# USING DEEP STACKING NETWORK TO IMPROVE STRUCTURED COMPRESSED SENSING WITH MULTIPLE MEASUREMENT VECTORS

*Hamid Palangi, Rabab Ward*

Department of Electrical and Computer Engineering
The University of British Columbia

*Li Deng*

Microsoft Research
Redmond, WA, USA

**ABSTRACT**

We study the MMV (Multiple Measurement Vectors) compressive sensing setting with a specific sparse structured support. The locations of the non-zero rows in the sparse matrix are not known. All that is known is that the locations of the non-zero rows have probabilities that vary from one group of rows to another. We propose two novel greedy algorithms for the exact recovery of the sparse matrix in this structured MMV compressive sensing problem. The first algorithm models the matrix sparse structure using a shallow non- linear neural network. The input of this network is the residual matrix after the prediction and the output is the sparse matrix to be recovered. The second algorithm improves the shallow neural network prediction by using the stacking operation to form a deep stacking network. Experimental evaluation demonstrates the superior performance of both new algorithms over existing MMV methods. Among all, the algorithm using the deep stacking network for modelling the structure in MMV compressive sensing performs the best.

***Index Terms***— Compressed Sensing, Multiple Measurement Vectors, Deep Learning

## 1. INTRODUCTION

In the general framework of Compressed Sensing (CS) [1],[2],[3], $N$ samples of $\mathbf{x} \in \Re^{N\times 1}$ can be exactly reconstructed from $M < N$ linear random measurements. This is expressed as:

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

where $\mathbf{y} \in \Re^{M\times 1}$ is the measured vector and $\mathbf{\Phi} \in \Re^{M\times N}$ is a random measurement matrix. An important underlying assumption for the decoder to uniquely recover $\mathbf{x}$ given $\mathbf{y}$ and $\mathbf{\Phi}$, is that $\mathbf{x}$ is sparse in a given basis $\mathbf{\Psi}$. This basis can be complete, i.e., $\mathbf{\Psi} \in \Re^{N\times N}$, or over-complete, i.e., $\mathbf{\Psi} \in \Re^{N\times N_1}$ where $N < N_1$ (compressed sensing for over-complete dictionaries was introduced in [4]). More accurately, assuming a complete basis $\mathbf{\Psi} \in \Re^{N\times N}$, then

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

where $\mathbf{s}$ is $K - sparse$, i.e., $\mathbf{s}$ has at most $K$ non-zero elements. Combining (1) and (2) results in $\mathbf{y} = \mathbf{\Phi\Psi s}$. The uniqueness conditions for the exact recovery of $\mathbf{s}$ based on the $spark$ and the mutual coherence of $\mathbf{\Phi\Psi}$ (denoted by $\mu(\mathbf{\Phi\Psi})$) criteria are expressed by corollary 4 of [5] and theorem 4 in [6] respectively. As discussed in [5],

$$spark(\mathbf{\Phi\Psi}) \leq min\{N, rank(\mathbf{\Phi\Psi}) + 1\} \tag{3}$$

and since $rank(\mathbf{\Phi\Psi}) \leq M$, the corollary 4 of [5] implies that $M \geq 2K$ measurements are needed to guarantee unique recovery at the decoder. In theorem 4 of [6], since $\mu(\mathbf{\Phi\Psi}) \geq \frac{1}{\sqrt{M}}$ (approximation of lower bound of $\mu(\mathbf{\Phi\Psi})$ for $M \ll N$ presented in [6]), the uniqueness of the recovered $\mathbf{s}$ is guaranteed when $\sqrt{M} > (2K - 1)$. From the above, it is clear that the conditions to recover a unique $\mathbf{s}$ are more optimistic when stated in terms of $spark(\mathbf{\Phi\Psi})$ than in terms of mutual coherence.

In the Multiple Measurement Vectors (MMV) problem, a set of $N_2$ sparse vectors $\{\mathbf{s}_i\}_{i=1}^{N_2}$ stacked as columns of a matrix $\mathbf{S}$ is to be jointly recovered from a set of $N_2$ measurement vectors $\{\mathbf{y}_i\}_{i=1}^{N_2}$ stacked as columns of a matrix $\mathbf{Y}$. The other two assumptions in the MMV problem are the joint sparsity of $\mathbf{S}$, i.e., the indices of non-zero entries are the same for every column in $\mathbf{S}$ (all sparse vectors have the same support), and $\mathbf{S}$ is $K - row - sparse$, i.e. at most $K$ rows of $\mathbf{S}$ have non-zero energy. Exploiting the structure and other prior information (beyond that of sparsity) about each vector in $\mathbf{S}$ leads to solvers with higher performance. An overview of this topic is presented in section V of [6].

In this paper, we focus on solving the MMV problem when the locations of the non-zero entries in the the sparse matrix $\mathbf{S}$ are not known . However, we know that $\mathbf{S}$ is a jointly sparse matrix and has the following structure: 1) each vector is sparse but the probability of having non-zero elements in each vector differs from one region of the vector to another, 2) the probability distribution of the support of each vector is unknown and 3) the probability distribution of the locations of non-zero regions in the different vectors are independent of each other. We propose two greedy solvers for the MMV problem with the above structure. These solvers are shown to outperform current MMV solvers. The essence of the proposed algorithms is the incorporation of a set of weights that enhances the impact of the columns of $\mathbf{\Phi\Psi}$ that

have a higher probability of contributing to the actual support of $\mathbf{S}$. To calculate these weights, we propose two approaches, a non-linear approach and a stacking non-linear approach. The stacking non-linear approach can deal with the huge amount of data more efficiently. It uses the Restricted Boltzmann Machine (RBM) [7],[8],[9] to initialize the weights and then a Deep Stacking Network (DSN) [10] to fine tune the weights. We experimentally show that the performance of the second method is close to the bound predicted by corollary 4 in [5], i.e., $M \simeq 2K$.

## 2. PROPOSED METHODOLOGIES

In the first part of this section, we propose an efficient and fast greedy solver. This solver which we call Non-linear Weighted Simultaneous Orthogonal Matching Pursuit (NWSOMP) relies on a non-linear model of the structure in the support. In the second part, we argue that NWSOMP can not be scaled across machines for a large scale problem. To solve this problem, we develop a stacking non-linear version of NWSOMP that relies on a deep neural network and that is suitable for large scale data and is easy to parallelize across different machines.

### 2.1. Non-linear Weighted Simultaneous Orthogonal Matching Pursuit (NWSOMP)

In the MMV problem, we want to recover a jointly sparse matrix $\mathbf{S}$ from the matrix of measurements $\mathbf{Y} = \mathbf{AS}$ where $\mathbf{A} = \mathbf{\Phi\Psi}$ and $\mathbf{A} = [\mathbf{a}_1\, \mathbf{a}_2\, \ldots\, \mathbf{a}_N]$ and $\mathbf{a}_i$ is $i$-th column of $\mathbf{A}$. In the Simultaneous Orthogonal Matching Pursuit (SOMP) algorithm [11] to recover $\mathbf{S}$ exactly, the atom selection criterion in each iteration is based on the $q$-norm[1] of the correlation between the residual matrix $\mathbf{R}$ and each atom ($\mathbf{a}_i$) of dictionary $\mathbf{A}$, i.e., $\|\mathbf{a}_i^T\mathbf{R}\|_q$. For the Single Measurement Vector (SMV) problem, a method that embeds prior information using neural networks has been proposed in [12]. We first extend this approach to the MMV problem and then propose our method. Suppose that $\Omega$ is the set of locations of the non-zero rows of $\mathbf{S}$ ($|\Omega| = K$ for a $K - row - sparse$ matrix $\mathbf{S}$), then

$$\mathbf{Y} = \mathbf{A}^{\Omega}\mathbf{S}^{\Omega} \tag{4}$$

where $\mathbf{A}^{\Omega}$ is the matrix that includes only those columns of $\mathbf{A}$ whose indices are in $\Omega$ and $\mathbf{S}^{\Omega}$ is vector that includes only those rows of $\mathbf{S}$ whose indices are in $\Omega$. $\Omega$ is not known in advance and the greedy algorithm completes it column by column. Suppose that at iteration $j$, the set of the locations of the non-zero rows is $\Omega_j$. Since $\Omega = \Omega_j \cup (\Omega \setminus \Omega_j)$, then

$$\mathbf{Y} = \mathbf{A}^{\Omega_j}\mathbf{S}^{\Omega_j} + \mathbf{A}^{\Omega\setminus\Omega_j}\mathbf{S}^{\Omega\setminus\Omega_j} \tag{5}$$

[1] $q$-norm of vector $\mathbf{v} \in \Re^{n\times 1}$ is defined as $\sum_{i=1}^{n}(|v_i|^q)^{\frac{1}{q}}$.

and therefore the residual matrix $\mathbf{R}_j$ will be:

$$\mathbf{R}_j = \mathbf{Y} - \mathbf{Y}_j = \mathbf{A}^{\Omega\setminus\Omega_j}\mathbf{S}^{\Omega\setminus\Omega_j} + \mathbf{A}^{\Omega_j}(\mathbf{S}^{\Omega_j} - \mathbf{S}_j^{\Omega_j}) \tag{6}$$

where $\mathbf{S}^{\Omega\setminus\Omega_j}$ is the set of non-zero rows of $\mathbf{S}$ that have not been identified yet and $(\mathbf{S}^{\Omega_j} - \mathbf{S}_j^{\Omega_j})$ is the estimation error at the $j$-th iteration. Hence, a neural network constructed with a set of weights calculated in the training stage can be used in the selection step of SOMP. These weights enforce the structure of support of the sparse matrix $\mathbf{S}$ in SOMP. The input of this network is the vector $vec(\mathbf{R})$ and the output is $vec(\hat{\mathbf{S}}_j)$ (estimation of $\mathbf{S}$ at $j$-th iteration). We formulate this network as follows:

$$vec(\hat{\mathbf{S}}) = \mathbf{W}_2 f_h(\mathbf{W}_1 vec(\mathbf{R}) + \mathbf{b}_1) + \mathbf{b}_2 \tag{7}$$

Where $f_h(.)$ is the activation function of the hidden layer neurons, $\mathbf{W}_1 \in \Re^{numhid\times MN_2}$ is the weight matrix between the input layer and the hidden layer ($numhid$ is the number of neurons in the hidden layer), $\mathbf{W}_2 \in \Re^{NN_2\times numhid}$ is the weight matrix between the hidden layer and the output layer, $\mathbf{b}_1 \in \Re^{numhid\times 1}$ is vector of bias values for the hidden layer neurons and $\mathbf{b}_2 \in \Re^{NN_2\times 1}$ is vector of bias values for the output layer neurons.

To find weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$, the following minimization problem is solved:

$$\begin{aligned}&(vec(\mathbf{W}_1), vec(\mathbf{W}_2)) = \\ &\underset{(vec(\mathbf{W}_1),vec(\mathbf{W}_2))}{\operatorname{argmin}} \frac{1}{2T}\sum_{i=1}^{T}\|vec(\mathbf{S})_i^t - vec(\hat{\mathbf{S}})_i\|_2^2\end{aligned} \tag{8}$$

Where $T$ is number of training samples, $\mathbf{S}_i^t$ is the ideal output for a certain input $\mathbf{R}_i$, $\hat{\mathbf{S}}_i$ is the approximate output for input $\mathbf{R}_i$ and the operator $vec(.)$ returns the vectorized form of a matrix. In this paper, a weighted module refers to a trained one layer feedforward neural network whose inputs are the normalized residual vector(s) and outputs are the approximated sparse signal(s). Now, instead of selecting the column of $\mathbf{A}$ that corresponds to the maximum element of vector $\mathbf{c} = [\|\mathbf{a}_1^T\mathbf{R}\|_q \|\mathbf{a}_2^T\mathbf{R}\|_q \ldots \|\mathbf{a}_N^T\mathbf{R}\|_q]^T$ (as done in SOMP), the column of $\mathbf{A}$ corresponding to the row of $\hat{\mathbf{S}}$ with the maximum $\ell_q$ norm is selected. It is important to note that to estimate $\mathbf{W}_1$ and $\mathbf{W}_2$, $\mathbf{S}_i^t$ should be known in advance for each $\mathbf{R}_i$. In the other words, $\mathbf{S}_i^t$ for $i = 1, 2, \ldots, T$ are the targets of the neural networks that are generated based on the prior information about $\mathbf{S}$. The inputs ($\mathbf{R}_i$) are generated by imitating the greedy algorithm.

With linear activation functions for the hidden layer and output layer neurons, (7) becomes a simple linear relationship. A more efficient choice for $f_h(.)$ is a non-linear function like the sigmoid function $\frac{1}{1+e^{-t}}$. However with a non-linear $f_h(.)$, solving (8) becomes a non trivial task. Existing algorithms in the neural networks literature for finding $\mathbf{W_1}$ and $\mathbf{W_2}$, such as the backpropagation and conjugate gradient backpropagation are very slow and inefficient when $f_h(.)$

is non-linear. Since we use one hidden layer only, there are more efficient ways, like the one in [13], to calculate the weights. In [13], the fact that $\mathbf{W}_2$ is dependent on $\mathbf{W}_1$ is taken into account when calculating the gradient. This leads to the following expression of the gradient of the energy function $E = \|\mathbf{S}^t - \hat{\mathbf{S}}\|_F^2$:

$$\frac{\partial E}{\partial \mathbf{W_1}} = 2\mathbf{R}[\mathbf{H}^T \circ (1-\mathbf{H})^T \circ [\mathbf{H}^\dagger (\mathbf{H}[\mathbf{S}^t]^T)(\mathbf{S}^t\mathbf{H}^\dagger) - [\mathbf{S}^t]^T(\mathbf{S}^t\mathbf{H}^\dagger) \tag{9}$$

where $\mathbf{H}^\dagger = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}$, $\circ$ is the Hadamard product operator and $\mathbf{H}$ is the output of the hidden layer:

$$\mathbf{H} = \frac{1}{1 + exp(-\mathbf{W}_1\mathbf{x_{in}})} \tag{10}$$

where $\mathbf{x_{in}} = vec(\mathbf{R})$. Then we should search in the opposite direction of the gradient, i.e.,

$$\mathbf{W_1^{k+1}} = \mathbf{W_1^k} - \rho \frac{\partial E}{\partial \mathbf{W_1^k}} \tag{11}$$

where $\rho$ is the learning rate. After updating $\mathbf{W}_1$ using (11), $\mathbf{W}_2$ is calculated using following closed form formulation:

$$\mathbf{W_2} = (\mu\mathbf{I} + \mathbf{H}\mathbf{H}^T)^{-1}\mathbf{H}[\mathbf{S}^t]^T \tag{12}$$

where $\mathbf{I}$ is the identity matrix and $\mu$ is the regularization parameter. The proposed greedy solver based on the weights calculated using (9), (10), (11) and (12) is presented in Algorithm 1.

**Algorithm 1** Non-linear Weighted Simultaneous Orthogonal Matching Pursuit (NWSOMP)

**Inputs**:The CS measurement matrix $\mathbf{A} \in \Re^{M\times N}$, The matrix of measurments $\mathbf{Y} \in \Re^{M\times N_2}$, The minimum $\ell_2$ norm of the residual matrix ($ResMin$) as the stopping criterion
**Output**: The matrix of sparse vectors $\hat{\mathbf{S}} \in \Re^{N\times N_2}$
**Initialization**: $\hat{\mathbf{S}} = 0$; $\hat{\mathbf{S}}_i = 0$; $i = 1$; $\Omega_i = \emptyset$; $\mathbf{R}_i = \mathbf{Y}$
**procedure** NWSOMP(**A**,**Y**)
  **while** $\|\mathbf{R}_i\|_2 \le ResMin$ **do**
    $i \leftarrow i + 1$
    $\mathbf{R}_i \leftarrow \frac{\mathbf{R}_{i-1}}{max(vec(\mathbf{R}_{i-1}))}$ ▷ residual normalization
    $vec(\hat{\mathbf{S}}) = \mathbf{W}_2 f_h(\mathbf{W}_1 vec(\mathbf{R_i}) + \mathbf{b_1}) + \mathbf{b_2}$ ▷ weight module
    $\mathbf{c} \leftarrow [\|\hat{\mathbf{S}}_{r1}\|_q \|\hat{\mathbf{S}}_{r2}\|_q \dots \|\hat{\mathbf{S}}_{rN}\|_q]^T$
    $idx \leftarrow Support(max(\mathbf{c}))$
    $\Omega_i \leftarrow \Omega_{i-1} \cup idx$
    $\hat{\mathbf{S}}_i^{\Omega_i} \leftarrow (\mathbf{A}^{\Omega_i})^\dagger \mathbf{Y}$
    $\hat{\mathbf{S}}_i^{\Omega_i^C} \leftarrow 0$
    $\mathbf{R}_i \leftarrow \mathbf{Y} - \mathbf{A}^{\Omega_i}\hat{\mathbf{S}}_i^{\Omega_i}$
  **end while**
$\hat{\mathbf{S}} \leftarrow \hat{\mathbf{S}}_i$
**end procedure**

### 2.2. Stacking non-linear NWSOMP using Deep Stacking Network (DSN-WSOMP)

There is a main deficiency in the NWSOMP proposed in the previous section. When we deal with a large scale problem, it is not easy to calculate $\mathbf{W_1}$ and $\mathbf{W_2}$ based on the method presented in the previous section. It is, in fact, computationally difficult because the computation of $\mathbf{W_1}$ and $\mathbf{W_2}$ is not easy to parallelize across different machines. In this section, we present a stacking non-linear version of NWSOMP that is suitable for large scale data and is easy to parallelize across different machines.

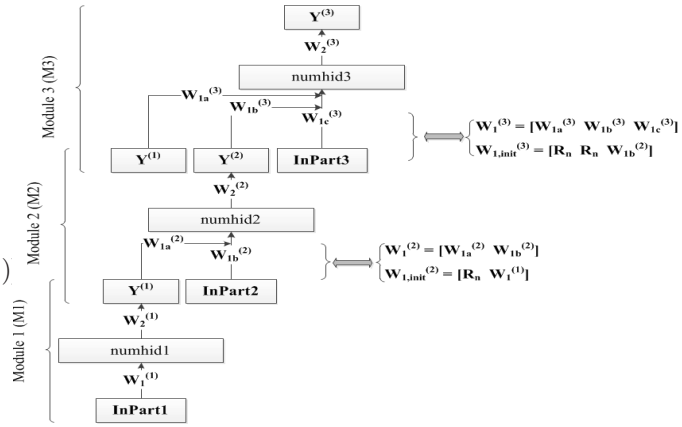


**Fig. 1**. Architecture of a deep stacking network with 3 modules, i.e., $D = 3$

To explain the stacking non-linear variant of NWSOMP, we should first describe the Deep Stacking Networks (DSN) introduced in [10]. A DSN is a layered and modular structure that mainly resolves the problem of difficult fine tuning in a Deep Neural Network (DNN) for large scale data. This difficulty arises from the fact that fine tuning a DNN requires performing stochastic gradient descent algorithm that is difficult to parallelize across machines. More details about DNNs can be found in [14],[15] and [16]. The structure of a sample DSN with three modules is illustrated in Fig. 1. In this figure, the input data is divided into 3 parts, denoted as $\mathbf{InPart1}$, $\mathbf{InPart2}$ and $\mathbf{InPart3}$ in Fig. 1. Each module of a DSN is a specialized one layer network and the output of each module forms part of the input of its upper module. Therefore, the dimension of the input of each module in a DSN is different from that of other modules. More precisely, the dimension of the input of the $j$-th module in a DSN with $D$ modules is:

$$n_j = n + m(j-1), \qquad j = 1, 2, \dots, D \tag{13}$$

Therefore for each module, the dimension of its input and of $\mathbf{W}_1$ is different from other modules. In the other words, in DSN-WSOMP, the bottom module is the same as in section 2.1. But at higher modules, the relationship (10) still holds, but $\mathbf{x_{in}}$ is different. The basic algorithm to find $\mathbf{W_2}$ given $\mathbf{W_1}$, is the same as (12). However, finding $\mathbf{W_1}$ and its initialization differs from one module to another. To initialize $\mathbf{W_1^{(1)}}$ in the bottom module (Module 1 in Fig. 1), a restricted Boltzmann machine is constructed separately using contrastive divergence [14]. A complete explanation about the design of RBM used in this work is not given due to lack of space. After initializing $\mathbf{W_1^{(1)}}$, we find $\mathbf{W_1^{(1)}}$ and $\mathbf{W_2^{(1)}}$

using the method described in the previous section pertaining to one layer networks. Now we freeze the value of weights for module 1, i.e., these are the final value of weights for module 1. For the second module, the part of $\mathbf{W_1^{(2)}}$ that corresponds to the output of module 1, i.e., $\mathbf{W_{1a}^{(2)}}$ in Fig. 1, is initialized with random numbers ($\mathbf{R_n}$ in Fig. 1). The part of $\mathbf{W_1^{(2)}}$ that corresponds to input data, i.e., $\mathbf{W_{1b}^{(2)}}$ in Fig. 1 is initialized with the final value of $\mathbf{W_1^{(1)}}$ that was calculated for module 1. After initializing $\mathbf{W_1^{(2)}}$, we use the same method that we used for first module to find $\mathbf{W_1^{(2)}}$ and $\mathbf{W_2^{(2)}}$ for the second module. Then we freeze these values for the second module and we continue to higher modules. The stacking non-linear version of NWSOMP, which we denote as, DSN-NWSOMP, is a solver that uses the architecture in Fig. 1 and the procedure described above to find $\mathbf{W}_1$ and $\mathbf{W}_2$ (assuming that the bias vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are absorbed in $\mathbf{W}_1$ and $\mathbf{W}_2$). Details of DSN-NWSOMP are not presented due to lack of space.

## 3. RESULTS

This section presents the numerical results of the proposed and existing solvers for the MMV problem addressed in this paper. Note that in the results presented in this section, exact recovery means:

$$\frac{\|\hat{\mathbf{S}} - \mathbf{S}\|}{\|\mathbf{S}\|} \leq N_2 \times 10^{-5} \tag{14}$$

where $\mathbf{S}$ is the actual sparse matrix, $\hat{\mathbf{S}}$ is the calculated solution and $N_2$ is the number of channels. In our experiments, we assume that we have three channels, i.e., $N_2 = 3$. The number of realizations for each experiment is $1000$. We use a vector of size $N = 100$ for each channel and a random Gaussian measurement matrix with normalized columns of size $45 \times 100$, i.e., $M = 45$ measurements per channel. The machine we used to perform the experiments has an Intel(R) Core(TM) i7 CPU with clock 2.93 GHz and with 16 GB RAM.

To generate a sparse matrix where the probability of having non-zero entries in its bands of rows differ from one band of rows to another, suppose that for each column the probability distribution of support is made of two Gaussian distributions with means $\mu_1 = 20$ and $\mu_2 = 70$ and standard deviations $\sigma_1 = 3$ and $\sigma_2 = 6$. Therefore, non-zero elements are concentrated in two regions, one around the $20^{th}$ entry and another one around the $70^{th}$ entry of each vector in $\mathbf{S}$. Also, we assume all entries of $\mathbf{S}$ are positive, because we want to show that the proposed solvers can exploit two structures at the same time, i.e., positivity and the structure in the support. Fig.2 shows the superior performance of the proposed solvers relative to SOMP and the mixed norm minimization methods. This was predictable because at each iteration the proposed solvers make a wiser atom selection than SOMP when finding the sparse solution (see section 2.1). Note that in this figure,

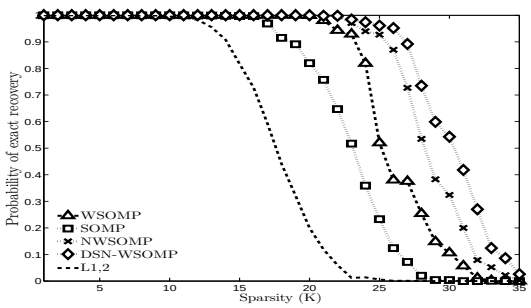


**Fig. 2**. Probability of exact recovery of NWSOMP, DSN-WSOMP, WSOMP, SOMP and mixed norm ($\ell_{1,2}$) minimization for different sparsity levels.

the WSOMP solver uses linear modelling and finds $\mathbf{W}_1$ and $\mathbf{W}_2$ using the least squares method ("lscov" in MATLAB). To calculate the weights in NWSOMP and DSN-WSOMP, we used $512$ neurons for the hidden layer, $55000$ training matrices[2], $\rho_1 = \rho_2 = \rho_3 = 0.0005$, the maximum number of iterations for each module was $100$ and the regularization coefficient for each module was $\mu = 100$. To make sure that we do not have over-fitting, we used cross validation.

## 4. DISCUSSION AND CONCLUSIONS

As observed in Fig. 2, the proposed NWSOMP and DSN-WSOMP methods start to fail when $K$ is increased beyond $K = 22$ (which is better than the WSOMP and SOMP solvers). Note that since $M = 45$, NWSOMP and DSN-WSOMP are very close to achieving the bound predicted by [5], i.e., $M = 2K$. The better performance of NWSOMP and DSN-WSOMP is predictable because $f_h(.)$ is a non-linear function. It is worth mentioning that, besides being efficient for large scale data, DSN-WSOMP has better performance than NWSOMP. It is important to note that using the method explained in section 2.1 makes the weight calculation very fast and practical. In fact, it is not practically possible to run conjugate gradient backpropagation ("traincgf" in MATLAB) on the machine used for the experiments of this paper.

In this work, we proposed two greedy solvers to solve the MMV problem when limited information about the structure of support of $\mathbf{S}$ is known. We showed that these methods outperform existing MMV solvers. We also discussed how the second proposed solver is suitable for large scale data while the first one is not. Potential applications of this work can be found in distributed compressed sensing of images and videos or in Electroencephalography (EEG) signals where the sparsity level up to which the practical solver gives exact recovery is a limitation. For example, having the prior information that in the transform domain, more non-zeros will occur in the low frequencies than in high frequencies can be helpful in improving the performance of the existing solvers using the proposed solvers in this paper.

[2] A complete description of how these training matrices should be generated is not presented due to lack of space

## 5. 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] Emmanuel J. Candes, Yonina C. Eldar, Deanna Needell, and Paige Randall, "Compressed sensing with coherent and redundant dictionaries," *Applied and Computational Harmonic Analysis*, vol. 31, no. 1, pp. 59 – 73, 2011.

[5] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via $\ell_1$ minimization," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 5, pp. 2197–2202, 2003, I have used the documnet in "http://www-stat.stanford.edu/ donoho/Reports/2002/OptSparse.pdf".

[6] M.F. Duarte and Y.C. Eldar, "Structured compressed sensing: From theory to applications," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4053 – 4085, sept. 2011.

[7] P. Smolensky, ," in *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1*, David E. Rumelhart and James L. McClelland, Eds., chapter Information processing in dynamical systems: foundations of harmony theory, pp. 194–281. MIT Press, Cambridge, MA, USA, 1986.

[8] Yoav Freund and David Haussler, "Unsupervised learning of distributions on binary vectors using two layer networks," Tech. Rep., University of California at Santa Cruz, Santa Cruz, CA, USA, 1994.

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

[10] Li Deng, Dong Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, march 2012, pp. 2133 – 2136.

[11] 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.

[12] D. Merhej, C. Diab, M. Khalil, and R. Prost, "Embedding prior knowledge within compressed sensing by neural networks," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1638 –1649, oct. 2011.

[13] Dong Yu and Li Deng, "Efficient and effective algorithms for training single-hidden-layer neural networks," *Pattern Recognition Letters*, vol. 33, no. 5, pp. 554–558, 2012.

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

[15] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, July 2006.

[16] 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.