# EFFICIENT CONVOLUTIONAL SPARSE CODING

*Brendt Wohlberg*

Theoretical Division
Los Alamos National Laboratory
Los Alamos, NM 87545, USA

## 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 the independent sparse coding of each patch results in a representation that is not optimal for the image as a whole. A recent development is *convolutional sparse coding*, in which a sparse representation for 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. A disadvantage of this formulation is its computational expense, but the development of efficient algorithms has received some attention in the literature, with the current leading method exploiting a Fourier domain approach. The present paper introduces a new way of solving the problem in the Fourier domain, leading to substantially reduced computational cost.

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

## 1. INTRODUCTION

Over the past 15 year or so, sparse representations [1] have become a very widely used technique for a variety of problems in image processing. There are numerous approaches to *sparse coding*, the inverse problem of computing a sparse representation of a particular signal or image vector $\mathbf{s}$, one of the most widely used being Basis Pursuit DeNoising (BPDN) [2]

$$\arg\min_{\mathbf{x}} \frac{1}{2} \|D\mathbf{x} - \mathbf{s}\|_2^2 + \lambda \|\mathbf{x}\|_1 \ , \tag{1}$$

where $D$ is a *dictionary* matrix, $\mathbf{x}$ is the sparse representation, and $\lambda$ is a regularization parameter. When applied to images, this decomposition is usually applied independently to a set of overlapping image patches covering the image; this approach is convenient, but often necessitates somewhat ad hoc subsequent handling of the overlap between patches, and results in a representation over the whole image that is suboptimal.

This research was supported by the U.S. Department of Energy through the LANL/LDRD Program.

More recently, these techniques have also begun to be applied, with considerable success, to computer vision problems such as face recognition [3] and image classification [4, 5, 6]. It is in this application context that *convolutional sparse representations* were introduced [7], replacing (1) with

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

where $\{\mathbf{d}_m\}$ is a set of $M$ dictionary *filters*, $*$ denotes convolution, and $\{\mathbf{x}_m\}$ is a set of coefficient maps, each of which is the same size as $\mathbf{s}$. Here $\mathbf{s}$ is a full image, and the $\{\mathbf{d}_m\}$ are usually much smaller. For notational simplicity $\mathbf{s}$ and $\mathbf{x}_m$ are considered to be $N$ dimensional vectors, where $N$ is the the number of pixels in an image, and the notation $\{\mathbf{x}_m\}$ is adopted to denote all $M$ of the $\mathbf{x}_m$ stacked as a single column vector. The derivations presented here are for a single image with a single color band, but the extension to multiple color bands (for both image and filters) and simultaneous sparse coding of multiple images is mathematically straightforward.

The original algorithm proposed for convolutional sparse coding [7] adopted a splitting technique with alternating minimization of two subproblems, the first consisting of the solution of a large linear system via an iterative method, and the other a simple shrinkage. The resulting alternating minimization algorithm is similar to one that would be obtained within an Alternating Direction Method of Multipliers (ADMM) [8, 9] framework, but requires continuation on the auxiliary parameter to enforce the constraint inherent in the splitting. All computation is performed in the spatial domain, the authors expecting that computation in the Discrete Fourier Transform (DFT) domain would result in undesirable boundary artifacts [7]. Other algorithms that have been proposed for this problem include coordinate descent [10], and a proximal gradient method [11], both operating in the spatial domain.

Very recently, an ADMM algorithm operating in the DFT domain has been proposed for dictionary learning for convolutional sparse representations [12]. The use of the Fast Fourier Transform (FFT) in solving the relevant linear systems is shown to give substantially better asymptotic performance than the original spatial domain method, and evidence is presented to support the claim that the resulting boundary

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

effects are not significant.

The present paper describes a convolutional sparse coding algorithm that is derived within the ADMM framework and exploits the FFT for computational advantage. It is very similar to the sparse coding component of the dictionary learning algorithm of [12], but introduces a method for solving the linear systems that dominate the computational cost of the algorithm in time that is linear in the number of filters, instead of cubic as in the method of [12].

## 2. ADMM ALGORITHM

Rewriting (2) in a form suitable for ADMM by introducing auxiliary variables $\{\mathbf{y}_m\}$, we have

$$\begin{aligned} \operatorname*{arg\,min}_{\{\mathbf{x}_m\},\{\mathbf{y}_m\}} \frac{1}{2} \left\| \sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \right\|_2^2 + \lambda \sum_m \|\mathbf{y}_m\|_1 \\ \text{such that } \mathbf{x}_m - \mathbf{y}_m = 0 \; \forall m \;, \end{aligned} \tag{3}$$

for which the corresponding iterations (see [8, Sec. 3]), with dual variables $\{\mathbf{u}_m\}$, are

$$\begin{aligned} \{\mathbf{x}_m\}^{(k+1)} &= \operatorname*{arg\,min}_{\{\mathbf{x}_m\}} \frac{1}{2} \left\| \sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \right\|_2^2 + \\ &\qquad \frac{\rho}{2} \sum_m \left\| \mathbf{x}_m - \mathbf{y}_m^{(k)} + \mathbf{u}_m^{(k)} \right\|_2^2 \end{aligned} \tag{4}$$

$$\begin{aligned} \{\mathbf{y}_m\}^{(k+1)} &= \operatorname*{arg\,min}_{\{\mathbf{y}_m\}} \lambda \sum_m \|\mathbf{y}_m\|_1 + \\ &\qquad \frac{\rho}{2} \sum_m \left\| \mathbf{x}_m^{(k+1)} - \mathbf{y}_m + \mathbf{u}_m^{(k)} \right\|_2^2 \end{aligned} \tag{5}$$

$$\mathbf{u}_m^{(k+1)} = \mathbf{u}_m^{(k)} + \mathbf{x}_m^{(k+1)} - \mathbf{y}_m^{(k+1)} \;. \tag{6}$$

Subproblem (5) is solved via shrinkage/soft thresholding as

$$\mathbf{y}_m^{(k+1)} = \mathcal{S}_{\lambda/\rho} \left( \mathbf{x}_m^{(k+1)} + \mathbf{u}_m^{(k)} \right) \;, \tag{7}$$

where

$$\mathcal{S}_\gamma(\mathbf{u}) = \operatorname{sign}(\mathbf{u}) \odot \max(0, |\mathbf{u}| - \gamma) \;, \tag{8}$$

with $\operatorname{sign}(\cdot)$ and $|\cdot|$ of a vector considered to be applied element-wise. The computational cost is $\mathcal{O}(MN)$.

The only computationally expensive step is solving (4), which is of the form

$$\operatorname*{arg\,min}_{\{\mathbf{x}_m\}} \frac{1}{2} \left\| \sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s} \right\|_2^2 + \frac{\rho}{2} \sum_m \|\mathbf{x}_m - \mathbf{z}_m\|_2^2 \;. \tag{9}$$

### 2.1. DFT Domain Formulation

An obvious approach is to attempt to exploit the FFT for efficient implementation of the convolution via the DFT convolution theorem. (This does involve some increase in memory requirement since the $\mathbf{d}_m$ are zero-padded to the size of the $\mathbf{x}_m$ before application of the FFT.)

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 (9) is equivalent to

$$\operatorname*{arg\,min}_{\{\hat{\mathbf{x}}_m\}} \frac{1}{2} \left\| \sum_m \hat{D}_m \hat{\mathbf{x}}_m - \hat{\mathbf{s}} \right\|_2^2 + \frac{\rho}{2} \sum_m \|\hat{\mathbf{x}}_m - \hat{\mathbf{z}}_m\|_2^2 \tag{10}$$

with the $\{\mathbf{x}_m\}$ minimizing (9) being given by the inverse DFT of the $\{\hat{\mathbf{x}}_m\}$ minimizing (10). Defining

$$\hat{D} = \begin{pmatrix} \hat{D}_0 & \hat{D}_1 & \dots \end{pmatrix} \;, \; \hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_0 \\ \hat{\mathbf{x}}_1 \\ \vdots \end{pmatrix} \;, \; \hat{\mathbf{z}} = \begin{pmatrix} \hat{\mathbf{z}}_0 \\ \hat{\mathbf{z}}_1 \\ \vdots \end{pmatrix} \;, \tag{11}$$

this problem can be expressed as

$$\operatorname*{arg\,min}_{\hat{\mathbf{x}}} \frac{1}{2} \left\| \hat{D} \hat{\mathbf{x}} - \hat{\mathbf{s}} \right\|_2^2 + \frac{\rho}{2} \|\hat{\mathbf{x}} - \hat{\mathbf{z}}\|_2^2 \;, \tag{12}$$

the solution being given by

$$(\hat{D}^H \hat{D} + \rho I) \hat{\mathbf{x}} = \hat{D}^H \hat{\mathbf{s}} + \rho \hat{\mathbf{z}} \;. \tag{13}$$

### 2.2. 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 $\hat{D}$ with its non-zero element at row $n$. As a result, there is no interaction between elements of $\hat{D}$ corresponding to different frequencies, so that (as pointed out in [12]) one need only solve $N$ independent $M \times M$ linear systems to solve (13). Bristow et al. [12] do not specify how they solve these linear systems (and their software implementation was not available for inspection), but since they rate the computational cost of solving them as $\mathcal{O}(M^3)$, it is reasonable to conclude that they apply a direct method such as Gaussian elimination. This can be very effective [8, Sec. 4.2.3] when it is possible to precompute and store a Cholesky or similar decomposition of the linear system(s), but in this case it is not practical unless $M$ is very small, having an $\mathcal{O}(M^2 N)$ memory requirement for storage of these decomposition. Nevertheless, this remains a reasonable approach, the only obvious alternative being an iterative method such as conjugate gradient (CG).

A more careful analysis of the unique structure of this problem, however, reveals that there is an alternative, and vastly more effective, solution. First, define the $m^{\text{th}}$ block of the right hand side of (13) as

$$\hat{\mathbf{r}}_m = \hat{D}_m^H \hat{\mathbf{s}} + \rho \hat{\mathbf{z}}_m \;, \tag{14}$$

so that

$$\begin{pmatrix} \hat{\mathbf{r}}_0 \\ \hat{\mathbf{r}}_1 \\ \vdots \end{pmatrix} = \hat{D}^H \hat{\mathbf{s}} + \rho \hat{\mathbf{z}} \; . \tag{15}$$

Now, denoting the $n^{\text{th}}$ element of a vector $\mathbf{x}$ by $\mathbf{x}(n)$ to avoid confusion between indexing of the vectors themselves and selection of elements of these vectors, define

$$\mathbf{v}_n = \begin{pmatrix} \hat{\mathbf{x}}_0(n) \\ \hat{\mathbf{x}}_1(n) \\ \vdots \end{pmatrix} \quad \mathbf{b}_n = \begin{pmatrix} \hat{\mathbf{r}}_0(n) \\ \hat{\mathbf{r}}_1(n) \\ \vdots \end{pmatrix} \; , \tag{16}$$

and define $\mathbf{a}_n$ as the column vector containing all of the non-zero entries from column $n$ of $\hat{D}^H$, i.e. writing

$$\hat{D} = \begin{pmatrix} \hat{d}_{0,0} & 0 & 0 & \dots & \hat{d}_{1,0} & 0 & 0 & \dots \\ 0 & \hat{d}_{0,1} & 0 & \dots & 0 & \hat{d}_{1,1} & 0 & \dots \\ 0 & 0 & \hat{d}_{0,2} & \dots & 0 & 0 & \hat{d}_{1,2} & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{17}$$

then

$$\mathbf{a}_n = \begin{pmatrix} \hat{d}^*_{0,n} \\ \hat{d}^*_{1,n} \\ \vdots \end{pmatrix} \; , \tag{18}$$

where $^*$ denotes complex conjugation. The linear system to solve corresponding to element $n$ of the $\{\mathbf{x}_m\}$ is

$$(\mathbf{a}_n \mathbf{a}_n^H + \rho I) \mathbf{v}_n = \mathbf{b}_n \; . \tag{19}$$

The critical observation is that the matrix on the left hand side of this system consists of a rank-one matrix plus a scaled identity. Applying the Sherman-Morrison formula

$$(A + \mathbf{u}\mathbf{v}^H)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^H A^{-1}}{1 + \mathbf{u}^H A^{-1} \mathbf{v}} \tag{20}$$

gives

$$(\rho I + \mathbf{a}\mathbf{a}^H)^{-1} = \rho^{-1} \left( I - \frac{\mathbf{a}\mathbf{a}^H}{\rho + \mathbf{a}^H \mathbf{a}} \right) \; , \tag{21}$$

so that the solution to (19) is

$$\mathbf{v}_n = \rho^{-1} \left( \mathbf{b}_n - \frac{\mathbf{a}_n^H \mathbf{b}_n}{\rho + \mathbf{a}_n^H \mathbf{a}_n} \mathbf{a}_n \right) \; . \tag{22}$$

The only vector operations here are inner products, element-wise addition, and scalar multiplication, so that this method is $\mathcal{O}(M)$ instead of $\mathcal{O}(M^3)$ as in [12]. The cost of solving $N$ of these systems is $\mathcal{O}(MN)$, and the cost of the FFTs is $\mathcal{O}(MN \log N)$. Here it is the cost of the FFTs that dominates, whereas in [12] the cost of solving the DFT domain linear systems dominates the cost of the FFTs.

This approach can be implemented in an interpreted language such as Matlab in a form that avoids explicit iteration over the N frequency indices by passing data for all N indices as a single array to the relevant linear-algebraic routines (commonly referred to as *vectorization* in Matlab terminology). Some additional computation time improvement is possible, at the cost of additional memory requirements, by precomputing $\mathbf{a}_n^H / (\rho + \mathbf{a}_n^H \mathbf{a}_n)$ in (22).

### 2.3. Algorithm Summary

The proposed algorithm is summarized in Alg. 1. The stopping criteria are those discussed in [8, Sec. 3.3], together with an upper bound on the number of iterations. The options for the $\rho$ update are (i) fixed $\rho$ (i.e. no update), (ii) the adaptive update strategy described in [8, Sec. 3.4.1], and the multiplicative increase scheme advocated in [12].

**Input**: image $\mathbf{s}$, filter dictionary $\{\mathbf{d}_m\}$, parameters $\lambda$, $\rho$
**Precompute:** FFTs of $\{\mathbf{d}_m\} \to \{\hat{D}_m\}$, FFT of $\mathbf{s} \to \hat{\mathbf{s}}$
**Initialize:** $\{\mathbf{y}_m\} = \{\mathbf{u}_m\} = 0$
**while** *stopping criteria not met* **do**
  Compute FFTs of $\{\mathbf{y}_m\} \to \{\hat{\mathbf{y}}_m\}$, $\{\mathbf{u}_m\} \to \{\hat{\mathbf{u}}_m\}$
  Compute $\{\hat{\mathbf{x}}_m\}$ using the method in Sec. 2.2
  Compute inverse FFTs of $\{\hat{\mathbf{x}}_m\} \to \{\mathbf{x}_m\}$
  $\{\mathbf{y}_m\} = \mathcal{S}_{\lambda/\rho}\left(\{\mathbf{x}_m\} + \{\mathbf{u}_m\}\right)$
  $\{\mathbf{u}_m\} = \{\mathbf{u}_m\} + \{\mathbf{x}_m\} - \{\mathbf{y}_m\}$
  Update $\rho$ if appropriate
**end**
**Output**: Coefficient maps $\{\mathbf{x}_m\}$

**Algorithm 1**: Summary of proposed ADMM algorithm

The computational cost of the algorithm components is $\mathcal{O}(MN \log N)$ for the FFTs, order $\mathcal{O}(MN)$ for the proposed linear solver, and $\mathcal{O}(MN)$ for both the shrinkage and dual variable update, so that the cost of the entire algorithm is $\mathcal{O}(MN \log N)$, dominated by the cost of FFTs. In contrast, the cost of the algorithm proposed in [12] is $\mathcal{O}(M^3 N)$ (there is also an $\mathcal{O}(MN \log N)$ cost for FFTs, but it is dominated by the $\mathcal{O}(M^3 N)$ cost of the linear solver), and the cost of the original spatial-domain algorithm [7] is $\mathcal{O}(M^2 N^2 L)$, where $L$ is the dimensionality of the filters.

## 3. DICTIONARY LEARNING

The extension of (2) to learning a dictionary from training data involves replacing the minimization with respect to $\mathbf{x}_m$ with minimization with respect to both $\mathbf{x}_m$ and $\mathbf{d}_m$. The optimization is invariably performed via alternating minimization between the two variables, the most common approach consisting of a sparse coding step followed by a dictionary update [13]. The commutativity of convolution suggests that the DFT domain solution of Sec. 2.1 can be directly applied in minimizing with respect to $\mathbf{d}_m$ instead of $\mathbf{x}_m$, but this is not possible since the $\mathbf{d}_m$ are of constrained size, and must be zero-padded to the size of the $\mathbf{x}_m$ prior to a DFT domain implementation of the convolution. If the size constraint is implemented in an ADMM framework [14], however, the problem is decomposed into a computationally cheap subproblem corresponding to a projection onto to constraint set, and another subproblem that can be efficiently solved by extending the method in Sec. 2.1. This iterative algorithm for the dictionary update can alternate with a sparse coding stage to form a

more traditional dictionary learning method [15], or the subproblems of the sparse coding and dictionary update algorithms can be combined into a single ADMM algorithm [12].

## 4. RESULTS

A comparison of execution times for the algorithm ($\lambda = 0.05$) with different methods of solving the linear system, for a set of overcomplete $8 \times 8$ DCT dictionaries and the $512 \times 512$ greyscale Lena image, is presented in Fig. 1. It is worth emphasizing that this is a large image by the standards of prior publications on convolutional sparse coding; the test images in [12], for example, are $50 \times 50$ and $128 \times 128$ pixels in size.

The Gaussian elimination solution is computed using a Cholesky decomposition (since it is, in general, impossible to cache this decomposition, it is necessary to recompute it at every solution), as implemented by the Matlab `mldivide` function, and is applied by iterating over all frequencies in the apparent absence of any practical alternative.

The conjugate gradient solution is computed using two different relative error tolerances. A significant part of the computational advantage here of CG over the direct method is that it is applied simultaneously over all frequencies.

The two curves for the proposed solver based on the Sherman-Morrison formula illustrate the significant gain from an implementation that simultaneously solves over all frequencies and that the relative advantage of doing so decreases with increasing $M$.

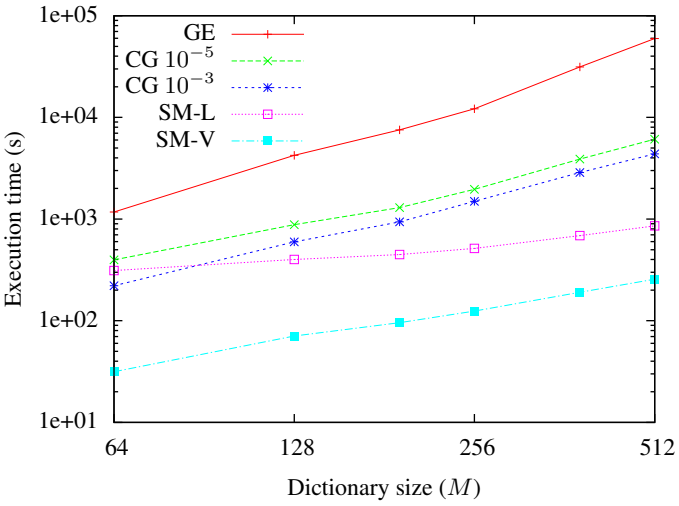


**Fig. 1**. A comparison of execution times for 10 steps of the ADMM algorithm for different methods of solving the linear system: Gaussian elimination (GE), Conjugate Gradient with relative error tolerance $10^{-5}$ (CG $10^{-5}$) and $10^{-3}$ (CG $10^{-3}$), and Sherman-Morrison implemented with a loop over frequencies (SM-L) or jointly over all frequencies (SM-V).

The performance of the three $\rho$ update strategies discussed in the previous section was compared by sparse coding a $256 \times 256$ Lena image using a $9 \times 9 \times 512$ dictionary (from [16], by the authors of [17]) with a fixed value of $\lambda = 0.02$ and a range of initial $\rho$ values $\rho_0$. The resulting values of the functional in (2) after 100, 500, and 1000 iterations of the proposed algorithm are displayed in Table 1. The adaptive update strategy uses the default parameters of [8, Sec. 3.4.1], and the increasing strategy uses a multiplicative update by a factor of 1.1 with a maximum of $10^5$, as advocated by [12].

In summary, a fixed $\rho$ can perform well, but is sensitive to a good choice of parameter. When initialized with a small $\rho_0$, the increasing $\rho$ strategy provides the most rapid decrease in functional value, but thereafter converges very slowly. Overall, unless rapid computation of an approximate solution is desired, the adaptive $\rho$ strategy appears to provide the best performance, with the least sensitivity to choice of $\rho_0$. This issue is complex, however, and further experimentation is necessary before drawing any general conclusions that could be considered valid over a broad range of problems.

| Iter. \ $\rho_0$ | $10^{-2}$ | $10^{-1}$ | $10^{0}$ | $10^{1}$ | $10^{2}$ | $10^{3}$ |
|---|---|---|---|---|---|---|
| Fixed $\rho$ | | | | | | |
| 100 | 28.27 | 27.80 | 18.10 | 10.09 | **9.76** | 11.60 |
| 500 | 28.05 | 22.25 | 11.11 | **8.89** | 9.11 | 10.13 |
| 1000 | 27.80 | 17.00 | 9.64 | **8.82** | 8.96 | 9.71 |
| Adaptive $\rho$ | | | | | | |
| 100 | 21.62 | 16.97 | 14.56 | **10.71** | 11.14 | 11.41 |
| 500 | 10.81 | 10.23 | 9.81 | **9.01** | 9.18 | 9.09 |
| 1000 | 9.44 | 9.21 | 9.06 | **8.83** | 8.87 | 8.84 |
| Increasing $\rho$ | | | | | | |
| 100 | 14.78 | 9.82 | **9.50** | 9.90 | 11.51 | 15.15 |
| 500 | 9.55 | **9.45** | 9.46 | 9.89 | 11.47 | 14.51 |
| 1000 | 9.53 | **9.44** | 9.45 | 9.88 | 11.41 | 13.97 |

**Table 1**. Comparison of functional value convergence for the same problem with three different $\rho$ update strategies.

## 5. CONCLUSION

A computationally efficient algorithm is proposed for solving the convolutional sparse coding problem in the Fourier domain. This algorithm has the same general structure as a previously proposed approach [12], but enables a very significant reduction in computational cost by careful design of a linear solver for the most critical component of the iterative algorithm. The theoretical computational cost of the algorithm is reduced from $\mathcal{O}(M^3)$ to $\mathcal{O}(MN \log N)$ (where $N$ is the dimensionality of the data and $M$ is the number of elements in the dictionary), and is also shown empirically to result in greatly reduced computation time. The significant improvement in efficiency of the proposed approach is expected to greatly increase the range of problems that can practically be addressed via convolutional sparse representations.

## 6. 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] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998. doi:10.1137/S1064827596304010

[3] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, February 2009. doi:10.1109/tpami.2008.79

[4] Y. Boureau, F. Bach, Y. A. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 2559–2566. doi:10.1109/cvpr.2010.5539963

[5] J. Yang, K. Yu, and T. S. Huang, "Supervised translation-invariant sparse coding," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3517–3524, 2010. doi:10.1109/cvpr.2010.5539958

[6] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, April 2012. doi:10.1109/tpami.2011.156

[7] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 2528–2535. doi:10.1109/cvpr.2010.5539957

[8] 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

[9] J. Eckstein, "Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results," Rutgers Center for Operations Research, Rutgers University, Rutcor Research Report RRR 32-2012, December 2012. [Online]. Available: http://rutcor.rutgers.edu/pub/rrr/reports2012/32_2012.pdf

[10] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. A. LeCun, "Learning convolutional feature hierachies for visual recognition," in *Advances in Neural Information Processing Systems (NIPS 2010)*, 2010.

[11] R. Chalasani, J. C. Principe, and N. Ramakrishnan, "A fast proximal method for convolutional sparse coding," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013, pp. 1–5. doi:10.1109/IJCNN.2013.6706854

[12] H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2013, pp. 391–398. doi:10.1109/CVPR.2013.57

[13] B. Mailhé and M. D. Plumbley, "Dictionary learning with large step gradient descent for sparse representations," in *Latent Variable Analysis and Signal Separation*, ser. Lecture Notes in Computer Science, F. J. Theis, A. Cichocki, A. Yeredor, and M. Zibulevsky, Eds. Springer Berlin Heidelberg, 2012, vol. 7191, pp. 231–238. doi:10.1007/978-3-642-28551-6_29

[14] 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 Transactions on Image Processing*, vol. 20, no. 3, pp. 681–695, March 2011. doi:10.1109/tip.2010.2076294

[15] K. Engan, S. O. Aase, and J. H. Husøy, "Method of optimal directions for frame design," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5, 1999, pp. 2443–2446. doi:10.1109/icassp.1999.760624

[16] J. Mairal, Software available from http://lear.inrialpes.fr/people/mairal/denoise_ICCV09.tar.gz.

[17] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2009, pp. 2272–2279. doi:10.1109/iccv.2009.5459452