EXPLOITING CORRELATIONS AMONG CHANNELS IN DISTRIBUTED COMPRESSIVE SENSING WITH CONVOLUTIONAL DEEP STACKING NETWORKS

Hamid Palangi[1], *Rabab Ward*[1], *Li Deng*[2]

[1]University of British Columbia, Vancouver, BC, Canada
[2]Microsoft Research, Redmond, WA, USA

ABSTRACT

This paper addresses the compressive sensing with Multiple Measurement Vectors (MMV) problem where the correlation amongst the different sparse vectors (channels) are used to improve the reconstruction performance. We propose the use of Convolutional Deep Stacking Networks (CDSN), where the correlations amongst the channels are captured by a moving window containing the "*residuals*" of different sparse vectors. We develop a greedy algorithm that exploits the structure captured by the CDSN to reconstruct the sparse vectors. Using a natural image dataset, we compare the performance of the proposed algorithm with two types of reconstruction algorithms: Simultaneous Orthogonal Matching Pursuit (SOMP) which is a greedy solver and the model-based Bayesian approaches that also exploit correlation among channels. We show experimentally that our proposed method outperforms these popular methods and is almost as fast as the greedy methods.

Index Terms— Convolutional Neural Network, Distributed Compressive Sensing, Deep Stacking Network.

1. INTRODUCTION

Compressive Sensing (CS) [1, 2, 3] is an effective framework where both sensing and compression are performed simultaneously. The only requirements are the sparsity of the signal in some basis, and the incoherence between this sparsifying basis and the measurement matrix. Since many natural signals are sparse in the time (or spatial) domain or in a transform domain, CS has found many applications including medical imaging, remote sensing, healthcare tele-monitoring, etc.

In the general CS framework, instead of acquiring N samples of a signal $\mathbf{x} \in \Re^{N\times 1}$, M random measurements are acquired where $M < N$. This is expressed by:

$$\mathbf{y} = \mathbf{\Phi x} \tag{1}$$

where $\mathbf{y} \in \Re^{M\times 1}$ is the known measured vector and $\mathbf{\Phi} \in \Re^{M\times N}$ is a random measurement matrix. To uniquely recover $\mathbf{x}$ given $\mathbf{y}$ and $\mathbf{\Phi}$, $\mathbf{x}$ must be sparse in a given basis $\mathbf{\Psi}$. This means that

$$\mathbf{x} = \mathbf{\Psi s} \tag{2}$$

From (1) and (2):

$$\mathbf{y} = \mathbf{As} \tag{3}$$

where $\mathbf{A} = \mathbf{\Phi\Psi}$. Since there is only one measurement vector, the above problem is usually called the Single Measurement Vector (SMV) problem in compressive sensing.

In distributed compressive sensing, also known as the Multiple Measurement Vectors (MMV) problem, a set of L measurement vectors $\{\mathbf{y}_i\}_{i=1,2,\dots,L}$ is given. A set of L sparse vectors $\{\mathbf{s}_i\}_{i=1,2,\dots,L}$ is to be jointly recovered from these measurement vectors. Let the L sparse vectors and the L measurement vectors be arranged as columns of matrices $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_L]$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_L]$, respectively. In the MMV problem, $\mathbf{S}$ is to be reconstructed given $\mathbf{Y}$:

$$\mathbf{Y} = \mathbf{AS} \tag{4}$$

Generally, solving the MMV problem jointly can lead to a better reconstruction performance than that obtained by solving the SMV problem for each vector independently [4].

To find $\mathbf{S}$ in (4), there are different approaches in the literature including the greedy approach [5] where a non-optimal subset selection is performed, relaxed mixed norm minimization approach [6] where a convex optimization problem is solved and the Bayesian approach [7, 8, 9] where a prior is given, i.e., $\mathbf{Y}$ is observed and $\mathbf{S}$ is sparse, then a posterior distribution is estimated for the entries of $\mathbf{S}$ given the prior information.

Recently a number of methods based on Deep Learning [10, 11, 12, 13] have been proposed for the SMV problem in CS [14] and also for the MMV problem [15, 16]. These methods are data driven methods that exploit the structure of the sparse vector(s) and not only their sparsity. In [15], a Deep Stacking Network (DSN) [17] was used to extract the structure of the sparse vectors in $\mathbf{S}$. To find the parameters of the DSN, a Restricted Boltzmann Machine (RBM) [18] was used for pre-training and then fine tuning was performed. In [14], a feedforward neural network was used to solve the SMV problem. Similar to [15] (but assuming there is only one sparse vector in [15]), a pre-training phase followed by a fine tuning was used in [14]. For pre-training, the Stacked Denoising Auto-encoder (SDA) [19] has been used. Note that an RBM with Gaussian visible units and binary hidden units (i.e., the one used in [15]) has the same energy function as an auto-encoder with sigmoid hidden units and real valued observations [20]. Therefore the extension of [14] to the MMV problem is expected to give similar performance to that of [15].

In [16], a different type of MMV problem where the sparse vectors are not jointly sparse was studied. A deep architecture based on Long Short Term Memory (LSTM) was used to capture the dependency among sparsity patterns of different channels. These sparsity patterns are generally not the same. In this paper, we address the MMV problem where we assume the sparsity patterns in different channels are similar. For example, all channels are DCT or wavelet transforms of images. This type of MMV problem has wide practical applications. The specific structure in the problem calls for the use of a different kind of deep architecture, the convolutional deep stacking network. This architecture will be described in detail next.

2. THE PROPOSED METHOD

To capture the correlations amongst the columns of the unknown matrix $\mathbf{S}$, we propose a greedy reconstruction algorithm with two steps. In the first step, at each iteration of the reconstruction algorithm, and for each column of $\mathbf{S}$ represented as $\mathbf{s}_i$, we first predict the location of next non-zero entry, given the residuals of the sparse vectors (columns) inside the convolution window at that iteration. Then we add the predicted location to the support of $\mathbf{s}_i$. The definition of the residual matrix at the $j-$th iteration is $\mathbf{R}_j = \mathbf{Y} - \mathbf{A}\mathbf{S}_j$ where $\mathbf{S}_j$ is the estimate of the sparse matrix $\mathbf{S}$ at the $j-$th iteration. In the second step, we find the value of the non-zero entry. This can be done by solving a least squares problem that finds $\mathbf{s}_i$ given $\mathbf{y}_i$ and $\mathbf{A}_{\Omega_i}$. $\mathbf{A}_{\Omega_i}$ is a matrix that includes only those atoms (columns) of $\mathbf{A}$ that are members of the support of $\mathbf{s}_i$. To predict the location of next non-zero entry at each iteration using residual vectors as input, we propose the use of the Convolutional version of Deep Stacking Network (CDSN).

Since we do not know the sparsity level in advance, and also since at each iteration of the proposed method we predict the location of one of the non-zero entries, we need to generate residual vectors corresponding to different sparsity levels. To generate the training data, i.e., the "*(residual,sparse vector)*" pairs, assume that a sparse vector in the training data of the i-th channel $\mathbf{s}_i$ has K non-zero entries. We find the location of the largest entry of $\mathbf{s}_i$ and add it to the support set of the i-th channel. Assume that the index of this location is k_0. Then we set the k_0-th entry of $\mathbf{s}_i$ to zero. Now we find the residual vector where the support set of the i-th channel has only one member, which is k_0:

$$\mathbf{r}_i = \mathbf{y}_i - \mathbf{A}_{\Omega_i} s(k_0) \tag{5}$$

It is obvious that this residual results from not knowing the locations of the remaining $k-1$ non-zero entries in the i-th channel. From these $k-1$ non-zero entries, the maximum

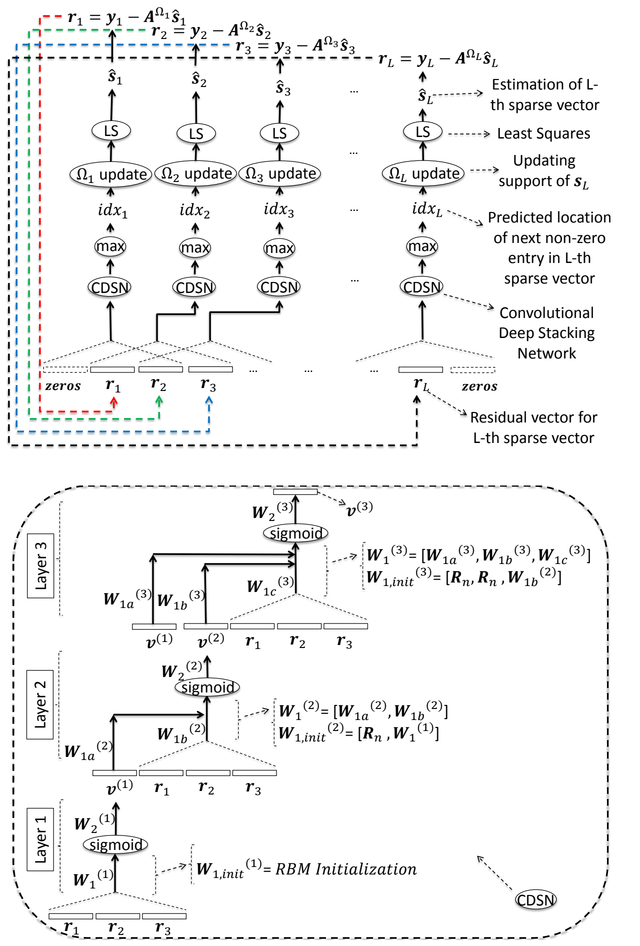
$r_1 = y_1 - A^{\Omega_1}\hat{s}_1$
$r_2 = y_2 - A^{\Omega_2}\hat{s}_2$
$r_3 = y_3 - A^{\Omega_3}\hat{s}_3$
$r_L = y_L - A^{\Omega_L}\hat{s}_L$
$\hat{s}_1$
$\hat{s}_2$
$\hat{s}_3$
$\hat{s}_L$
Estimation of L-th sparse vector
LS
Least Squares
Ω_1 update
Ω_2 update
Ω_3 update
Ω_L update
Updating support of s_L
idx_1
idx_2
idx_3
idx_L
Predicted location of next non-zero entry in L-th sparse vector
max
CDSN
Convolutional Deep Stacking Network
zeros
r_1
r_2
r_3
r_L
zeros
Residual vector for L-th sparse vector
Layer 3
$W_2^{(3)}$
$v^{(3)}$
sigmoid
$W_1^{(3)} = [W_{1a}^{(3)}, W_{1b}^{(3)}, W_{1c}^{(3)}]$
$W_{1,init}^{(3)} = [R_n, R_n, W_{1b}^{(2)}]$
$W_{1a}^{(3)}$
$W_{1b}^{(3)}$
$W_{1c}^{(3)}$
$v^{(1)}$
$v^{(2)}$
Layer 2
$W_2^{(2)}$
$W_1^{(2)} = [W_{1a}^{(2)}, W_{1b}^{(2)}]$
$W_{1,init}^{(2)} = [R_n, W_1^{(1)}]$
$W_{1a}^{(2)}$
$W_{1b}^{(2)}$
Layer 1
$W_2^{(1)}$
$W_1^{(1)}$
$W_{1,init}^{(1)}$ = RBM Initialization
CDSN

Fig. 1. Up: Block diagram of the proposed method with window size 3. Bottom: Block diagram of the convoltional deep stacking network with 3 layers. This diagram shows CDSN for channel 2, and $\mathbf{R}_n$ denotes a random matrix.

contribution to the residual in (5) is from the second largest entry in $\mathbf{s}_i$. Assume that it is the k_1-th entry of $\mathbf{s}_i$. We normalize all entries in $\mathbf{s}_i$ with respect to the value in the k_1-th entry. Therefore the training pair is $\mathbf{r}_i$ in (5) as input and the normalized $\mathbf{s}_i$ with k_0-th entry set to zero as target. We continue this procedure up to the point where $\mathbf{s}_i$ does not have any non-zero entry. Then we continue with the next training sample. We do the same procedure for each channel.

The training of CDSN is done only once. Then the trained network is used in the reconstruction algorithm. The block diagram of the proposed method is presented in Fig. 1 where the forward pass for the l-th layer is:

$$\mathbf{h}^{(l)} = \frac{1}{1 + e^{-\mathbf{W}_1^{(l)}\mathbf{z}^{(l)}}}$$
$$\mathbf{v}^{(l)} = [\mathbf{W}_2^{(l)}]^T \mathbf{h}^{(l)} \tag{6}$$

where $\mathbf{v}^{(l)}$ is the output and $\mathbf{z}^{(l)}$ is the input of l-th layer and is defined as follows:

$$\mathbf{z}^{(l)} = [\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(l-1)}, \mathbf{r}] \tag{7}$$

In (7), $\mathbf{r}$ is the concatenation of all residual vectors in each convolution window.

To find the CDSN unknown parameters $\mathbf{W}_1^{(l)}$ and $\mathbf{W}_2^{(l)}$ for each layer, l, We solve the following problem:

$$\{\mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}\} = \underset{\{\mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}\}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{V}^{(l)} - \mathbf{T}\|_2^2 \tag{8}$$

where $\mathbf{T}$ is a matrix whose columns are target vectors in the training set and $\mathbf{V}^{(l)}$ is a matrix whose columns are the corresponding output vectors from l-th layer. Please note that in CDSN, similar to DSN discussed in [17], the problem in (8) is solved for each layer separately. Each layer of CDSN in Fig. 1 is a neural network with a non-linear hidden layer and a linear output layer. The linearity of the output layer makes it possible to calculate a closed form formulation for $\mathbf{W}_2^{(l)}$ given $\mathbf{W}_1^{(l)}$ and $\mathbf{T}$ [17]:

$$\mathbf{W}_2^{(l)} = \left[\mu\mathbf{I} + \mathbf{H}^{(l)}[\mathbf{H}^{(l)}]^T\right]^{-1}\mathbf{H}^{(l)}\mathbf{T}^T \tag{9}$$

where $\mathbf{H}^{(l)}$ is a matrix whose columns are $\mathbf{h}^{(l)}$ in (6) corresponding to different training samples in the training set and μ is the regularization parameter. Now, given $\mathbf{W}_2^{(l)}$ in (9), and considering the fact that $\mathbf{W}_2^{(l)}$ also depends on $\mathbf{W}_1^{(l)}$, it can be shown [21] that the gradient of the cost function in (8) with respect to $\mathbf{W}_1^{(l)}$ is:

$$\begin{aligned}\frac{\partial\|\mathbf{V}^{(l)} - \mathbf{T}\|_2^2}{\partial\mathbf{W}_1^{(l)}} &= \mathbf{Z}^{(l)}\Bigg[[\mathbf{H}^{(l)}]^T \circ [1 - \mathbf{H}^{(l)}]^T \circ \\ &\Big[\left[\mathbf{H}^{(l)}\right]^{\dagger}\left[\mathbf{H}^{(l)}\mathbf{T}^T\right]\left[\mathbf{T}[\mathbf{H}^{(l)}]^{\dagger}\right] - \mathbf{T}^T\left[\mathbf{T}[\mathbf{H}^{(l)}]^{\dagger}\right]\Big]\Bigg]\end{aligned} \tag{10}$$

where $\mathbf{Z}^{(l)}$ is a matrix whose columns are $\mathbf{z}^{(l)}$ in (7) corresponding to different training samples in the training set and $\circ$ is the Hadamard product operator.

For each layer, we use stochastic gradient descent to find $\mathbf{W}_1^{(l)}$. Usually exploiting the past gradient information can improve the convergence speed in convex problems [22]. Although the problem in (8) due to stack of non-linear hidden layers is not necessarily convex, but we experimentally found out that the past gradient information can be helpful here as well. Similar to [21], we use the FISTA algorithm to accelerate fine tuning. Therefore, the update equations for $\mathbf{W}_1^{(l)}$ at k-th iteration is as follows:

$$\begin{aligned}\mathbf{W}_{1,k}^{(l)} &= \hat{\mathbf{W}}_{1,k}^{(l)} - \rho\frac{\partial\|\mathbf{V}^{(l)} - \mathbf{T}\|_2^2}{\partial\hat{\mathbf{W}}_{1,k}^{(l)}} \\ m_{k+1} &= \frac{1}{2}(1 + \sqrt{1 + 4m_k^2}) \\ \hat{\mathbf{W}}_{1,k+1}^{(l)} &= \hat{\mathbf{W}}_{1,k}^{(l)} + \frac{m_{k-1}}{m_{k+1}}(\mathbf{W}_{1,k}^{(l)} - \mathbf{W}_{1,k-1}^{(l)})\end{aligned} \tag{11}$$

After finding $\mathbf{W}_1^{(l)}$ from above update equations, we use the closed form formulation in (9) to find $\mathbf{W}_2^{(l)}$.

Since the problem is not necessarily convex and we have not used a large dataset for training, initialization of $\mathbf{W}_1^{(l)}$ is important. We used Restricted Boltzmann Machine (RBM)[18] for initialization of the input weights in the first layer as proposed and shown to be helpful in [23] and [17]. Similar to DSN, in CDSN we fix the learned parameters of each layer when training of that layer is done. Then we initialize the input weights of the upper layer, i.e., $\mathbf{W}_1^{(l+1)}$, with learned input weights of the lower layer, i.e., with $\mathbf{W}_1^{(l)}$. We also concatenate all sparse channels in the convolution window with outputs of all lower layers and use it as input to upper layer of CDSN. This is shown in the block diagram of CDSN in Fig. 1.

After finding the parameters of CDSN, we use the algorithm presented in Algorithm 1 to find the sparset solution $\mathbf{S}$ given $\mathbf{Y}$ and $\mathbf{A}$ in (4). We refere to this algorithm as CDSN-CS since we use Convolutional DSN to find the sparse solution for compressive sensing with multiple measurement vectors.

Algorithm 1 Distributed Compressive Sensing using Covolutional Deep Stacking Network (CDSN-CS)

Inputs: CS measurement matrix $\mathbf{A} \in \Re^{M\times N}$; matrix of measurements $\mathbf{Y} \in \Re^{M\times L}$; minimum ℓ_2 norm of residual matrix "$resMin$" as stopping criterion; Trained "$cdsn$" model; Convolution window size "w"
Output: Matrix of sparse vectors $\hat{\mathbf{S}} \in \Re^{N\times L}$
Initialization: $\hat{\mathbf{S}} = 0$; $j = 1$; $i = 1$; $\Omega = \emptyset$; $\mathbf{R} = \mathbf{Y}$.

```
 1: procedure CDSN-CS($\mathbf{A}$,$\mathbf{Y}$, $cdsn$)
 2:     while $i \le N$ and $\|\mathbf{R}\|_2 \le resMin$ do
 3:         $i \leftarrow i + 1$
 4:         for $j = 1 \rightarrow L$ do
 5:             $\mathbf{R}(:,j)_i \leftarrow \frac{\mathbf{R}(:,j)_{i-1}}{max(|\mathbf{R}(:,j)_{i-1}|)}$
 6:             $\mathbf{v}_j \leftarrow$
    $cdsn([\mathbf{R}(:,j-\frac{w}{2})_i, \mathbf{R}(:,j-\frac{w}{2}+1)_i, \dots, \mathbf{R}(:,j+\frac{w}{2}-1)_i, \mathbf{R}(:,j+\frac{w}{2})_i])$
 7:             $idx \leftarrow Support(max(\mathbf{v}_j))$
 8:             $\Omega_i \leftarrow \Omega_{i-1} \cup idx$
 9:             $\hat{\mathbf{S}}^{\Omega_i}(:,j) \leftarrow (\mathbf{A}^{\Omega_i})^{\dagger}\mathbf{Y}(:,j)$          ▷ Least Squares
10:             $\hat{\mathbf{S}}^{\Omega_i^C}(:,j) \leftarrow 0$
11:             $\mathbf{R}(:,j)_i \leftarrow \mathbf{Y}(:,j) - \mathbf{A}^{\Omega_i}\hat{\mathbf{S}}^{\Omega_i}(:,j)$
12:         end for
13:     end while
14: end procedure
```

3. EXPERIMENTAL EVALUATION AND CONCLUSIONS

We performed experiments on three different classes of images from a natural image dataset provided by Microsoft Research in Cambridge [24]. This was to evaluate the performance of different reconstruction algorithms for the MMV problem, including the proposed method when DCT or Wavelet transform were applied on images. We also compared the CPU time of these methods.

The reconstruction error was defined as $MSE = \frac{\|\hat{\mathbf{S}}-\mathbf{S}\|}{\|\mathbf{S}\|}$, where $\mathbf{S}$ is the actual sparse matrix and $\hat{\mathbf{S}}$ is the recovered sparse matrix from random measurements by the reconstruction algorithm. The machine used to perform the experiments has an Intel(R) Core(TM) i7 CPU with clock 2.93 GHz and with 16 GB RAM.

Ten randomly selected test images belonging to three classes of this dataset (flowers, buildings, cows) were used for experiments. We have used 64×64 images. Each image was divided into 8×8 blocks. After reconstructing all blocks of an image in the decoder, the MSE for the reconstructed image is calculated. The task was to simultaneously encode 8 blocks ($L = 8$) of an image and reconstruct them in the decoder. This meant that $\mathbf{S}$ in (4) had 8 columns each one having $N = 64$ entries. We used 40% measurements, thus $\mathbf{Y}$ in (4) had 8 columns each one having $M = 25$ entries. The encoder was a typical compressive sensing encoder, i.e., a randomly generated matrix $\mathbf{A}$. We normalized each column of $\mathbf{A}$ to have unit norm. To simulate the measurement noise, we added a Gaussian noise with standard deviation 0.005 to the measurement matrix $\mathbf{Y}$ in (4).

We compared the performance of the proposed algorithm, CDSN-CS, with BCS[25] applied per channel, SOMP [5], MT-BCS[8], T-SBL[9] and NWSOMP[15]. For each class of images, we used just 55 images for training set and 5 images for validation set which do not include any of 10 images used for test. We used 25 epochs for training the CDSN-CS. The experiments were performed for two popular transforms, DCT and Wavelet, for all of the above reconstruction algorithms. For the wavelet transform we used Haar wavelet transform with 3 levels of decomposition. For CDSN we used 3 layers and 64 neurons per layer and a window size of 5. For NWSOMP we used 3 layers and 512 neurons per layers. For all other methods, we used the MATLAB codes released by the authors and the settings they recommended. Due to lack of space, only the results for one class of images, i.e., cows, are presented. Results for DCT transform and wavelet transform are shown in Fig. 2. The results from the other two classes of images are similar to what is presented here.

As observed in Fig.2, CDSN-CS outperforms the other methods discussed in this paper for almost all sparsity levels. For computational efficiency, since all methods are implemented in MATLAB and run on the same machine, the CPU time shown in Fig.2 demonstrates that the proposed method is substantially faster than the Bayesian methods discussed in this paper and is almost as fast as the greedy method SOMP.

In conclusion, this paper presents a method based on convolutional deep stacking networks to reconstruct sparse vectors in the MMV problem. The convolution window helps to capture the correlation among different sparse vectors. We experimentally showed that the proposed method outperforms the popular SOMP and Bayesian methods for the MMV problem. We also showed that it is almost as fast as greedy methods. Our future work includes applying the proposed method for distributed compressive sensing of different frames in video and different channels of recorded electroencephalogram (EEG) signals where sparse vectors are highly correlated.

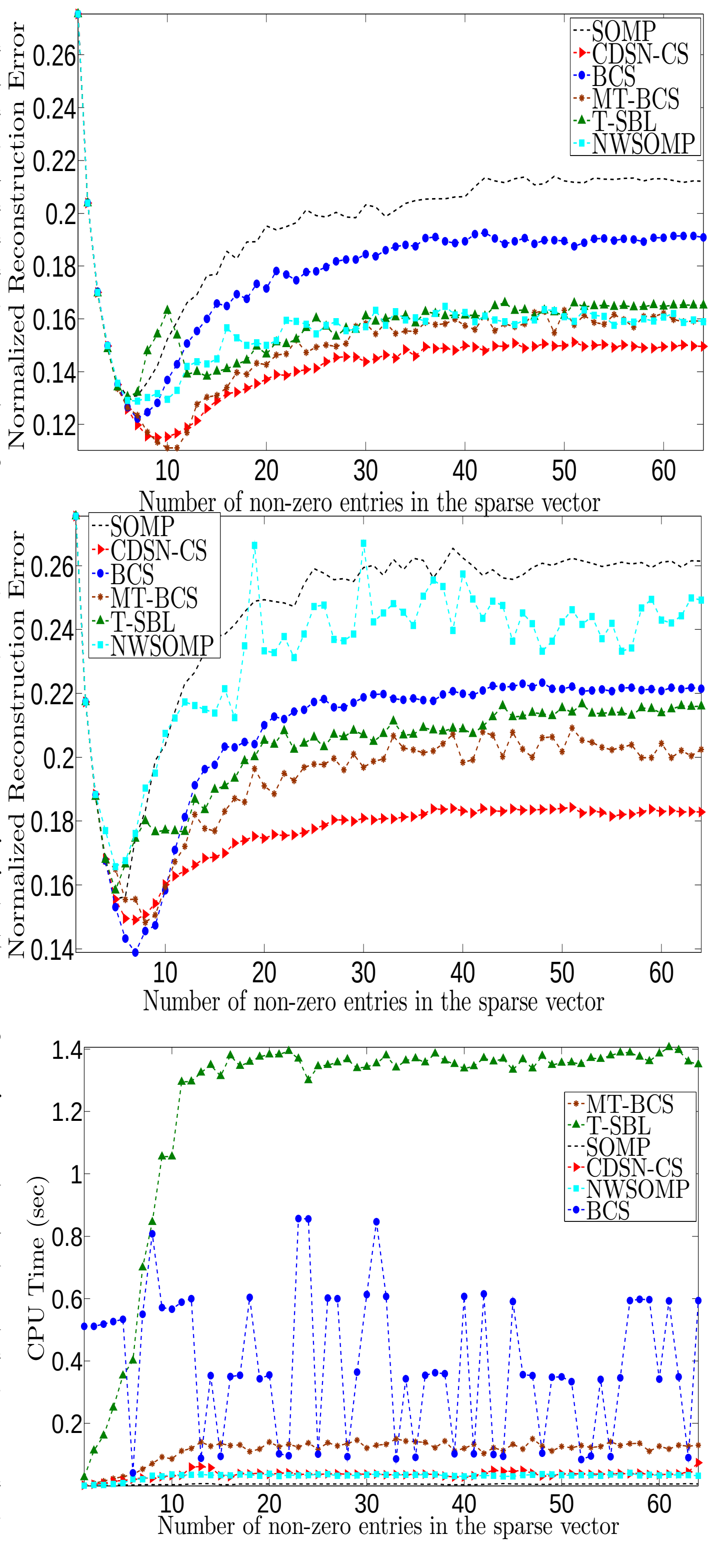

Fig. 2. Up: Comparative reconstruction performance using DCT transform. Middle: Reconstruction performance using Wavelet transform. Bottom: CPU time. Note that the time is reported for T-MSBL which is a faster version of T-SBL.

4. REFERENCES

[1] D.L. Donoho, “Compressed sensing,” *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289 –1306, april 2006.

[2] E. Candes, J. Romberg, and T. Tao, “Stable signal recovery from incomplete and inaccurate measurements,” *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[3] R.G. Baraniuk, “Compressive sensing [lecture notes],” *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118 –121, july 2007.

[4] Y.C. Eldar and H. Rauhut, “Average case analysis of multichannel sparse recovery using convex relaxation,” *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 505 –519, Jan. 2010.

[5] J.A. Tropp, A.C. Gilbert, and M.J. Strauss, “Algorithms for simultaneous sparse approximation. part I: Greedy pursuit,” *Signal Processing*, vol. 86, no. 3, pp. 572 – 588, 2006.

[6] J.A. Tropp, “Algorithms for simultaneous sparse approximation. part II: Convex relaxation,” *Signal Processing*, vol. 86, no. 3, pp. 589 – 602, 2006.

[7] David P Wipf and Bhaskar D Rao, “An empirical bayesian strategy for solving the simultaneous sparse approximation problem,” *Signal Processing, IEEE Transactions on*, vol. 55, no. 7, pp. 3704–3716, 2007.

[8] Shihao Ji, David Dunson, and Lawrence Carin, “Multitask compressive sensing,” *Signal Processing, IEEE Transactions on*, vol. 57, no. 1, pp. 92–106, 2009.

[9] Zhilin Zhang and Bhaskar D Rao, “Sparse signal recovery with temporally correlated source vectors using sparse bayesian learning,” *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 5, pp. 912–926, 2011.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[11] Dong Yu and Li Deng, “Deep learning and its applications to signal and information processing [exploratory dsp],” *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 145 –154, jan. 2011.

[12] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, November 2012.

[13] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward, “Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2016, To appear.

[14] Ali Mousavi, Ankit B. Patel, and Richard G. Baraniuk, “A deep learning approach to structured signal recovery,” *arXiv*, vol. abs/1508.04065, 2015, http://arxiv.org/abs/1508.04065.

[15] Hamid Palangi, Rabab Ward, and Li Deng, “Using deep stacking network to improve structured compressed sensing with multiple measurement vectors,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[16] Hamid Palangi, Rabab Ward, and Li Deng, “Distributed compressive sensing: A deep learning approach,” *arXiv*, vol. abs/1508.04924, 2015, http://arxiv.org/abs/1508.04924.

[17] L. Deng, D. Yu, and J. Platt, “Scalable stacking and learning for building deep architectures,” in *Proc. ICASSP*, march 2012, pp. 2133 –2136.

[18] Geoffrey E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002.

[19] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, “Extracting and composing robust features with denoising autoencoders,” *ICML*, pp. 1096–1103, 2008.

[20] Pascal Vincent, “A connection between score matching and denoising autoencoders,” *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, July 2011.

[21] Dong Yu and Li Deng, “Efficient and effective algorithms for training single-hidden-layer neural networks,” *Pattern Recognition Letters*, pp. 554–5580, 2012.

[22] Amir Beck and Marc Teboulle., *Gradient-based algorithms with applications to signal-recovery problems*, Cambridge University Press, 2009.

[23] G. Hinton and R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[24] Microsoft Research, “http://research.microsoft.com/en-us/projects/objectclassrecognition,” .

[25] Shihao Ji, Ya Xue, and Lawrence Carin, “Bayesian compressive sensing,” *Signal Processing, IEEE Transactions on*, vol. 56, no. 6, pp. 2346–2356, 2008.