the K-SVD of updating a single dictionary element at a time. The MoTIF algorithm [31] is rather unusual in its strategy of iteratively growing the dictionary with a procedure reminiscent of Gram-Schmidt. These methods tend to be paired with greedy MP/OMP sparse coding algorithms for the coefficient update step.

The most widely used dictionary updates are convolutional forms of gradient descent [14], [26]–[29], [37], [39]–[41], [47] and Method of Optimal Directions (MOD) [12], [32], [34] or variants thereof [21], [35]. Other methods that have been considered include a quasi-Newton method [20], stochastic gradient descent [11], [16], stochastic Levenberg-Marquardt [42], and block coordinate relaxation [44].

Most dictionary learning methods are composed of alternation at the outer level between a sparse coding stage and a dictionary update stage, each of which may consist of iterations over a set of inner update steps. An alternative approach is a more tightly coupled algorithm directly alternating between the inner sparse coding and dictionary update steps [21], [37], the more recent of which [21] is derived with an Augmented Lagrangian framework.

As in the case of sparse coding, most methods operate in the spatial domain, but [21], [35], [37], [43], [44] perform the dictionary update in the DFT domain.

III. ADMM Algorithm for CBPDN

Given the very good performance of ADMM for the standard BPDN problem, it is a natural choice to consider for CBPDN. The general outline of the method derived here is very similar to the sparse coding component of the Augmented Lagrangian dictionary learning algorithm proposed by Bristow et al. [21] and described in more detail, with some errors corrected, in an independent technical report [48]. At a superficial level, the only difference is that here the ADMM algorithm is derived in the spatial domain, with one of the sub-problems being solved in the frequency domain, whereas they directly derive the algorithm using a mixture of spatial and frequency domain variables. A much more important difference, however, is the use of a far more efficient method (first introduced in [8]) for solving the linear system that represents the bulk of the computational cost of the algorithm.

The ADMM iterations for solving the optimization

$$\arg\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) + g(\mathbf{y}) \;\text{ such that }\; A\mathbf{x} + B\mathbf{y} = \mathbf{c} \tag{5}$$

are, in *scaled form* [3, Sec. 3.1.1]

$$\mathbf{x}^{(j+1)} = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\rho}{2} \left\| A\mathbf{x} + B\mathbf{y}^{(j)} - \mathbf{c} + \mathbf{u}^{(j)} \right\|_2^2 \tag{6}$$

$$\mathbf{y}^{(j+1)} = \arg\min_{\mathbf{y}} g(\mathbf{y}) + \frac{\rho}{2} \left\| A\mathbf{x}^{(j+1)} + B\mathbf{y} - \mathbf{c} + \mathbf{u}^{(j)} \right\|_2^2 \tag{7}$$

$$\mathbf{u}^{(j+1)} = \mathbf{u}^{(j)} + A\mathbf{x}^{(j+1)} + B\mathbf{y}^{(j+1)} - \mathbf{c} \;. \tag{8}$$

Rewriting Eq. (4) in a suitable form by introducing an auxiliary variable with a constraint, we have

$$\arg\min_{\{\mathbf{x}_m\},\{\mathbf{y}_m\}} \frac{1}{2} \Big\| \sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \Big\|_2^2 + \lambda \sum_m \|\mathbf{y}_m\|_1 \;\text{ s.t. }\; \mathbf{x}_m - \mathbf{y}_m = 0 \;, \tag{9}$$

so the ADMM iterations for our problem are

$$\{\mathbf{x}_m\}^{(j+1)} = \arg\min_{\{\mathbf{x}_m\}} \frac{1}{2} \Big\| \sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \Big\|_2^2 + \frac{\rho}{2} \sum_m \left\| \mathbf{x}_m - \mathbf{y}_m^{(j)} + \mathbf{u}_m^{(j)} \right\|_2^2 \tag{10}$$

$$\{\mathbf{y}_m\}^{(j+1)} = \arg\min_{\{\mathbf{y}_m\}} \lambda \sum_m \|\mathbf{y}_m\|_1 + \frac{\rho}{2} \sum_m \left\| \mathbf{x}_m^{(j+1)} - \mathbf{y}_m + \mathbf{u}_m^{(j)} \right\|_2^2 \tag{11}$$

$$\mathbf{u}_m^{(j+1)} = \mathbf{u}_m^{(j)} + \mathbf{x}_m^{(j+1)} - \mathbf{y}_m^{(j+1)} \;. \tag{12}$$

The stopping criteria discussed in [3, Sec. 3.3] provide a more effective means of ensuring convergence to suitable accuracy than simply specifying a maximum allowed number of iterations[3].

Sub-problem Eq. (11) is solved via shrinkage/soft thresholding as

$$\mathbf{y}_m^{(j+1)} = \mathcal{S}_{\lambda/\rho} \left(\mathbf{x}_m^{(j+1)} + \mathbf{u}_m^{(j)} \right) \;, \tag{13}$$

where

$$\mathcal{S}_\gamma(\mathbf{u}) = \operatorname{sign}(\mathbf{u}) \odot \max(0, |\mathbf{u}| - \gamma) \;, \tag{14}$$

with $\operatorname{sign}(\cdot)$ and $|\cdot|$ of a vector considered to be applied element-wise, and $\odot$ denoting element-wise multiplication. The computational cost of this sub-problem is $\mathcal{O}(MN)$. The only computationally expensive step is solving Eq. (10), which is of the form

$$\arg\min_{\{\mathbf{x}_m\}} \frac{1}{2} \Big\| \sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \Big\|_2^2 + \frac{\rho}{2} \sum_m \|\mathbf{x}_m - \mathbf{z}_m\|_2^2 \;. \tag{15}$$

A. DFT Domain Formulation

An obvious approach is to attempt to exploit the Fast Fourier Transform (FFT) for efficient implementation of the convolution via the DFT convolution theorem. Define linear operators D_m such that $D_m \mathbf{x}_m = \mathbf{d}_m * \mathbf{x}_m$, and denote the variables D_m, $\mathbf{x}_m$, $\mathbf{s}$, and $\mathbf{z}_m$ in the DFT domain by $\hat{D}_m$, $\hat{\mathbf{x}}_m$, $\hat{\mathbf{s}}$, and $\hat{\mathbf{z}}_m$ respectively. It is easy to show via the DFT convolution theorem that Eq. (15) is equivalent to

$$\arg\min_{\{\hat{\mathbf{x}}_m\}} \frac{1}{2} \Big\| \sum_m \hat{D}_m \hat{\mathbf{x}}_m - \hat{\mathbf{s}} \Big\|_2^2 + \frac{\rho}{2} \sum_m \|\hat{\mathbf{x}}_m - \hat{\mathbf{z}}_m\|_2^2 \;, \tag{16}$$

[3] See Alg. 1 in the Supplementary Material.

with the $\{\mathbf{x}_m\}$ minimizing Eq. (15) being given by the inverse DFT of the $\{\hat{\mathbf{x}}_m\}$ minimizing Eq. (16). Defining

$$\hat{D} = \begin{pmatrix} \hat{D}_0 & \hat{D}_1 & \dots \end{pmatrix} \quad \hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_0 \\ \hat{\mathbf{x}}_1 \\ \vdots \end{pmatrix} \quad \hat{\mathbf{z}} = \begin{pmatrix} \hat{\mathbf{z}}_0 \\ \hat{\mathbf{z}}_1 \\ \vdots \end{pmatrix} \;, \quad (17)$$

this problem can be expressed as

$$\arg\min_{\hat{\mathbf{x}}} \frac{1}{2} \left\| \hat{D}\hat{\mathbf{x}} - \hat{\mathbf{s}} \right\|_2^2 + \frac{\rho}{2} \left\| \hat{\mathbf{x}} - \hat{\mathbf{z}} \right\|_2^2 \;, \quad (18)$$

the solution being given by the linear system

$$(\hat{D}^H \hat{D} + \rho I)\hat{\mathbf{x}} = \hat{D}^H \hat{\mathbf{s}} + \rho \hat{\mathbf{z}} \;. \quad (19)$$

B. Independent Linear Systems

Matrix $\hat{D}$ has a block structure consisting of M concatenated $N \times N$ diagonal matrices, where M is the number of filters and N is the number of samples in $\mathbf{s}$. $\hat{D}^H \hat{D}$ is an $MN \times MN$ matrix, but due to the diagonal block (*not* block diagonal) structure of $\hat{D}$, a row of $\hat{D}^H$ with its non-zero element at column n will only have a non-zero product with a column of $\hat{D}$ that has its non-zero element at row n. As a result, there are no products between elements of $\hat{D}$ corresponding to different frequency indices n, so that, as pointed out in [21], one need only solve N independent $M \times M$ linear systems to solve Eq. (19). The cost of the FFTs and these linear solves dominate the computational cost of the algorithm. Bristow et al. [21] do not specify how they solve these linear systems, but since they rate the computational cost of solving them as $\mathcal{O}(M^3)$, one must conclude that they apply a direct method such as Gaussian elimination. This can be a very effective solution when it is possible to precompute and cache an LU or Cholesky decomposition of the system matrix, but in this case that is, in general, impossible due to the $\mathcal{O}(M^2 N)$ storage requirements for these decompositions.

A careful analysis of the unique structure of this problem, however, reveals that there is an alternative, and vastly more effective solution. The independent systems, each of which has the sum of a diagonal and a rank-one matrix on the left hand side, can be solved very efficiently by re-arranging the independent systems in Eq. (19) into the form of Eq. (47), as described in Appendix A, and then solving using Eq. (51), derived by applying the Sherman-Morrison formula as described in Appendix B. The only vector operations in Eq. (51) are scalar multiplication, subtraction, and inner products, so that this method is $\mathcal{O}(M)$, instead of $\mathcal{O}(M^3)$ as in [21]. The cost of solving such a system at all M spatial indices is $\mathcal{O}(MN)$, and the cost of the FFTs is $\mathcal{O}(MN \log N)$; it is the cost of the FFTs that dominates the computational complexity, whereas in [21] the cost of the solutions of the linear systems in the DFT domain dominate the cost of the FFTs. Further computational improvement is possible, at the cost of additional memory requirements, by precomputing components of Eq. (51), e.g. $\mathbf{a}_n^H / (\rho + \mathbf{a}_n^H \mathbf{a}_n)$.

C. Performance Comparison: Linear Solvers for ADMM

The execution times and solution accuracies of four different methods of solving the linear system in single precision[4] are compared in Fig. 1. Computations were performed in both single and double precision arithmetic to allow evaluation of the corresponding execution time and solution accuracy trade-off. Dictionaries of different sizes were constructed by selecting subsets of a $12 \times 12 \times 512$ dictionary[5] learned using standard (non-convolutional) methods, and $\mathbf{s}$ was the 512×512 greyscale Lena image. After an initial 20 iterations of the ADMM algorithm (to avoid any unique characteristics of the linear system directly after initialisation of the iterations), the four different methods were used to solve Eq. (19) for the following iteration, recording the time required by each. Reconstruction error was computed as the relative residual error[6] with which solution $\hat{\mathbf{x}}$ satisfied the linear equation Eq. (19).

The following observations can be made from these results:

- Gaussian elimination (GE) gives solutions accurate to floating point precision, but is the slowest of all the methods, being vastly slower than SM, and having much worse scaling with M.
- Conjugate Gradient (CG) provides a better accuracy/time trade-off than GE, being substantially faster, while still providing solutions of acceptable accuracy.
- Sherman-Morrison (SM) is by far the fastest method, giving acceptable solution accuracy in single precision, and very high accuracy in double precision. The negligible variation in run time with precision suggests that use of a double precision solution might be preferred, but empirical evidence (see below) indicates that, in practice, the single precision solution accuracy is sufficient, and gives approximately the same rate of convergence for the ADMM algorithm as double precision solution.

Since these methods all exhibit a different trade-off between accuracy and speed, it is reasonable to ask what the effect is on algorithm performance. It is not worth expending a large amount of time to compute a very accurate solution since ADMM algorithms only require approximate solutions of each step [3, Sec. 3.4.4], but convergence will be slow or unreliable if the solutions are too inaccurate. The results presented in Fig. 2 show that SM gives a faster reduction in functional value[7] with time than CG with any choice of tolerance[8], even for the smallest M considered in Fig. 1. In this case CG with a relative residual tolerance of 10^{-1} is not much

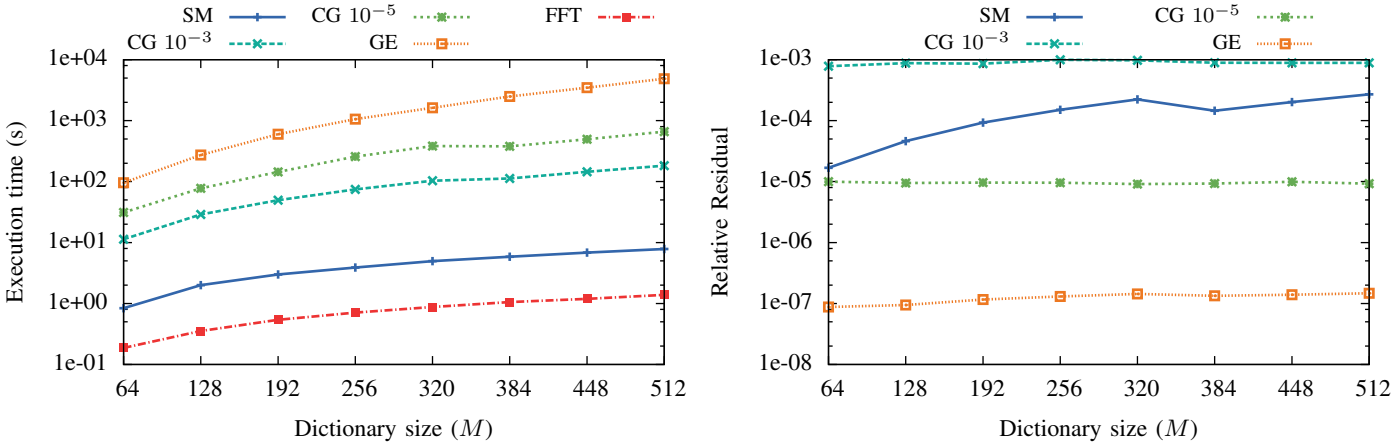
[4] A corresponding comparison for double precision is presented in Fig. 1 in the Supplementary Material.

[5] Included in the demonstration software distributed by the authors of [49].

[6] The relative residual error of $\mathbf{x}$ in the linear equation $A\mathbf{x} = \mathbf{b}$ is $\|A\mathbf{x} - \mathbf{b}\|_2 / \|\mathbf{b}\|_2$.

[7] When evaluating the functional in an ADMM algorithm such as this, different values will be obtained for computations using the primary ($\{\mathbf{x}_m\}$) and auxiliary ($\{\mathbf{y}_m\}$) variables. The auxiliary $\{\mathbf{y}_m\}$ is used for the experiments reported here since (i) $\{\mathbf{x}_m\}$ is merely an intermediate result and not part of the algorithm state [3, pg. 14], and (ii) it has been found empirically to give a curve that decays faster and is less subject to oscillations. Corresponding plots for primal and dual residuals for the ADMM algorithm are provided in Fig. 2–3 in the Supplementary Material.

[8] Smaller CG tolerances than in Fig. 1 are used in Fig. 2 since it is already clear from Fig. 1 that tolerances smaller than 10^{-3} are not competitive.



(a) Run time

(b) Solution error

Fig. 1. A comparison of execution times and solution errors for single-precision solution of a linear system for the 512×512 greyscale Lena image with dictionaries consisting of 12×12 filters, with $\lambda = 0.01$. Solution methods are Gaussian elimination (GE), Conjugate Gradient (CG) with relative residual tolerances of 10^{-3} and 10^{-5}, and Sherman-Morrison (SM). The combined run time for a forward and inverse FFT is provided as a reference.

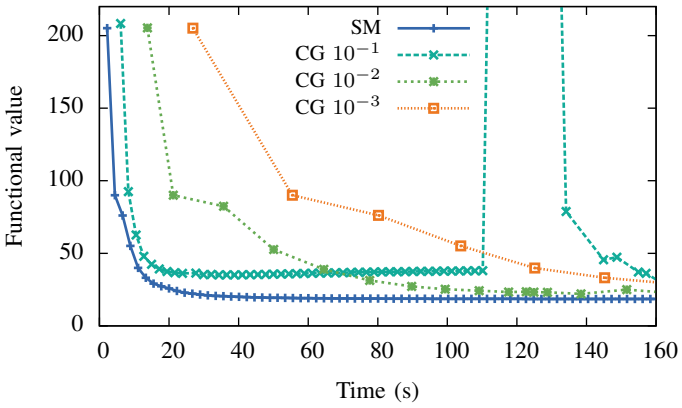


Fig. 2. A comparison of functional value evolution with time for the ADMM algorithm with Eq. (19) solved in single precision using Sherman-Morrison and CG with three different relative residual tolerances. The test image was the 512×512 greyscale Lena image with a learned $8 \times 8 \times 64$ dictionary and $\lambda = 0.01$.

slower than SM, but the algorithm does not converge; larger tolerance values give more reliable decrease in functional value with each iteration (note, though, that on a longer timescale than displayed in Fig. 2, CG with tolerances 10^{-2} and 10^{-3} also begin to exhibit some oscillations), but are very much slower. If plotted on the same graph, the SM result for double precision is indistinguishable from the single precision SM result shown.

D. ADMM Parameter Selection

The selection of a suitable penalty parameter ρ is critical to obtaining a good convergence rate. Two different strategies for selecting ρ are compared in this section. The first of these is the increasing parameter scheme, advocated in [21]

$$\rho_{j+1} = \begin{cases} \tau \rho_j & \text{if } \rho_j < \rho_{\max} \\ \rho_j & \text{otherwise} \end{cases} , \tag{20}$$

where ρ_j is the value of ρ at iteration j, and τ and $\rho_{\max}$ are fixed parameters, with typical values $\tau \in [1.01, 1.50]$ and $\rho_{\max} = 10^5$. The other strategy is the adaptive method proposed in [50] and described in [3, Sec. 3.4.1]

$$\rho_{j+1} = \begin{cases} \tau \rho_j & \text{if } \left\|\mathbf{r}^{(j)}\right\|_2 > \mu \left\|\mathbf{s}^{(j)}\right\|_2 \\ \tau^{-1} \rho_j & \text{if } \left\|\mathbf{s}^{(j)}\right\|_2 > \mu \left\|\mathbf{r}^{(j)}\right\|_2 \\ \rho_j & \text{otherwise} \end{cases} , \tag{21}$$

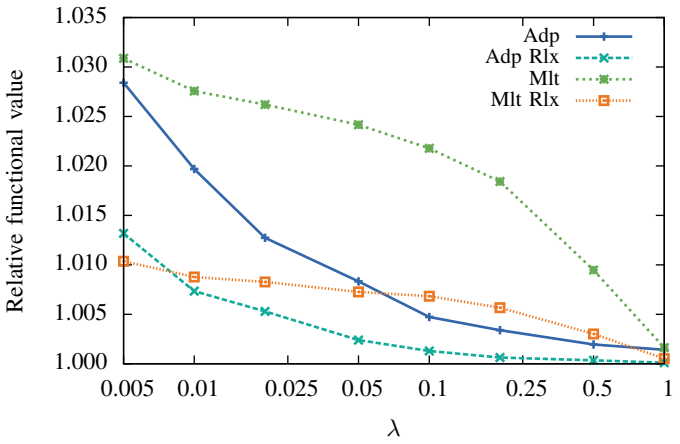
where τ and μ are constants, typical values being $\tau = 2$ and $\mu = 10$ [3], [50], and $\mathbf{r}^{(k)} = \mathbf{x}^{(k)} - \mathbf{y}^{(j)}$ and $\mathbf{s}^{(j)} = \rho_k(\mathbf{y}^{(j-1)} - \mathbf{y}^{(j)})$ are the primal and dual residuals [3, Sec. 3.3] respectively An additional variation for both methods is comparing the results with and without the relaxation method described in [3, Sec. 3.4.3], which involves replacing $\mathbf{x}_m^{(j+1)}$ in Eq. (11)–(12) with

$$\mathbf{x}_{\text{relax},m}^{(j+1)} = \alpha \mathbf{x}_m^{(j+1)} + (1 - \alpha) \mathbf{y}_m^{(j)} \ . \tag{22}$$

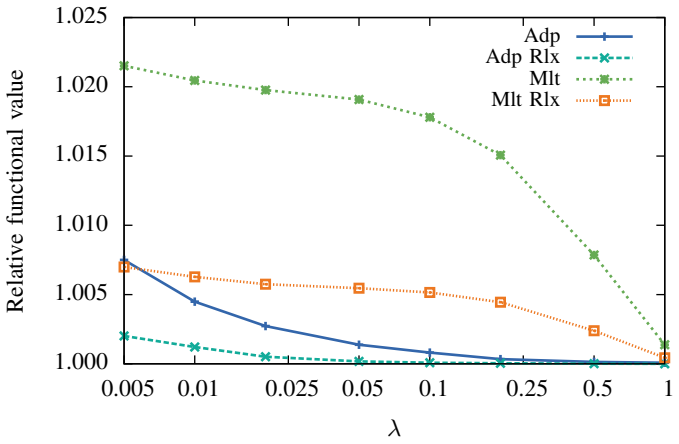
Relaxation parameter $\alpha = 1.8$ was found to give the best results, and is used whenever this method is applied, except where otherwise specified. All parameter selection experiments described in this section used the 512×512 greyscale Lena test image with a learned $8 \times 8 \times 64$ dictionary.

The first set of experiments, reported in Fig. 3, compare the different methods across different values of λ using relative functional values, computed by dividing functional values for each λ by the smallest functional value attained for that λ, by any method, at the maximum number of iterations (500). These results show that:

- The multiplicative update scheme without over-relaxation is not at all competitive with other methods.
- Including over-relaxation with the multiplicative update scheme substantially improves performance, but it remains inferior to the automatic scheme for all but the smallest values of λ when performance is compared at 50 iterations, and is uniformly inferior when comparisons are made at larger iteration counts.

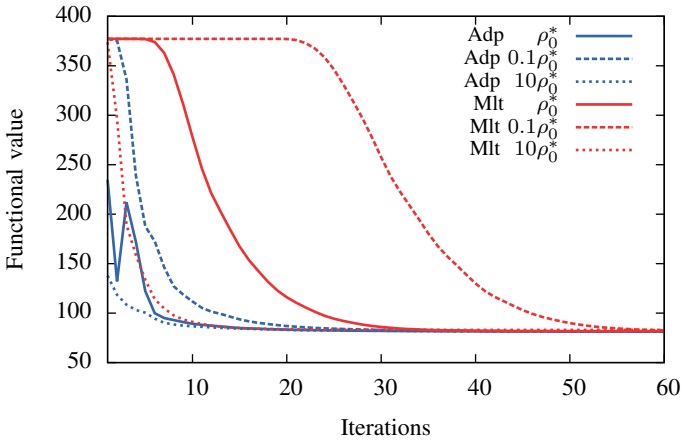


(a) Best ρ_0 at 50 iterations

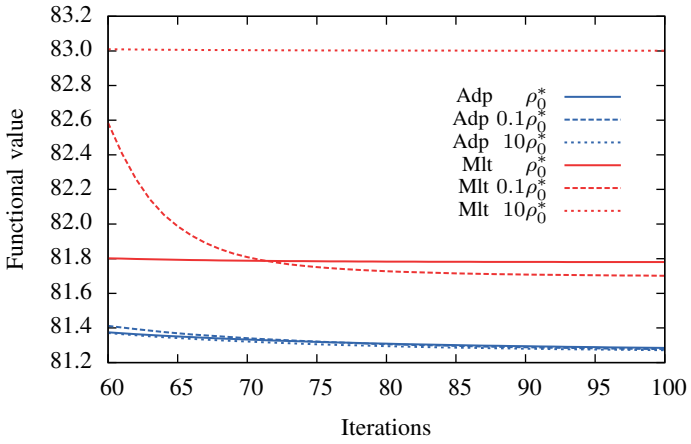


(b) Best ρ_0 at 100 iterations

Fig. 3. Relative functional values for different ρ update strategies with ρ_0 selected to minimize the functional value at 50 and 100 iterations. The update strategies are denoted by "Adp" (adaptive) and "Mlt" (fixed multiplicative update), with "Rlx" indicating the use of over-relaxation with the parameter $\alpha = 1.8$.



(a) 1 to 60 iterations



(b) 60 to 100 iterations

Fig. 4. Functional value behaviour for $\lambda = 0.05$ and different ρ update strategies with ρ_0 selected to minimize the functional value at 50 iterations. Both "Adp" (adaptive) and "Mlt" (fixed multiplicative update) strategies use over-relaxation parameter $\alpha = 1.8$. The evolution of the functional value for each method is shown for the optimum choice of ρ_0 *for that method* at 50 iterations, denoted by ρ_0^* as well as for ρ_0 set to 0.1 and 10 times ρ_0^*.

- Over-relaxation substantially improves performance for the adaptive scheme.

The second set of experiments, reported in Fig. 4, evaluate the sensitivities of the different methods to the correct choice of ρ_0 by comparing the decrease in functional value with iteration number for the optimum choice of ρ_0, as well as the optimum value multiplied by 0.1 and 10. These experiments show that:

- The multiplicative update scheme with ρ_0 chosen for the smallest functional value at 50 iterations converges substantially slower than the adaptive scheme. Initial convergence is much faster if ρ_0 is chosen to be 10 times larger, but then the functional value stagnates at a level substantially above the minimum. (If ρ_0 is instead chosen for best performance at 100 iterations then the stagnation effect is reduced, but at the expense of even slower decrease in the functional value, i.e. the initial plateau becomes larger.) The performance of the multiplicative update scheme with respect to the adaptive method, both in terms of decay rate and stagnation effect, appears to improve slightly for smaller values of λ.

By inspection of 2D arrays of relative functional values for different λ and ρ_0 values[9], it was determined that $\rho_0 = 100\lambda + 0.5$ is a simple heuristic that provides good performance for the experimental conditions – type of data, data scaling, and preprocessing, etc. – considered here.

IV. CBPDN Algorithm Comparison

Sec. II-C includes a brief summary of the different approaches that have been proposed for CBPDN. The earliest algorithms [23], [29], [30], [47], based on a heuristic subset selection stage, have been shown to be substantially slower

[9]See Fig. 4–6 in the Supplementary Material.

than a convolutional variant of the feature-sign search algorithm [35]. Of the more recent methods, the block coordinate relaxation algorithms [39], [40] do not solve the full CBPDN problem, and even the earlier variant of the frequency domain ADMM algorithm [21] is shown to be substantially faster, in most circumstances, than the coordinate descent algorithm [11]. Since FISTA is known to be substantially more effective than ISTA, it is reasonable to conclude that the FISTA algorithm for convolutional BPDN [16], [20] outperforms those based on ISTA [12], [37], [41], and FISTA has also been shown [20] to be faster than the coordinate descent algorithm [11].

Given these previous performance comparison results in the literature, as well as the demonstration in the preceding section of the performance of the ADMM variant proposed here with respect to the earlier variant [21], it remains only to compare the performance of this proposed algorithm with that of the feature-sign search algorithm [35] and of FISTA [16], [20].

A. Feature-Sign Search

Since the feature-sign search (FSS) algorithm [35] operates by iteratively estimating the set of non-zero coefficients, and is able to compute the solution on that set to floating point precision, it is not straightforward to compare its performance with a method such as ADMM in terms of functional value decay with time. Instead, the total computation time for the FSS algorithm is compared with that of ADMM with three different relative error stopping tolerances [3, Sec. 3.3], the smallest of which, $\epsilon_{\mathrm{rel}} = 10^{-4}$, is more than adequate for most image processing applications. All FSS results were computed using a publicly available implementation [51] by the authors of [35].

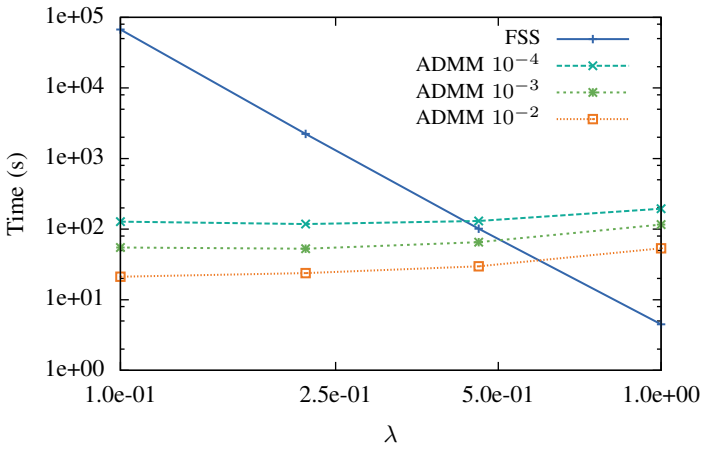


Fig. 5. A comparison of computation times for the feature-sign search (FSS) algorithm and the ADMM algorithm with three different relative error tolerances. The sparse coding was applied to the greyscale Lena image downsampled to a size of 256×256 pixels, and the dictionary was $8 \times 8 \times 64$.

It can be seen[10] from Fig. 5 that the performance of the feature-sign search algorithm is strongly dependent on λ. The method substantially outperforms ADMM for large values of λ and small N, but performance falls off very rapidly as λ is decreased and as the image size is increased. It is therefore a good choice for the type of computer vision problem in which large reconstruction error is acceptable, and applied to relatively small images, but for image processing problems requiring a low reconstruction error, and therefore much smaller λ (even the smallest value of $\lambda = 0.1$ in Fig. 5 is usually far too large for image restoration problems), ADMM is faster by a few orders of magnitude.

B. FISTA

While there is evidence that ADMM is significantly faster than FISTA for at least some problems [52], an empirical comparison with the recently proposed FISTA algorithm for CBPDN [20] is warranted. The main computational cost of this algorithm is in the computation of the gradient of the data fidelity term, which is performed in the spatial domain. This gradient can, however, easily be computed in the frequency domain as $\hat{D}^H(\hat{D}\hat{\mathbf{x}} - \hat{\mathbf{s}})$, i.e. the gradient of the term $\frac{1}{2}\left\|\hat{D}\hat{\mathbf{x}} - \hat{\mathbf{s}}\right\|_2^2$ in Eq. (18). A new variant of the FISTA algorithm that exploits the frequency domain gradient computation is also included in the performance comparisons reported here. These comparisons were performed for a set of different dictionaries (learned on a separate set of images) and λ values, using the 512×512 greyscale Lena test image. The ADMM results were generated using over-relaxation with $\alpha = 1.8$, with $\rho_0 = 100\lambda + 0.5$, and with adaptive ρ. FISTA parameters were selected by a grid search over a range of values around the fixed $L_0 = 1$ and $\eta = 5$ values of [20]; the best values of L_0 were 0.05, 0.5, or 1, and the best values of η were 2 or 5. (Note that ADMM parameters were all automatically set, or set via an effective heuristic, whereas it is not clear how to select the best FISTA parameters without a search over the parameter space.)

The results reported in Fig. 6 indicate that FISTA with spatial domain computation of the gradient is substantially slower than ADMM. FISTA with frequency domain computation of the gradient is competitive with ADMM for larger values of λ and smaller dictionaries (i.e. smaller M), but is also substantially slower than ADMM for smaller λ and larger M values[11].

V. Dictionary Learning

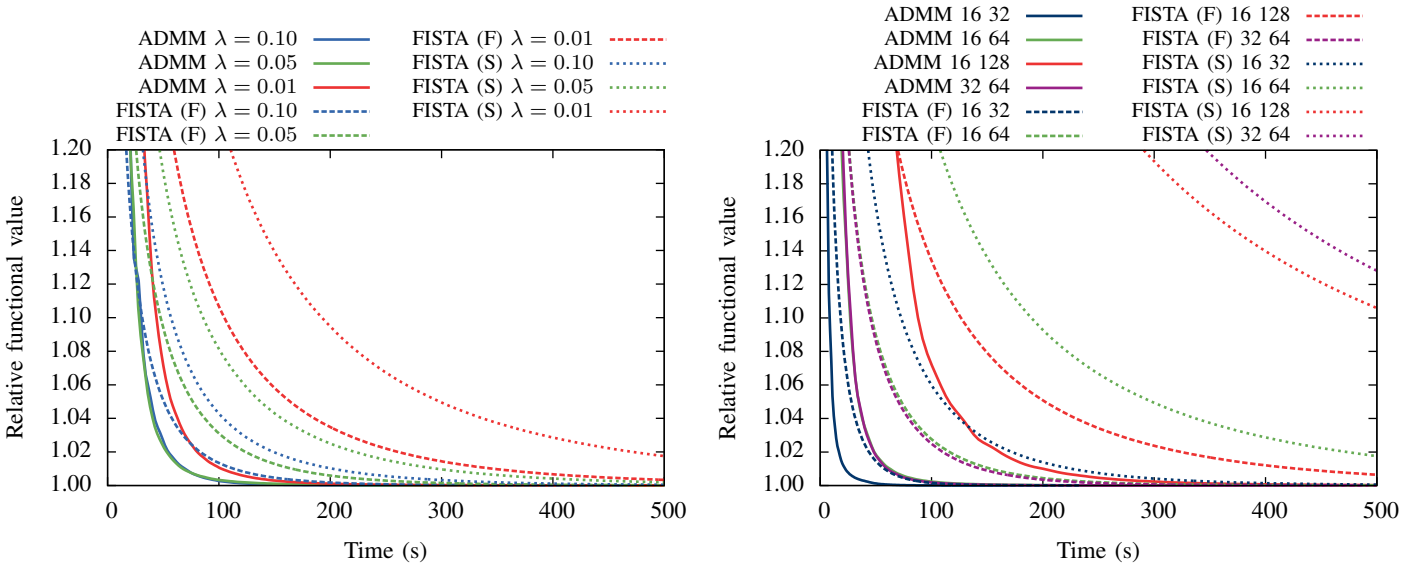
The natural dictionary learning extension of CBPDN is

$$\underset{\{\mathbf{d}_m\},\{\mathbf{x}_{k,m}\}}{\arg\min} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{d}_m * \mathbf{x}_{k,m} - \mathbf{s}_k \right\|_2^2 + \lambda \sum_k \sum_m \|\mathbf{x}_{k,m}\|_1 \quad \text{such that} \quad \|\mathbf{d}_m\|_2 = 1 \;\forall m \;, \tag{23}$$

where the constraint on the norms of filters $\mathbf{d}_m$ is required to avoid the scaling ambiguity between filters and coefficients. The standard approach is to solve this problem via alternating minimization with respect to coefficients and dictionary. The

[10] A corresponding comparison for varying number of pixels N in the image is presented in Fig. 7 in the Supplementary Material.

[11] Additional experiments reported in Sec. III-B in the Supplementary Material indicate that similar relative performance is observed when comparing these two algorithms in the context of some image reconstruction applications.



(a) $8 \times 8 \times 64$ dictionary

(b) $\lambda = 0.05$

Fig. 6. Comparison of time evolution of relative functional values for ADMM, and FISTA with the gradient computed in the spatial domain "(S)" and frequency domain "(F)". (a) provides a comparison for different values of λ with a dictionary consisting of 64 filters of size 8×8, and (b) provides a comparison for different dictionary sizes ("method n m" indicates a dictionary of m filters of size $n \times n$) and $\lambda = 0.05$.

minimization with respect to $\{\mathbf{x}_{k,m}\}$ involves solving the Multiple Measurement Vector (MMV) extension of CBPDN, which is trivial since the problems for each k are decoupled from one another, but $\{\mathbf{d}_m\}$ is more challenging since the problems for different k are coupled.

Ignoring the constraint on the norm of $\mathbf{d}_m$, which is usually applied as a post-processing normalization step after the update, the minimization with respect to $\{\mathbf{d}_m\}$ can be expressed as

$$\operatorname*{arg\,min}_{\{\mathbf{d}_m\}} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{d}_m * \mathbf{x}_{k,m} - \mathbf{s}_k \right\|_2^2 , \tag{24}$$

which is a convolutional form of MOD [53]. When computing the convolutions $\mathbf{d}_m * \mathbf{x}_{k,m}$ in the DFT domain, there is an implicit zero-padding of the filters $\mathbf{d}_m$ to the size of the coefficient maps $\mathbf{x}_{k,m}$; this can be overlooked when minimizing with respect to the coefficient maps, but must be explicitly represented when minimizing with respect to the filters to ensure that the filters resulting from the optimization have an appropriately constrained support in the spatial domain. Defining zero-padding operator P, Eq. (24) can be expressed in the DFT domain as

$$\operatorname*{arg\,min}_{\{\mathbf{d}_m\}} \frac{1}{2} \sum_k \left\| \sum_m (\widehat{P\mathbf{d}_m}) \odot \hat{\mathbf{x}}_{k,m} - \hat{\mathbf{s}}_k \right\|_2^2 . \tag{25}$$

Unfortunately, the spatial-domain operator P does not have a compact representation in the DFT domain, making an efficient direct DFT domain solution impossible. A variable splitting approach, however, makes it possible to solve this problem, including dictionary norm constraints, via an ADMM algorithm.

A. Constrained MOD Update

The desired filters can be obtained as $P^T \mathbf{d}_m$ after solving the constrained problem

$$\operatorname*{arg\,min}_{\{\mathbf{d}_m\}} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_m - \mathbf{s}_k \right\|_2^2 \text{ s. t. } \mathbf{d}_m \in C_{\mathrm{P}} \; \forall m \;, \tag{26}$$

where the $\mathbf{d}_m$ have the same spatial support as the $\mathbf{x}_{k,m}$, and

$$C_{\mathrm{P}} = \{\mathbf{x} \in \mathbb{R}^N : (I - PP^T)\mathbf{x} = 0\} \;. \tag{27}$$

Since we are setting up a constrained problem requiring an iterative solution, however, it is reasonable to also include the normalisation $\|\mathbf{d}_m\|_2 = 1$ (or $\|\mathbf{d}_m\|_2 \leq 1$) of the dictionary elements that is often, and suboptimally, performed as a postprocessing step after the dictionary update. Including the normalisation requirement $\|\mathbf{d}_m\|_2 = 1$, the constraint set becomes

$$C_{\mathrm{PN}} = \{\mathbf{x} \in \mathbb{R}^N : (I - PP^T)\mathbf{x} = 0, \; \|\mathbf{x}\|_2 = 1\} \;. \tag{28}$$

Employing the indicator function[12] $\iota_{C_{\mathrm{PN}}}$ of the constraint set C_{PN}, the constrained problem can be written in unconstrained form [54]

$$\operatorname*{arg\,min}_{\{\mathbf{d}_m\}} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_m - \mathbf{s}_k \right\|_2^2 + \sum_m \iota_{C_{\mathrm{PN}}}(\mathbf{d}_m) \;, \tag{29}$$

[12]The indicator function of a set S is defined as

$$\iota_S(X) = \begin{cases} 0 & \text{if } X \in S \\ \infty & \text{if } X \notin S \end{cases} .$$

and rewriting with an auxiliary variable in a form suitable for ADMM gives

$$\begin{aligned} \operatorname*{arg\,min}_{\{\mathbf{d}_m\},\{\mathbf{g}_m\}} \frac{1}{2} \sum_k \Big\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_m - \mathbf{s}_k \Big\|_2^2 + \sum_m \iota_{C_{\text{PN}}}(\mathbf{g}_m) \\ \text{such that } \; \mathbf{d}_m - \mathbf{g}_m = 0 \; \forall m \; . \end{aligned} \tag{30}$$

This problem can be solved via an ADMM algorithm

$$\begin{aligned} \{\mathbf{d}_m\}^{(j+1)} = \operatorname*{arg\,min}_{\{\mathbf{d}_m\}} \frac{1}{2} \sum_k \Big\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_m - \mathbf{s}_k \Big\|_2^2 + \\ \frac{\sigma}{2} \sum_m \Big\| \mathbf{d}_m - \mathbf{g}_m^{(j)} + \mathbf{h}_m^{(j)} \Big\|_2^2 \end{aligned} \tag{31}$$

$$\begin{aligned} \{\mathbf{g}_m\}^{(j+1)} = \operatorname*{arg\,min}_{\{\mathbf{g}_m\}} \sum_m \iota_{C_{\text{PN}}}(\mathbf{g}_m) + \\ \frac{\sigma}{2} \sum_m \Big\| \mathbf{d}_m^{(j+1)} - \mathbf{g}_m + \mathbf{h}_m^{(j)} \Big\|_2^2 \end{aligned} \tag{32}$$

$$\mathbf{h}_m^{(j+1)} = \mathbf{h}_m^{(j)} + \mathbf{d}_m^{(j+1)} - \mathbf{g}_m^{(j+1)} \; . \tag{33}$$

The $\{\mathbf{g}_m\}$ update is of the form

$$\operatorname*{arg\,min}_{\mathbf{x}} \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 + \iota_{C_{\text{PN}}}(\mathbf{x}) = \text{prox}_{\iota_{C_{\text{PN}}}}(\mathbf{y}) \; . \tag{34}$$

It is immediately clear from the geometry of the problem that

$$\text{prox}_{\iota_{C_{\text{PN}}}}(\mathbf{y}) = \frac{P P^T \mathbf{y}}{\|P P^T \mathbf{y}\|_2} \; , \tag{35}$$

or, if the normalisation $\|\mathbf{d}_m\|_2 \leq 1$ is desired instead,

$$\text{prox}_{\iota_{C_{\text{PN}}}}(\mathbf{y}) = \begin{cases} P P^T \mathbf{y} & \text{if } \; \big\|P P^T \mathbf{y}\big\|_2 \leq 1 \\ \frac{P P^T \mathbf{y}}{\|P P^T \mathbf{y}\|_2} & \text{if } \; \big\|P P^T \mathbf{y}\big\|_2 > 1 \end{cases} \; . \tag{36}$$

The problem for the $\{\mathbf{d}_m\}$ update is of form

$$\operatorname*{arg\,min}_{\{\mathbf{d}_m\}} \frac{1}{2} \sum_k \Big\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_m - \mathbf{s}_k \Big\|_2^2 + \frac{\sigma}{2} \sum_m \|\mathbf{d}_m - \mathbf{z}_m\|_2^2 \; . \tag{37}$$

In the DFT domain, where $\hat{X}_{k,m} = \text{diag}(\hat{\mathbf{x}}_{k,m})$, this becomes

$$\operatorname*{arg\,min}_{\{\hat{\mathbf{d}}_m\}} \frac{1}{2} \sum_k \Big\| \sum_m \hat{X}_{k,m} \hat{\mathbf{d}}_m - \hat{\mathbf{s}}_k \Big\|_2^2 + \frac{\sigma}{2} \sum_m \Big\| \hat{\mathbf{d}}_m - \hat{\mathbf{z}}_m \Big\|_2^2 \; . \tag{38}$$

Defining

$$\hat{X}_k = \begin{pmatrix} \hat{X}_{k,0} & \hat{X}_{k,1} & \dots \end{pmatrix} \; \hat{\mathbf{d}} = \begin{pmatrix} \hat{\mathbf{d}}_0 \\ \hat{\mathbf{d}}_1 \\ \vdots \end{pmatrix} \; \hat{\mathbf{z}} = \begin{pmatrix} \hat{\mathbf{z}}_0 \\ \hat{\mathbf{z}}_1 \\ \vdots \end{pmatrix} \tag{39}$$

this problem can be expressed as

$$\operatorname*{arg\,min}_{\hat{\mathbf{d}}} \frac{1}{2} \sum_k \big\| \hat{X}_k \hat{\mathbf{d}} - \hat{\mathbf{s}}_k \big\|_2^2 + \frac{\sigma}{2} \big\| \hat{\mathbf{d}} - \hat{\mathbf{z}} \big\|_2^2 \tag{40}$$

with solution

$$\Big(\sum_k \hat{X}_k^H \hat{X}_k + \sigma I \Big) \hat{\mathbf{d}} = \sum_k \hat{X}_k^H \hat{\mathbf{s}}_k + \sigma \hat{\mathbf{z}} \; . \tag{41}$$

This linear system can be solved by iterated application of the Sherman-Morrison formula, as described in Appendices C and D.

B. Interleaved $\mathbf{x}_{k,m}$ and $\mathbf{d}_m$ Updates

Given iterative algorithms for the $\mathbf{x}_{k,m}$ and $\mathbf{d}_m$ updates (i.e. the ADMM algorithms for CBPDN and Constrained MOD respectively), the immediate question is how these should be combined into a full dictionary learning algorithm. The standard approach is to alternate between sparse coding and dictionary update, solving the sparse coding problem with reasonable accuracy before moving on to the dictionary update. Since each sub-problem is an iterative algorithm, this would entail performing multiple iterations of each sub-problem before switching to the other. A reasonable strategy is to maintain the auxiliary variables for each sub-problem across the top-level iterations since performing a cold start at each switch of sub-problems would entail substantially reduced efficiency. The necessity of retaining these variables, in turn, suggests interleaving the ADMM iterations for each sub-problem into a single set of iterations, rather than combining the $\mathbf{x}_{k,m}$ and $\mathbf{d}_m$ updates in a way that treats each as a single functional unit[13]. A single iteration of the resulting algorithm consists of updates Eq. (10), (11), (12), (31), (32), and (33) in sequence.

The most obvious way of combining these updates is to transfer the primary variables across to the other update steps, i.e. $\mathbf{d}_m$ represent the dictionary in the sparse coding steps, and $\mathbf{x}_{k,m}$ represent the sparse code in the dictionary update steps. Such a combination turns out to be quite unstable in practice, with convergence being very sensitive to suitable choices of the ρ and σ parameters for each update. A far more stable algorithm is obtained if the updates are interleaved on their auxiliary variables, i.e. $\mathbf{g}_m$ represent the dictionary in the sparse coding steps, and $\mathbf{y}_{k,m}$ represent the sparse code in the dictionary update steps[14]. It is worth noting that such a combination is not derivable from single Augmented Lagrangian functional, as is the otherwise-similar dictionary learning approach of Bristow et al. [21]. Additional differences in derivation and algorithm are:

- they construct the Augmented Lagrangian using a mixture of spatial and frequency domain variables, while the derivation presented here poses the problem in the spatial domain, switching into the frequency domain where appropriate for efficient solution of relevant sub-problems,
- they derive the ADMM algorithm in unscaled rather than scaled form [3, Sec. 3.1.1], and
- most significantly, they propose the use of a direct method for solving Eq. (41).

The first two of these choices appears to lead to a slightly more complicated path to deriving solutions to at least one of the sub-problems (see section "Subproblem s" in [21], describing the sub-problem corresponding to the $\{\mathbf{g}_m\}$ update here), but do not have a significant effect on the resulting algorithm. The final choice is compared with alternatives in the following section.

[13]This general strategy, of interleaving the algorithms for sparse code and dictionary updates, has previously been proposed, but with substantially different algorithms for each update [55].

[14]See Alg. 2 in the Supplementary Material.

C. Performance Comparison: Linear Solvers for ADMM

The execution times and solution accuracies of three different methods of solving the linear system are compared[15] for fixed $M = 64$ and varying K in Fig. 7. Dictionaries of different sizes, M, were constructed by initialising with Gaussian random dictionaries of the appropriate sizes, and the different training image sets of size, K, were selected from a set of 256×256 training images derived from the MIRFlickr dataset [56]. After an initial 20 iterations of the ADMM algorithm, the different methods were used to solve Eq. (41) for the following iteration, recording the time required by each. Reconstruction error was computed as the relative residual error with which solution $\hat{\mathbf{d}}$ satisfies the linear equation Eq. (41).

The following observations can be made from these results:

- The CG 10^{-1} method is by far the fastest, but the solution accuracy is inadequate (see below).
- The CG 10^{-5} method has very similar solution accuracy to SM, but is substantially slower for small K values.
- In the sparse coding problem, the SM solution greatly outperforms the alternatives, but that is not always the case here, with other methods, including GE, becoming competitive for larger K values. This should not be surprising since GE has $\mathcal{O}(KM^3N)$ cost, while SM has $\mathcal{O}(K^2MN)$ cost (see Alg. 1 in Appendix D).

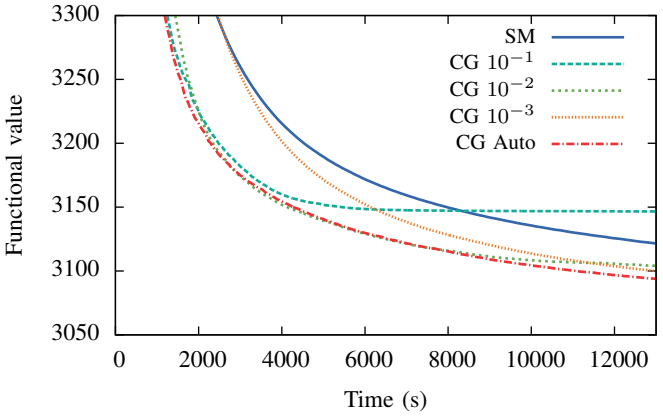


Fig. 8. A comparison of functional value evolution with time for the ADMM algorithm with Eq. (41) solved in single precision using Sherman-Morrison and CG with three different relative residual tolerances as well as an automatic tolerance selection method. The dictionary was $8 \times 8 \times 64$ and $\lambda = 0.1$.

The effects of the different trade-offs between time and accuracy represented by the different solution methods can only be evaluated by comparing performance within a number of iterations of the full dictionary learning algorithm. Such a comparison was performed using the same dictionary size, λ value, and training data (with $K = 32$) as for the experiments above, the results being presented in Fig. 8.

The double precision SM result is not shown since it would be indistinguishable from the single precision version on this graph; for this experiment the maximum fractional difference between the two (occurring within the first few iterations) is 3.3×10^{-5}, and the median is 3.8×10^{-6}. Thus double precision SM gives effectively the same solution, but at the cost of doubling the memory requirements (and is approximately 30% slower than single precision). Since CG with a sufficiently large tolerance gives approximately the same solution accuracy independent of precision, at an even greater computation time penalty, double precision CG is also not a useful option.

In addition to three different CG tolerances, these experiments also include an automatic CG tolerance method that starts with a tolerance of 10^{-1}, at each iteration setting it to $1/20$ times the ℓ^2 norm of the primal residual for the $\mathbf{d}_m$ update if that quantity is smaller than the current tolerance. In this experiment that auto-tolerance method is quite competitive, giving the overall best performance in terms of functional value reduction with time[16].

D. Multi-scale Dictionaries

The vast majority of dictionary learning research has focused on the learning of single-scale dictionaries (i.e. all dictionary elements are of the same size) within a patch-based context. Given the obvious benefits of a multi-scale representation (see e.g. [57, Sec. 3] for a discussion), it is not surprising, however, that the construction of multi-scale dictionaries has received at least some attention. The approaches that have been considered follow a few basic schemes:

Analytically defined The simplest approach is to analytically define a multi-scale family of basis functions, without any learning or adaptation of the dictionary from data. The design of a fast sparse coding algorithm using a multi-scale Gaussian chirp dictionary was described in [58].

Learned wavelet filters A wavelet basis can be optimised for sparse representation of images by a learning procedure applied to the associated filter bank [59], [60].

Transform domain dictionaries Standard dictionary learning applied in the wavelet transform domain [61], [62] or in a Laplacian pyramid [63] provides the dictionary with a multi-scale structure in the spatial domain.

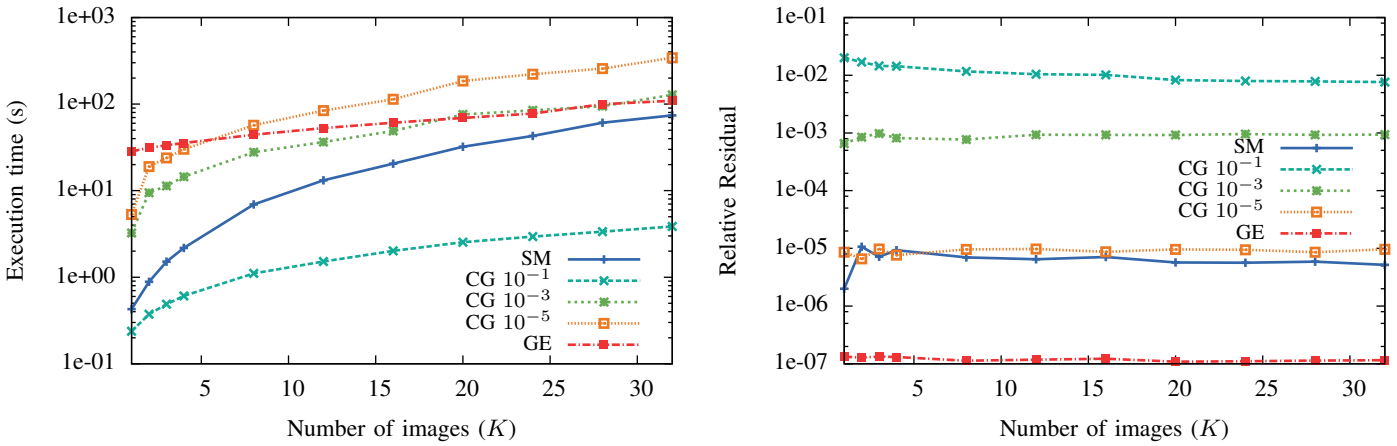
Quadtree spatial decomposition A set of dictionaries can be learned to give an image representation at the multiple patch sizes occurring within a quadtree decomposition [57], [64].

These methods all have structural constraints (either being imposed by the properties of the transform domain within which the sparse representation is computed, or from the quadtree spatial structure imposed on the dictionary), ultimately resulting from the difficulty of applying a multi-scale dictionary in a natural way within a patch-based framework.

In the convolutional sparse representation framework, in contrast, there is absolutely no reason why the dictionary filters should be of the same size, and multi-scale dictionaries can be defined in a very natural way, without any structural constraints on their form. Learning of such dictionaries is no more difficult than learning a single-scale dictionary, simply

[15] A corresponding comparison for fixed $K = 32$ and varying M is presented in Fig. 14 in the Supplementary Material.

[16] In contrast, the best reduction with respect to number of iterations is provided by SM and CG 10^{-3} (see Fig. 15 in the Supplementary Material).



(a) Run time (b) Solution error

Fig. 7. A comparison of execution times and solution errors for single-precision solution of a linear system corresponding to an $8 \times 8 \times 64$ dictionary with $\lambda = 0.1$. Solution methods are Gaussian elimination (GE), Conjugate Gradient (CG), and Sherman-Morrison (SM).

by replacing P by P_m in Eq. (25), (27), (28), (35), and (36). It is surprising, therefore, that the learning and use of multi-scale convolutional dictionaries has not previously been considered in imaging applications, and has only received very limited attention in a signal processing context (the possibility of using filters of different lengths is pointed out in [42], but not discussed in any detail).

The utility of this type of multi-scale dictionary ultimately depends on the performance in actual applications, a proper evaluation of which is beyond the scope of the present paper, and will be addressed in future work. An initial demonstration of the potential advantages is, however, provided by some simple experiments comparing the minimal CBPDN functional values for different dictionaries, and the trade-off of representation accuracy and number of non-zero coefficients for different dictionaries.

A set of five different dictionaries, all with 72 filters, was learned using the same set of 16 images, and then used to solve a CBPDN problem using a separate set of 16 image as a test set. A corresponding experiment was also performed using a set of four dictionaries, all with 96 filters, trained on the same set of 32 images and tested on the same set of 16 images. The results in Table I show that a multi-scale dictionary can provide a lower cost representation than a single-scale dictionary with the same number of filters of the same size as one of the filter sizes in the multi-scale dictionary.

A further experiment compares the variation of reconstruction error with number of non-zero components in the sparse representations for each of the dictionaries in the lower half of Table I. The reconstruction error curves were computed by sorting coefficients by their absolute value and successively zeroing them from smallest to largest, computing the new reconstruction error at each stage. The curves in Fig. 9 represent the reconstructions SNR *difference* between the reconstructions for the specified dictionary and the $8 \times 8 \times 96$ dictionary. It can be seen that the $12 \times 12 \times 96$ dictionary gives a higher SNR reconstruction for a small number of non-zero coefficients, but is inferior to the $8 \times 8 \times 96$ for many non-zero coefficients. The $16 \times 16 \times 96$ dictionary is initially superior to the $12 \times 12 \times 96$ dictionary, but its performance more quickly drops below that of the $8 \times 8 \times 96$ dictionary, and the maximum performance deficit is much greater. The multi-scale dictionary provides a good compromise between the three other dictionaries, giving the best reconstruction SNR at low numbers of non-zero coefficients, and remains at least slightly superior to the $8 \times 8 \times 96$ dictionary even at the maximum number of non-zero coefficients.

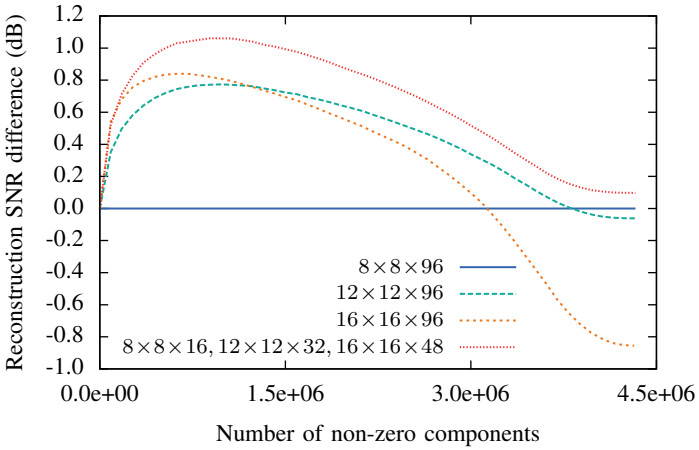


Fig. 9. Reconstruction SNR difference with that of the $8 \times 8 \times 96$ dictionary.

VI. Conclusion

At a high level, the derivation of ADMM algorithms for convolutional sparse coding and the dictionary update for dictionary learning is straightforward, but correct choice of methods for solving the sub-problems is critical to the design of efficient algorithms, with vastly inferior computational performance being possible if these methods are not properly chosen. The results presented here show that the proposed Sherman-Morrison solution of the main linear system is clearly

Dict.	$8 \times 8 \times 72$	$12 \times 12 \times 72$	8×8×24,12×12×48	$16 \times 16 \times 72$	8×8×24,16×16×48
Func.	70.60	68.73	**68.00**	68.71	**67.60**
Dict.	$8 \times 8 \times 96$	$12 \times 12 \times 96$	$16 \times 16 \times 96$	8×8×16,12×12×32,16×16×48	
Func.	66.78	64.71	64.92	**63.87**	

TABLE I
A COMPARISON OF SOLUTION FUNCTIONAL VALUES FOR THE CBPDN PROBLEM COMPUTED WITH $\lambda = 0.001$ FOR THE SAME SET OF IMAGES AND WITH DIFFERENT DICTIONARIES. ALL SPARSE DECOMPOSITIONS WERE COMPUTED ON THE SAME SET OF 16 IMAGES.

the best choice for the CBPDN sparse coding problem, giving much better asymptotic performance, $\mathcal{O}(M)$ instead of $\mathcal{O}(M^3)$, than the previously proposed direct method [21], and is also much faster in practical application. Furthermore, over-relaxation methods are shown to improve performance, and effective heuristic techniques for selection of the penalty parameter ρ are presented.

The resulting ADMM algorithm is compared with alternatives, both via computational comparisons in the literature, and new computational experiments. The leading alternatives are FSS [35] and FISTA [20]. FSS is the fastest algorithm for very large values of regularization parameter λ, but is not at all competitive in the λ value regime necessary for reconstruction problems. The proposed ADMM algorithm is much faster than the previously proposed FISTA approach [20], and faster to varying degrees depending on λ when the gradient required by FISTA is computed in the DFT domain.

In the case of CBPDN dictionary learning, an iterated Sherman-Morrison solution of the main linear system in the dictionary update stage is the best choice for a small number of training images K, but other methods become competitive for larger K. Overall, a CG solution with a moderate accuracy tolerance, or with an adaptive tolerance, appears to be the best choice when K is large. It is also demonstrated that the proposed dictionary learning algorithm can be used to construct convolutional multi-scale dictionaries that do not have any of the structural constraints or complications inherent in previous approaches to constructing multi-scale dictionaries within the standard patch-based framework.

In the interests of reproducible research, software implementations of the main algorithms proposed here are made publicly available [65].

ACKNOWLEDGMENT

The author is grateful to R. Chalasani for kindly providing a copy of his implementation of the method described in [20], and to A. Kucukelbir and J. Theiler for their helpful comments on draft versions of this manuscript.

APPENDIX A
DIAGONAL BLOCK LINEAR SYSTEMS

Given sets of vectors $\mathbf{a}_m \in \mathbb{C}^N$, $\mathbf{b}_m \in \mathbb{C}^N$, and $\mathbf{c}_m \in \mathbb{C}^N$, and unknown vectors $\mathbf{x}_m \in \mathbb{C}^N$, define $A_m = \mathrm{diag}(\mathbf{a}_m)$, $B_m = \mathrm{diag}(\mathbf{b}_m)$, $A = \begin{pmatrix} A_0 & A_1 & \dots & A_{M-1} \end{pmatrix}$, and

$$B = \begin{pmatrix} B_0 & 0 & \dots & 0 \\ 0 & B_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B_{M-1} \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{M-1} \end{pmatrix} \quad \mathbf{c} = \begin{pmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_{M-1} \end{pmatrix} . \tag{42}$$

The A_m are $N \times N$ matrices and A contains M blocks so that A is $N \times MN$. In the following, a^* denotes the complex conjugate of a, and A^H denotes the Hermitian transpose (conjugate transpose) of A. The goal is to solve the linear system $(A^H A + B)\mathbf{x} = \mathbf{c}$, which can be expanded as

$$\begin{pmatrix} A_0^H A_0 + B_0 & A_0^H A_1 & A_0^H A_2 & \dots \\ A_1^H A_0 & A_1^H A_1 + B_1 & A_1^H A_2 & \dots \\ A_2^H A_0 & A_2^H A_1 & A_2^H A_2 + B_2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \end{pmatrix} . \tag{43}$$

Since the A_m are diagonal, so are the blocks $A_m^H A_{m'}$, and it is not difficult to confirm that each row of $A^H A$ has non-zero entries corresponding to a single index in each of the $\mathbf{x}_m$. Denoting entry n of vector $\mathbf{x}_m$ by $\mathbf{x}_m(n)$, the rows of Eq. (43) can be written as a set of equations

$$\sum_m \Big(\mathbf{a}_{m'}(n)^* \mathbf{a}_m(n) + \mathbf{b}_{m'}(n) \Big) \mathbf{x}_m(n) = \mathbf{c}_{m'}(n) \;, \tag{44}$$

which can be simplified by swapping the vector and entry indexing by defining

$$\begin{aligned} \tilde{\mathbf{a}}_n(m) &= \mathbf{a}_m(n)^* & \tilde{\mathbf{b}}_n(m) &= \mathbf{b}_m(n) \\ \tilde{\mathbf{x}}_n(m) &= \mathbf{x}_m(n) & \tilde{\mathbf{c}}_n(m) &= \mathbf{c}_m(n) \;, \end{aligned} \tag{45}$$

and re-ordering the equations to give

$$\sum_m \Big(\tilde{\mathbf{a}}_n(m') \tilde{\mathbf{a}}_n(m)^* + \tilde{\mathbf{b}}_n(m') \Big) \tilde{\mathbf{x}}_n(m) = \tilde{\mathbf{c}}_n(m') \;, \tag{46}$$

which can now be written using vector products as

$$\Big(\tilde{\mathbf{a}}_n \tilde{\mathbf{a}}_n^H + \mathrm{diag}(\tilde{\mathbf{b}}_n) \Big) \tilde{\mathbf{x}}_n = \tilde{\mathbf{c}}_n \;. \tag{47}$$

The $MN \times MN$ system $(A^H A + B)\mathbf{x} = \mathbf{c}$ has been replaced by N independent linear systems of size $M \times M$, each of which consists of a rank one component plus a diagonal component. Systems of this form can be efficiently solved by application of the Sherman-Morrison formula, as shown in Appendix B.

APPENDIX B
SHERMAN-MORRISON SOLUTION

Consider the solution of linear systems of the form

$$\left(J + \mathbf{a}\mathbf{a}^H \right) \mathbf{x} = \mathbf{b} \tag{48}$$

where J is a diagonal matrix. Applying the Sherman-Morrison formula [66]

$$(A + \mathbf{u}\mathbf{v}^H)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^H A^{-1}}{1 + \mathbf{u}^H A^{-1} \mathbf{v}} \tag{49}$$

to derive $\left(J + \mathbf{a}\mathbf{a}^H \right)^{-1}$, and re-arranging to avoid matrix-vector products and vector outer products (noting that

$\mathbf{a}^H J^{-1}\mathbf{b}$ is a scalar so that $\mathbf{a}\mathbf{a}^H J^{-1}\mathbf{b} = (\mathbf{a}^H J^{-1}\mathbf{b})\mathbf{a}$), gives

$$\mathbf{x} = J^{-1}\left(\mathbf{b} - \frac{\mathbf{a}^H J^{-1}\mathbf{b}}{1 + \mathbf{a}^H J^{-1}\mathbf{a}}\mathbf{a}\right) . \tag{50}$$

If $J = \rho I$, i.e. a scaled identity matrix, then

$$\mathbf{x} = \rho^{-1}\left(\mathbf{b} - \frac{\mathbf{a}^H \mathbf{b}}{\rho + \mathbf{a}^H \mathbf{a}}\mathbf{a}\right) . \tag{51}$$

Appendix C
Multiple Diagonal Block Linear Systems

Consider a generalization of the situation in Appendix A such that vectors $\mathbf{a}_{k,m} \in \mathbb{C}^N$ take an additional index, with $A_{k,m} = \mathrm{diag}(\mathbf{a}_{k,m})$ and $A_k = \begin{pmatrix} A_{k,0} & A_{k,1} & \dots & A_{k,M-1} \end{pmatrix}$, with other variables as before, and the goal now being to solve the linear system $(\sum_k A_k^H A_k + B)\mathbf{x} = \mathbf{c}$, which can be expanded as

$$\begin{pmatrix} A_{0,0}^H A_{0,0} + A_{1,0}^H A_{1,0} + \ldots + B_0 & A_{0,0}^H A_{0,1} + A_{1,0}^H A_{1,1} + \ldots & \ldots \\ A_{0,1}^H A_{0,0} + A_{1,1}^H A_{1,0} + \ldots & A_{0,1}^H A_{0,1} + A_{1,1}^H A_{1,1} + \ldots + B_1 & \ldots \\ A_{0,2}^H A_{0,0} + A_{1,2}^H A_{1,0} + \ldots & A_{0,2}^H A_{0,1} + A_{1,2}^H A_{1,1} + \ldots & \ldots \\ \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \end{pmatrix} . \tag{52}$$

By the same re-indexing and reordering as in Appendix A, but now with $\tilde{\mathbf{a}}_{k,n}(m) = \mathbf{a}_{k,m}(n)^*$, this system can be represented by the equations

$$\sum_m \Big(\sum_k \tilde{\mathbf{a}}_{k,n}(m')\tilde{\mathbf{a}}_{k,n}(m)^* + \tilde{\mathbf{b}}_n(m') \Big) \tilde{\mathbf{x}}_n(m) = \tilde{\mathbf{c}}_n(m') , \tag{53}$$

which can be written using vector products as

$$\Big(\sum_k \tilde{\mathbf{a}}_{k,n}\tilde{\mathbf{a}}_{k,n}^H + \mathrm{diag}(\tilde{\mathbf{b}}_n) \Big) \tilde{\mathbf{x}}_n = \tilde{\mathbf{c}}_n . \tag{54}$$

Systems of this form can be efficiently solved by iterated application of the Sherman-Morrison formula, as shown in Appendix D.

Appendix D
Iterated Sherman-Morrison Solution

Consider the solution of problems of the form

$$\left(J + \mathbf{a}_0\mathbf{a}_0^H + \mathbf{a}_1\mathbf{a}_1^H + \ldots + \mathbf{a}_{K-1}\mathbf{a}_{K-1}^H\right)\mathbf{x} = \mathbf{b} . \tag{55}$$

Define $A_0 = J$ and $A_{k+1} = A_k + \mathbf{a}_k\mathbf{a}_k^H$. From the Sherman-Morrison formula we have

$$A_{k+1}^{-1} = \left(A_k + \mathbf{a}_k\mathbf{a}_k^H\right)^{-1} = A_k^{-1} - \frac{A_k^{-1}\mathbf{a}_k\mathbf{a}_k^H A_k^{-1}}{1 + \mathbf{a}_k^H A_k^{-1}\mathbf{a}_k} . \tag{56}$$

Now define $\boldsymbol{\alpha}_{l,k} = A_l^{-1}\mathbf{a}_k$ and $\boldsymbol{\beta}_k = A_k^{-1}\mathbf{b}$ so that $\boldsymbol{\alpha}_{0,k} = J^{-1}\mathbf{a}_k$ and $\boldsymbol{\beta}_0 = J^{-1}\mathbf{b}$, so that

$$\boldsymbol{\beta}_{k+1} = A_{k+1}^{-1}\mathbf{b} \tag{57}$$

$$= A_k^{-1}\mathbf{b} - \frac{A_k^{-1}\mathbf{a}_k\mathbf{a}_k^H A_k^{-1}\mathbf{b}}{1 + \mathbf{a}_k^H A_k^{-1}\mathbf{a}_k} \tag{58}$$

$$= \boldsymbol{\beta}_k - \frac{\boldsymbol{\alpha}_{k,k}\mathbf{a}_k^H\boldsymbol{\beta}_k}{1 + \mathbf{a}_k^H\boldsymbol{\alpha}_{k,k}} , \tag{59}$$

and

$$\boldsymbol{\alpha}_{l+1,k} = A_{l+1}^{-1}\mathbf{a}_k \tag{60}$$

$$= A_l^{-1}\mathbf{a}_k - \frac{A_l^{-1}\mathbf{a}_l\mathbf{a}_l^H A_l^{-1}\mathbf{a}_k}{1 + \mathbf{a}_l^H A_l^{-1}\mathbf{a}_l} \tag{61}$$

$$= \boldsymbol{\alpha}_{l,k} - \frac{\boldsymbol{\alpha}_{l,l}\mathbf{a}_l^H\boldsymbol{\alpha}_{l,k}}{1 + \mathbf{a}_l^H\boldsymbol{\alpha}_{l,l}} . \tag{62}$$

An iterative algorithm to compute the solution for the system Eq. (55), given by β_K, is easily derived from these equations. (This algorithm is equivalent to one previously proposed by Egidi and Maponi [67].) An algorithm for solving Eq. (55) when $J = \rho I$ is presented in Alg. 1.

Input: vectors $\{\mathbf{a}_k\}$, parameter ρ
Initialize: $\boldsymbol{\alpha} = \rho^{-1}\mathbf{a}_0$, $\boldsymbol{\beta} = \rho^{-1}\mathbf{b}$
for $k \in \{1, \ldots, K\}$ **do**
  $\boldsymbol{\gamma}_{k-1} = \dfrac{\boldsymbol{\alpha}}{1 + \mathbf{a}_{k-1}^H\boldsymbol{\alpha}}$
  $\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{\gamma}_{k-1}\mathbf{a}_{k-1}^H\boldsymbol{\beta}$
  if $k \leq K - 1$ **then**
    $\boldsymbol{\alpha} = \rho^{-1}\mathbf{a}_k$
    for $l \in \{1, \ldots, k\}$ **do**
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} - \boldsymbol{\gamma}_{l-1}\mathbf{a}_{l-1}^H\boldsymbol{\alpha}$
    end
  end
end
Output: linear equation solution $\boldsymbol{\beta}$

Algorithm 1: Iterated Sherman-Morrison

References

[1] 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*, vol. 51, no. 1, pp. 34–81, 2009. doi:10.1137/060657704

[2] 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

[3] 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*, vol. 3, no. 1, pp. 1–122, 2010. doi:10.1561/2200000016

[4] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993. doi:10.1109/78.258082

[5] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, Nov. 1993, pp. 40–44. doi:10.1109/acssc.1993.342465

[6] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proc. IEEE*, vol. 98, no. 6, pp. 1045–1057, Jun. 2010. doi:10.1109/jproc.2010.2040551

[7] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998. doi:10.1137/S1064827596304010

[8] B. Wohlberg, "Efficient convolutional sparse coding," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, May 2014, pp. 7173–7177. doi:10.1109/ICASSP.2014.6854992

[9] P.-K. Jao, Y.-H. Yang, and B. Wohlberg, "Informed monaural source separation of music based on convolutional sparse coding," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, May 2015.

[10] A. Cogliati, Z. Duan, and B. Wohlberg, "Piano music transcription with fast convolutional sparse coding," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Sep. 2015.

[11] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. A. LeCun, "Learning convolutional feature hierachies for visual recognition," in *Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 1090–1098.

[12] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Proc. IEEE Int. Conf. Comp. Vis. (ICCV)*, 2011, pp. 2018–2025. doi:10.1109/iccv.2011.6126474

[13] Y. A. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. doi:10.1109/5.726791

[14] 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

[15] J. Yang, K. Yu, and T. S. Huang, "Supervised translation-invariant sparse coding," *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, pp. 3517–3524, 2010. doi:10.1109/cvpr.2010.5539958

[16] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. A. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. IEEE Conf. Comp. Vis. Pat. Recog. (CVPR)*, Jun. 2013, pp. 3626–3633. doi:10.1109/cvpr.2013.465

[17] B. Chen, G. Polatkan, G. Sapiro, D. Blei, D. Dunson, and L. Carin, "Deep learning with hierarchical convolutional factor analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1887–1901, Aug. 2013. doi:10.1109/tpami.2013.19

[18] A. D. Szlam, K. Kavukcuoglu, and Y. A. LeCun, "Convolutional matching pursuit and dictionary training," *Computing Research Repository (arXiv)*, 2010, arXiv:1010.0422.

[19] M. Pachitariu, A. M. Packer, N. Pettit, H. Dalgleish, M. Hausser, and M. Sahani, "Extracting regions of interest from biological images with convolutional sparse block coding," in *Adv. Neural Inf. Process. Syst.*, 2013, vol. 26, pp. 1745–1753.

[20] R. Chalasani, J. C. Principe, and N. Ramakrishnan, "A fast proximal method for convolutional sparse coding," in *Proc. Int. Joint Conf. Neural Net. (IJCNN)*, Aug. 2013.

[21] 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

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

[23] M. S. Lewicki and T. J. Sejnowski, "Coding time-varying signals using sparse, shift-invariant representations," in *Adv. Neural Inf. Process. Syst.*, vol. 11, 1999, pp. 730–736.

[24] W. Hashimoto and K. Kurata, "Properties of basis functions generated by shift invariant sparse representations of natural images," *Biological Cybernetics*, vol. 83, no. 2, pp. 111–118, 2000. doi:10.1007/s004220000149

[25] H. Wersing, J. Eggert, and E. Körner, "Sparse coding with invariance constraints," in *Artificial Neural Networks and Neural Information Processing – ICANN/ICONIP 2003*, ser. Lecture Notes in Computer Science, 2003, vol. 2714, pp. 385–392. doi:10.1007/3-540-44989-2_46

[26] B. A. Olshausen, "Learning sparse, overcomplete representations of time-varying natural images," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, vol. 1, Sep. 2003, pp. I–41–4. doi:10.1109/icip.2003.1246893

[27] T. Blumensath and M. E. Davies, "On shift-invariant sparse coding," in *Independent Component Analysis and Blind Signal Separation*, ser. Lecture Notes in Computer Science, 2004, vol. 3195, pp. 1205–1212. doi:10.1007/978-3-540-30110-3_152

[28] ——, "Shift-invariant sparse coding for single channel blind source separation," in *Signal Processing with Adaptative Sparse Structured Representations (SPARS)*, Nov. 2005.

[29] M. D. Plumbley, S. A. Abdallah, T. Blumensath, and M. E. Davies, "Sparse representations of polyphonic music," *Signal Processing*, vol. 86, no. 3, pp. 417–431, 2006. doi:10.1016/j.sigpro.2005.06.007

[30] T. Blumensath and M. E. Davies, "Sparse and shift-invariant representations of music," *IEEE Trans. Audio, Speech, Language Process.*, vol. 14, no. 1, pp. 50–57, Jan. 2006. doi:10.1109/tsa.2005.860346

[31] P. Jost, P. Vandergheynst, S. Lesage, and R. Gribonval, "MoTIF: An efficient algorithm for learning translation invariant dictionaries," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, vol. 5, May 2006. doi:10.1109/icassp.2006.1661411

[32] K. Skretting, J. H. Husøy, and S. O. Aase, "General design algorithm for sparse frame expansions," *Signal Process.*, vol. 86, no. 1, pp. 117–126, Jan. 2006. doi:10.1016/j.sigpro.2005.04.013

[33] M. Aharon, "Overcomplete dictionaries for sparse representation of signals," Ph.D. dissertation, Department of Computer Science, Israel Institute of Technology, Nov. 2006.

[34] K. Engan, K. Skretting, and J. H. Husøy, "Family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation," *Digital Signal Processing*, vol. 17, no. 1, pp. 32–49, 2007. doi:10.1016/j.dsp.2006.02.002

[35] R. Grosse, R. Raina, H. Kwong, and A. Y. Ng, "Shift-invariant sparse coding for audio classification," in *Proc. Twenty-Third Conf. on Uncertainty in Artificial Intell. (UAI)*, Jul. 2007, pp. 149–158.

[36] B. Mailhé, S. Lesage, R. Gribonval, F. Bimbot, and P. Vandergheynst, "Shift-invariant dictionary learning for sparse representations: extending K-SVD," in *European Signal Processing Conference (EUSIPCO)*, 2008.

[37] M. Mørup, M. N. Schmidt, and L. K. Hansen, "Shift invariant sparse coding of image and music data," Technical University of Denmark, Tech. Rep. IMM2008-04659, 2008.

[38] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Shift-invariant sparse representation of images using learned dictionaries," in *IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Oct. 2008, pp. 145–150. doi:10.1109/mlsp.2008.4685470

[39] M. Nakashizuka, H. Nishiura, and Y. Iiguni, "Sparse image representations with shift-invariant tree-structured dictionaries," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Nov. 2009, pp. 2145–2148. doi:10.1109/icip.2009.5414319

[40] Y.-k. Tomokusa, M. Nakashizuka, and Y. Iiguni, "Sparse image representations with shift and rotation invariance constraints," in *Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS)*, Jan. 2009, pp. 256–259. doi:10.1109/ispacs.2009.5383854

[41] M. Mørup and M. N. Schmidt, "Transformation invariant sparse coding," in *IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, 2011, pp. 1–6. doi:10.1109/mlsp.2011.6064547

[42] Q. Barthélemy, A. Larue, A. Mayoue, D. Mercier, and J. I. Mars, "Shift & 2d rotation invariant sparse coding for multivariate signals," *IEEE Trans. Signal Process.*, vol. 60, no. 4, pp. 1597–1611, Apr. 2012. doi:10.1109/tsp.2012.2183129

[43] G. Pope, C. Aubel, and C. Studer, "Learning phase-invariant dictionaries," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, May 2013, pp. 5979–5983. doi:10.1109/icassp.2013.6638812

[44] C. Rusu, B. Dumitrescu, and S. A. Tsaftaris, "Explicit shift-invariant dictionary learning," *IEEE Signal Process. Lett.*, vol. 21, no. 1, pp. 6–9, Jan. 2014. doi:10.1109/lsp.2013.2288788

[45] R. P. N. Rao and D. H. Ballard, "Development of localized oriented receptive fields by learning a translation-invariant code for natural images," *Network: Computation in Neural Systems*, vol. 9, no. 2, pp. 219–234, 1998. doi:10.1088/0954-898x_9_2_005

[46] C. Ekanadham, D. Tranchina, and E. P. Simoncelli, "Recovery of sparse translation-invariant signals with continuous basis pursuit," *IEEE Trans. Signal Process.*, vol. 59, no. 10, pp. 4735–4744, Oct. 2011. doi:10.1109/tsp.2011.2160058

[47] T. Blumensath and M. E. Davies, "Unsupervised learning of sparse and shift-invariant decompositions of polyphonic music," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, vol. 5, May 2004, pp. 497–500. doi:10.1109/icassp.2004.1327156

[48] B. Kong and C. C. Fowlkes, "Fast convolutional sparse coding (FCSC)," Department of Computer Science, University of California, Irvine, Tech. Rep., May 2014.

[49] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. IEEE Int. Conf. Comp. Vis. (ICCV)*, 2009, pp. 2272–2279. doi:10.1109/iccv.2009.5459452

[50] B.-S. He, H. Yang, and S.-L. Wang, "Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities," *J. Optim. Theor. Appl.*, vol. 106, pp. 337–356, 2000. doi:10.1023/a:1004603514434

[51] R. Grosse, "Shift-Invariant Sparse Coding," Matlab library available from http://www.cs.toronto.edu/~rgrosse/uai07-sisc-code.tar.gz, 2007.

[52] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *IEEE Trans. Image Process.*, vol. 19, no. 9, pp. 2345–2356, 2010. doi:10.1109/tip.2010.2047910

[53] 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

[54] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, "An Augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems," *IEEE Trans. Image Process.*, vol. 20, no. 3, pp. 681–695, Mar. 2011. doi:10.1109/tip.2010.2076294

[55] Q. Liu, J. Luo, S. Wang, M. Xiao, and M. Ye, "An augmented Lagrangian multi-scale dictionary learning algorithm," *EURASIP J. Adv. Signal Process.*, vol. 2011, pp. 1–16, 2011. doi:10.1186/1687-6180-2011-58

[56] M. J. Huiskes and M. S. Lew, "The MIR Flickr retrieval evaluation," in *Proc. ACM Int. Conf. Multimedia Information Retrieval (MIR)*, 2008. doi:10.1145/1460096.1460104

[57] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *Multiscale Model. Simul.*, vol. 7, no. 1, pp. 214–241, 2008. doi:10.1137/070697653

[58] R. Gribonval, "Fast matching pursuit with a multiscale dictionary of Gaussian chirps," *IEEE Trans. Signal Process.*, no. 5, pp. 994–1001, 2001. doi:10.1109/78.917803

[59] B. A. Olshausen, P. Sallee, and M. S. Lewicki, "Learning sparse image codes using a wavelet pyramid architecture," in *Adv. Neural Inf. Process. Syst.*, 2000, vol. 13, pp. 887–893.

[60] P. Sallee and B. A. Olshausen, "Learning sparse multiscale image representations," in *Adv. Neural Inf. Process. Syst.*, 2002, vol. 15, pp. 1351–1358.

[61] B. Ophir, M. Lustig, and M. Elad, "Multi-scale dictionary learning using wavelets," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 5, pp. 1014–1024, Sep. 2011. doi:10.1109/JSTSP.2011.2155032

[62] K. Skretting and K. Engan, "Image compression using learned dictionaries by RLS-DLA and compared with K-SVD," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, May 2011, pp. 1517–1520. doi:10.1109/icassp.2011.5946782

[63] J. M. Hughes, D. N. Rockmore, and Y. Wang, "Bayesian learning of sparse multiscale image representations," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 4972–4983, Dec. 2013. doi:10.1109/tip.2013.2280188

[64] J. Mairal, G. Sapiro, and M. Elad, "Multiscale sparse image representation with learned dictionaries," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, vol. 3, Sep. 2007, pp. 105–108. doi:10.1109/icip.2007.4379257

[65] B. Wohlberg, "SParse Optimization Research COde (SPORCO)," Matlab library available from http://math.lanl.gov/~brendt/Software/SPORCO/, 2015, version 0.0.2.

[66] W. W. Hager, "Updating the inverse of a matrix," *SIAM Review*, vol. 31, no. 2, pp. 221–239, Jun. 1989. doi:10.1137/1031049

[67] N. Egidi and P. Maponi, "A Sherman-Morrison approach to the solution of linear systems," *Journal of Computational and Applied Mathematics*, vol. 189, no. 1-2, pp. 703–718, May 2006. doi:10.1016/j.cam.2005.02.013

Brendt Wohlberg received the BSc(Hons) degree in applied mathematics, and the MSc(Applied Science) and PhD degrees in electrical engineering from the University of Cape Town, South Africa, in 1990, 1993 and 1996 respectively. He is currently a staff scientist in Theoretical Division at Los Alamos National Laboratory, Los Alamos, NM. His primary research interest is in signal and image processing inverse problems, with an emphasis on sparse representations and exemplar-based methods. He was an associate editor for IEEE TRANSACTIONS ON IMAGE PROCESSING from 2010 to 2014, and is currently chair of the Computational Imaging Special Interest Group of the IEEE Signal Processing Society and an associate editor for IEEE TRANSACTIONS ON COMPUTATIONAL IMAGING.

Supplementary Material for "Efficient Algorithms for Convolutional Sparse Representations"

Brendt Wohlberg

I. Introduction

This document provides additional detail and results that were omitted from the main document due to space restrictions. Sections are named corresponding to relevant section titles in the main document.

II. ADMM Algorithm for Convolutional BPDN

A. Algorithm Summary

The proposed ADMM algorithm for Convolutional BPDN is summarised in Alg. 1. Typical values for the penalty auto-update parameters are $J_{\mathrm{p}} = 1$, $\mu = 10$, and $\tau = 2$.

B. Performance Comparison: Linear Solvers for ADMM

The execution times and solution accuracies of four different methods of solving the linear system in single precision are compared in Fig. 1 in the main document. A corresponding comparison for double-precision solution of these systems is presented here in Fig. 1. It is clear from a comparison of these two figures that the variation in execution times between single and double precision is far smaller than the variation between the different solution methods. SM run time has very small variation with precision, while CG can be almost twice as slow in double precision as in single, and GE run time dependence on precision increases with M.

In the main document, Fig. 2 presents a comparison of functional value evolution with time for the proposed sparse coding ADMM algorithm with different solution methods for the main linear system to be solved. The corresponding primal and dual residual evolution is provided here in Figs. 2 and 3. As in the case of functional value evolution, the Sherman-Morrison solution gives by far the best performance.

C. ADMM Parameter Selection

The heuristic choice of $\rho_0 = 100\lambda + 0.5$ proposed in Sec. III-D of the main document is motivated in Figs. 4–6, this choice representing a compromise between selecting for good performance at 50, 100, and 500 iterations. The relative functional values, which are required to allow comparisons across different values of λ, are computed by dividing functional values for each λ by the smallest functional value attained for that λ at the specified number of iterations. General experimental parameters and test data are the same as for Figs. 3 and 4 in the main document.

The author is with Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA (e-mail: brendt@lanl.gov).

Input: image $\mathbf{s}$ (N pixels), dictionary $\mathbf{d}_m$ (M filters), regularization parameter λ, initial penalty parameter ρ_0, penalty auto-update parameters J_{p}, μ, τ, relaxation parameter α, maximum iterations $J_{\max}$, absolute and relative stopping tolerances ϵ_{abs}, ϵ_{rel}

Precompute: $\hat{\mathbf{s}} = \mathrm{FFT}(\mathbf{s})$, $\hat{D}_m = \mathrm{FFT}(\mathbf{d}_m) \;\; \forall m$

Initialize: $\mathbf{y}_m = \mathbf{y}_m^{\mathrm{prev}} = \mathbf{u}_m = 0 \;\; \forall m$, $\rho = \rho_0$, $j = 1$

repeat

- $\hat{\mathbf{z}}_m = \mathrm{FFT}(\mathbf{y}_m - \mathbf{u}_m) \;\; \forall m$
- Compute $\hat{\mathbf{x}}_m \; \forall m$ as in Eq. (16)–(19) and Sec. III-B of main document
- $\mathbf{x}_m = \mathrm{IFFT}(\hat{\mathbf{x}}_m)$
- $\mathbf{x}_{\mathrm{relax},m} = \alpha \mathbf{x}_m + (1 - \alpha)\mathbf{y}_m \;\; \forall m$
- $\mathbf{y}_m = \mathcal{S}_{\lambda/\rho}\left(\mathbf{x}_{\mathrm{relax},m} + \mathbf{u}_m\right) \;\; \forall m$
- $\mathbf{u}_m = \mathbf{u}_m + \mathbf{x}_{\mathrm{relax},m} - \mathbf{y}_m \;\; \forall m$
- $r = \|\mathbf{x} - \mathbf{y}\|_2$
- $s = \rho \left\|\mathbf{y}^{\mathrm{prev}} - \mathbf{y}\right\|_2$
- $\epsilon_{\mathrm{pri}} = \epsilon_{\mathrm{abs}} \sqrt{MN} + \epsilon_{\mathrm{rel}} \max\{\|\mathbf{x}\|_2, \|\mathbf{y}\|_2\}$
- $\epsilon_{\mathrm{dua}} = \epsilon_{\mathrm{abs}} \sqrt{MN} + \epsilon_{\mathrm{rel}} \rho \|\mathbf{u}\|_2$
- $\mathbf{y}_m^{\mathrm{prev}} = \mathbf{y}_m \;\; \forall m$
- **if** $j \neq 1$ *and* $j \mod J_{\mathrm{p}} = 0$ **then**
  - **if** $r > \mu s$ **then**
    - $\rho = \tau \rho$
    - $\mathbf{u}_m = \mathbf{u}_m / \tau \;\; \forall m$
  - **else if** $s > \mu r$ **then**
    - $\rho = \rho / \tau$
    - $\mathbf{u}_m = \tau \mathbf{u}_m \;\; \forall m$
  - **end**
- **end**
- $j = j + 1$

until $j > J_{\max}$ *or* ($r \leq \epsilon_{\mathrm{pri}}$ *and* $s \leq \epsilon_{\mathrm{dua}}$)

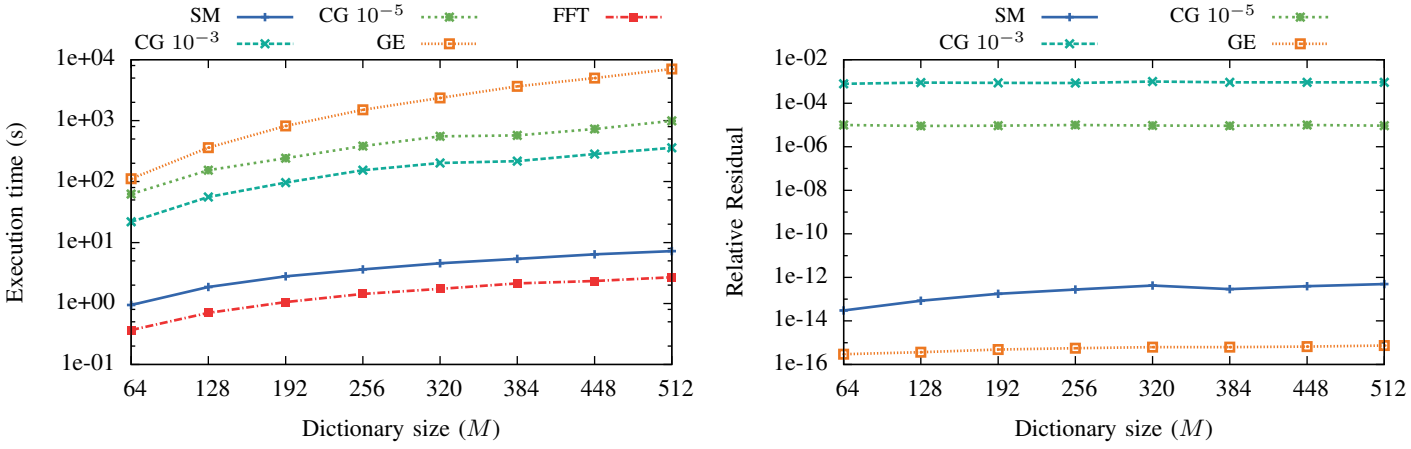
Output: Coefficient maps $\mathbf{y}_m$

Algorithm 1: Summary of proposed ADMM algorithm for Convolutional BPDN. A subscript indexed variable written without the subscript denotes the entire set of vectors concatenated as a single vector, e.g. $\mathbf{x}$ denotes the vector constructed by concatenating all vectors $\mathbf{x}_m$.

III. CBPDN Algorithm Comparison

A. Feature-Sign Search

In the main document, Fig. 5 compares the performance of the feature-sign search algorithm and ADMM for varying λ. A corresponding comparison for varying image sizes is presented in Fig. 7 here.



(a) Run time

(b) Solution error

Fig. 1. A comparison of execution times and solution errors for double-precision solution of a linear system for the 512×512 greyscale Lena image with dictionaries consisting of 12×12 filters, with $\lambda = 0.01$. Solution methods are Gaussian elimination (GE), Conjugate Gradient (CG) with relative residual tolerances of 10^{-3} and 10^{-5}, and Sherman-Morrison (SM). The combined run time for a forward and inverse FFT is provided as a reference.

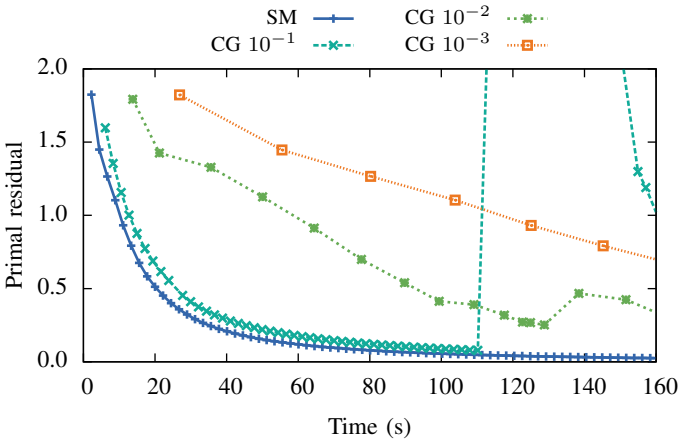


Fig. 2. A comparison of primal residual evolution with time for the ADMM algorithm with Eq. (19) solved in single precision using Sherman-Morrison and CG with three different relative residual tolerances. The primal residual is $r = \|\mathbf{x} - \mathbf{y}\|_2$. The test image was the 512×512 greyscale Lena image with a learned $8 \times 8 \times 64$ dictionary and $\lambda = 0.01$. The corresponding functional value evolution is displayed in Fig. 2 in the main document.

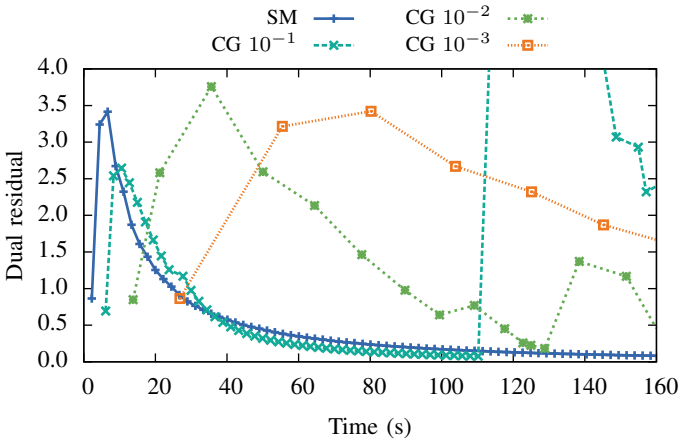


Fig. 3. A comparison of dual residual evolution with time for the ADMM algorithm with Eq. (19) solved in single precision using Sherman-Morrison and CG with three different relative residual tolerances. The dual residual is $s = \rho \|\mathbf{y}^{\text{prev}} - \mathbf{y}\|_2$ where $\mathbf{y}^{\text{prev}}$ and $\mathbf{y}$ are from the previous and current iterations respectively. The test image was the 512×512 greyscale Lena image with a learned $8 \times 8 \times 64$ dictionary and $\lambda = 0.01$. The corresponding functional value evolution is displayed in Fig. 2 in the main document.

B. FISTA

It is clear from the summary of the literature and the experiments reported in Sec. IV in the main document that FISTA with frequency domain computation of the gradient is the most viable alternative to ADMM when not operating in the narrow range of problem sizes and representation sparsity levels in which the FSS algorithm is competitive. The experiments reported in Fig. 6 in that section compare the performance of ADMM and FISTA in sparse coding of a test image that has not been subjected to any additional degradation, so that it could be argued that those results may not be representative of the relative performance that would be observed when sparse coding in the context of an image reconstruction problem, where the image to be sparse coded or the dictionary could have substantially different properties. The results reported in Fig. 8 here show that a similar advantage to ADMM persists in two distinct reconstruction problems. As in Sec. IV in the main document, FISTA parameters were selected by a grid search over a range of values around the standard values $L_0 = 1$ and $\eta = 5$. The same $16 \times 16 \times 128$ dictionary was used for all of these experiments.

The first of these problems is denoising of Gaussian white noise. Four different test cases were constructed by adding noise of progressively high variance to the same test image, giving noisy images with SNRs with respect to the original image of 25.5dB, 19.4dB, 13.4dB, and 7.4dB. The corresponding λ values of $2.63 \times 10^{-2}, 6.05 \times 10^{-2}, 1.46 \times 10^{-1}$, and 6.05×10^{-2} respectively were selected according to the

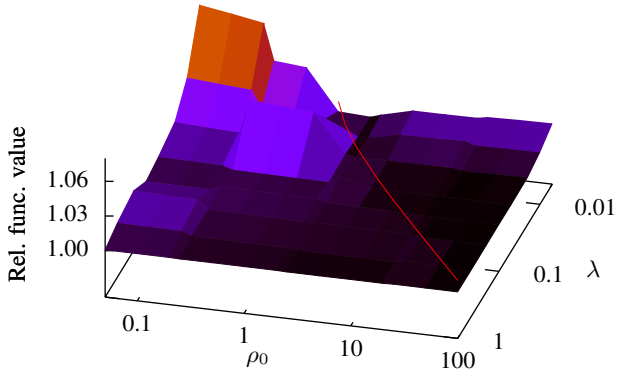


Fig. 4. Relative functional values at 50 iterations for different choices of λ and ρ_0, with an adaptive ρ update strategy with period 1, and with over-relaxation with $\alpha = 1.8$. The curve $\rho_0 = 100\lambda + 0.5$ is overlaid in red.

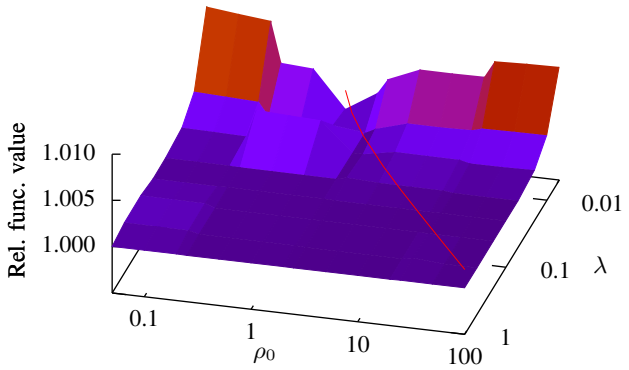


Fig. 5. Relative functional values at 100 iterations for different choices of λ and ρ_0, with an adaptive ρ update strategy with period 1, and with over-relaxation with $\alpha = 1.8$. The curve $\rho_0 = 100\lambda + 0.5$ is overlaid in red.

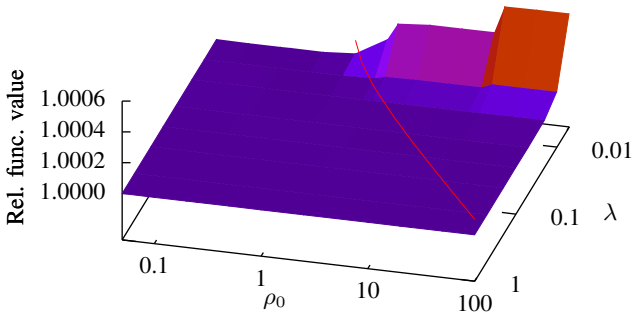


Fig. 6. Relative functional values at 500 iterations for different choices of λ and ρ_0, with an adaptive ρ update strategy with period 1, and with over-relaxation with $\alpha = 1.8$. The curve $\rho_0 = 100\lambda + 0.5$ is overlaid in red.

well-known discrepancy principle, which has the advantage of simplicity, and is more than adequate for this demonstration. As usual (see Sec. I in the main document), a highpass filtering preprocessing step was applied before sparse coding. The FISTA parameter search found different parameters for each of these problems that provided very slightly faster convergence than the standard values $L_0 = 1$ and $\eta = 5$. The relative performance presented in Fig. 8(a) mirrors that of Fig. 6(a) in

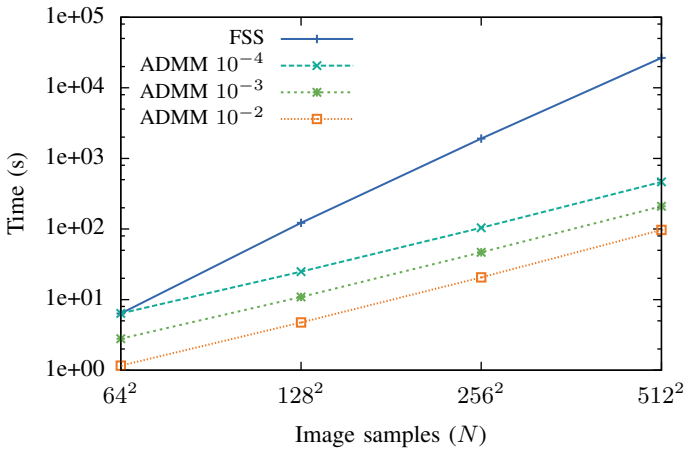


Fig. 7. A comparison of computation times for the feature-sign search (FSS) algorithm and the ADMM algorithm with three different relative error tolerances. The sparse coding was applied with $\lambda = 0.22$ to the greyscale Lena image downsampled to the different sizes, and the dictionary was $8 \times 8 \times 64$.

the main document, with FISTA providing competitive (albeit somewhat slower) convergence than ADMM for larger values of λ, and exhibiting an increasing lag in convergence rate with decreasing λ.

The second problem type is a form of image inpainting in which a randomly distributed set of pixels, of known location, have values set to zero. This problem can be addressed using CBPDN with a dictionary that has an additional impulse filter added to represent that unknown values, and with a weighted ℓ^1 norm where the weighting is derived from a mask indicating the locations of pixels of unknown values. In this problem the highpass preprocessing step was performed using Total Variation denoising with a weighted data fidelity term since the usual highpass filtering is problematic in the presence of unknown pixels. Four different test cases were constructed with 20%, 40%, 60%, and 80% of the pixels set to zero. A fixed λ value of 1.0×10^{-2} was used for all four cases since it was found empirically to provide good results, and so that the performance comparison would only exhibit differences resulting from differences between the problems. The FISTA parameter search revealed that choosing $L_0 = 5$ and $\eta = 10$ gave a relatively small but significant improvement in the convergence rate for all four of the test cases. The relative performance presented in Fig. 8(b) shows that FISTA is not at all competitive with ADMM in this type of problem, with much slower convergence and a performance gap that grows with the difficulty of the problem (i.e. the fraction of unknown pixels).

IV. Dictionary Learning

A. Algorithm Summary

The proposed algorithm for Convolutional BPDN dictionary learning is summarised in Alg. 2. Typical values for the penalty auto-update parameters are $J_{\mathrm{x,p}} = J_{\mathrm{d,p}} = 10$, $\mu_{\mathrm{x}} = \mu_{\mathrm{d}} = 10$, and $\tau_{\mathrm{x}} = \tau_{\mathrm{d}} = 2$.

Input: images $\mathbf{s}_k$ (K images of N pixels each), initial dictionary $\mathbf{d}_m^0$ (M filters), regularization parameter λ, initial penalty parameters ρ_0, σ_0, penalty auto-update parameters $J_{\mathrm{x,p}}$, μ_{x}, τ_{x}, $J_{\mathrm{d,p}}$, μ_{d}, τ_{d}, relaxation parameters α_{x}, α_{d}, maximum iterations $J_{\max}$, absolute and relative stopping tolerances ϵ_{abs}, ϵ_{rel}

Precompute: $\hat{\mathbf{s}}_k = \mathrm{FFT}(\mathbf{s}_k) \;\; \forall k$

Initialize: $\mathbf{y}_{k,m} = \mathbf{y}_{k,m}^{\mathrm{prev}} = \mathbf{u}_{k,m} = 0 \;\; \forall k, m \;\; \mathbf{h}_m = 0 \;\; \mathbf{g}_m = \mathbf{g}_m^{\mathrm{prev}} = \mathbf{d}_m^0 \;\; \forall m$, $\rho = \rho_0$, $\sigma = \sigma_0$, $j = 1$

repeat

$\hat{\mathbf{g}}_m = \mathrm{FFT}(\mathbf{g}_m) \;\; \forall m$

$\hat{\mathbf{z}}_{k,m} = \mathrm{FFT}(\mathbf{y}_{k,m} - \mathbf{u}_{k,m}) \;\; \forall k, m$

Compute $\hat{\mathbf{x}}_{k,m} \; \forall k, m$ as in Eq. (16)–(19) and Sec. III-B of main document, using $\hat{\mathbf{g}}_{k,m}$ as the dictionary

$\mathbf{x}_{k,m} = \mathrm{IFFT}(\hat{\mathbf{x}}_{k,m})$

$\mathbf{x}_{\mathrm{relax},k,m} = \alpha_{\mathrm{x}} \mathbf{x}_{k,m} + (1 - \alpha_{\mathrm{x}}) \mathbf{y}_{k,m} \;\; \forall k, m$

$\mathbf{y}_{k,m} = \mathcal{S}_{\lambda/\rho} \left(\mathbf{x}_{\mathrm{relax},k,m} + \mathbf{u}_{k,m} \right) \;\; \forall k, m$

$\mathbf{u}_{k,m} = \mathbf{u}_{k,m} + \mathbf{x}_{\mathrm{relax},k,m} - \mathbf{y}_{k,m} \;\; \forall k, m$

$\hat{\mathbf{y}}_{k,m} = \mathrm{FFT}(\mathbf{y}_{k,m}) \;\; \forall k, m$

$\hat{\mathbf{z}}_m = \mathrm{FFT}(\mathbf{g}_m - \mathbf{h}_m) \;\; \forall m$

Compute $\hat{\mathbf{d}}_m \; \forall m$ as in Eq. (38)–(41) and Sec. V-A of main document, using $\hat{\mathbf{y}}_{k,m}$ as the coefficient maps

$\mathbf{d}_m = \mathrm{IFFT}(\hat{\mathbf{d}}_m)$

$\mathbf{d}_{\mathrm{relax},m} = \alpha_{\mathrm{d}} \mathbf{d}_m + (1 - \alpha_{\mathrm{d}}) \mathbf{g}_m \;\; \forall m$

$\mathbf{g}_m = \mathrm{prox}_{\iota_{C_{\mathrm{PN}}}} (\mathbf{d}_{\mathrm{relax},m} + \mathbf{h}_m) \;\; \forall m$

$\mathbf{h}_m = \mathbf{h}_m + \mathbf{d}_{\mathrm{relax},m} - \mathbf{x}_m \;\; \forall m$

$r_{\mathrm{x}} = \left\| \mathbf{x} - \mathbf{y} \right\|_2$

$s_{\mathrm{x}} = \rho \left\| \mathbf{y}^{\mathrm{prev}} - \mathbf{y} \right\|_2$

$r_{\mathrm{d}} = \left\| \mathbf{d} - \mathbf{g} \right\|_2$

$s_{\mathrm{d}} = \sigma \left\| \mathbf{g}^{\mathrm{prev}} - \mathbf{g} \right\|_2$

$\epsilon_{\mathrm{x,pri}} = \epsilon_{\mathrm{abs}} \sqrt{KMN} + \epsilon_{\mathrm{rel}} \max\{ \left\| \mathbf{x} \right\|_2, \left\| \mathbf{y} \right\|_2 \}$

$\epsilon_{\mathrm{x,dua}} = \epsilon_{\mathrm{abs}} \sqrt{KMN} + \epsilon_{\mathrm{rel}} \rho \left\| \mathbf{u} \right\|_2$

$\epsilon_{\mathrm{d,pri}} = \epsilon_{\mathrm{abs}} \sqrt{MN} + \epsilon_{\mathrm{rel}} \max\{ \left\| \mathbf{d} \right\|_2, \left\| \mathbf{g} \right\|_2 \}$

$\epsilon_{\mathrm{d,dua}} = \epsilon_{\mathrm{abs}} \sqrt{MN} + \epsilon_{\mathrm{rel}} \sigma \left\| \mathbf{h} \right\|_2$

$\mathbf{y}_m^{\mathrm{prev}} = \mathbf{y}_m \;\; \forall m$

$\mathbf{g}_m^{\mathrm{prev}} = \mathbf{g}_m \;\; \forall m$

if $j \neq 1$ *and* $j \mod J_{\mathrm{x,p}} = 0$ **then**

  if $r_{\mathrm{x}} > \mu_{\mathrm{x}} s_{\mathrm{x}}$ **then**

    $\rho = \tau_{\mathrm{x}} \rho$

    $\mathbf{u}_m = \mathbf{u}_m / \tau_{\mathrm{x}} \;\; \forall m$

  else if $s_{\mathrm{x}} > \mu_{\mathrm{x}} r_{\mathrm{x}}$ **then**

    $\rho = \rho / \tau_{\mathrm{x}}$

    $\mathbf{u}_m = \tau_{\mathrm{x}} \mathbf{u}_m \;\; \forall m$

  end

end

if $j \neq 1$ *and* $j \mod J_{\mathrm{d,p}} = 0$ **then**

  if $r_{\mathrm{d}} > \mu_{\mathrm{d}} s_{\mathrm{d}}$ **then**

    $\sigma = \tau_{\mathrm{d}} \sigma$

    $\mathbf{h}_m = \mathbf{h}_m / \tau_{\mathrm{d}} \;\; \forall m$

  else if $s_{\mathrm{d}} > \mu_{\mathrm{d}} r_{\mathrm{d}}$ **then**

    $\sigma = \sigma / \tau_{\mathrm{d}}$

    $\mathbf{h}_m = \tau_{\mathrm{d}} \mathbf{h}_m \;\; \forall m$

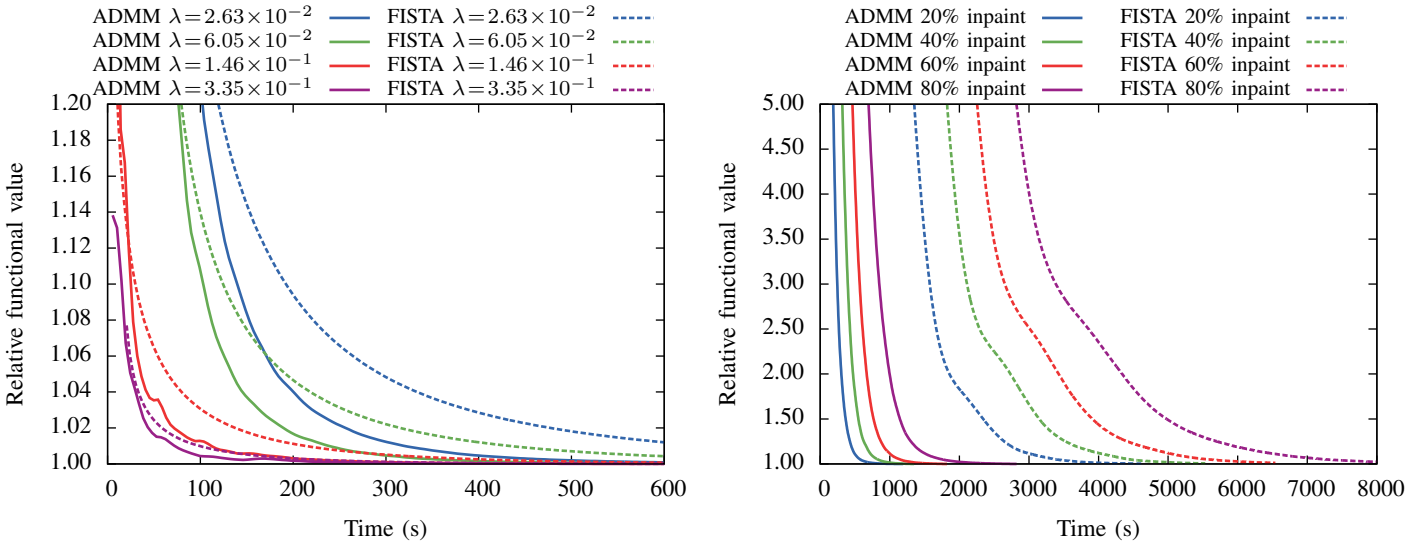
  end

end

until $j > J_{\max}$ *or* ($r_{\mathrm{x}} \leq \epsilon_{\mathrm{x,pri}}$ *and* $s_{\mathrm{x}} \leq \epsilon_{\mathrm{x,dua}}$ *and* $r_{\mathrm{d}} \leq \epsilon_{\mathrm{d,pri}}$ *and* $s_{\mathrm{d}} \leq \epsilon_{\mathrm{d,dua}}$)

Output: Dictionary $\{\mathbf{g}_m\}$, coefficient maps $\{\mathbf{y}_m\}$

Algorithm 2: Summary of proposed ADMM algorithm for Convolutional BPDN dictionary learning. A subscript indexed variable written without the subscript denotes the entire set of vectors concatenated as a single vector, e.g. $\mathbf{x}$ denotes the vector constructed by concatenating all vectors $\mathbf{x}_{k,m}$.



(a) Denoising

(b) Random corruption inpainting

Fig. 8. Comparison of time evolution of relative functional values for ADMM, and FISTA with the gradient computed in the frequency domain. (a) provides a comparison for four Gaussian white noise denoising problems with varying noise variance and correspondingly selected λ values, and (b) provides a comparison for four inpainting-type problems with different fractions of randomly distributed corrupted pixels and fixed $\lambda = 1.0\times10^{-2}$.

The effects of different algorithm parameters are explored in the experiments reported in Figs. 9–11.

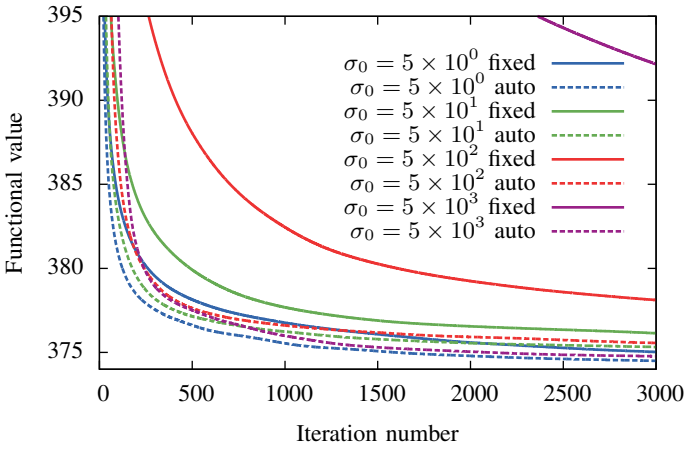


Fig. 9. Dictionary learning functional value evolution for different choices of σ_0, with and without the automatic parameter adaptation scheme. The dictionary was $8\times8\times64$ with Gaussian random initialisation, the training data was a set of five 256×256 images from the MIRFlickr dataset, and $\lambda = 0.1$, $\rho_0 = 10.5$. Penalty auto-update parameters (for the curves for which the automatic adaptation was active) were $J_{\mathrm{x,p}} = J_{\mathrm{d,p}} = 10$, $\mu_{\mathrm{x}} = \mu_{\mathrm{d}} = 10$, and $\tau_{\mathrm{x}} = \tau_{\mathrm{d}} = 2$, and relaxation parameters were $\alpha_{\mathrm{x}} = \alpha_{\mathrm{d}} = 1.8$. Note that the apparent relative differences between these curves is exaggerated by the scaling of the vertical axis; if the vertical axis were expanded to include the entire range of functional values, most of these differences would be observed to be relatively small, with no perceptible difference between any of the curves beyond 2000 iterations except for "$\sigma = 5\times10^2$ fixed" and "$\sigma = 5\times10^3$ fixed".

In Fig. 9, $\sigma_0 = 5$ with automatic adaptation gives the best overall performance. Performance varies widely for different values of σ_0 when σ is fixed, but the effect of σ_0 is small when automatic adaptation is enabled. Note that the effect of parameter σ scales linearly with the number of training images K so that $\sigma_0 \approx K$ is a reasonable choice when applying the same parameters to a training set of a different size than the $K = 5$ in this experiment.

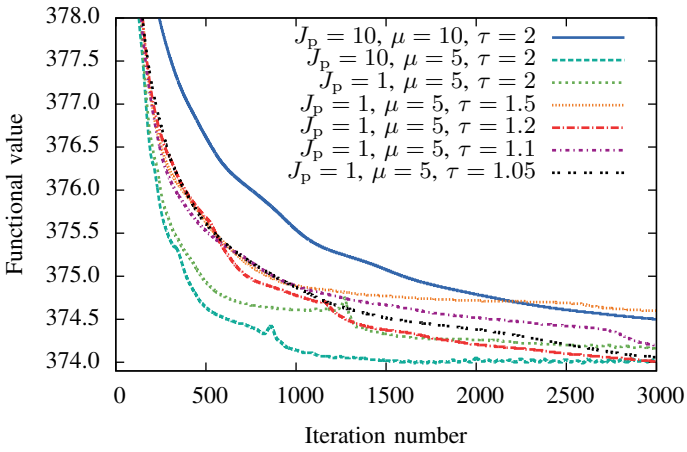


Fig. 10. Dictionary learning functional value evolution for different penalty auto-update parameters (x and d subscripts are omitted since values are the same for both updates). The dictionary was $8\times8\times64$ with Gaussian random initialisation, the training data was a set of five 256×256 images from the MIRFlickr dataset, and $\lambda = 0.1$, $\rho_0 = 10.5$, $\sigma_0 = 5$. Relaxation parameters were $\alpha_{\mathrm{x}} = \alpha_{\mathrm{d}} = 1.8$. Note that the apparent relative differences between these curves is exaggerated by the scaling of the vertical axis; if the vertical axis were expanded to include the entire range of functional values, these differences would be almost imperceptible.

In Fig. 10, the best performance is obtained for $J_{\mathrm{x,p}} = J_{\mathrm{d,p}} = 10$, $\mu_{\mathrm{x}} = \mu_{\mathrm{d}} = 5$, and $\tau_{\mathrm{x}} = \tau_{\mathrm{d}} = 2$, but the relative performance difference between the different choices in this graph is very small, with the specific choice of automatic parameter adaptation parameters making a smaller difference than the choice of whether to use fixed or automatically adapted parameters (see Fig. 9).

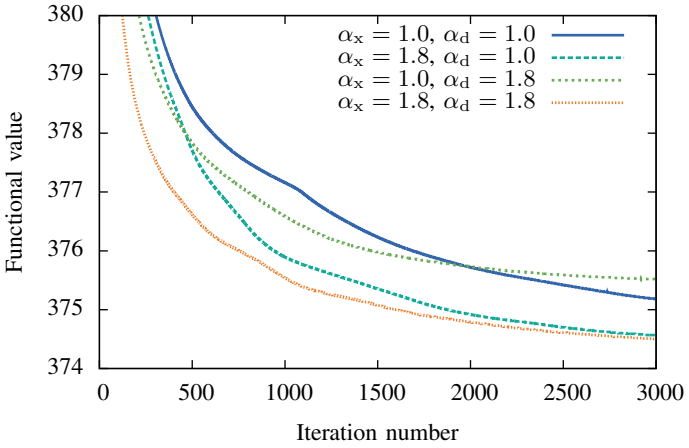


Fig. 11. Dictionary learning functional value evolution for different relaxation parameters. The dictionary was $8\times8\times64$ with Gaussian random initialisation, the training data was a set of five 256×256 images from the MIRFlickr dataset, and $\lambda = 0.1$, $\rho_0 = 10$, $\sigma_0 = 5$. Penalty auto-update parameters were $J_{\mathrm{x,p}} = J_{\mathrm{d,p}} = 10$, $\mu_{\mathrm{x}} = \mu_{\mathrm{d}} = 10$, and $\tau_{\mathrm{x}} = \tau_{\mathrm{d}} = 2$. Note that the apparent relative differences between these curves is exaggerated by the scaling of the vertical axis; if the vertical axis were expanded to include the entire range of functional values, these differences would be observed to be relatively small beyond 200 iterations, and barely perceptible beyond 2000.

Fig. 11 shows that there is a small but clear advantage to the use of over-relaxation in both $\mathbf{x}$ and $\mathbf{d}$ updates in the proposed dictionary learning algorithm.

B. Performance Comparison: Linear Solvers for ADMM

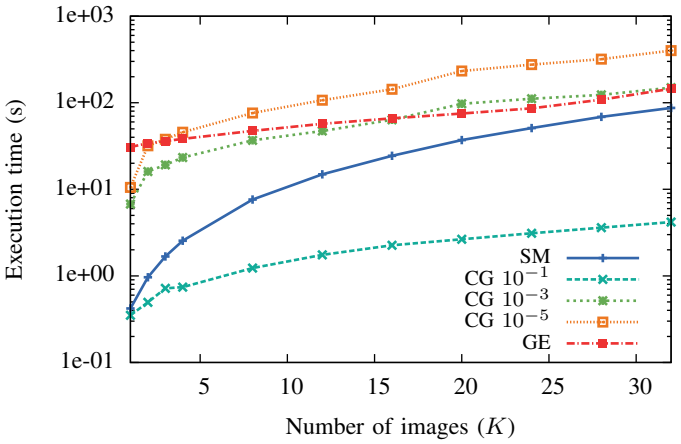


Fig. 12. A comparison of execution times for double-precision solution of a linear system corresponding to an $8 \times 8 \times 64$ dictionary with $\lambda = 0.1$. Solution methods are Gaussian elimination (GE), Conjugate Gradient (CG), and Sherman-Morrison (SM). The corresponding single-precision comparison is displayed in Fig. 7 in the main document.

A comparison of single-precision solution methods for the main linear system in the ADMM dictionary update algorithm is presented in Fig. 7 in the main document. The corresponding double-precision comparison is presented here in Figs. 12 and 13.

The execution times and solution accuracies of three different methods of solving the main linear system in the dictionary update are compared for fixed $K = 32$ and varying M in Fig. 14.

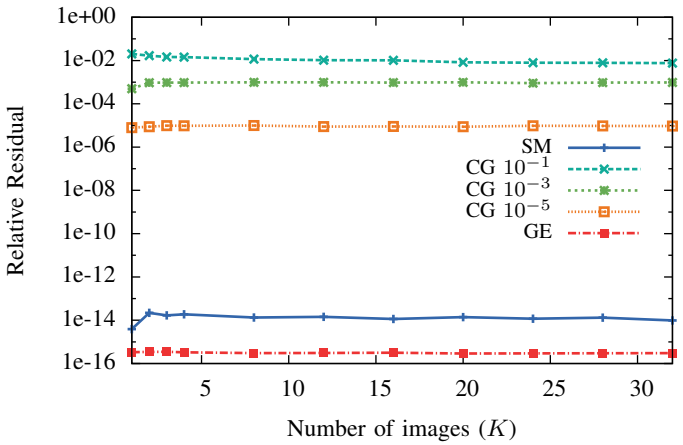
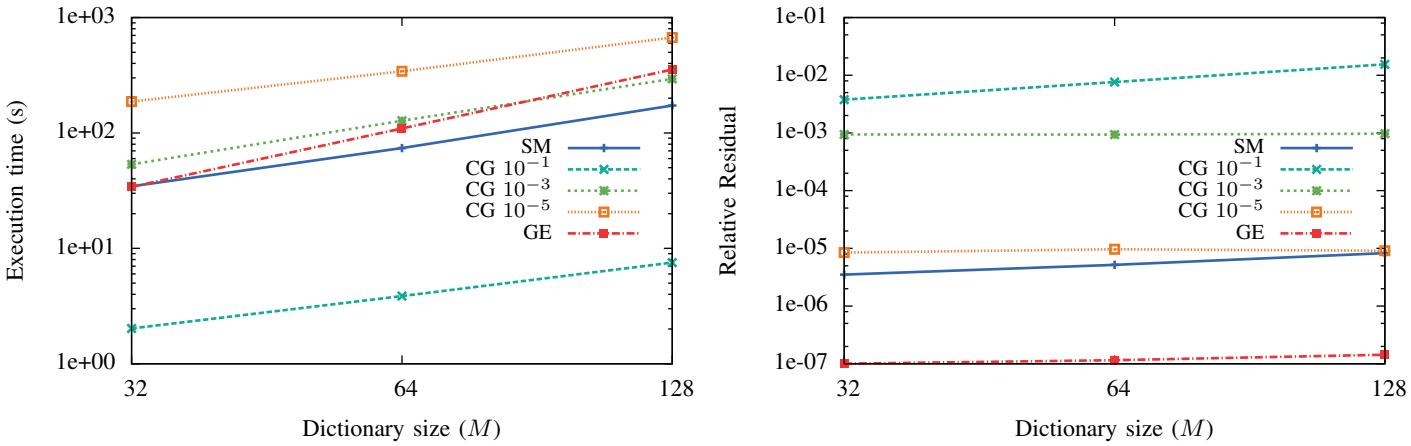


Fig. 13. A comparison of solution errors for double-precision solution of a linear system corresponding to an $8 \times 8 \times 64$ dictionary with $\lambda = 0.1$. Solution methods are Gaussian elimination (GE), Conjugate Gradient (CG), and Sherman-Morrison (SM). The corresponding single-precision comparison is displayed in Fig. 7 in the main document.

In the main document, Fig. 8 presents a comparison of functional value evolution with time for the proposed dictionary learning ADMM algorithm with different solution methods for the main linear system to be solved in the dictionary update. The corresponding comparison against iteration number instead of time is provided here in Fig. 15.

C. Multi-scale Dictionaries

An example multi-scale dictionary is depicted in Fig. 16.



(a) Run time

(b) Solution error

Fig. 14. A comparison of execution times for single precision solution of linear system corresponding to an $8 \times 8 \times M$ dictionary with $K = 32$ and $\lambda = 0.1$. Solution methods are Gaussian elimination (GE), Conjugate Gradient (CG), and Sherman-Morrison (SM).

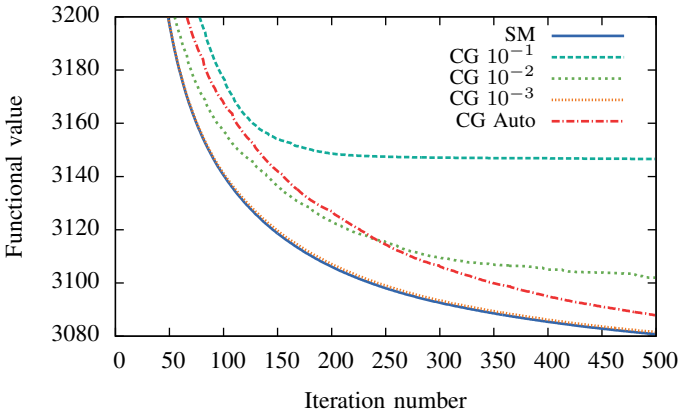


Fig. 15. A comparison of functional value evolution with iteration number the ADMM algorithm with Eq. (41) solved in single precision using Sherman-Morrison and CG with three different relative residual tolerances as well as an automatic tolerance selection method. The dictionary was $8 \times 8 \times 64$ and $\lambda = 0.1$. A corresponding comparison against time is displayed in Fig. 8 in the main document.

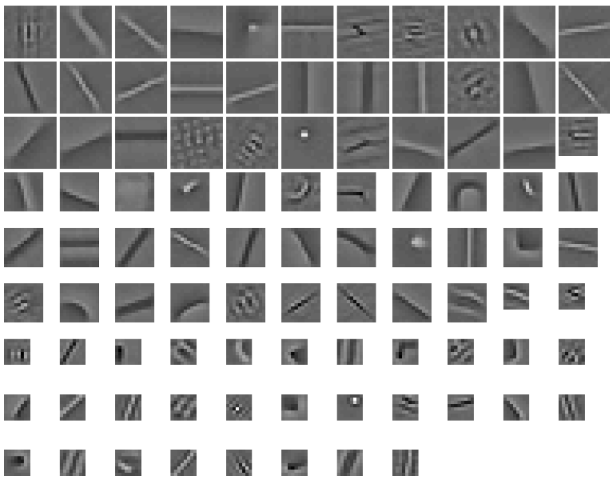


Fig. 16. Example multi-scale dictionary with 8×8, 12×12, and 16×16 filters learned with $\lambda = 0.1$ from a set of 32 512×512 training images derived from a selection of FlickR Creative Commons images.