Memory-Augmented Deep Unfolding Network for Compressive Sensing

Jiechong Song  
Shenzhen Graduate School, Peking University  
Shenzhen, China

Bin Chen  
Shenzhen Graduate School, Peking University  
Shenzhen, China

Jian Zhang*  
zhangjian.sz@pku.edu.cn  
Shenzhen Graduate School, Peking University  
Peng Cheng Laboratory  
Shenzhen, China

ABSTRACT

Mapping a truncated optimization method into a deep neural network, deep unfolding network (DUN) has attracted growing attention in compressive sensing (CS) due to its good interpretability and high performance. Each stage in DUNs corresponds to one iteration in optimization. By understanding DUNs from the perspective of the human brain’s memory processing, we find there exists two issues in existing DUNs. One is the information between every two adjacent stages, which can be regarded as short-term memory, is usually lost seriously. The other is no explicit mechanism to ensure that the previous stages affect the current stage, which means memory is easily forgotten. To solve these issues, in this paper, a novel DUN with persistent memory for CS is proposed, dubbed Memory-Augmented Deep Unfolding Network (MADUN). We design a memory-augmented proximal mapping module (MAPMM) by combining two types of memory augmentation mechanisms, namely High-throughput Short-term Memory (HSM) and Cross-stage Long-term Memory (CLM). HSM is exploited to allow DUNs to transmit multi-channel short-term memory, which greatly reduces information loss between adjacent stages. CLM is utilized to develop the dependency of deep information across cascading stages, which greatly enhances network representation capability. Extensive CS experiments on natural and MR images show that our MADUN outperforms existing state-of-the-art methods by a large margin. The source code is available at https://github.com/jianzhanggcs/MADUN/.

KEYWORDS

compressive sensing, deep unfolding network, memory mechanisms, information augmentation

ACM Reference Format:
Jiechong Song, Bin Chen, and Jian Zhang. 2021. Memory-Augmented Deep Unfolding Network for Compressive Sensing. In Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3474085.3475562

1 INTRODUCTION

Compressive sensing (CS) is a novel methodology of acquisition and reconstruction. Signal is first sampled and compressed with linear random transformations. Then, the original signal can be reconstructed from far fewer measurements than required by the sub-Nyquist sampling rate [25][20]. Due to its attractive merits of the simple/fast sampling process and the low demand for data transmission and storage, the CS method has spawned many applications, including but not limited to single-pixel imaging [6][24], accelerated magnetic resonance imaging (MRI) [21], wireless remote monitoring [47], and snapshot compressive imaging [36][35].

Mathematically, a random linear measurements $y \in \mathbb{R}^M$ can be formulated as

$$y = \Phi x,$$

where $x \in \mathbb{R}^N$ is the original signal and $\Phi \in \mathbb{R}^{M \times N}$ is the measurement matrix with $M \ll N$. $\frac{M}{N}$ is the CS ratio. Obviously, CS reconstruction is an ill-posed inverse problem. To obtain a reliable reconstruction, the conventional CS methods commonly solve an energy function,

$$\arg\min_x \frac{1}{2} \left\| \Phi x - y \right\|_2^2 + \lambda \mathcal{F}(x),$$

where $\lambda \mathcal{F}(x)$ denotes a prior term with regularization parameter $\lambda$. For the traditional CS methods [12][18][43][42][7][22][52], the prior term can be the sparsifying operator corresponding to some pre-defined transform basis, such as discrete cosine transform (DCT) and wavelet [49][50]. They enjoy the advantages of strong convergence and theoretical analysis in most cases, but usually inevitably suffer from high computational complexity and face the trouble of choosing optimal transforms and parameters [51].

Recently, fueled by the powerful learning ability of deep networks, several deep network-based image CS reconstruction algorithms have been proposed [16][32][8][2], which can be generally grouped into two categories: deep non-unfolding networks (DNUNs) and deep unfolding networks (DUNs). For DNUN, it aims to directly learn the inverse mapping from the CS measurement domain to the original signal domain [23][11]. For DUN, it combines deep network with optimization and trains a truncated unfolding...
Augmented Deep Unfolding Network (MADUN), focusing on CS satisfaction of interpretability and the ensurance of information network representation ability by augmenting entire informationous optimization-inspired DUNs and meanwhile, improving the is a comprehensive framework, considering the merits of previ-justable long-term information path across stages. Our MADUN mechanism (CLM) is exploited to explicitly construct the deep and ad-\text{memory augmentation mechanisms, as shown in Figure 1. To proximal Mapping Module (MAPMM) which contains two different memory augmentation mechanisms.}

Figure 1: Architecture of our proposed MADUN, which consists of $K$ stages. $x$ denotes the full-sampled image for training, $\Phi$ denotes measurement matrix, $y$ is the under-sampled data, $x^{(0)}$ denotes the initialization, $x^{(K)}$ denotes the recovered result of MADUN and $x^{(k)}$ stands for the output images of $(k)$-th stage. The dotted red line denotes High-throughput Short-term Memory (HSM) transmission and the dotted green line denotes Cross-stage Long-term Memory (CLM) transmission.

## 2 RELATED WORK

### 2.1 Deep Non-Unfolding Network

Deep non-unfolding network (DNUN) directly learns mapping functions from the CS sampling image $\Phi x$ to the full-sampled image $x$, resulting in an end-to-end network. Kulkarni et al. [16] develop a CNN-based CS algorithm, dubbed ReconstructNet, which learns to regress an image block from its CS measurement. Sun et al. [32] design a dual-path network to learn the structure-texture decomposition in a data-driven manner. Residual learning and U-Net structure are adopted in CS-MRI to successfully learn the aliasing artifacts [10]. Some works jointly learn the measurement by a sampling sub-network and the recovery by a reconstruction sub-network from the training data. CSNet [27] can avoid blocking artifacts by learning an end-to-end mapping between measurements and the whole reconstructed images. Shi et al. [28] exploit a convolutional neural network for scalable sampling and quality scalable reconstruction. Apparently compared with traditional methods, DNUNs can represent image information flexibly with the fast inferencing efficiency. However, in DNUNs, the learning performance seriously depends on the careful tuning and the sampling matrix is not well-embedded in the reconstruction process, which not only result in the very tricky training schemes but also drag down the network performance due to the difficulty of directly learning the recovery mapping without explicit matrix guidance.

### 2.2 Deep Unfolding Network

Deep unfolding networks (DUNs) have been proposed to solve different image inverse tasks, such as denoising [3][17], deblurring [15][34], and demosaicking [14]. DUN has friendly interpretability on training data pairs $\{(y_j, x_j)\}_{j=1}^{N_d}$, which is usually formulated on CS construction as the bi-level optimization problem:

$$\min_{\hat{x}_j} \sum_{j=1}^{N_d} L(\hat{x}_j, x_j),$$

subject to $\hat{x}_j = \arg \min_x \frac{1}{2}||\Phi x - y_j||_2^2 + \lambda F(x).$ (2)

DUNs on CS and compressive sensing MRI (CS-MRI) usually integrate some effective convolutional neural network (CNN) denoisers into some optimization methods including half quadratic splitting (HQS) [46][5][1], alternating minimization (AM) [26][31][53], iterative shrinkage-thresholding algorithm (ISTA) [40][8][41], approximate message passing (AMP) [48][54], alternating direction
As we all know, DUN is a kind of CNN method that combines HSM. (d) is the PMM with Cross-stage Long-term Memory (CLM). (e) is our proposed Memory-Augmented Proximal Mapping (CM) module which can augment the sequential links across stages.

ISTA solves the CS reconstruction problem in Eq. (1) by iterating an efficient iterative algorithm. Iterative shrinkage-thresholding methods usually lead to different optimization-inspired DUNs.

Although existing DUNs inherit a good structure from optimization and exhibit friendly interpretability, we find that, from the perspective of neural networks, DUNs’ inherent design of taking one-channel images as inputs and outputs for each stage will greatly hinder network information transmission and lose much more image details. Chen et al. [2] propose a contextual memory (CM) module which can augment the sequential links across stages. However, the CM module is still hard to maintain abundant details due to the poor storage capacity of one-channel stage outputs.

3 PROPOSED METHOD

In this section, we will elaborate on the design of our proposed Memory-Augmented Deep Unfolding Network (MADUN) for CS.

3.1 Basic Network Architecture

As we all know, DUN is a kind of CNN method that combines an efficient iterative algorithm. Iterative shrinkage-thresholding algorithm (ISTA) is well suited for solving many large-scale linear inverse problems [40], e.g., general CS and CS-MRI. Traditional ISTA solves the CS reconstruction problem in Eq. (1) by iterating between the following update steps:

\[
\begin{align*}
    r^{(k)} &= x^{(k-1)} - \rho \Phi^T \left( \Phi x^{(k-1)} - y \right), \\
    x^{(k)} &= \text{arg min}_x \frac{1}{2} ||x - r^{(k)}||^2_2 + \lambda F(x),
\end{align*}
\]

where \( \rho \) is the step size. Inspired by ISTA-Net * [40] which is a DUN unfolded by ISTA, Eq. (3) and Eq. (4) can be expressed as two modules in the \( (k) \)-th stage \( 0 < k \leq K \) as shown in Figure 2 (a):

\[
\begin{align*}
    r^{(k)} &= x^{(k-1)} - \rho \Phi^T \left( \Phi x^{(k-1)} - y \right), \\
    x^{(k)} &= H_{\text{PMM}}^{(k)}(r^{(k)}).
\end{align*}
\]

Eq. (5) is called gradient descent module (GDM) where \( r^{(k)} \) is a learnable parameter, and Eq. (6) is called proximal mapping module (PMM) which is actually a CNN-based denoiser.

To keep a simple structure with the high recovery accuracy, PMM uses basic convolution layers (Conv) and residual blocks (RB) which generate residual outputs by the structure of Conv-ReLU-Conv, and adopts an efficient reconstruction network [45] as shown in Figure 2 (b), which can be divided into four parts:

- A convolution layer receives one-channel network inputs \( r^{(k)} \) and generates multi-channel outputs, yielding Conv1 \( (r^{(k)}) \).
- A mapping layer extracts deep representation, which consists of two residual blocks (RB1 (·) and RB2 (·)) in PMM due to optimizing easily [9]. Here, \( s^{(k)} \) is the intermediate result produced between the two residual blocks.
- A convolution layer Conv2 (·) outputs residual result by the feature conversions from multi-channel to one-channel.
- A residual learning strategy produces final module outputs.

Accordingly, PMM can be formulated \( x^{(k)} = H_{\text{PMM}}^{(k)}(r^{(k)}) \) as:

\[
x^{(k)} = r^{(k)} + \text{Conv2}(\text{RB1}(\text{Conv1}(r^{(k)}))).
\]

3.2 Memory-Augmented DUN

Motivated by the viewpoint that the human brain achieves memory by preserving and storing what they acquire or are informed previously [4], the information flow across stages can also be regarded as

Figure 2: (a) is the illustration of \( (k)-\text{th} \) stage in MADUN, where \( r^{(k)} \) is the outputs of the Gradient Descent Module (GDM) and the inputs of the Proximal Mapping Module (PMM). (b) is the PMM. (c) is the PMM with High-throughput Short-term Memory (HSM). (d) is the PMM with Cross-stage Long-term Memory (CLM). (e) is our proposed Memory-Augmented Proximal Mapping Module (MAPMM) which combines HSM and CLM with PMM. Here, “+” is a residual learning strategy, “⊕” is the concatenation along channel dimension, \( H \) and \( W \) denote the height and width of the image and \( C \) is the filter number.
short-term memory transmission in D UN s, where $x^{(k)}$ in Figure 2 is the link between each two adjacent stages. However, Conv2 (·) in Eq. (7) is a lossy conversion process which would cause memory loss. Here, we propose a Memory-Augmented Deep Unfolding Network (MADUN) which augments memory across stages, each of which shares a unified two-step recovering scheme as follows:

\[
\begin{align*}
\mathbf{r}^{(k)} &= \mathcal{H}^{(k)}_{\text{CLM}}(\mathbf{x}^{(k-1)}), \\
\mathbf{x}^{(k)} &= \mathcal{H}^{(k)}_{\text{MAPMM}}(\mathbf{r}^{(k)}).
\end{align*}
\]

Here, GDM is trivial [40] and is the same as Eq. (3), and Memory-Augmented Proximal-Mapping Module (MAPMM) is introduced with two types of memory augmentation mechanisms which can highly strengthen the information transmission among all stages.

### 3.3 Memory-Augmented Proximal Mapping

On the basis of PMM, MAPMM introduces High-throughput Short-term Memory (HSM) and Cross-stage Long-term Memory (CLM), which greatly enhance the network representation capability.

#### 3.3.1 High-throughput Short-term Memory (HSM)

To address the information-lossy intra-stage transmission, we introduce HSM with multi-channel $\mathbf{z}^{(k-1)}$ (of size $H \times W \times C$) to bridge a high-capacity path between adjacent stages. As shown in Figure 2 (c) and as the output of $\text{RB}_2(\cdot)$ at $(k - 1)$-th stage, $\mathbf{z}^{(k-1)}$ is:

\[
\mathbf{z}^{(k-1)} = \text{RB}_2(\mathbf{z}^{(k-1)}).
\]

Concatenating $\mathbf{z}^{(k-1)}$ with $\mathbf{r}^{(k)}$ followed by $\text{Conv}_3(\cdot)$ generates $C$-channel features. The remaining process is the same with Eq. (7). Thus, PMM with HSM, denoted by $\mathbf{x}^{(k)} = \mathcal{H}^{(k)}_{\text{PMM-HSM}}(\mathbf{r}^{(k)})$, is:

\[
\mathbf{x}^{(k)} = \mathbf{r}^{(k)} + \text{Conv}_2(\text{RB}_2(\text{Conv}_1(\text{Conv}_3(r^{(k)} \| z^{(k-1)})�)))
\]

where $\|$ denotes concatenating the feature maps. Obviously, $\mathbf{z}^{(k-1)}$ transmits high-throughput information from the previous stage to the current stage, achieving multi-channel short-term memory which ensures maximum flow and reduces information loss caused by the conventional channel-shrinking transformation.

#### 3.3.2 Cross-stage Long-term Memory (CLM)

For modeling explicitly the long-range dependencies among all cascading stages, we utilize a ConvLSTM layer [29], which has been well-validated to be a stable and powerful way to balance the past and current states, to develop the cross-stage long-term memory (CLM) and further enhance the signal conversion and transmission.

In the process of achieving CLM, the ConvLSTM layer is sandwiched between two residual blocks ($\text{RB}_1(\cdot)$ and $\text{RB}_2(\cdot)$), and updates cell outputs $c^{(k)}$ and hidden states $h^{(k)}$ with the inputs of the intermediate result $s^{(k)} = \text{RB}_1(\text{Conv}_1(r^{(k)}))$ in Figure 2 (d).

\[
[h^{(k)}, c^{(k)}] = \text{ConvLSTM}(s^{(k)}, [h^{(k-1)}, c^{(k-1)})
\]

where CLM $[h^{(k-1)}, c^{(k-1)}]$ updates the current state of deep information, predicts multi-stage information and achieves long-term memory across cascading stages. The architecture of ConvLSTM with inputs $s^{(k)}$ can be expressed as:

\[
\begin{align*}
\mathbf{i}^{(k)} &= \sigma(W_{si} \ast s^{(k)} + W_{hi} \ast h^{(k-1)} + b_i), \\
\mathbf{f}^{(k)} &= \sigma(W_{sf} \ast s^{(k)} + W_{hf} \ast h^{(k-1)} + b_f), \\
\mathbf{c}^{(k)} &= \mathbf{f}^{(k)} \odot c^{(k-1)} + \mathbf{i}^{(k)} \odot \tanh(W_{sc} \ast s^{(k)} + W_{hc} \ast h^{(k-1)} + b_c), \\
\mathbf{o}^{(k)} &= \sigma(W_{so} \ast s^{(k)} + W_{ho} \ast h^{(k-1)} + b_o), \\
\mathbf{h}^{(k)} &= \mathbf{o}^{(k)} \odot \tanh(c^{(k)}),
\end{align*}
\]

where $\ast$ denotes the convolution operator, $\odot$ denotes the Hadamard product, $W_{si}$, $W_{hi}$, ..., $W_{ho}$ are the filter weights, $b_i$, $b_f$, ..., $b_o$ are the bias, $\mathbf{i}^{(k)}$, $\mathbf{f}^{(k)}$, $\mathbf{o}^{(k)}$ denote the input gate, the forget gate and the output gate respectively, $\sigma(\cdot)$ and $\tanh(\cdot)$ denote the sigmoid function and the tanh function respectively. $\mathbf{c}^{(k)}$ acts as an accumulator of the state information and $\mathbf{h}^{(k)}$ is further controlled by the lastest cell outputs $\mathbf{c}^{(k)}$ and the output gate $\mathbf{o}^{(k)}$.

$\mathbf{h}^{(k)}$ is directly utilized as the inputs of $\text{RB}_2(\cdot)$ in the current stage and $[h^{(k)}, c^{(k)}]$ transmits deep features at the same position across stages to augment the dependency of deep information. And the rest process of achieving CLM in PMM is the same with Eq. (7). So the process of $\mathbf{x}^{(k)} = \mathcal{H}^{(k)}_{\text{MAPMM-CLM}}(\mathbf{r}^{(k)})$ can be reformulated as:

\[
\mathbf{x}^{(k)} = \mathbf{r}^{(k)} + \text{Conv}_2(\text{RB}_2(\text{ConvLSTM}(\text{RB}_1(\text{Conv}_1(r^{(k)}))))).
\]

Therefore, incorporating both HSM $\mathbf{z}^{(k-1)}$ and CLM $[h^{(k-1)}, c^{(k-1)}]$ into PMM and with the inputs $\mathbf{r}^{(k)}$ in the $(k-1)$-th stage, the proposed memory-augmented proximal mapping module (MAPMM) in Figure 2 (c) and denoted by $\mathbf{x}^{(k)} = \mathcal{H}^{(k)}_{\text{MAPMM}}(\mathbf{r}^{(k)})$ is formulated as:

\[
\mathbf{x}^{(k)} = \mathbf{r}^{(k)} + \text{Conv}_2(\text{RB}_2(\text{ConvLSTM}(\text{RB}_1(\text{Conv}_3(r^{(k)} \| z^{(k-1)})))))
\]

Comparing Eq. (15) with Eq. (7), it is obvious to observe that $\mathbf{z}^{(k-1)}$ is able to effectively add high-throughput information between each two adjacent stages, while $[h^{(k-1)}, c^{(k-1)}]$ can augment mid/high-frequency information by applying ConvLSTM layers across cascading stages. Therefore, our proposed MAPMM enhances information transmission among all stages by adaptively learning these two types of memory mechanisms.

### 3.4 Initialization of HSM and CLM

Given the measurements $\mathbf{y}$ as the known information, HSM $\mathbf{z}^{(k)}$ is initialized by applying a light-weight one-convolution layer $\text{Conv}_{0}(\cdot)$ on the image $\Phi^{T} \mathbf{y}$ to generate $C$ feature maps, yielding:

\[
\mathbf{z}^{(0)} = \text{Conv}_{0}(\Phi^{T} \mathbf{y}).
\]

Considering that CLM $[h^{(k)}, c^{(k)}]$ has its physical meaning and it is practical to assume no prior knowledge is used at the beginning, therefore, we simply initialize $[h^{(k)}, c^{(k)}]$ to be zero.

### 3.5 Network Parameters and Loss Function

The learnable parameter set in MADUN, denoted by $\Theta$, can be expressed as $\Theta = \{\text{Conv}_{1}(\cdot), \mathcal{H}^{(k)}_{\text{MAPMM}}(\cdot)\}_{k=1}^{K} \cup \{\text{Conv}_{0}(\cdot)\}$. Conv$_{0}(\cdot)$ is a one-convolution layer with one input channel and $C$ output channels. $\mathcal{H}^{(k)}_{\text{MAPMM}}(\cdot)$ consists of Conv$_{(1)}$, Conv$_{(2)}$, RB$_{(1)}$, RB$_{(2)}$ and ConvLSTM$(\cdot)$ whose parameters are different in each stage. Conv$_{(2)}$ with $C + 1$ input channels and $C$ output channels receives
networks inputs, and Conv$_2(\cdot)$ with C input channels and one output channel outputs residual results. Also, the parameters of RB$_1(\cdot)$ and RB$_2(\cdot)$ are both from two convolution layers without the change of the channel number (with C input channels and C output channels), while ConvLSTM$(\cdot)$ is same with [19]. Here, all the convolutions in MADUN adopt $3 \times 3$ filter kernels.

Given a set of full-sampled images $\{x_j\}_{j=1}^{N_0}$ and some sampling patterns with CS ratio $\gamma$, the under-sampled $k$-space data is obtained by $y_j = \Phi x_j$, producing the train data pairs $\{(y_j, x_j)\}_{j=1}^{N_0}$. Our MADUN takes $y_j$ as inputs and generates the reconstruction result $x_j^{(K)}$ as outputs with the initialization $x_j^{(0)} = \Phi^\top y_j$. The loss function is designed to use $L_1$ loss between $x_j$ and $x_j^{(K)}$ as:

$$L(\Theta) = \frac{1}{N N_0} \sum_{j=1}^{N_0} \| x_j - x_j^{(K)} \|_1,$$

where $N_0$, $N$ and $K$ represent the number of training images, the size of each image and the stage number of MADUN respectively.

### 4 EXPERIMENT

#### 4.1 Implementation Details

We use the 400 training images of size $180 \times 180$ [3], generating the training data pairs $\{(y_j, x_j)\}_{j=1}^{N_0}$ by extracting the luminance component of each image block of size $33 \times 33$, i.e. $N = 1,089$. Meanwhile, we apply in the data augmentation technique to increase the data diversity. For a given CS ratio, the corresponding measurement matrix $\Phi \in \mathbb{R}^{M \times N}$ is constructed by generating a random Gaussian matrix or a jointly learned matrix and then orthogonalizing its rows, i.e. $\Phi \Phi^\top = I$, where $I$ is the identity matrix. Applying $y_j = \Phi x_j$ yields the set of CS measurements, where $x_j$ is the vectorized version of an image block.

To train the network, we use Adam optimization [13] with a learning rate of 0.0001, and a batch size of 64. We also use momentum of 0.9 and weight decay of 0.999. For MADUN, the models are trained with 410 epochs separately for each CS ratio. Each image block of size $33 \times 33$ is sampled and reconstructed independently for the first 400 epochs, and for the last ten epochs, we adopt larger image blocks of size $99 \times 99$ as inputs to further fine-tune the model.

To alleviate blocking artifacts, we firstly unfold the blocks of size $33 \times 33$ and then fold the blocks of size $99 \times 99$ as inputs to further fine-tune the model. For MADUN, the models are trained with 410 epochs separately for each CS ratio. Each image block of size $33 \times 33$ is sampled and reconstructed independently for the first 400 epochs, and for the last ten epochs, we adopt larger image blocks of size $99 \times 99$ as inputs to further fine-tune the model.

For MADUN, the models are trained with 410 epochs separately for each CS ratio. Each image block of size $33 \times 33$ is sampled and reconstructed independently for the first 400 epochs, and for the last ten epochs, we adopt larger image blocks of size $99 \times 99$ as inputs to further fine-tune the model.
Table 3: Average PSNR/SSIM performance comparisons on Set11, CBSD68 and Urban100 datasets with different CS ratios. We compare MADUN with two DNUNs and six DUNs with fixed random gaussian matrix. The best and second best results are highlighted in red and blue colors, respectively.

| Dataset   | Methods       | 10%       | 25%       | 30%       | 40%       | 50%       |
|-----------|---------------|-----------|-----------|-----------|-----------|-----------|
|           |               | CS Ratio  |           |           |           |           |
|           |               | 10%       | 25%       | 30%       | 40%       | 50%       |
| Set11     | ReconNet [16] | 23.96/0.7172 | 26.38/0.7883 | 28.20/0.8424 | 30.02/0.8837 | 30.62/0.8983 |
|           | DPA-Net [32]  | 27.66/0.8530 | 32.38/0.9311 | 33.35/0.9425 | 35.21/0.9580 | 36.80/0.9685 |
|           | IRCNN [46]    | 23.05/0.6789 | 28.42/0.8382 | 29.55/0.8606 | 31.30/0.8898 | 32.59/0.9075 |
|           | ISTA-Net* [40]| 26.58/0.8066 | 32.48/0.9242 | 33.81/0.9395 | 36.04/0.9581 | 38.06/0.9706 |
|           | DPDNN [5]     | 26.23/0.7992 | 31.71/0.9153 | 33.16/0.9338 | 35.29/0.9534 | 37.63/0.9693 |
|           | GDN [8]       | 23.90/0.6927 | 29.20/0.8600 | 30.26/0.8833 | 32.31/0.9137 | 33.31/0.9285 |
|           | MAC-Net [2]   | 27.68/0.8182 | 32.91/0.9244 | 33.96/0.9372 | 35.94/0.9560 | 37.67/0.9688 |
|           | iPiano-Net [30]| 28.05/0.8460 | 33.53/0.9359 | 34.78/0.9472 | 37.01/0.9631 | 38.94/0.9737 |
|           | MADUN         | 29.44/0.8807 | 34.88/0.9496 | 36.07/0.9582 | 38.13/0.9700 | 39.92/0.9779 |
| CBSD68    | ReconNet [16] | 24.02/0.6414 | 26.01/0.7498 | 27.20/0.7999 | 28.71/0.8409 | 29.32/0.8642 |
|           | DPA-Net [32]  | 25.47/0.7372 | 29.01/0.8595 | 29.73/0.8827 | 31.17/0.9156 | 32.55/0.9386 |
|           | IRCNN [46]    | 23.07/0.5580 | 26.44/0.7206 | 27.31/0.7543 | 28.76/0.8042 | 30.00/0.8398 |
|           | ISTA-Net* [40]| 25.37/0.7020 | 29.28/0.8513 | 30.39/0.8807 | 32.21/0.9171 | 34.27/0.9455 |
|           | DPDNN [5]     | 23.41/0.6011 | 27.11/0.7636 | 27.52/0.7745 | 30.14/0.8649 | 30.88/0.8763 |
|           | GDN [8]       | 25.80/0.7024 | 29.42/0.8469 | 30.28/0.8713 | 32.02/0.9085 | 33.68/0.9352 |
|           | MAC-Net [2]   | 26.34/0.7431 | 30.16/0.8711 | 31.24/0.8964 | 33.14/0.9298 | 34.98/0.9521 |
|           | iPiano-Net [30]| 26.83/0.7620 | 30.81/0.8844 | 31.87/0.9078 | 33.81/0.9376 | 35.82/0.9587 |
|           | MADUN         | 21.49/0.6223 | 23.31/0.7107 | 24.72/0.7697 | 26.44/0.8250 | 27.06/0.8447 |
|           | DPA-Net [32]  | 24.55/0.7841 | 28.80/0.8944 | 29.47/0.9034 | 31.09/0.9311 | 32.08/0.9447 |
|           | IRCNN [46]    | 21.62/0.6137 | 26.38/0.7955 | 27.47/0.8248 | 29.22/0.8645 | 30.63/0.8900 |
|           | ISTA-Net* [40]| 23.61/0.7238 | 28.93/0.8840 | 30.21/0.9079 | 32.43/0.9377 | 34.43/0.9571 |
|           | DPDNN [5]     | 23.69/0.7211 | 28.70/0.8798 | 30.06/0.9005 | 32.27/0.9350 | 34.81/0.9579 |
|           | GDN [8]       | 21.48/0.5958 | 25.75/0.7704 | 26.72/0.7950 | 28.96/0.8698 | 29.89/0.8768 |
|           | MAC-Net [2]   | 24.21/0.7445 | 28.79/0.8798 | 29.99/0.9017 | 31.94/0.9272 | 34.03/0.9513 |
|           | iPiano-Net [30]| 25.67/0.7963 | 30.87/0.9157 | 32.16/0.9320 | 34.27/0.9531 | 36.22/0.9675 |
|           | MADUN         | 27.13/0.8393 | 32.54/0.9347 | 33.77/0.9472 | 35.80/0.9633 | 37.75/0.9746 |

4.2 Ablation Study

4.2.1 Performance Effect of Stage Number. The average PSNR curve in Figure 3 is the results when ratio is 25% on Set11 dataset. To get a better trade-off between model performance and complexity, we choose $K = 25$ as the default setting in all experiments.

4.2.2 Performance Effect of MADUN Components. Based on the PMM structure, we do ablation experiments on the HSM and CLM at ratio=10% on Set11 and Urban100 dataset, as shown in Table 1. To demonstrate the effectiveness of HSM and CLM, we remove them from MADUN to form two simplified versions, corresponding to Figure 2 (c) and Figure 2 (d) respectively. We conclude that they are all effective ways to compensate the signal flow among stages, and HSM plays a more important role by bridging each two adjacent stages with a high-throughput short-term transmission.

4.3 Analysis of Memory Augmentation

We now separately illustrate how our different memory augmentation mechanisms affect information transmission.

For HSM, high-throughput information which is added to the original one-channel short-memory between each two adjacent stages significantly reduces information loss and improves performance, as shown in Table 1. For further analysis of the effect on different positions of linking memory in HSM, we make two comparative experiments (c) and (d), where $\circ$HSM represents that the outputs of Conv$_3$ (·) are the points of memory transmission and $\bullet$HSM denotes the outputs of RB$_1$ (·). As we can see, our default version of HSM transmits more refined memory than others.

And for CLM, we do experiments in different memory methods to prove the superiority of ConvLSTM, as shown in Table 2. In Figure 4, on the basis of PMM, (a) is the "Plus" strategy which takes a direct addition operation, (b) is the "Concat" strategy which adopts a single convolution layer to merge the all memory into the main intermediate features, and (c) is the implementation of CLM. CLM achieves better performance than others, which reveals that information is selectively retained by applying LSTM is useful for CS. To get deeper insights of the long-term information flow established by our CLM, as Figure 5 illustrates, we give the average kernel weight norms of the three convolution gates in each CLM, and plot the 1-D spectral density curves of memory features $\mathbf{h}(k)$ in different CLM variants by integrating the 2-D power spectrums along each concentric circle [33] (i.e. by averaging the feature components with the same frequency in the DCT domain and normalizing the frequency range into [0, 1]). From the upper subplot, we observe that the weight norms of the three convolution gates increase as the stage index increases, and the average norm of the forget gate is larger than the other two in most stages, which means that the cell
outputs $c^{(k)}$ get close attention in the main trunk of MADUN and CLM plays a more important role in later stages. From the spectral density curves, we can see that, without long-term memory makes the network hard to keep the mid/high-frequency information; the “Concat” strategy also causes the information loss due to its weak adaptability; the “Plus” strategy introduces much feature space noise and weakens the proximal mapping process. However, our CLM with a flexible gated memory mechanism, achieves a better balance in the long-term information flow, which obtains the higher performances compared with others.

### 4.4 Qualitative Evaluation

We conduct two types of experiments on fixed random Gaussian sampling matrix and jointly learned sampling matrix, and compare them with the corresponding methods respectively.
### Methods

| Dataset | Methods       | CS Ratio | 10% | 25% | 30% | 40% | 50% |
|---------|---------------|----------|-----|-----|-----|-----|-----|
| Set11   | CSNet⁺ [27]   | 28.34/0.8580 | 33.34/0.9387 | 34.27/0.9492 | 36.44/0.9690 | 38.47/0.9796 |
|         | [28]          | 38.52/0.8616 | 33.43/0.9373 | 34.64/0.9511 | 36.92/0.9666 | 39.01/0.9769 |
|         | BCS-Net⁺ [54] | 29.42/0.8673 | 34.20/0.9408 | 35.63/0.9495 | 36.68/0.9667 | 39.58/0.9734 |
|         | [41]          | 39.81/0.8904 | 34.86/0.9509 | 35.79/0.9541 | 37.96/0.9633 | 40.19/0.9800 |
|         | AMP-Net⁺ [48]  | 29.42/0.8782 | 34.60/0.9469 | 35.91/0.9541 | 37.96/0.9633 | 40.26/0.9786 |
|         | MADUN         | 29.91/0.8906 | 35.66/0.9601 | 36.94/0.9676 | 39.13/0.9772 | 40.77/0.9832 |
| CBSD68  | CSNet⁺ [27]   | 27.91/0.7938 | 31.12/0.9060 | 32.20/0.9220 | 35.01/0.9258 | 36.76/0.9638 |
|         | [28]          | 28.02/0.8042 | 31.15/0.9058 | 32.64/0.9237 | 35.03/0.9214 | 36.27/0.9593 |
|         | BCS-Net⁺ [54] | 27.98/0.8015 | 31.29/0.8846 | 32.70/0.9301 | 35.14/0.9397 | 36.85/0.9682 |
|         | [41]          | 27.82/0.8045 | 31.51/0.9061 | 32.35/0.9215 | 34.95/0.9261 | 36.35/0.9660 |
|         | AMP-Net⁺ [48]  | 27.79/0.7853 | 31.37/0.8749 | 32.68/0.9291 | 35.06/0.9395 | 36.59/0.9620 |
|         | MADUN         | 28.15/0.8229 | 32.66/0.9221 | 33.35/0.9379 | 35.42/0.9606 | 37.11/0.9730 |

### Table 5: Average PSNR/SSIM performance comparisons on testing brain MR images with one DNUN and six DUNs.

| CS Ratio | Hyun et al. [10] | Schlemper et al. [26] | ADMM-Net [37] | RDN [31] | CDDN [53] | ISTA-Net⁺ [40] | MoDL [1] | MADUN |
|----------|-----------------|----------------------|----------------|----------|-----------|----------------|---------|-------|
| 10%      | 32.78/0.8385    | 34.23/0.8921         | 34.42/0.8971   | 34.59/0.8968 | 34.63/0.9002 | 34.65/0.9038 | 35.16/0.9091 | 36.13/0.9237 |
| 20%      | 36.36/0.9070    | 38.47/0.9457         | 38.60/0.9478   | 38.58/0.9470 | 38.59/0.9474 | 38.67/0.9480 | 38.51/0.9457 | 39.44/0.9542 |
| 30%      | 38.85/0.9383    | 40.85/0.9628         | 40.87/0.9633   | 40.82/0.9625 | 40.89/0.9633 | 40.91/0.9631 | 40.97/0.9636 | 41.48/0.9666 |
| 40%      | 40.65/0.9539    | 42.63/0.9724         | 42.58/0.9726   | 42.62/0.9723 | 42.59/0.9725 | 42.65/0.9727 | 42.38/0.9705 | 43.06/0.9746 |
| 50%      | 42.35/0.9662    | 44.19/0.9794         | 44.19/0.9796   | 44.18/0.9793 | 44.15/0.9795 | 44.24/0.9798 | 44.20/0.9776 | 44.60/0.9810 |

### 4.4.1 Fixed Random Gaussian Matrix

We compare our proposed MADUN with eight recent representative CS reconstruction methods, including two DNUNs and six DUNs. The average PSNR reconstruction performances on Set11, CBSD68 and Urban100 dataset with respect to five CS ratios are summarized in Table 3. The models of ReconNet [16], ISTA-Net⁺ [40], DPA-Net [32], IRCNN [46], MAC-Net [2] and iPiano-Net [30] are trained by the same methods and training datasets with the corresponding works, and DPDNN [5] and GDN [8] utilize the same training dataset with our method due to no CS reconstruction task in their original works. One can observe that our MADUN outperforms all the other competing methods in PSNR and SSIM across all the cases. Figure 6 further show the visual comparisons on challenging images on Set11 dataset and Urban100 dataset, respectively. Our MADUN generates images that are visually pleasant and faithful to the ground-truth.

### 4.4.2 Jointly Learned Matrix

We mimic the CS sampling process \( y = \Phi x \in \mathbb{R}^M \) using a convolution layer, and utilize another convolution layer with N filters from \( \Phi^\top \in \mathbb{R}^{NxM} \) to obtain initialization \( \Phi^\top y \), like [41]. We compare MADUN with some methods which jointly learn the sampling matrix and the reconstruction process. Table 4 presents quantitative results on Set11 and CBSD68 dataset. As we can see, our MADUN achieves the best performance on all ratios. Figure 7 further shows the visual comparisons on challenging images, one can see that MADUN recovers richer image structures and texture details than the other methods.

### 4.5 Application to Compressive Sensing MRI

To demonstrate the generality of MADUN, we directly extend MADUN to the specific problem of CS-MRI reconstruction, which aims at reconstructing MR images from a small number of under-sampled data in k-space. In this application, we set the sampling process \( \Phi \) in Eq. (5) to \( \Phi = BF \), where \( B \) is an under-sampling matrix and \( F \) is the discrete Fourier transform. In this case, we compare against seven recent methods for the CS-MRI domain. Utilizing the same training and testing brain medical images as ADMM-Net [37], the CS-MRI results of MADUNs are summarized in Table 4 for different CS ratios. From Table 5, we can see that our proposed method outperforms the state-of-the-art methods on testing brain dataset with all tested CS ratios, especially when ratios are smaller.

### 5 CONCLUSION

We propose a novel Memory-Augmented Deep Unfolding Network (MADUN) for CS, which addresses the problem of the information-lossy short-term transmission between each two adjacent stages in traditional DUNs by establishing and strengthening both the short-term and long-term memory flows with large capacity. Our two types of memory augmentation mechanisms, dubbed High-throughput Short-term Memory (HSM) and Cross-stage Long-term Memory (CLM), are developed and integrated into our Memory-Augmented Proximal Mapping Module (MAPMM). HSM and CLM collaboratively achieve persistent and adaptive memory by bridging adjacent stages with the multi-channel signal path and developing the dependency of deep information across all cascading stages, respectively. Extensive CS experiments on both natural and MR images exhibit that the proposed MADUN achieves better signal balance based on both memory mechanisms and outperforms existing state-of-the-art deep network-based methods with large margins. In the future, we will further extend our MADUN to other image inverse problems and video applications.
[49] Chen Zhao, Siwei Ma, and Wen Gao. 2014. Image compressive-sensing recovery using structured laplacian sparsity in DCT domain and multi-hypothesis prediction. In Proceedings of IEEE International Conference on Multimedia and Expo (ICME).

[50] Chen Zhao, Siwei Ma, Jian Zhang, Ruixin Xiong, and Wen Gao. 2016. Video compressive sensing reconstruction via reweighted residual sparsity. IEEE Transactions on Circuits and Systems for Video Technology 27, 6 (2016), 1182–1195.

[51] Chen Zhao, Jian Zhang, Siwei Ma, and Wen Gao. 2016. Nonconvex lp nuclear norm based admm framework for compressed sensing. In Proceedings of Data Compression Conference (DCC).

[52] Chen Zhao, Jian Zhang, Ronggang Wang, and Wen Gao. 2018. CREAM: CNN-Regularized ADMM framework for compressively-sensed image reconstruction. IEEE Access 6 (2018), 76838–76853.

[53] Hao Zheng, Faming Fang, and Guixu Zhang. 2019. Cascaded dilated dense network with two-step data consistency for MRI reconstruction. In Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS).

[54] Siwang Zhou, Yan He, Yonghe Liu, Chengqing Li, and Jianming Zhang. 2020. Multi-channel deep networks for block-based image compressive sensing. IEEE Transactions on Multimedia (2020).