RECONSTRUCTION OF SPARSE VECTORS IN COMPRESSIVE SENSING WITH MULTIPLE MEASUREMENT VECTORS USING BIDIRECTIONAL LONG SHORT-TERM MEMORY

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

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

ABSTRACT

In this paper we address the problem of compressive sensing with multiple measurement vectors. We propose a reconstruction algorithm which learns sparse structure inside each sparse vector and among sparse vectors. The learning is based on a cross entropy cost function. The model is the Bidirectional Long Short-Term Memory that is deep in time. All modifications are done at the decoder so that the encoder remains the general compressive sensing encoder, i.e., wide random matrix. Through numerical experiments on a real world dataset, we show that the proposed method outperforms the traditional greedy algorithm SOMP as well as a number of model based Bayesian methods including Multitask Compressive Sensing and Compressive Sensing with Temporally Correlated Sources. We emphasize that since the proposed method is a learning based method, its performance depends on the availability of training data. Nevertheless, in many applications huge dataset of offline training data is usually available.[1]

Index Terms— Deep Learning, Compressive Sensing, Sparse Reconstruction.

1. INTRODUCTION

Compressive Sensing (CS) [1] is a framework where both sensing and compression are performed at the same time. This has been made possible by exploiting the sparsity of a signal, either in time or spatial domain, or in a transform domain like DCT or Wavelet. Given the fact that there are many natural signals that are sparse in one of the above domains, CS has found numerous applications.

In compressive sensing with one measurement vector, instead of acquiring N samples of a signal $\mathbf{x} \in \Re^{N\times 1}$, M random measurements are acquired where $M < N$:

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

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

$$\mathbf{y} = \mathbf{\Phi}(\mathbf{\Psi s}) = \mathbf{As} \tag{2}$$

where $\mathbf{A} = \mathbf{\Phi\Psi}$. This problem is also called the Single Measurement Vector (SMV) problem.

In the Multiple Measurement Vectors (MMV) problem, L measurement vectors $\{\mathbf{y}_i\}_{i=1,2,\ldots,L}$ are given.The goal is to jointly reconstruct L sparse vectors $\{\mathbf{s}_i\}_{i=1,2,\ldots,L}$ from the measurement vectors. Let the L sparse vectors and the L measurement vectors be arranged as columns of matrices $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_L]$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L]$, respectively. In the MMV problem, $\mathbf{S}$ is to be reconstructed given $\mathbf{Y}$ and $\mathbf{A}$:

$$\mathbf{Y} = \mathbf{AS} \tag{3}$$

To reconstruct $\mathbf{S}$ in (3), there are a number of approaches in the literature, the greedy approach [2] where a subset selection is performed that is not necessarily optimal, the relaxed mixed norm minimization approach [3] where by relaxing $\ell_{0,p}$ to $\ell_{1,p}$, a convex problem is solved and the Bayesian approach [4, 5, 6] where given a prior, i.e., $\mathbf{Y}$ is observed and $\mathbf{S}$ is sparse, a posterior distribution is estimated for the entries of $\mathbf{S}$.

Recently a number of methods based on Deep Learning [7, 8, 9, 10, 11, 12] have been proposed for the MMV problem [13, 14, 15, 16]. These methods are learning based methods that exploit the structure of the sparse vectors as well as their sparsity. In [13], 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) was used for pre-training and then fine tuning was performed. In [14], 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)[18] was used to capture the dependency among sparsity patterns of different channels.

In this paper, we address the MMV problem where the sparsity patterns in different channels are not much different. For example, all channels are DCT or Wavelet transforms of images. This problem has wide practical applications. Given

[1]This work was made possible by NPRP grant 7-684-1-127 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

the fact that in the MMV problem, usually all measurement vectors are given, we can use both past and future information about the structure of the sparse vectors in $\mathbf{S}$. This means that we can perform support prediction for a given column of $\mathbf{S}$, based on both previous columns and future columns. This calls for a bidirectional learning architecture. We propose a sparse reconstruction algorithm that addresses this problem. We experimentally show that by using a bidirectional learning method in the proposed reconstruction algorithm, the proposed method outperforms [5, 6, 13] and [14].

2. THE PROPOSED METHOD

As a high level picture, we can think of a greedy sparse reconstruction problem as a classification task plus a least squares step. Different classes in this task are different atoms of a dictionary, i.e., different columns of the matrix $\mathbf{A}$ in (3). The goals of a sparse reconstruction method are: (i): to find the "*correct*" subset of classes (atoms) and report them as the support of the reconstructed sparse vector(s). (ii): find the values of corresponding entries in sparse vector(s). In many applications a huge amount of data is available. For example, an offline dataset, recorded image frames of a security camera, recorded biological signals of a patient over time. Now the question is, considering the recent success of deep learning methods in difficult classification tasks on huge datasets [7, 19, 9], can we use these methods to extract high level features for above sparse reconstruction classification task?

We continue the presentation using the diagram of the proposed method in Fig. 1. If we detect the classes (the non-zero entries) one by one, we can use the remaining residuals after finding each class (non-zero entry) as an appropriate input to a deep model for feature extraction. The extracted feature vectors are represented by $\{\overrightarrow{\mathbf{v}_1}, \overrightarrow{\mathbf{v}_2}, \ldots, \overrightarrow{\mathbf{v}_L}\}$ for left-to-right model and $\{\overleftarrow{\mathbf{v}_1}, \overleftarrow{\mathbf{v}_2}, \ldots, \overleftarrow{\mathbf{v}_L}\}$ for right-to-left model in Fig. 1.

Since we are interested in reconstructing multiple sparse vectors, it is desirable to have a model that generates high level feature vectors by treating residual vectors of different sparse vectors as a sequence. A good candidate for this sequence model is Long Short-Term Memory (LSTM) [18] given its recent success in difficult sequence modeling tasks [20, 21]. Since in the MMV problem, we usually have access to measurement vectors from the previous and future columns of $\mathbf{Y}$ in (3), it is more efficient to use a bidirectional model that captures the features from both left and right columns. Feature vectors from both directions can be concatenated for the next step of the algorithm.

We initialize the residual vector, $\mathbf{r}$, for each channel by the measurement vector, $\mathbf{y}$, of that channel. These residual vectors, represented as $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_L$ in Fig. 1, serve as the inputs to the bidirectional LSTM model. The bidirectional LSTM model captures features of the residual vectors using input weight matrices ($\mathbf{W}_1$,$\mathbf{W}_2$,$\mathbf{W}_3$,$\mathbf{W}_4$) as well as the dependency among the residual vectors using recurrent weight matrices ($\mathbf{W}_{rec1}$,$\mathbf{W}_{rec2}$,$\mathbf{W}_{rec3}$,$\mathbf{W}_{rec4}$) and the central memory unit shown in Fig. 1. A transformation matrix $\mathbf{U}$ is then used to transform, $[\overrightarrow{\mathbf{v}}, \overleftarrow{\mathbf{v}}]^T \in \Re^{2ncell\times 1}$, the outputs of each memory cell after gating for both left-to-right and right-to-left models, into the sparse vectors space, i.e., $\mathbf{z} \in \Re^{N\times 1}$. "$ncell$" is the number of cells in the LSTM model. Then a softmax layer is used for each channel to find the probability of each entry of each sparse vector being non-zero. For example, for channel 1, the j-th output of the softmax layer is:

$$P(s_1(j)|\mathbf{r}_1) = \frac{e^{z(j)}}{\sum_{k=1}^{N} e^{z(k)}} \tag{4}$$

Then for each channel, the entry with the maximum probability value is selected and added to the support set of that channel. After that, given the new support set, the following least squares problem is solved to find an estimate of the sparse vector for the j-th channel:

$$\hat{\mathbf{s}}_j = \underset{\mathbf{s}_j}{\operatorname{argmin}} \|\mathbf{y}_j - \mathbf{A}^{\Omega_j}\mathbf{s}_j\|_2^2 \tag{5}$$

where Ω_j is the support set of the j-th channel. Using $\hat{\mathbf{s}}_j$, the new residual value for the j-th channel is calculated as follows:

$$\mathbf{r}_j = \mathbf{y}_j - \mathbf{A}^{\Omega_j}\hat{\mathbf{s}}_j \tag{6}$$

This residual serves as the input to the bidirectional LSTM model at the next iteration of the algorithm. The pseudo-code of the proposed method is presented in Algorithm 1.

Algorithm 1 Pseudo-code of the Proposed Method

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 "$lstm$" model
Output: Matrix of sparse vectors $\hat{\mathbf{S}} \in \Re^{N\times L}$
Initialization: $\hat{\mathbf{S}} = 0; j = 1; i = 1; \Omega = \emptyset; \mathbf{R} = \mathbf{Y}$.

procedure LSTM-CS($\mathbf{A}$,$\mathbf{Y}$, $lstm$)
  while $i \leq N$ or $\|\mathbf{R}\|_2 \leq resMin$ **do**
    $i \leftarrow i + 1$
    for $j = 1 \rightarrow L$ **do**
      $\mathbf{R}(:,j)_i \leftarrow \frac{\mathbf{R}(:,j)_{i-1}}{max(|\mathbf{R}(:,j)_{i-1}|)}$
      $\mathbf{v}_j \leftarrow blstm(\mathbf{R}(:,j)_i, \mathbf{v}_{j-1}, \mathbf{c}_{j-1})$ ▷ Bidirectional LSTM
      $\mathbf{z}_j \leftarrow \mathbf{U}\mathbf{v}_j$
      $\mathbf{c} \leftarrow softmax(\mathbf{z}_j)$
      $idx \leftarrow Support(max(\mathbf{c}))$
      $\Omega_i \leftarrow \Omega_{i-1} \cup idx$
      $\hat{\mathbf{S}}^{\Omega_i}(:,j) \leftarrow (\mathbf{A}^{\Omega_i})^{\dagger}\mathbf{Y}(:,j)$ ▷ Least Squares
      $\hat{\mathbf{S}}^{\Omega_i^C}(:,j) \leftarrow 0$
      $\mathbf{R}(:,j)_i \leftarrow \mathbf{Y}(:,j) - \mathbf{A}^{\Omega_i}\hat{\mathbf{S}}^{\Omega_i}(:,j)$
    end for
  end while
end procedure

Please note that 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 will also 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 j-th channel $\mathbf{s}_j$ has K non-zero

entries. We find the location of the largest entry of $\mathbf{s}_j$ and add it to the support set of the j-th channel. Assume that the index of this location is k_0. Then we set the k_0-th entry of $\mathbf{s}_j$ to zero. Now we find the residual vector where the support set of the j-th channel has only one member, which is k_0:

$$\mathbf{r}_j = \mathbf{y}_j - \mathbf{A}_{\Omega_j} s(k_0) \tag{7}$$

It is obvious that this residual results from not knowing the locations of the remaining $k-1$ non-zero entries in the j-th channel. From these $k-1$ non-zero entries, the maximum contribution to the residual in (7) is from the second largest entry in $\mathbf{s}_j$. Assume that it is the k_1-th entry of $\mathbf{s}_j$. We normalize all entries in $\mathbf{s}_j$ with respect to the value in the k_1-th entry. Therefore the training pair is $\mathbf{r}_j$ in (7) as input and the normalized $\mathbf{s}_j$ with k_0-th entry set to zero as target. We continue this procedure up to the point where $\mathbf{s}_j$ does not have any non-zero entry. Then we continue with the next training sample. We do the same procedure for each channel.

To find the bidirectional LSTM parameters, i.e., $\mathbf{W}_i$, $\mathbf{W}_{reci}$, $\mathbf{b}_i$, $i = 1, 2, 3, 4$, we minimized a cross entropy cost function on the training pairs, i.e., the "*(residual,sparse vector)*" pairs. We used backpropagation through time and Adam [22] for minimizing the cost function. Please note that the training is done only once, after that the trained model is used in the reconstruction algorithm (Algorithm 1).

3. EXPERIMENTAL EVALUATION

We performed experiments on three different classes of images from a natural image dataset provided by Microsoft Research in Cambridge [23]. This was to evaluate the performance of different reconstruction algorithms for the MMV problem, including the proposed method when Wavelet or DCT transform were applied on images. We also compared the CPU time of these methods.

The reconstruction error was defined as:

$$NMSE = \frac{\|\hat{\mathbf{S}} - \mathbf{S}\|}{\|\mathbf{S}\|} \tag{8}$$

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.

Five randomly selected test images belonging to three classes of this dataset (flowers, buildings, cows) were used for test experiments. 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 5 images used for test. We resized images to 128×128 images. Each image was divided into 8×8 blocks. After reconstructing all blocks of an image in the decoder, the $NMSE$ for the reconstructed image was calculated. The task was to simultaneously encode 4 blocks ($L = 4$) of an image and reconstruct them in the decoder. This meant that $\mathbf{S}$ in (3) had 4 columns each one having $N = 64$ entries. We used 40% measurements, thus $\mathbf{Y}$ in (3) had 4 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 (3).

We compared the performance of the proposed algorithm, BLSTM-CS, with SOMP [2], MT-BCS[5], T-SBL[6], NWSOMP[13] and LSTM-CS[14]. For MT-BCS we set the parameters of the Gamma prior on noise variance to $a = 100/0.1$ and $b = 1$ which are the values suggested by the authors. We set the stopping threshold to 10^{-8} as well. For T-SBL, we used the default values proposed by the authors. We used T-MSBL which is a faster version of T-SBL. For NWSOMP, during training, we used one layer, 512 neurons and 15 epochs of parameters update. 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 both LSTM-CS and BLSTM-CS, we used a small model with 16 cells. For NWSOMP we used 3 layers and 512 neurons per layers. Due to lack of space, only the results for one class of images, i.e., buildings, are presented. To monitor and prevent overfitting, we used 5 images per channel as the validation set and we used early stopping if necessary. Please note that the images used for validation were not used in the training set or in the test set. 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, BLSTM-CS outperforms the other methods discussed in this paper for different sparsity levels. To evaluate run time of different methods, considering the fact that 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 faster than the Bayesian methods discussed in this paper and is almost as fast as the greedy method SOMP.

4. CONCLUSION

In this paper, a sparse reconstruction method for the MMV problem was proposed. The main difference of the proposed approach with other methods was that it was a learning based method that exploit the dependencies among sparse vectors in both directions. Through numerical experiments, it was shown that the proposed method outperforms the traditional greedy methods, a number of model based Bayesian methods and two of the previous learning based methods. Our future work is to use the proposed method for applications where there is high correlations among sparse vectors, e.g., compressive sensing of electroencephalogram (EEG) signals for health telemonitoring and compressive sensing of different frames in a video.

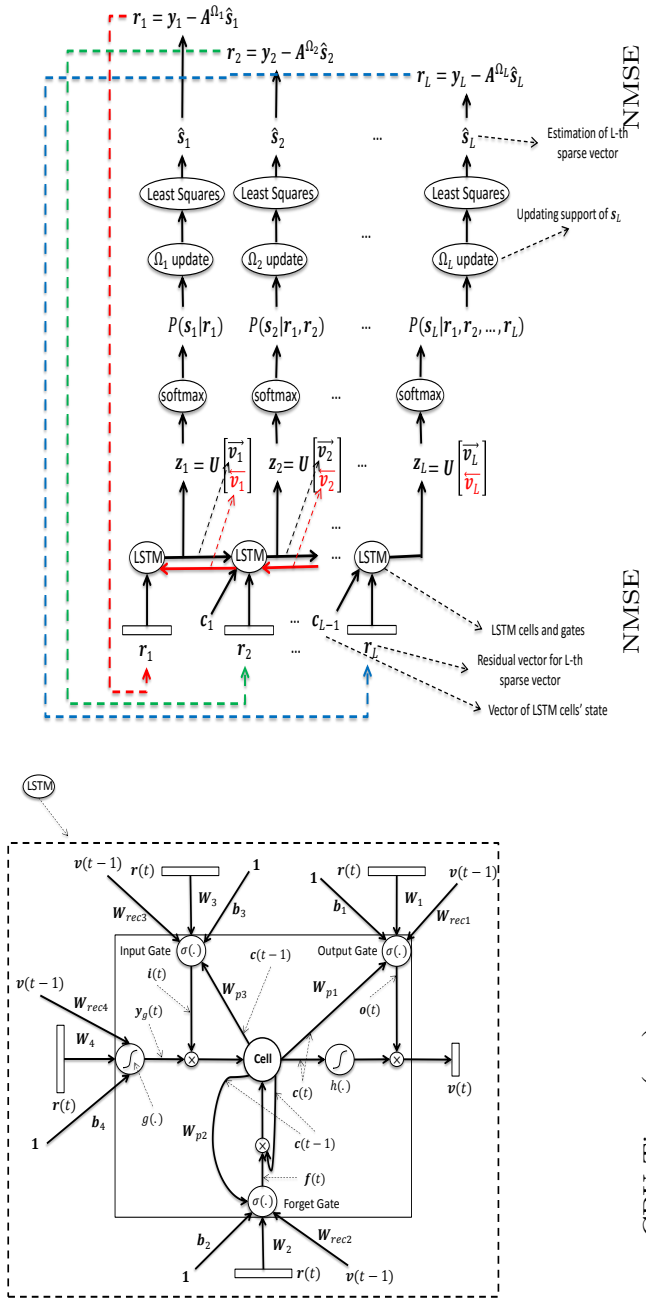

Fig. 1. Up: Block diagram of the proposed method unfolded over channels. Bottom: Block diagram of the Long Short-Term Memory (LSTM).

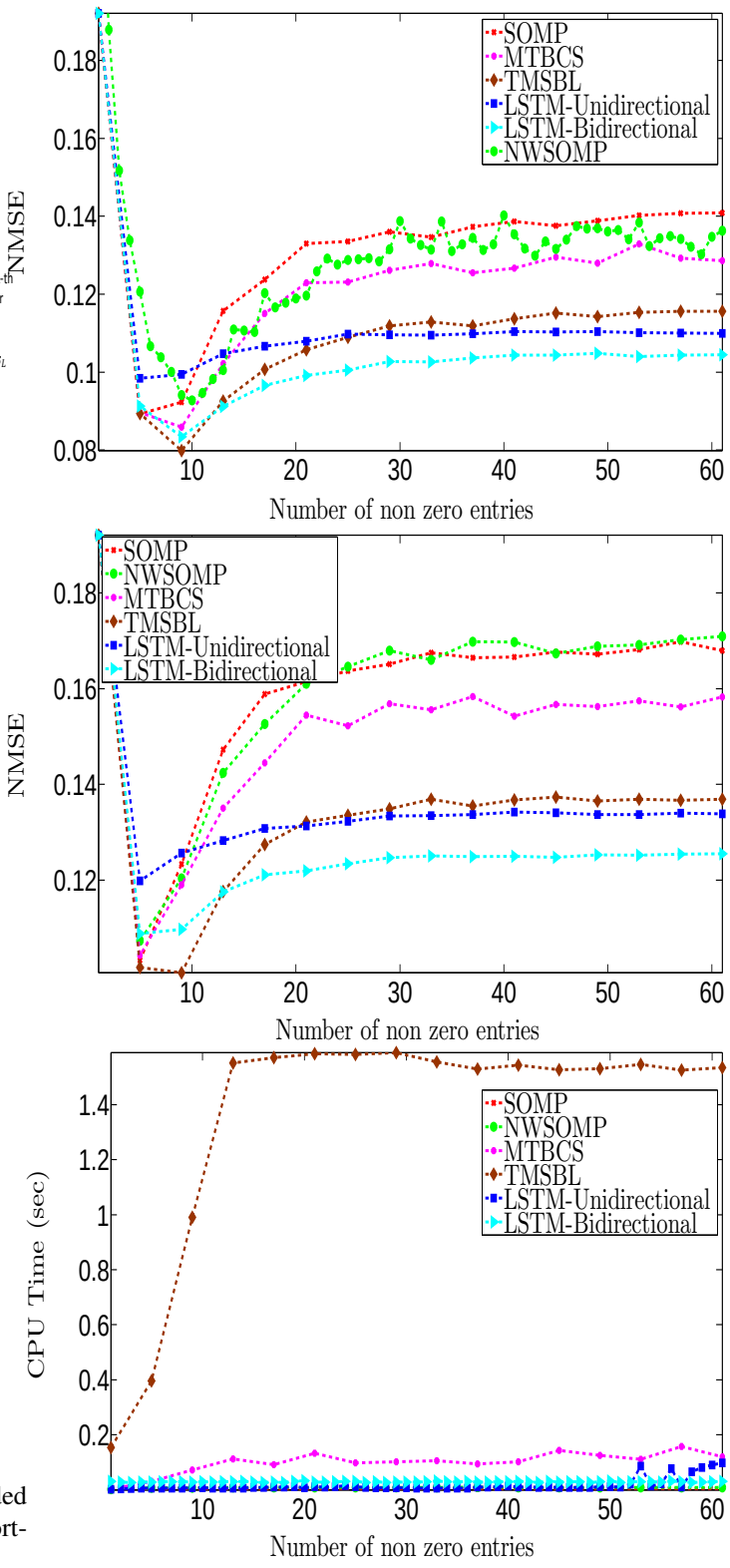

Fig. 2. Up: Comparative reconstruction performance using DCT transform. Middle: Reconstruction performance using Wavelet transform. Bottom: CPU time. Note that the time is reported for T-MSBL which is a faster version of T-SBL.

5. REFERENCES

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

[2] 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.

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

[4] 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.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. 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*, vol. 24, no. 4, pp. 694–707, April 2016.

[11] H. Palangi, Li Deng, and R.K. Ward, "Recurrent deep-stacking networks for sequence classification," in *Signal and Information Processing (ChinaSIP), 2014 IEEE China Summit International Conference on*, July 2014, pp. 510–514.

[12] H. Palangi, L. Deng, and R. Ward, "Learning input and recurrent weight matrices in echo state networks," in *NIPS Workshop on Deep Learning*, December 2013, http://research.microsoft.com/apps/pubs/default.aspx?id=204701.

[13] H. Palangi, R. Ward, and L. Deng, "Using deep stacking network to improve structured compressed sensing with multiple measurement vectors," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 3337–3341.

[14] H. Palangi, R. Ward, and L. Deng, "Distributed compressive sensing: A deep learning approach," *IEEE Transactions on Signal Processing*, vol. 64, no. 17, pp. 4504–4518, Sept 2016.

[15] H. Palangi, R. Ward, and L. Deng, "Exploiting correlations among channels in distributed compressive sensing with convolutional deep stacking networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 2692–2696.

[16] H. Palangi, R. Ward, and L. Deng, "Convolutional deep stacking networks for distributed compressive sensing," *Signal Processing*, vol. 131, pp. 181 – 189, 2016.

[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] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

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

[20] Hasim Sak, Andrew W. Senior, and Francoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Proc. Interspeech*, 2014.

[21] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *ICLR2015*, 2015, http://arxiv.org/abs/1409.0473.

[22] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2014, http://arxiv.org/abs/1412.6980.

[23] Microsoft Research, "http://research.microsoft.com/en-us/projects/objectclassrecognition," .