LR-CSNet
Low-Rank Deep Unfolding Network for Image Compressive Sensing
Zhang, Tianfang; Li, Lei; Igel, Christian; Oehmcke, Stefan; Gieseke, Fabian; Peng, Zhenming

Published in:
2022 IEEE International Conference on Computer and Communications (ICCC), Chengdu, China

DOI:
10.1109/ICCC56324.2022.10065722

Publication date:
2023

Document version
Early version, also known as pre-print

Document license:
CC BY

Citation for published version (APA):
LR-CSNet: Low-Rank Deep Unfolding Network for Image Compressive Sensing

1st Tianfang Zhang1,2, 2nd Lei Li2, 3rd Christian Igel2, 4th Stefan Oehmcke2, 5th Fabian Gieseke2, 6th Zhenming Peng1,*
1School of Information and Communication Engineering, University of Electronic Science and Technology of China
2Computer Science, University of Copenhagen
sparkcarleton@gmail.com, lilei@di.ku.dk, igel@di.ku.dk
stefan.oehmcke@di.ku.dk, fabian.gieseke@di.ku.dk, zmpeng@uestc.edu.cn

Abstract—Deep unfolding networks (DUNs) have proven to be a viable approach to compressive sensing (CS). In this work, we propose a DUN called low-rank CS network (LR-CSNet) for natural image CS. Real-world image patches are often well-represented by low-rank approximations. LR-CSNet exploits this property by adding a low-rank prior to the CS optimization task. We derive a corresponding iterative optimization procedure using variable splitting, which is then translated to a new DUN architecture. The architecture uses low-rank generation modules (LRGMs), which learn low-rank matrix factorizations, as well as gradient descent and proximal mappings (GDPMs), which are proposed to extract high-frequency features to refine image details. In addition, the deep features generated at each reconstruction stage in the DUN are transferred between stages to boost the performance. Our extensive experiments on three widely considered datasets demonstrate the promising performance of LR-CSNet compared to state-of-the-art methods in natural image CS.

Index Terms—Image Compressive Sensing, Deep Learning, Deep Unfolding Network, Low-Rank Prior, Image Restoration

I. INTRODUCTION

Compressive sensing (CS) has become an important tool in modern signal processing. It allows to identify sparse solutions of underdetermined linear systems [4]. Under the assumption that the original signal is sparse in some transform domain [17], CS requires fewer measurements to reconstruct the original signal than expected by the Nyquist sampling theorem [4]. Compressive sensing methods have successfully been applied in various fields, including single-pixel cameras [5], magnetic resonance imaging [16], and seismic imaging [8].

Mathematically, for an original signal \( \bar{x} \in \mathbb{R}^N \), the observation \( y = \Phi \bar{x} \in \mathbb{R}^M \) is obtained after sampling through a measurement matrix \( \Phi \in \mathbb{R}^{M \times N} \), where \( M \ll N \). Here, \( \frac{M}{N} \) is denoted as the so-called CS ratio. Given a matrix \( \Phi \) and \( y \), the goal of compressed sensing is to reconstruct \( \bar{x} \) under some sparseness assumptions, such as structural sparsity [23].

Corresponding author: Zhenming Peng
This work was supported by Natural Science Foundation of Sichuan Province of China (Grant No.2022NSFSC040574) and partially by National Natural Science Foundation of China (Grant No.61775030, Grant No.61571096).

Note that for \( \bar{x} \in \mathbb{R}^{\sqrt{N} \times \sqrt{N}} \) and \( \text{vec}(\bar{x}) \in \mathbb{R}^N \) being the vectorization of \( \bar{x} \) we have \( \|\bar{x}\| = \|\text{vec}(\bar{x})\|_2 \).

dictionary sparsity [20], and low-rankness [21]. We can write CS as an optimization problem of the form

\[
\hat{x} = \arg \min_{x \in \mathbb{R}^N} \left[ \frac{1}{2} \| \Phi x - y \|_F^2 + \lambda \mathcal{G}(x) \right]
\]

for \( y = \Phi \bar{x} \), where \( \|\cdot\|_F \) is the Frobenius-norm, \( \mathcal{G} \) a sparseness constraint function, and \( \lambda \) controls the penalty strength.

In recent years, with the advancement of neural networks, data-driven CS reconstruction methods have made great progress. In general, they can be divided into two categories: deep non-unfolding networks (DNUNs) and deep unfolding networks (DUNs). DNUNs learn the mapping between the observed signal \( y \) and the original signal \( \bar{x} \) directly from training examples [22]. In contrast, DUNs consider the optimization problem given by Eq. (1) and map the iterative optimization algorithm used to solve Eq. (1) to a deep neural network architecture. Usually, \( K \) optimization steps are mimicked by means of \( K \) sequential blocks (reconstruction stages) in the network [34]. DUNs learn the matrix \( \Phi \), the regularization
function $G$, and parameters of the underlying optimization process simultaneously end-to-end by minimizing an objective function of the form

$$\frac{1}{2\ell} \sum_{i=1}^{\ell} \| \hat{x}_i - f_{\text{DUN}}(\Phi \hat{x}_i) \|_F^2,$$ (2)

where $f_{\text{DUN}}$ denotes the neural network and $\{\hat{x}_i\}_{i=1}^{\ell}$ are $\ell$ training instances. The architecture of $f_{\text{DUN}}(\hat{x}_i)$ results from unfolding the iterative optimization of Eq. (1), the parameters of the network that encode (among others) $\Phi$, as well as $G$ (in our case, $\nabla G$). Thus, in contrast to model-based CS, the measurement matrix as well as the sparsity regularization are not given a priori but are learned from data. Because of their excellent reconstruction performance [29], [30], DUNs have become state of the art in image CS.

However, DUNs usually constrain the signals $x$ to be sparse in some transform domain, ignoring other intrinsic properties, such as low-rankness. The manipulation of image patches in CS has become a common practice making the low-rank property more prominent. The singular value curves of several image patches are shown in Fig. [1] The trend indicated in the graph and the convergence of the singular values to 0 indicate the low-rankness property of the images at hand. We extend Eq. (1) by an additional term that reflects the low-rankness of the input signals, i.e.,

$$\hat{x} = \arg \min_{x \in \mathbb{R}^N} \left[ \frac{1}{2} \| \Phi x - y \|_F^2 + \lambda G(x) + \mu \mathcal{R}(x) \right],$$ (3)

where $\mathcal{R}(x)$ is the function increasing in the rank of the signal $x$ and $\mu$ controls the penalty degree.

In this paper, we propose an optimization-based deep unfolding network for image compressive sensing, dubbed LR-CSNet, by exploring the low-rank prior of the input images from the perspective of neural networks. Our main contributions can be summarized as follows:

1) For the problem formulation, we establish an achievable constraint on the low-rank component and demonstrate its iterative optimization process by variable splitting.

2) We propose LR-CSNet to simulate the iterative optimization process into multiple reconstruction stages and learn an end-to-end mapping between observations and original signal.

3) We design a low-rank generation module (LRGM) to learn the low-rank components as well as gradient descent and proximal mapping (GDPM) to refine details of the reconstructed image. Furthermore, we enhance the network representation by feature transmission.

4) We demonstrate via extensive experiments that LR-CSNet exhibits a superior performance on natural image datasets compared to state-of-the-art approaches.

II. RELATED WORK

A. Deep Unfolding Networks

DUNs emulate iterative optimization methods through neural networks and have been successfully applied to image inverse problems [14]. For CS, neural networks were combined with the alternating direction method of multipliers (ADMM) for efficient MRI reconstruction [26]. [1] learn the sparse linear inverse problem from the perspective of approximate message passing (AMP) and follow-up work showed intensive studies [34] on image CS. The ISTA-Net+ [29] focuses on modelling the iterative shrinkage thresholding algorithm (ISTA) with neural networks, whereas the OPINE-Net [30] obtained satisfactory results using a binary trainable sampling matrix. While the other models were trained with a fixed CS ratio, the ISTA-Net++ [27] model trains at multiple CS ratios, reducing computational cost. Finally, COAST [28] is able to handle arbitrary sampling matrices and achieves promising results.

B. Low-Rank Representation

Low-rank representations characterize high-dimensional data with fewer vectors and effectively reveal the overall data structure [11], [13]. They are widely applied in the fields such as image restoration [6], [7], background modelling [25], infrared small target detection [24], [31], [32], and image compressive sensing [21]. Whereas, due to the non-convex nature, its optimal convex approximation is widely adopted, i.e. the nuclear norm $\| x \|_* = \sum \sigma_i$, where $\sigma$ is the singular value. Even though low-rank representations helped model-driven approaches to achieve great success, the necessity of the singular value decomposition (SVD) greatly limits computational efficiency. The SVD also complicates the integration of the low-rankness condition into a neural network. [2] consider the low-rank term to be the product of two sub-matrices. Inspired by the tensor CP (CANDECOMP/PARAFAC) decomposition, Chen et al. [3] treats a tensor of rank $r$ as the sum of multiple rank unit tensors and apply it as attention to semantic segmentation. Although this problem has been partially studied, it remains a challenging task to learn low-rank priors more efficiently and mapping them into DUNs reasonably.

III. PROPOSED LR-CSNet

In this section, we first define the low rank constrained CS problem and then present our specific update process in terms of optimization. Thereafter, we describe the process of mapping the optimization into neural networks and give details on the LR-CSNet. Finally, the training parameters and loss function are described in general.

We use the following notation. Plain font $\rho$ indicates scalars, bold lowercase $\mathbf{x}$ indicates matrices and vectors, bold capital $\mathbf{F}$ indicates deep features, and calligraphic font $\mathcal{G}$ indicates functions.

A. Problem Formulation

We constrain the L2 norm of the low-rank component $\mathbf{I}$ and signal $\mathbf{x}$, rather than using the nuclear norm directly, in order to circumvent the costly SVD, i.e. $\mathcal{R}(\mathbf{x}) = \frac{1}{2} \| \mathbf{x} - \mathbf{I} \|_F^2$. Then we perform variable splitting to reduce the complex operations during optimization, specifically, we introduce an auxiliary variable $\mathbf{z}$ as follows:
\[
\begin{align*}
\mathcal{L}(x, z) &= \frac{1}{2} \|\Phi z - y\|_F^2 + \lambda \mathcal{G}(x) + \frac{\mu}{2} \|z - l\|_F^2 + \frac{\beta}{2} \|x - z\|_F^2 \\
&= \frac{1}{2} \|\Phi z - y\|_F^2 + \frac{1}{4} \|\Phi z - y\|_F^2 + \lambda \mathcal{G}(x) + \frac{\mu}{2} \|z - l\|_F^2 + \frac{\beta}{2} \|x - z\|_F^2 \\
&\quad + \|l\|_F^2 - \frac{1}{4} \|\Phi z - y\|_F^2 + \lambda \mathcal{G}(x) + \frac{\mu}{2} \|z - l\|_F^2 + \frac{\beta}{2} \|x - z\|_F^2.
\end{align*}
\]

In order to avoid the complex operations such as matrix inversion that occur in the update process, we perform the Taylor expansion at \(z^{k-1}\) for the first term in Eq. (6), i.e. we replace \(\frac{1}{2} \|\Phi z - y\|_F^2\) by \(\frac{1}{4} \|\Phi z - y\|_F^2 + \lambda \mathcal{G}(x) + \frac{\mu}{2} \|z - l\|_F^2 + \frac{\beta}{2} \|x - z\|_F^2\) and \(\|l\|_F^2\), where \(\mathcal{G}(x)\) is not specifically mathematically constrained. Similarly, we perform a Taylor expansion of \(\mathcal{G}(x)\) at \(x^{k-1}\), which is converted into the \(\nabla \mathcal{G}(x)\) form with L2 norm constraints and arrive at

\[
x^k = \frac{\lambda \mathcal{G}(x)}{\lambda \mathcal{G}(x) + \|x - z^k\|_F^2}.
\]

Fig. 2. Illustration of the LR-CSNet network architecture and modules, which includes \(K\) reconstruction stages in total. From top to bottom is the overall network architecture, LRGM as well as GDPM in the \(k\)th reconstruction stage, with the legend and dense block (DB) in GDPM on the right. Specifically, \(v^k = \rho_1^k x^{k-1} + \rho_2^k z^{k-1} + (1 - \rho_1^k - \rho_2^k) l^k\). + and − indicates the positive and negative sign of the branch when adding up.
c) Overall: In end-to-end learning, we can set complex penalty parameters as learnable variables, so the overall optimization steps are
\[
\begin{align*}
\mathbf{z}^k &= \rho_1 \mathbf{z}^{k-1} + \rho_2 \mathbf{x}^{k-1} + (1 - \rho_1 - \rho_2) \mathbf{I}^k - \eta \Phi^T \mathbf{F} \mathbf{z}^k - \eta \Phi^T \mathbf{y} \quad (10) \\
\mathbf{x}^k &= \alpha_k \mathbf{z}^{k-1} + (1 - \alpha_k) \mathbf{x}^k - \gamma_k \nabla \mathcal{G} (\mathbf{x}^{k-1})
\end{align*}
\]
where \( \rho_1 = \frac{2\beta}{l_1 + 2\mu + 2\beta}, \rho_2 = \frac{l_1 + 2\mu + 2\beta}{l_1 + 2\mu + 2\beta}, \eta = \frac{1}{l_1 + 2\mu + 2\beta}, \alpha = \frac{\lambda_2}{\lambda_2 + \gamma}, \) and \( \gamma = \frac{\lambda_2}{\lambda_2 + \gamma}. \) These parameters are trained independently in each reconstruction stage.

B. Network Architecture

In this section we elaborate on the network architecture and module design of LR-CSNet based on the optimization process of Eq. (10). As shown in Fig. 2, given an original signal \( \mathbf{x} \in \mathbb{R}^{N \times N} \), we perform sampling and end-to-end image reconstruction through the network.

During sampling, the original image \( \mathbf{x} \) passes through a convolutional layer with a kernel size and step size of 33, where the input and output channels are 1 and \( M \), respectively. In this way, the sampling process for \( \mathbf{y} = \Phi \mathbf{x} \) is simulated and the observation \( \mathbf{y} \in \mathbb{R}^{1 \times 1 \times M} \) is obtained.

In the reconstruction phase, \( \mathbf{y} \) is passed through convolutional layers with kernel size and stride of 1, where the input and output channels are \( M \) and \( 33 \times 33 \) respectively. This operation is used to simulate \( \mathbf{x}^0 = \Phi^T \mathbf{y} \), where the convolutional layer share weights with the one in the sampling process. Then the reconstructed signal is reshaped to \( \mathbf{x}^0 \in \mathbb{R}^{N \times N} \) by PixelShuffle(33) [30]. The reconstructed image is then passed through \( K \) reconstruction stages to simulate the iterative updates. Each reconstruction stage consists of two modules: LRGM and GDPM.

1) Low-Rank Generation Module (LRGM): LRGM is used to generate the low-rank matrix \( \mathbf{I}^k \) of the current stage, which contains the majority of the information in the background. A low-rank matrix can be considered as the result of multiplying two sub-matrices together, i.e. \( \mathbf{I} = \mathbf{pq} \), where \( \mathbf{p} \in \mathbb{R}^{\sqrt{N} \times r} \), \( \mathbf{q} \in \mathbb{R}^{r \times \sqrt{N}} \), and \( r \) is the rank number. LRGM takes the updated variables from the previous stage as input and concatenates it with the transferred tensor \( \mathbf{F}^{k-1} \) after one convolutional layer, as shown in Fig. 2. Subsequently, the deep feature is adaptively pooled into two tensors of scale \( \sqrt{N} \times r \times C \) and \( r \times \sqrt{N} \times C \) according to the rank number \( r \) respectively, where \( C \) is the channel number. The two sub-matrices \( \mathbf{p} \) and \( \mathbf{q} \) are obtained through two 1 \( \times 1 \) convolutional layers for feature separation and dimensionality reduction. Finally, these sub-matrices are multiplied to obtain the updated low-rank matrix \( \mathbf{I}^k \). In this way LRGM is able to guarantee that \( \text{rank}(\mathbf{I}^k) \leq r \).

2) Gradient Descent and Proximal Mapping (GDPM): GDPM is used to update the variables \( \mathbf{z}^k \) and \( \mathbf{x}^k \) according to Eq. (10), whereby the scalars are learnable variables. In Fig. 2, after obtaining \( \mathbf{x}^k \), it is concatenated with the low-rank matrix \( \mathbf{I}^k \) and passed through a convolution layer. \( \mathbf{I}^k \) contains more structured image information and can provide guidance for the neural network in learning image details. Then we simulate the function \( \nabla \mathcal{G} \) with two dense blocks (DBs) [33]. Since \( \nabla \mathcal{G} \) is learning high-frequency details in the image, the residual connections by [33] were not applied here. It is worth mentioning that DB is essentially an accumulation of multiple convolutional layers, which clearly satisfies Lipschitz continuous, guaranteeing the validity of this module. In addition, the last deep feature is concatenated with \( \mathbf{F}^{k-1} \) and the transferred tensor \( \mathbf{F}^k \) is updated through a 1 \( \times 1 \) convolutional layer. Finally, the transferred tensor \( \mathbf{F}^k \), the updated variables \( \mathbf{z}^k \), and \( \mathbf{x}^k \) are delivered to the next reconstruction stage.

C. Network Parameter and Loss Function

The trainable parameters in LR-CSNet consist of four components: 1) the same measurement matrix \( \Phi \) in each reconstruction stage, 2) the auxiliary scalars \( \rho_1, \rho_2, \eta, \alpha, \gamma, \) 3) the network weights \( \Theta_i \) in LRGM, and 4) the weights \( \Theta_g \) in GDPM. Thus, all training parameters are denoted as \( \Theta = \{ \Phi \} \cup \{ \rho_1, \rho_2, \eta, \alpha, \gamma \} \cup \{ \Theta_i \} \cup \{ \Theta_g \} \), where \( K \) is the total reconstruction stages. \( \Phi \) and \( \Phi^T \) share weights [30].

The loss function of the network, as it is common practice [30], consists of two components for the given training data \( \{ \mathbf{x}_i \}_{i=1}^\ell \); the fidelity loss \( \mathcal{L}_{\text{fidelity}} \) to ensure that the reconstruction result \( \mathbf{x}_i^k \) closely approximates the input \( \mathbf{x}_i \), and the orthogonal loss \( \mathcal{L}_{\text{orth}} \) to impose an orthogonal constraint on the measurement matrix. The combined loss function is
\[
\mathcal{L}(\Theta) = \mathcal{L}_{\text{fidelity}} + \tau \mathcal{L}_{\text{orth}}
\]
\[
= \frac{1}{N\ell} \sum_{i=1}^\ell \| \mathbf{x}^k_i - \tilde{\mathbf{x}}_i \|_F^2 + \frac{\tau}{M^2} \| \Phi \Phi^T - \mathbf{E} \|_F^2 ,
\]
where \( \ell \) is the total amount of training data, \( \mathbf{E} \) the unit matrix, and \( \tau \) a constant (set to 0.01 for our experimental evaluation).

IV. Experiments

In this section we first give details on the widely applied datasets, the evaluation metrics, and the network implementation. Then, we demonstrate the validity of each module in this paper through extensive ablation studies and investigate the effect of key parameters. Finally, we compare LR-CSNet with other state-of-the-art methods in both quantitative and qualitative aspects to validate the performance of our approach.
TABLE II
QUANTITIVE COMPARISON OF AVERAGE PSNR/SSIM FOR DIFFERENT CS RATIOS ON THE Set11, BSD68, AND Urban100.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ratio</th>
<th>ISTA-Net+</th>
<th>CSNet+</th>
<th>AdapRecon</th>
<th>OPINE-Net+</th>
<th>AMP-Net</th>
<th>LR-CSNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set11</td>
<td>1%</td>
<td>17.42/0.4029</td>
<td>19.87/0.4977</td>
<td>19.63/0.4848</td>
<td>20.02/0.5362</td>
<td>20.04/0.5132</td>
<td>20.85/0.5583</td>
</tr>
<tr>
<td></td>
<td>4%</td>
<td>21.32/0.6037</td>
<td>23.93/0.7338</td>
<td>23.87/0.7279</td>
<td>25.69/0.7920</td>
<td>24.64/0.7527</td>
<td>26.16/0.8040</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>26.64/0.8087</td>
<td>26.04/0.7971</td>
<td>27.39/0.8521</td>
<td>29.81/0.8884</td>
<td>28.84/0.8765</td>
<td>30.35/0.8987</td>
</tr>
<tr>
<td></td>
<td>25%</td>
<td>32.59/0.9254</td>
<td>29.98/0.8932</td>
<td>31.75/0.9257</td>
<td>34.86/0.9509</td>
<td>34.42/0.9153</td>
<td>35.64/0.9573</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>38.11/0.9707</td>
<td>34.61/0.9435</td>
<td>35.87/0.9625</td>
<td>40.17/0.9797</td>
<td>40.12/0.9818</td>
<td>41.03/0.9826</td>
</tr>
<tr>
<td>BSD68</td>
<td>1%</td>
<td>19.14/0.4158</td>
<td>21.91/0.4958</td>
<td>21.50/0.4825</td>
<td>21.88/0.5162</td>
<td>21.97/0.5086</td>
<td>22.32/0.5282</td>
</tr>
<tr>
<td></td>
<td>4%</td>
<td>22.17/0.5486</td>
<td>24.63/0.6564</td>
<td>24.30/0.6491</td>
<td>25.20/0.6825</td>
<td>25.40/0.6985</td>
<td>25.53/0.6972</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>25.32/0.7022</td>
<td>27.02/0.7864</td>
<td>26.72/0.7821</td>
<td>27.82/0.8045</td>
<td>27.41/0.8036</td>
<td>28.21/0.8159</td>
</tr>
<tr>
<td></td>
<td>25%</td>
<td>29.36/0.8525</td>
<td>30.22/0.8918</td>
<td>30.10/0.8901</td>
<td>31.51/0.9061</td>
<td>31.56/0.9121</td>
<td>32.12/0.9162</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>34.04/0.9424</td>
<td>34.82/0.9590</td>
<td>33.60/0.9479</td>
<td>36.35/0.9660</td>
<td>36.64/0.9707</td>
<td>37.29/0.9720</td>
</tr>
<tr>
<td>Urban100</td>
<td>1%</td>
<td>16.90/0.3846</td>
<td>19.26/0.4632</td>
<td>19.14/0.4510</td>
<td>19.38/0.4872</td>
<td>19.62/0.4967</td>
<td>19.65/0.4971</td>
</tr>
<tr>
<td></td>
<td>4%</td>
<td>19.83/0.5377</td>
<td>21.96/0.6430</td>
<td>21.92/0.6390</td>
<td>23.36/0.7114</td>
<td>22.82/0.6963</td>
<td>23.41/0.7210</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>24.04/0.7378</td>
<td>24.76/0.7899</td>
<td>24.55/0.7801</td>
<td>26.93/0.8397</td>
<td>26.05/0.8287</td>
<td>27.41/0.8547</td>
</tr>
<tr>
<td></td>
<td>25%</td>
<td>29.78/0.8954</td>
<td>28.13/0.8827</td>
<td>28.21/0.8841</td>
<td>31.86/0.9308</td>
<td>30.94/0.9273</td>
<td>32.50/0.9391</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>35.24/0.9614</td>
<td>32.97/0.9503</td>
<td>31.88/0.9434</td>
<td>37.23/0.9741</td>
<td>36.54/0.9744</td>
<td>37.87/0.9776</td>
</tr>
</tbody>
</table>

A. Datasets and Evaluation Metrics

We test LR-CSNet on three natural image dataset benchmarks that are widely used in CS: Set11 [12], BSD68 [18], and Urban100 [9]. As training data, we use image patches \( \{ \bar{x}_i \} \) of size 33 × 33 as published in [29], where the total number is \( \ell = 88912 \). For fine-tuning, we train with an additional 36000 image patches of size 99 × 99 from BSD300 [18], which is also publicly available. As evaluation metrics, we choose peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM), which are widely adopted in image restoration, with higher values of both indicating better reconstruction results.

B. Implementation Details

Our implementation is based on PyTorch [19] and all experiments are performed on NVIDIA Titan RTX. We train the network on a set of CS ratios \{1\%, 4\%, 10\%, 25\%, 50\%\}, where we train 150 epochs using 33 × 33 image patches with batch size of 128, followed by a fine-tuning phase of 100 epochs using 99 × 99 image patches with batch size of 32. We optimize the parameters using Adam [10] with a momentum of 0.9 and weight decay of 0.999. The learning rate was set to a constant \( 10^{-4} \).

C. Ablation Study and Parameter Setting

We present the ablation study in Fig. 3 and Table 1 to explore the impact of each module and the changes in key parameters.
1) Impact of LRGM: Table I explores the effectiveness of each module by comparing each possible combination. Removing ’LRGM’ means removing the constraint on low-rank from the problem formulation in Eq. (4), with the derivation and settings remaining the same as before. The results show that LRGM always contributes to the performance.

2) Impact of Dense: Removing ’Dense’ is to replace the dense block in Fig. 2 with 6 convolutional layers, which implies a reduction in network parameter number and feature reuse capability. Again, the results demonstrate a degradation in performance without ’Dense’.

3) Impact of Transmission: Transmission is the integration of deep features from the previous reconstruction stages into the current stage, which theoretically enables a more effective aggregation of information. Removing ’Trans’ is removing the F in Fig. 2. The experiments show that ‘Trans’ does improve the reconstruction accuracy of the network.

4) Rank Number r: From Table III we employ ranks $r \in \{4, 8, 12, 16, 24, 32\}$ for the LRGM module to analyze their impact. In general, a larger rank number indicates that more information can potentially be learned. However, Fig. 3 shows that $r = 8$ performs best. This indicates that larger ranks lead to redundant information that does not help improve network performance. We therefore set $r = 8$.

5) Stage Number K: We explore the performance gain from the number of reconstruction stages $K$. As shown in Fig. 3 we set $K \in \{1, 3, 6, 9, 12\}$. The process of making $K$ larger brings a significant gain and also increases the number of network parameters. We found a trade-off at $K = 9$.

D. Comparison With State-of-the-Art

1) Quantitative Evaluation: We compare LR-CSNet with five state-of-the-art methods, including two DNUNs: CSNet+ [22], AdapRecon [15], and three DUNs: ISTA-Net+ [29], OPINE-Net+ [30], and AMP-Net [34]. We summarize the evaluation metrics of these methods on multiple datasets in Table II. It can be seen that deep non-folding networks are stacking more convolutional layers which does not increase performance. Whereas ISTA-Net+ operates directly on deep features to simulate soft-thresholding, which limits the representation capability, resulting in poor performance. Meanwhile, OPINE-Net+ uses convolutional layers to simulate the analytical solution of an optimization problem such as the sum of L2 norm and L1 norm. AMP-Net focuses on removing the boundary effects between image patches using denoising techniques. These approaches ignore the low-rank properties of the image patches, resulting in networks that capture structural information only to a limited extent. As shown in the table, LR-CSNet achieves the best reconstruction results at multiple CS ratios.

2) Qualitative Evaluation: Fig. 4 visualizes the reconstruction results at each stage, where higher stages are reconstructed more accurately and the information learnt by $\nabla G^k$ becomes increasingly more detailed. In addition, to illustrate the reconstruction effect of LR-CSNet more intuitively, we show the reconstruction effect of state-of-the-art approaches and LR-CSNet on two images as in Fig. 5, where the red-boxed parts are enlarged and placed on the right side. The corresponding method and evaluation metrics are listed below.

Fig. 5. Visual comparison on ‘img_011’ with CS ratio of 10% (upper) and on ‘img_059’ with CS ratio of 50% (lower). The best performance is highlighted.
and the best value is highlighted. Compared to the other methods, LR-CSNet is better at capturing the overall structure of the image and retains detailed information. This is due to the network taking into account the low-rank attributes of the image patches, together with $\nabla G$ to learn high-frequency information, leading to its ability to obtain better reconstruction accuracy.

V. CONCLUSION

In this paper, we propose a deep unfolding network for natural image compressive sensing (CS) called LR-CSNet. As real-world image patches are often well-represented by low-rank approximations, we add a low-rank prior to the CS reconstruction. We unfold the corresponding iterative optimization problem using variable splitting, leading to a neural network for CS that can be trained end-to-end. Extensive experiments support the effectiveness of our approach.

REFERENCES


