LEARNING SPARSIFYING TRANSFORMS FOR IMAGE PROCESSING

Saiprasad Ravishankar and Yoram Bresler

Department of Electrical and Computer Engineering and the Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign, IL, USA

ABSTRACT

The sparsity of signals and images in a certain analytically defined transform domain or dictionary such as discrete cosine transform or wavelets has been exploited in many applications in signal and image processing. Recently, the idea of learning a dictionary for sparse representation of data has become popular. However, while there has been extensive research on learning synthesis dictionaries, the idea of learning analysis sparsifying transforms has received only little attention. We propose a novel problem formulation and an alternating algorithm for learning well-conditioned square sparsifying transforms from data. We show the superiority of our approach for image representation over analytical sparsifying transforms such as the DCT. We also show promise in image denoising. Denoising using the learnt analysis transforms is not only better than by synthesis dictionaries learnt using the K-SVD algorithm but also faster.

Index Terms— Analysis transforms, Image representation, Image denoising, Dictionary learning, Sparse representation.

1. INTRODUCTION

Sparse representation of signals in certain transform domains or dictionaries has become widely popular in recent years. It is well known that natural signals and images have an essentially sparse representation (few significant non-zero coefficients) in analytical transform domains such as Wavelets and discrete cosine transform (DCT). This property has been exploited for image denoising and compression.

Alternatively, a signal $y \in \mathbb{R}^n$ may be represented as a linear combination of a small number of atoms/columns from a dictionary $D \in \mathbb{R}^{n \times K}$. Here, it is assumed that $y \approx Dx$, where $x \in \mathbb{R}^K$ is sparse, i.e., $\|x\|_0 \ll K$. The l_0 quasi norm counts the number of non-zeros in x (or measures the sparsity of x). When $n = K$, and D is full rank, the dictionary D is a basis, whereas when $K > n$, the dictionary is said to be overcomplete. This approach of representing a signal in a dictionary D, also known as the *synthesis model* [1], has received considerable attention recently. In particular, the idea of learning such a dictionary from training signals has been exploited (cf. [2] and the references therein).

Adaptive synthesis dictionaries have been shown to be useful in a variety of applications such as image denoising, image inpainting, deblurring, and demosaicing [3, 4]. Recently, we have shown impressive performance for learnt dictionaries in magnetic resonance imaging (MRI) involving highly undersampled k-space data [5].

On the other hand, the *analysis model* [1] has been quite popular in applications such as compression. This model suggests that a signal y is *sparsifiable* using a transform $W \in \mathbb{R}^{m \times n}$, that is $Wy \approx x$, where $x \in \mathbb{R}^m$ is sparse, i.e., $\|x\|_0 \ll m$ (x is referred to as the sparse code for y). Elad et al. [1] derive conditions under which analysis-based priors and synthesis-based priors approximate one another. Well known examples of analysis transforms for natural signals include Wavelets and DCT. These transforms are square, i.e., $m = n$, and orthonormal. When a suitable sparsifying transform or analysis dictionary W is known for the signal y, the process of obtaining a sparse code x of given sparsity T_0 merely involves thresholding the product Wy. In contrast, sparse coding with the synthesis dictionary involves an NP-hard problem that is typically solved using greedy algorithms such as orthogonal matching pursuit (OMP) [6]. Thus, an analysis dictionary is much simpler and faster to use in practice.

Learning the sparsifying transform from the data could make it more effective in applications. In this adaptive setting, given a matrix $Y \in \mathbb{R}^{n \times N}$ whose columns represent training signals, we would like to learn a transform W such that the *sparsification error* denoted as $\|WY - X\|_F^2$ is minimized. Here, X is an unknown matrix containing the sparse codes of the training signals as columns. This idea of learning a sparsifying transform, or an analysis model, is the subject of this paper. In this work, we study the learning of square and well-conditioned transforms.

The idea of learning analysis dictionaries has received only little recent attention. Yaghoobi et al. [7] and Elad et al. [8] attempt to minimize the l_0 or l_1 norm of WY (or alternatively of WZ where Z approximates Y [8]). However, it is unclear whether these methods can outperform synthesis based approaches like K-SVD [2] in performance and speed. Furthermore, the promise of learnt transforms in image processing applications has not been adequately explored.

In this paper, we show that sparsifying transforms that are adapted to the image data provide significantly better representations than fixed transforms such as the DCT. We also demonstrate the promise of adaptive transforms in image denoising where the results are better than those obtained using the overcomplete K-SVD [9]. Moreover, our analysis transform learning algorithm is much faster than the synthesis K-SVD.

2. TRANSFORM LEARNING

Given the training matrix $Y \in \mathbb{R}^{n \times N}$, we propose learning a transform $W \in \mathbb{R}^{n \times n}$ adapted to the data as follows.

$$(P1)\ \min_{W,X} \|WY - X\|_F^2 \quad s.t.\ \ \|X_i\|_0 \leq T_0\ \ \forall\ i \qquad (1)$$

Here, the columns X_i of matrix $X \in \mathbb{R}^{n \times N}$ with maximum allowed sparsity level T_0 are the sparse codes of the signals/columns in Y. The objective function in (P1) is the sparsification error, which measures the deviation of the data in the transform domain from perfect sparsity at sparsity level T_0. This choice of the objective function is motivated by the fact that natural signals and images can be reasonably approximated in a transform domain (such as Wavelets) using few (here T_0) significant non-zero transform coefficients.

This work was supported in part by the National Science Foundation (NSF) under grant CCF 10-18660.

In order to avoid the trivial solution $W = 0, X = 0$, as well as solutions that have repeated rows, or linearly dependent rows (since we want the transform to convey maximum information), we modify Problem (P1) to obtain the following formulation, where λ is a positive constant.

$$(P2) \;\; \min_{W,X} \; \|WY - X\|_F^2 - \lambda \log \det W \qquad (2)$$
$$s.t. \;\; \|X_i\|_0 \leq T_0 \;\; \forall \; i$$

The $\log \det W$ penalty restricts the solution of (P2) to full rank transforms. Problem (P2) is non-convex. One could constrain W to be positive definite, $W \succ 0$. If in addition, the l_0 quasi norm for X is relaxed to an l_1 norm, the resulting problem would be jointly convex in W and X. However, enforcing the transform to be positive definite is too restrictive in general. Many well-known sparsifying transforms such as DCT and Wavelets are not positive definite. Therefore, we do not use such a constraint. Instead, we only require $\det W > 0$ so that $\log \det W$ is well-defined (the cost remains non-convex in this case). This constraint is not restrictive since the sign of $\det W$ can be trivially changed by e.g., row permutation. However, although the constraint $\det W > 0$ is required/implied, we do not explicitly enforce it, except to initialize the algorithm with W_0 such that $\det W_0 > 0$. This is because the cost function has log-barriers in the space of matrices at W for which the determinant is zero. These log-barriers help prevent optimization algorithms that minimize the objective function from getting into the infeasible regions where $\det W < 0$.

When the data Y admits an exact representation, i.e., if there exists a pair $(\tilde{W}, \tilde{X})$ with $\tilde{X}$ sparse such that $\tilde{W}Y = \tilde{X}$, then the cost in (P2) can be made arbitrarily small by pre-multiplying $\tilde{W}$ and $\tilde{X}$ by a scalar α (or equivalently, by a diagonal matrix Γ with entries $\pm\alpha$ which is such that $\det(\Gamma\tilde{W}) > 0$) with $\alpha \to \infty$. The cost becomes unbounded from below in this case, which spells trouble for optimization algorithms. In order to address this scaling ambiguity, we introduce an additional norm penalty, which is convex.

$$(P3) \;\; \min_{W,X} \; \|WY - X\|_F^2 - \lambda \log \det W + \mu \|W\|_F^2 \qquad (3)$$
$$s.t. \;\; \|X_i\|_0 \leq T_0 \;\; \forall \; i$$

The cost function in (P3) is lower-bounded since $\|W\|_F^2$ grows faster than $\log \det W$. Moreover, it can be shown that together, the $\|W\|_F^2$ and the $-\log \det W$ penalties provide complete control of the condition number of the learnt transform. This is important, because badly conditioned transforms typically convey little information and may degrade performance in applications.

Algorithm and Properties

Our algorithm for solving Problem (P3) alternates between updating X and W. In step 1 (Sparse Update Step), we solve Problem (P3) with fixed W.

$$\min_{X} \; \|WY - X\|_F^2 \;\; s.t. \;\; \|X_i\|_0 \leq T_0 \;\; \forall \; i \qquad (4)$$

The solution X can be computed exactly by thresholding WY, and retaining only the T_0 largest coefficients in each column. In step 2 of the algorithm (Transform Update Step), we solve Problem (P3) with fixed X. This involves the following unconstrained minimization

$$\min_{W} \; \|WY - X\|_F^2 - \lambda \log \det W + \mu \|W\|_F^2 \qquad (5)$$

This problem can be easily solved using iterative methods such as steepest descent or conjugate gradients (CG). CG typically converges faster than steepest descent. We can employ CG with the Armijo step size rule which guarantees reduction of cost function. Fixed step size rules were also observed to work well and faster in practice. CG typically converges quickly and a fixed number of CG iterations were empirically observed to work well. For our algorithm to work, W must be initialized to have a positive determinant.

The proposed algorithm alternates between sparse update and transform update steps. The solution for the sparse update step is exact. Thus, the cost function can only decrease in this step. For the transform update step, the solution is obtained by CG (for instance with Armijo step size rule). Thus, in this step too, the cost function decreases. Since the cost function is monotone decreasing and lower bounded, it converges. The computational cost per iteration (of sparse update and transform update) of the proposed algorithm scales as $O(Nn^2)$. This compares favorably with the $O(Nn^3)$ cost per iteration of K-SVD for square synthesis dictionaries [9].

3. EXPERIMENTS

3.1. Convergence and Learning

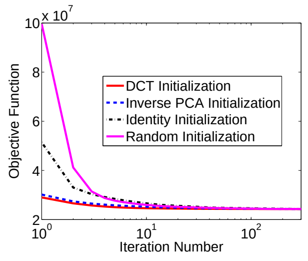
For the first experiment, we extract the 8×8 non-overlapping patches from the image Barbara [2]. The data matrix Y in this case has 4096 training signals (patches represented as vectors) and we work with $T_0 = 11$. The means (or DC values) of the patches are removed and we only sparsify the mean-subtracted patches (mean removal is typically adopted in image processing applications such as K-SVD based image denoising). Problem (P3) is solved to learn a square transform W that is adapted to this data. The algorithm parameters are $\lambda = \mu = 4 \times 10^5$. The CG algorithm was run for 128 iterations for each transform update step, with a fixed step size of 10^{-8} (performance is similar even with 20 or less CG iterations).

We consider four different initializations (initial transforms) for the algorithm. The first is the 64×64 2D DCT matrix (defined as $W_0 \otimes W_0$ where W_0 is the 8×8 1D DCT matrix and "$\otimes$" denotes the Kronecker product). The second initialization called 'inverse PCA' is obtained by inverting the left singular matrix of Y. The third and fourth initializations are the identity matrix and a random matrix with i.i.d. gaussian entries (zero mean and standard deviation 0.2) respectively. Note that any initial matrix with a negative determinant can be made to have a positive determinant by switching the signs of the entries on one row, or by exchanging two rows.

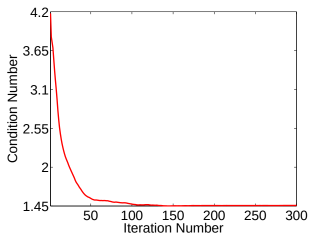
Figure 1 (a) shows the objective function of Problem (P3) over the iterations of the algorithm for the different initializations. The cost function converges monotonically and although different initializations have different initial rates of convergence, the costs have nearly identical final values. This indicates that our alternating algorithm is robust to initialization. The sparsification error (Figures 1 (c) and 1 (d)) too converges quickly and to similar values for all initializations. The horizontal lines in Figures 1 (c) and 1 (d) denote the sparsification errors of the 2D DCT, inverse PCA, identity, and random gaussian transforms (i.e., the sparsification error at iteration zero). Our algorithm reduces the sparsification error by 5.98 dB, 7.14 dB, 14.77 dB, and 18.72 dB respectively compared to the values for the initial transforms.

We also define the 'normalized sparsification error' as the ratio $\|WY - X\|_F^2 \, / \, \|WY\|_F^2$. This measures the fraction of energy lost in sparse fitting. The normalized sparsification error for the learnt transform W (with X being the sparse code obtained by thresholding WY) with the 2D DCT initialization is 0.0437. The values corresponding to the other initializations are only slightly different.

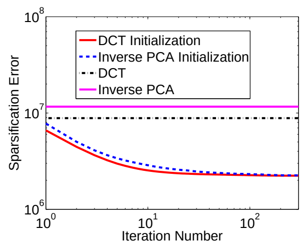
Finally, the condition number shown in Figure 1 (b) for the random gaussian initialization, also converges quickly to a low value of 1.46. Thus, image patches are well sparsified by well-conditioned



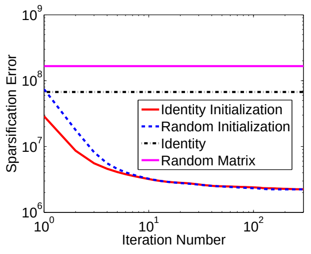
(a)



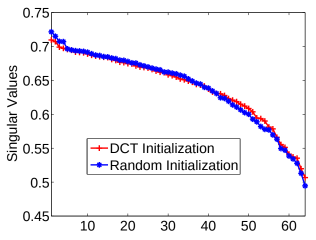
(b)



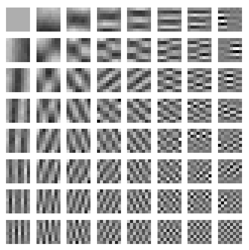
(c)



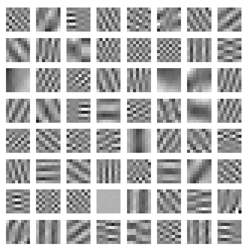
(d)



(e)



(f)



(g)

Fig. 1. Effect of different Initializations: Objective function vs. iterations (a); Condition number vs. iterations for random gaussian initialization (b); Sparsification error vs. iterations for DCT, Inverse PCA (c) and identity and random gaussian (d) initializations, along with sparsification errors (horizontal lines) of the initial transforms; Singular values of the transforms learnt with DCT and random gaussian initializations (e); Transforms learnt with DCT (f) and random gaussian (g) initializations.

transforms.

To study further the effect of different initializations, Figure 1 (e) compares the singular values of the transforms learnt with 2D DCT and with random gaussian initializations. The singular values for the two cases are almost identical with condition numbers of 1.40 and 1.46 respectively. For direct comparison, the transforms learnt with the two different initializations are compared in Figures 1 (f) and 1 (g), with each row of W displayed as an image patch. While the two learnt transforms appear different, they belong to the same 'equivalence class' in the sense that they produce very similar sparsification errors and have almost identical singular values (and thus, almost identical condition numbers). The existence of alternative but equivalent solutions to (P3) suggests that additional application-specific performance criteria may be used to select between equivalent transforms, or the problem formulation may be modified to incorporate additional preferences.

	NSE (Learnt)	CN (Learnt)	NSE (2D DCT)
Barbara	0.0437	1.40	0.0676
Lena	0.0376	1.16	0.0474
Peppers	0.0343	1.17	0.0448
Cameraman	0.0088	1.13	0.0191

Table 1. Normalized sparsification errors (NSE) and condition numbers (CN) for adapted transforms and 2D DCT.

Next, we study the behavior of our algorithm on different images, by solving (P3) to learn a transform for each of four different 512×512 test images. The transforms are directly adapted to the non-overlapping patches of the images. All algorithm parameters are the same as for the experiment of Figure 1. Table 1 lists the normalized sparsification errors of the adapted transforms and the patch-based 2D DCT, along with the condition numbers of the adapted transforms. The learnt transforms are seen to be well-conditioned for all the test images. The corresponding normalized sparsification errors are small and, moreover, better than those of 2D DCT by upto 3.4 dB for the test images.

Note that while we considered adapting the transform to specific images, a transform adapted to a data-set of training images also gives promising improvements over DCT and Wavelets in test images – a property that can be exploited for image compression.

3.2. Preliminary Results for Image Denoising

The goal of image denoising is to estimate an image $x \in \mathbb{R}^P$ (2D image represented as a vector) from its measurement $y = x + n$, that is corrupted by noise n. While there are numerous learned synthesis dictionary-based methods for image denoising [4, 9], we focus here on adaptive sparsifying transform-based image denoising. Our goal here is an initial exploration of the potential of the proposed framework and algorithm for learning an analysis transform in this classical and prototypical application.

Similar to the previous methods, we work with overlapping image patches. We propose a simple problem formulation for denoising as follows

$$(P4) \quad \min_{x_i, \alpha_i} \sum_{i=1}^{M} \|W x_i - \alpha_i\|_2^2 + \tau \sum_{i=1}^{M} \|R_i\, y - x_i\|_2^2 \qquad s.t. \quad \|\alpha_i\|_0 \leq T_i \;\; \forall\, i \tag{6}$$

Here, $R_i\, y$ denotes the i^{th} patch from the noisy image y (M overlapping patches are assumed). The operator $R_i \in \mathbb{R}^{n \times P}$ extracts a $\sqrt{n} \times \sqrt{n}$ patch as a vector from the image. Vector x_i represents the denoised image patch corresponding to $R_i\, y$. We assume that the noisy patch $R_i\, y$ can be approximated by a noiseless patch x_i that is sparsifiable. The vector $\alpha_i \in \mathbb{R}^{n \times n}$ denotes the sparse representation of x_i in the transform W that has T_i non-zeros. The parameter τ is typically inversely proportional to the noise level σ [9]. Once, the denoised patches x_i are found, the denoised image x is obtained by averaging the x_i's at their respective locations in the image (cf. [5]). Problem formulation (P4) denoises the patches using a given W. This W can be learnt from the patches of the noisy image.

The steps of our denoising method are outlined as follows. In step 1, we extract the noisy image patches $R_i\, y$ and subtract away each of their mean/DC values. In step 2, we learn the transform W from the noisy image patches (with fixed sparsity levels). In step 3, we use the learnt transform within (P4) to obtain the denoised patches. Problem (P4) is non-convex. We solve it in two steps wherein the x_i's are initially set to be equal to $R_i\, y$.

The α_i's are first updated in (P4) by thresholding $Wx_i = WR_i\, y$. Then, for fixed α_i, Problem (P4) reduces to a least squares problem in the x_i's. Each x_i can be independently updated as follows where † denotes the pseudo-inverse (note that the two step update for solving (P4) could be iterated but we found that a single iteration is sufficient in practice, which also reduces run-time).

$$x_i = \begin{bmatrix} \sqrt{\tau} I \\ W \end{bmatrix}^{\dagger} \begin{bmatrix} \sqrt{\tau} R_i y \\ \alpha_i \end{bmatrix} \tag{7}$$

The sparsity level T_i at which $WR_i\, y$ should be thresholded (or α_i is updated) is typically unknown apriori. The sparse coding step in K-SVD based denoising [9] stops when the patch-based fitting error (between the noisy patch and its dictionary representation) falls below $nC^2\sigma^2$, where C is a fixed parameter. This stopping rule is crucial for the success of patch-based denoising strategies. Hence, here we choose the level T_i for the i^{th} patch such that the error term $\|R_i\, y - x_i\|_2^2$ computed after updating x_i by (7) is below $nC^2\sigma^2$. This requires repeating the two step solution of (P4) for patch i at various sparsity levels to determine the level at which the error term falls below the required threshold. However, this process can be done efficiently by adding one non-zero element at a time from $WR_i\, y$ (elements chosen in descending order) to α_i in equation (7) until the error measure $\|R_i\, y - x_i\|_2^2$ (with the newly updated x_i) falls below the required threshold.

The matrix $G = \begin{bmatrix} \sqrt{\tau} I \\ W \end{bmatrix}^{\dagger}$ can be pre-computed and the addition of a new coefficient to α_i leads to x_i being updated by adding a scaled column of G (scaled by the new coefficient of α_i). In step 4 of the denoising algorithm, we add back the mean values to each x_i. Step 5 computes x by patch-averaging and the updated x is then restricted to its range (e.g., 0-255), if known.

We present preliminary results for our image denoising framework. We add i.i.d. gaussian noise at noise level $\sigma = 10$ to the 'peppers' image [4]. The noisy image is shown in Figure 2. We work with a patch size of 8×8 ($n = 64$) and include all overlapping patches. We choose parameters $\mu = \lambda = 2 \times 10^6$ and initialize transform learning with the (patch-based) 2D DCT with sparsity level set as $n/5$ (rounded to nearest integer). The denoising algorithm parameters were set as $C = 1.08$ and $\tau = 0.01/\sigma$.

The denoised image with the adapted transform (well-conditioned, with condition number of 2.66) is shown in Figure 2 and has a peak signal to noise ratio (PSNR) of 34.43 dB. This is better than the denoising PSNR of 34.28 dB obtained with the 64×256 K-SVD overcomplete synthesis dictionary [9] (trained on noisy image). Moreover, computationally, the sparse update step in transform learning (thresholding) is much faster than the sparse coding step in K-SVD (computational cost of K-SVD dominated by sparse coding). Currently, our code takes 4.6 minutes to denoise the peppers image (this time can be significantly reduced by code optimization).

We also repeat the experiment with the same parameters for the image Barbara [4] but using a larger, 9×9 patch ($n = 81$). Moreover, only a fraction 26% of all overlapping noisy patches is used for training to ensure faster learning [4]. The noisy and denoised images for this case are shown in Fig. 2. The denoised image has a PSNR of 34.48 dB. This is better than the denoising PSNR of 34.42 dB obtained with the 64×256 K-SVD dictionary [9]. Note that an 81×81 transform still has fewer free parameters than a 64×256 synthesis dictionary. The denoising PSNRs here can be further improved by optimal choice of parameters.

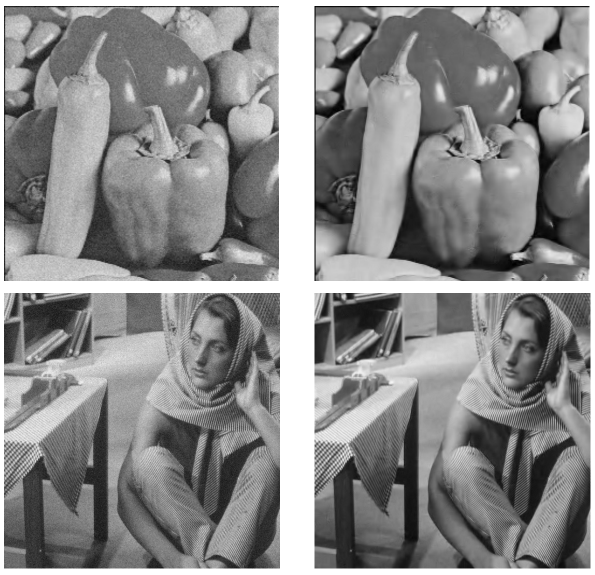


Fig. 2. Noisy Images (Left), Denoised Images (Right).

4. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel problem formulation for learning square sparsifying transforms. The proposed algorithm involves alternations between thresholding and the conjugate gradient method. Our formulation gives rise to transforms with much lower sparsification errors compared to analytical transforms. The algorithm was shown to provide monotonic decrease in the cost function and is insensitive to initialization. We demonstrated the promise of transform learning for image denoising, achieving results better than K-SVD on the examples in this paper. A detailed study of the effectiveness of transform learning in denoising, and in other image processing applications is left for future work.

5. REFERENCES

[1] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," *Inverse Problems*, vol. 23, no. 3, pp. 947–968, 2007.

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

[3] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 53–69, 2008.

[4] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *SIAM Multiscale Modeling and Simulation*, vol. 7, no. 1, pp. 214–241, 2008.

[5] S. Ravishankar and Y. Bresler, "MR image reconstruction from highly undersampled k-space data by dictionary learning," *IEEE Trans. Med. Imag.*, vol. 30, no. 5, pp. 1028–1041, 2011.

[6] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inform. Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[7] M. Yaghoobi, S. Nam, R. Gribonval, and M. Davies, "Analysis operator learning for overcomplete cosparse representations," in *EUSIPCO*, 2011.

[8] R. Rubinstein and M. Elad, "K-SVD dictionary-learning for analysis sparse models," in *Proc. SPARS11*, June 2011.

[9] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, 2006.