STP-Net: Semi-Tensor Product Neural Network for Image Compressive Sensing

Youhao Yu[1,2], Richard M. Dansereau[1]

Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada[1]
School of Information Engineering, Putian University, Fujian, China[2]
e-mail: youhaoyu@sce.carleton.ca e-mail: rdanse@sce.carleton.ca

***Abstract*—Semi-tensor product (STP) is developed into a neural network in this paper and applied to image compressive sensing (CS). Large matrix computation for fully connected layers results in a large number of weight coefficients that need long training times. Instead of using an *M*×*N* measurement matrix, according to the theory of STP a smaller measurement matrix of size *M/t*×*N/t* can be applied, where *t* is a shrinkage factor. STP only needs *N/t* elements of the original signal for one measurement and the measurement matrix is shrunk to 1/*t*^{2} that of traditional CS. The shrinkage factor *t* is adjustable. To demonstrate the effectiveness of the STP-based neural network, we apply it to image reconstruction. The goal is to sample and recover larger images, without partitioning into smaller blocks that introduces block artifacts, and provide good initial reconstruction for subsequent networks.**

Keywords-compressive sensing; convolutional neural network; semi-tensor product; image reconstruction.

I. Introduction

Compressive sensing (CS) has become a significant research field in analog information processing, image compression, machine learning and so on [1]–[3]. The fundamental issue in CS is to reconstruct a sparse signal from few measurements. Since natural images are intrinsically sparse in some domain, they can be restored efficiently from CS measurements. Sparse signal reconstruction is an inverse problem that can be solved by techniques like basis pursuit (BP) [4], matching pursuit (MP) [5], orthogonal matching pursuit (OMP) [6], approximate message passing (AMP) [7], and least absolute shrinkage and selection operator (LASSO) [8], to name a few, but these tend to be time-consuming.

CS acquires signals that are sparse in a certain basis in a compressed form. The sparsifying basis and the measurement matrix should be incoherent [3]. According to CS theory, a high dimension sparse signal x is sampled by a measurement matrix Φ resulting in a low dimension measurement y as

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

where x is an $N\times 1$ vector, y is an $M\times 1$ vector and Φ is an $M\times N$ matrix ($M<<N$). The measurement matrix should satisfy the restricted isometry property (RIP) [9].

Each measurement y_i is the linear combination of the elements in x through a row of Φ as

$$y_i = \phi_{i,1}x_1 + \cdots + \phi_{i,j}x_j + \cdots + \phi_{i,N}x_N \tag{2}$$

where x_j, y_i, and $\phi_{i,j}$ are elements of x, y, and Φ. In (2), all N elements in x are used to obtain one measurement y_i, which causes large computational cost when x is long since the measurement matrix Φ will be large.

There are numerous data reconstruction approaches for compressive sensing. In [10][11], STP is adopted and an iterative optimization approach used for image CS reconstruction. While the approach in [10][11] produces good results at a high measurement rate, it is time consuming, needs many iterations, and a wavelet transform is used before CS and an inverse wavelet transform after CS reconstruction.

In this paper, we propose developing the Semi-Tensor product into a neural network (NN). Such a network uses fewer parameters to train and provides theoretical foundation for a layer to be designed. The proposed NN needs fewer layers and less training time for efficient measurement and a good initial reconstruction compared to others, performing better than other full convolutional NN systems. Given the efficiency, the developed NN is used for whole image CS reconstruction using no block partitioning.

The rest of this paper is organized as follows. Section II is an overview of previously proposed CS reconstruction methods. Section III describes the measurement and the initial reconstruction of CS based on STP. Section IV gives a detailed description of STP-Net. Our experiments are set up to demonstrate the outstanding performance of STP-Net in Section V. Finally, we conclude with a discussion of our findings in Section VI.

II. Related Work

Many NN algorithms have been studied for CS measurement reconstruction, such as [12]–[18]. Among them, [12] develops a good framework for sensing and recovering structured signals, but a few full connection layers make it less efficient. [13] and [14] use convolutional layers or residual blocks to refine the initial reconstruction of every image block. After that, they use block-matching and 3-D filtering (BM3D) [15] to remove block artifacts, which is an image denoising strategy based on an enhanced sparse representation in the transform domain. [16][17] give novel methods that measure an image using convolutional layers. However, our experiments show that when the sampling rate is 1% the reconstructed image is affected by block artifacts, especially at the edges of the image, even with a residual network (ResNet) after initial reconstruction as they did to improve performance. [18] introduces an interpretable optimization-inspired deep network for image compressive sensing. They cast the iterative shrinkage-threshold algorithm (ISTA) into a deep network that produces good performance.

To overcome the limitation of GPU memory, existing methods usually divide an image into blocks and vectorize the blocks before measurement. Then, the image is reconstructed block-by-block, making block artifacts inevitable [16] and requiring denoising to remove block artifacts.

Our method is different from others since we process an image as a whole rather than block-by-block, which avoids the block artifacts because the structure information of an image is preserved. Normally, operating on the full image would be computationally costly but the proposed STP-Net helps reduce that computational cost.

III. STP Approach for CS

STP approach can be applied for CS measurement and its initial reconstruction of 1D and 2D signal.

A. Measurement

According to the theory of Semi-Tensor product [19], a smaller measurement matrix $\Phi(t)$ can be obtained with dimensions $M/t \times N/t$. Here, *t* is a shrinkage factor which is a common divisor of M and *N*. ($M, N, t, M/t$ and N/t are all positive integers) [11]. Only N/t elements of the signal are needed for one measurement. The dimension of the measurement matrix is shrunk by $1/t^2$ of that for traditional CS. Using the left product operator $\ltimes$ for STP, (1) is rewritten as

$$y = \Phi(t) \ltimes x. \tag{3}$$

For clarity, the signal x and its measurements y can be segmented into groups and every group only has t points. So, x is divided into N/t fragments and y is divided into M/t fragments. Reshaping the vectors $x \in \mathbb{R}^{N\times 1}$ and $y \in \mathbb{R}^{M\times 1}$ into matrix form $X \in \mathbb{R}^{t\times\frac{N}{t}}$ and $Y \in \mathbb{R}^{t\times\frac{M}{t}}$, we rewrite (3) to maintain column-wise order as

$$Y = X \cdot \Phi(t)^T. \tag{4}$$

Considering X as an image, this means an image can be sampled directly by matrix multiplication.

For our work, t is set to the number of rows the image. Our method takes an image as a whole without dividing it into blocks or vectorizing the image.

B. Initial Reconstruction

Typically, the rows of the measurement matrix Φ are chosen to be orthonormal and the least-squares solution (minimum energy reconstruction) as the initial estimate for *x* [20]. Let Φ be an $M\times N$ matrix and let *y* be a vector in R^M. The least-squares solution of $\Phi x = y$ is the solution of $\Phi^T\Phi x = \Phi^T y$ [21]. If Φ has orthonormal rows, $\Phi\Phi^T$ is an $M\times M$ identity matrix. Using $\Phi^T y$ is a common initial reconstruction.

STP maintains the above-mentioned property [11]. If $\Phi(t)$ is a matrix with orthonormal rows and $y = \Phi(t) \ltimes x$, the least-squares solution for *x* is the solution of $[\Phi(t)]^T \ltimes y = [\Phi(t)]^T \ltimes \Phi(t) \ltimes x$. Since $\Phi(t) \ltimes [\Phi(t)]^T = \Phi(t)[\Phi(t)]^T = I$, $\tilde{x} = [\Phi(t)]^T \ltimes y$ is an initial estimate. The result can be written in matrix form as $\tilde{X} = Y \cdot \Phi(t)$.

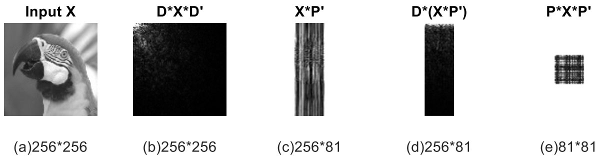

Figure 1. Image measurements and sparsity: (a) original image, (b) 1D sparsifying basis *D* applied along rows and columns, (c) measurement of (a) using random matrix *P*, (d) result showing measurement is still sparse, and (e) image sampled along rows and columns. Sampling rate = $(81/256)^2 \approx$ 10%.

C. 2D Compressive Sensing

According to compressive sensing theory [1]–[3], a signal that is sparse in a certain domain can be sampled at a rate less than the Nyquist sampling rate. In Fig. 1, (a) is an image with size 256×256. Usually, a natural image is sparse in the frequency domain, as in Fig. 1(b) where a 2D discrete cosine transform (DCT) has been applied using matrix *D.* Most energy is concentrated on top-left corner. If measurement matrix $\Phi(t) = P$, where *P* is a random matrix, then the image measured by (4) produces Fig. 1(c) with smaller size.

Since t is set to the number of rows, the measurement matrix has effectively measured each column separately. As shown in Fig. 1(d), the measurement results are still sparse with the right choice of sparsifying basis *D*; most energy concentrates in the top area. Following CS theory, the measurements can be sampled and compressed again using

$$Y = \Phi_1(t) \cdot X \cdot \Phi_2(t)^T. \tag{5}$$

In Fig. 1, since the image is square, we suppose $\Phi_1(t) = \Phi_2(t) = P$, where P is a random matrix. Let us now map image X and its measurements Y into vectors x and y by column ordering, it is equivalent to (1) when $\Phi = \Phi_1(t) \otimes \Phi_2(t)$ where $\otimes$ is the Kronecker product. In this case, the 1D operation of (1) is expressed as the separable 2D operation that reduces the computational complexity [22].

It has been shown that a sparse matrix $\hat{X}$ (i.e, $\hat{X} = DXD^T$) can be recovered from its matrix sketching $Y = \Phi_1 \cdot \hat{X} \cdot \Phi_2^T$ [23][24][25]. Here, we assume X has size $t \times t$ where $t^2 = N$. The dimension of Y is $m \times m$ with $m = M/t$ and $m \ll t$. Φ_1 and Φ_2 are two $m \times t$ matrices that can be seen as measurement matrices. It is equivalent to $y = (\Phi_1 \otimes \Phi_2) \cdot x$. The initial estimate of the image can then be $\tilde{X} = \Phi_1^T \cdot Y \cdot \Phi_2$.

As shown in the dashed box of Fig. 2, an image is effectively measured and compressed twice and its initial reconstruction uses two corresponding steps.

IV. STP-Net: Neural Network Layer

Inspired by the flexibility of STP, we build a neural network layer to implement it. The sampling and initial recovery process of the proposed STP neural network (STP-Net) for compressive sensing can be implemented by building layers of an NN in two ways. One is defining a custom deep learning layer with learnable parameters in forward and backward propagation [26], and the other is by means of ready-made convolutional layers, like in Matlab,

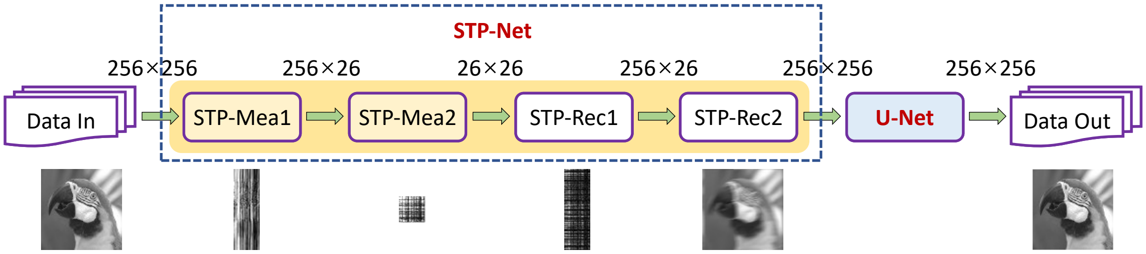

Figure 2. STP-Net connected with U-Net for deblurring (Sampling rate:1%).

TensorFlow, or PyTorch, followed by a custom reshape layer.

A. Defining a Custom Deep Learning Layer

A custom STP layer has a forward pass, a backward pass, and learnable parameters including weights and biases. It can be used as a measurement matrix by setting the biases to zero and the learning rates to zero so that the set weights remain static. It also can be used in initial reconstruction through a least-squares like solution.

Reconstruction of a signal is a regression problem, so the output layer is a regression layer. The loss function of the regression layer is the half-mean-square-error (HMSE) [27]

$$L = \frac{1}{2S}\sum_{i=1}^{S}\sum_{j=1}^{R}\frac{(t_{ij}-y_{ij})^2}{R} \tag{6}$$

where S is the number of observations in a mini-batch, R is the number of responses, t_{ij} is the target output, and y_{ij} is the network's prediction for the response variable corresponding to observation i.

For a semi-tensor product layer, the derivative of the loss function with respect to the input data of the custom layer are

$$\frac{\partial L}{\partial x_{i,j}} = \sum_{a=1}^{M/t}\frac{\partial L}{\partial y_{i,a}}\phi_{a,j} \tag{7}$$

where $i = 1,\dots,t$, $a = 1,\dots,M/t$, and $j = 1,\dots,N/t$.

The derivatives of the loss with respect to the weights are

$$\frac{\partial L}{\partial \phi_{a,j}} = \sum_{i=1}^{t}\frac{\partial L}{\partial y_{i,a}}x_{i,j} \tag{8}$$

where $i = 1,\dots,t$, $a = 1,\dots,M/t$, and $j = 1,\dots,N/t$.

For better performance, it could have biases B added to the STP layer. Since $Y = X\cdot\Phi(t)^T + B$, the derivatives of the loss with respect to the biases B have the same size with output Y and the values are

$$\frac{\partial L}{\partial B_{ij}} = \frac{\partial L}{\partial y_{ij}} \tag{9}$$

where B_{ij} and y_{ij} are the elements of B and Y ($i = 1,\dots,t$ and $j = 1,\dots,M/t$).

Obtaining the derivative of the loss with respect to the measurement matrix and the biases, the learnable parameters are updated with

$$\phi_{a,b}^{k} = \phi_{a,b}^{k-1} - \eta\frac{\partial L}{\partial \phi_{a,b}} \tag{10}$$

$$B_{ij}^{k} = B_{ij}^{k-1} - \eta\frac{\partial L}{\partial B_{ij}} \tag{11}$$

where k is the iteration number, η is the learning rate, $a = 1,\dots,M/t$, $b = 1,\dots,N/t$, $i = 1,\dots,t$ and $j = 1,\dots,M/t$.

The initial reconstruction would be $\tilde{X} = Y\cdot\Phi(t) + B$. The traditional compressive sensing measurement paradigm applies fixed linear measurement [12], which is easy to implement in practical applications. Biases are not added with the measurement results, but they are added during initial reconstruction to have better results.

B. Ready-made Convolutional Layer

The CS measurement process based on STP is similar to dilated convolution (also known as "à trous" convolution). STP can be implemented by means of a dilated convolutional layer with factor t used to increase the receptive field (the area of the input signal which the layer can detect) of the layer without increasing the number of parameters and computation [28]. Since the number of rows of the measurement matrix correspond to the number of neurons in the convolutional layer and every filter produces one channel output, this layer should be followed by a reshaping, which makes the measurements in the same channel.

V. Experiments

In this section, we conduct a series of experiments to test the measurement and reconstruction performance of STP-Net.

A. Implementation Details

The experiments are conducted on MATLAB R2019a. The computer is equipped with Intel i7-8700K, GeForce GTX 1080 CPU with frequency of 3.7 GHz and 16 GB RAM. Natural images from the ILSVRC2014 ImageNet dataset are adopted. Here, 20k images are chosen: 14k (70%) for training, 3k (15%) for validation, and 3k (15%) for testing. We extracted the central 256×256 part of each image and converted them to 8-bit grayscale. Training used stochastic gradient descent with momentum, minibatch size of 64, maximum epoch of 40, learning rate of 2e-03, drop factor of 0.10, and drop period of 15. For comparison, we also utilize the widely used benchmark dataset Set11 [13] during testing.

ISBN: 978-1-61208-970-6

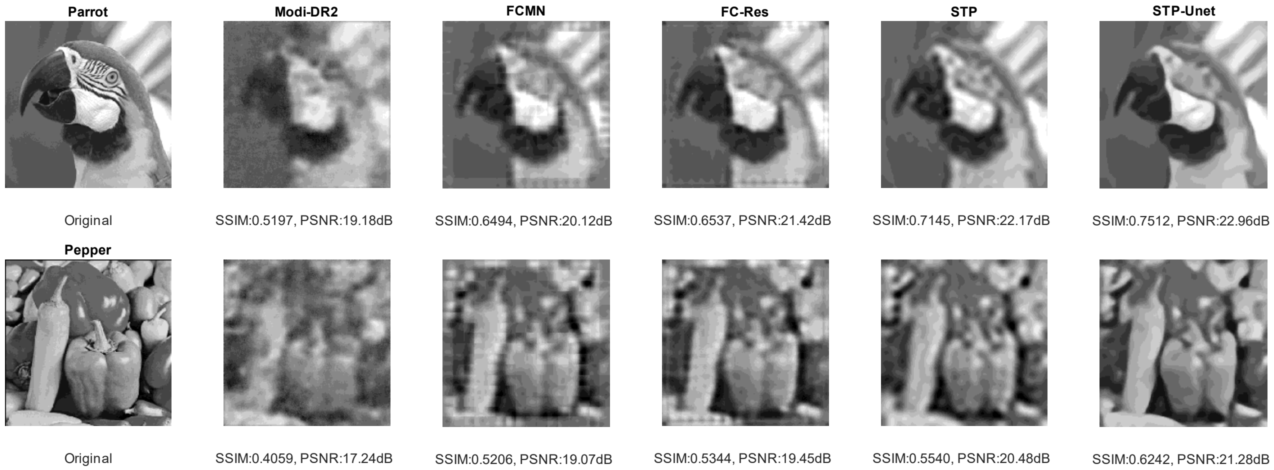
Parrot
Modi-DR2
FCMN
FC-Res
STP
STP-Unet
Original
SSIM:0.5197, PSNR:19.18dB
SSIM:0.6494, PSNR:20.12dB
SSIM:0.6537, PSNR:21.42dB
SSIM:0.7145, PSNR:22.17dB
SSIM:0.7512, PSNR:22.96dB
Pepper
Original
SSIM:0.4059, PSNR:17.24dB
SSIM:0.5206, PSNR:19.07dB
SSIM:0.5344, PSNR:19.45dB
SSIM:0.5540, PSNR:20.48dB
SSIM:0.6242, PSNR:21.28dB

Figure 3. Reconstruction results for parrot and pepper from noiseless CS measurements at measurement rate of 1%. It is evident that STP based method restores more visually appealing images than the competitors.

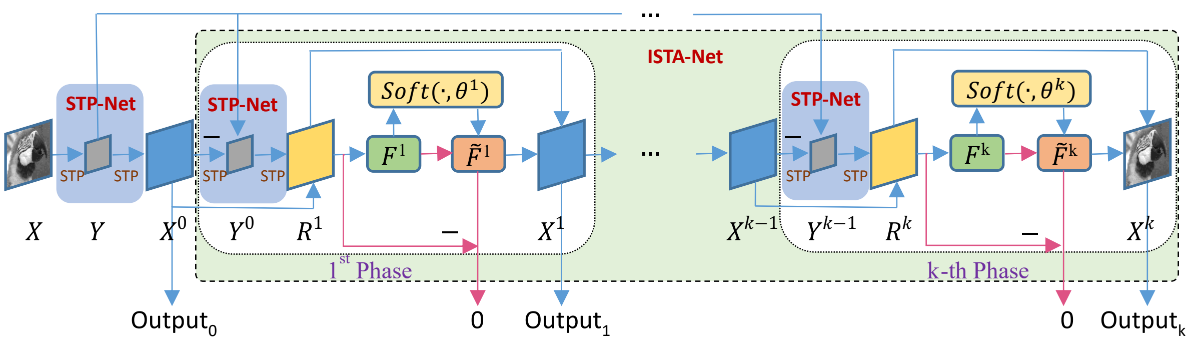

Figure 4. Connect STP-Net with ISTA-Net and use STP layers for measurement and reconstruction in every phase.

B. Measurement Matrix Based on STP

Every image has 65536 pixels. A sampling rate of 1% will produce 655 measurements. The size of such a measurement matrix is 655×65536, which needs more than 300 MB of memory to store the measurement matrix (with double precision matrix elements). According to the STP method, if the parameter t is 256, then a measurement matrix with size equal to *ceiling*(256×1%)×256 can be used that satisfies RIP [9]. It needs 6 kB of memory and is much smaller than the full measurement matrix. Too small of a measurement matrix leads to excessive information loss, so we implement CS measurement in two steps. As mentioned in Section III, we can use two 1D projections for the 2D image, which can be done by applying STP twice. For a target of 1%, one option is that the first pass compresses the signal by 10% and the second pass compresses by another 10%. The size of measurement matrix would be *ceiling*(256×10%)×256 with t equal to 256.

C. Proposed Structure

Fig. 2 shows the structure of the neural network. STP-Net samples the image and provides initial reconstruction that can be passed to subsequent networks. U-Net [29] has proved to work well with semantic segmentation, which motivates us to apply it for reconstruction as a deblurring step. To set up an image regression network, we remove the last two layers (soft max and segmentation layers) of a U-Net with an encoder-decoder depth of 3 and add a regression layer. In Fig. 4, STP-Net is connected with ISTA-Net and STP layers are used repeatedly as measurement and reconstruction for each iteration. Our experiments adopt five ISTA iterations (phases) and every convolutional layer uses 16 kernels.

D. Experimental Results

For the sampling process, we applied different sampling rate combinations (20%+5%, 10%+10% and 5%+20%) in the two measurement layers to reach total measurement rate of 1%. The mean peak signal-to-noise ratio (PSNR) of the 3000 test images with these sampling rate combinations are 20.55, 21.50 and 20.64 dB, respectively, without U-Net. It seems that the square root (10%+10%) of the total measurement rate could be a better choice. Tables I to III show that the proposed method has better performance than other methods and has higher PSNR and structural similarity measure (SSIM).

From Fig. 3, we see that other methods have block artifacts. For DR^2-Net [14], the image is measured block-by-block and reconstructed with 4 residual blocks. Then, they use BM3D [15] to remove the artifacts caused by block-wise processing. We modified the process by composing an intermediate reconstructed image with their initial reconstruction and then using the 4 residual blocks to remove block artifacts. It has better performance than their original method even without

BM3D. Full convolutional measurement network (FCMN) has block artifacts too, especially at the four edges of a reconstructed image. We apply one residual block after it as the authors did in [16]. The block artifacts are alleviated, but the four corners of restored image are darker. From Tables I to IV, it shows that STP-Net provides an attractive initial reconstruction quality for another network. Table III shows that STP-Net works well in less time. In Table IV, ISTA-Net and ISTA-Net+ adopt nine ISTA iterations with every convolutional layer having 32 kernels. STP-Net provides ISTA-Net with information-rich measurements and reasonable initial reconstruction, that enable it to simplify its structure with fewer ISTA iterations and kernels to have better performance.

VI. Conclusion

We have presented an STP-based neural network to the problem of CS image reconstruction. The measurement and initial reconstruction process are efficiently implemented through STP without dividing the image into blocks and vectorizing. At different measurement rates, our algorithm yields superior quality reconstructions than other methods. The method does not have block artifacts that many people try to solve. It makes the sampling process convenient and provides good initial reconstruction for subsequent network, such as U-Net or ISTA-Net. In future work, we are going to apply it in deep equilibrium architecture to develop an efficient, high performance fixed-point iteration layer [30].

Table I. PSNR Values in dB on Set11 with Different Algorithms at 1% Measurement Rate

Image Name	**ReconNet +BM3D** [13]	**DR^2-Net** [14]	**DR^2+BM3D** [14]	**Modified DR^2-Net** [14]	**FCMN** [16]	**FC-Res** [16]	**STP-Net**	**STP-UNet**
Barbara	19.08	18.65	19.10	19.02	20.38	20.97	21.83	**22.10**
Boat	18.83	18.67	18.95	18.82	19.96	20.57	21.46	**22.23**
Cameraman	17.49	17.08	17.34	17.72	19.16	19.68	20.14	**21.25**
Fingerprint	14.88	14.73	14.95	14.92	15.56	15.83	**16.16**	**16.16**
Flintstones	14.08	14.01	14.18	13.29	14.46	14.77	15.28	**15.37**
Foreman	20.33	20.59	21.08	22.54	21.08	23.72	**27.15**	27.00
House	19.52	19.61	19.99	20.61	20.93	22.38	23.16	**24.47**
Lena	18.05	17.97	18.40	18.51	20.49	21.15	21.95	**22.72**
Monarch	15.49	15.33	15.50	15.52	17.20	17.58	18.28	**18.79**
Parrot	18.30	18.01	18.41	19.18	20.12	21.42	22.17	**22.96**
Pepper	16.96	16.90	17.11	17.24	19.07	19.45	20.48	**21.28**

(For ReconNet, we use the results reported in [13]. For DR^2-Net and DR^2+BM3D, we use the results reported in [14]. For the other algorithms, the experiments use Matlab with networks trained from the same dataset with the same images.)

Table II. SSIM Value for 11 Extra Images

Image Name	**ReconNet** [13]	**Modified DR^2-Net** [14]	**FCMN** [16]	**FC-Res** [16]	**STP-Net**	**STP-UNet**
Barbara	0.3730	0.3578	0.4555	0.4575	0.5024	**0.5271**
Boat	0.4140	0.3838	0.4729	0.4771	0.4950	**0.5587**
Cameraman	0.4517	0.4391	0.4998	0.5389	0.5503	**0.6565**
Fingerprint	0.1641	0.0708	0.0853	0.0858	0.0884	**0.0886**
Flintstones	0.2733	0.1789	0.2386	0.2429	0.2580	**0.2871**
Foreman	0.5647	0.6078	0.6680	0.6849	0.7536	**0.7869**
House	0.5278	0.5282	0.5809	0.5948	0.6291	**0.7056**
Lena	0.4418	0.4344	0.5364	0.5489	0.5765	**0.6324**
Monarch	0.3802	0.3427	0.4683	0.4816	0.5003	**0.5578**
Parrot	0.5329	0.5197	0.6494	0.6537	0.7145	**0.7512**
Pepper	0.4002	0.4059	0.5206	0.5344	0.5540	**0.6242**
Mean SSIM	0.4112	0.3881	0.4705	0.4819	0.5111	**0.5615**

(For ReconNet, we calculate the values of SSIM from the images the authors provide.)

Table III. Results of 3000 Test Images

Evaluation index	**Modified DR^2-Net** [14]	**FCMN** [16]	**FC-Res** [16]	**STP-Net**	**STP-UNet**
Mean SSIM	0.3696	0.4347	0.4563	0.4911	**0.5301**
Mean PSNR	18.72	19.26	20.45	21.50	**22.06**
Elapsed Time(s)	56.62	10.70	22.02	**9.36**	41.33

(The number in the table are the mean of 10 times experiments.)

Table IV. Average PSNR (dB) Performance Comparisons on Set11 with Different CS Ratios

Sampling rate	**ReconNet +BM3D** [13]	**DR^2-Net** [14]	**DR^2+BM3D** [14]	**FCMN** [16]	**FC-Res** [16]	**ISTA-Net** [18]	**ISTA-Net+** [18]	**STP-Net**	**STP-ISTA-Net**
1%	17.55	17.44	17.73	18.95	19.77	17.30	17.34	20.65	**21.30**
4%	20.44	20.80	21.29	23.14	24.22	21.23	21.31	23.39	**24.92**
10%	23.23	24.32	24.71	25.36	27.30	25.80	26.64	26.02	**28.65**
25%	25.92	28.66	29.06	28.69	31.15	31.53	32.57	30.06	**33.54**

(The best performance is labeled in bold.)

Acknowledgement

This research was partially funded through a grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] E. J. Candes and J. K. Romberg, "Signal recovery from random projections," *Comput. Imaging III*, vol. 5674, pp. 76–86, 2005.

[2] R. G. Baraniuk, V. Cevher, and M. F. Duarte, "Model-based compressive sensing," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1982–2001, 2010.

[3] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[4] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM*, vol. 43, pp. 129–159, 2001.

[5] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, 1993.

[6] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," *Conf. Rec. Asilomar Conf. Signals, Syst. Comput.*, vol. 1, pp. 40–44, 1993.

[7] J. Zammit and I. J. Wassell, "Adaptive block compressive sensing: Toward a real-time and low-complexity implementation," *IEEE Access*, vol. 8, pp. 120999–121013, 2020.

[8] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *J. R. Stat. Soc. Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.

[9] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Commun. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, 2006.

[10] J. Wang, S. Ye, Y. Ruan, and C. Chen, "Low storage space for compressive sensing: Semi-tensor product approach," *Eurasip J. Image Video Process.*, vol. 2017, no. 1, pp. 1-13, 2017.

[11] J. Wang, Z. Xu, Z. Wang, S. Xu, and J. Jiang, "Rapid compressed sensing reconstruction: A semi-tensor product approach," *Inf. Sci. (Ny).*, vol. 512, pp. 693–707, 2020.

[12] A. Mousavi, A. Patel, and R. Baraniuk, "A deep learning approach to structured signal recovery," *53rd Annu. Allert. Conf. Commun. Control. Comput. Allert. 2015*, pp. 1336–1343, 2016.

[13] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "ReconNet: Non-iterative reconstruction of images from compressively sensed measurements," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016, pp. 449–458, 2016.

[14] H. Yao, F. Dai, S. Zhang, Y. Zhang, Q. Tian, and C. Xu, "DR2-Net: Deep residual reconstruction network for image compressive sensing," *Neurocomputing*, vol. 359, pp. 483–493, 2019.

[15] K. Dabov, A. Foi, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. IMAGE Process.*, vol. 16, no. 8, pp. 2080–2095, 2007.

[16] J. Du, X. Xie, C. Wang, G. Shi, X. Xu, and Y. Wang, "Fully convolutional measurement network for compressive sensing image reconstruction," *Neurocomputing*, vol. 328, pp. 105–112, 2019.

[17] J. Du, X. Xie, C. Wang, and G. Shi, "Perceptual compressive sensing," *Lect. Notes Comput. Sci*, vol. 11258 LNCS, pp. 268–279, 2018.

[18] J. Zhang and B. Ghanem, "ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing," *IEEE/CVF Conf. Computer Vision and Pattern Recognition*, pp. 1828–1837, 2018.

[19] Qi, Hongsheng and Daizhan Cheng. "Analysis and control of boolean networks: A semi-tensor product approach." *2009 7th Asian Control Conference*. IEEE, pp. 1352–1356, 2009.

[20] Candes E, Romberg J. "L1-magic : Recovery of sparse signals via convex programming," April 2022. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.212.9120&rep=rep1&type=pdf.

[21] D. Margalit and J. Rabinoff, *Interactive Linear Algebra*. Georgia Institute of Technology, 2019.

[22] A. Akansu, R. Haddad, and P. Haddad, *Multiresolution signal decomposition: transforms, subbands, and wavelets*. Academic Press, 2001.

[23] T. Wimalajeewa, Y. C. Eldar, and P. K. Varshney, "Recovery of sparse matrices via matrix sketching," *CoRR*, vol. 2, no. 5, pp. 1–5, 2013.

[24] G. Dasarathy, P. Shah, B. Bhaskar, and R. Nowak, "Sketching sparse matrices," *arXiv Prepr. arXiv1303.6544*, pp. 1–33, 2013.

[25] G. Dasarathy, P. Shah, B. Bhaskar, and R. Nowak, "Sketching sparse matrices, covariances, and graphs via tensor products," *IEEE Trans. Inf. Theory*, vol. 61, no. 3, pp. 1373–1388, 2015.

[26] MathWorks, "Define Custom Deep Learning Layer with Learnable Parameters," April 2022. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/define-custom-deep-learning-layer.html.

[27] MathWorks, "Specify Layers of Convolutional Neural Network," April 2022. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html.

[28] MathWorks, "2-D Convolutional Layer," April 2022. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.convolution2dlayer.html.

[29] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *Int. Conf. Med. Image Comput. Comput. Interv.*, pp. 234–241, 2015.

[30] D. Gilton, G. Ongie, and R. Willett, "Deep equilibrium architectures for inverse problems in imaging," *arXiv Prepr. arXiv2102.07944*, pp. 1–21, 2021.