ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9414093

GRAPH SIGNAL DENOISING USING NESTED-STRUCTURED DEEP ALGORITHM UNROLLING

Masatoshi Nagahama[1], Koki Yamada[1], Yuichi Tanaka[1], Stanley H. Chan[2], and Yonina C. Eldar[3]

[1]Tokyo University of Agriculture and Technology, Tokyo, Japan
[2]Purdue University, West Lafayette, IN, USA
[3]Weizmann Institute of Science, Rehovot, Israel

ABSTRACT

In this paper, we propose a deep algorithm unrolling (DAU) based on a variant of the alternating direction method of multiplier (ADMM) called Plug-and-Play ADMM (PnP-ADMM) for denoising of signals on graphs. DAU is a trainable deep architecture realized by unrolling iterations of an existing optimization algorithm which contains trainable parameters at each layer. We also propose a *nested-structured* DAU: Its submodules in the unrolled iterations are also designed by DAU. Several experiments for graph signal denoising are performed on synthetic signals on a community graph and U.S. temperature data to validate the proposed approach. Our proposed method outperforms alternative optimization- and deep learning-based approaches.

Index Terms— Graph signal processing, optimization algorithm, deep learning, signal denoising, deep algorithm unrolling.

1. INTRODUCTION

Many sensing devices capture and store various signals. Since observed signals via such devices frequently suffer from noise or missing values as a consequence of the sensing process, signal restoration methods have been intensively studied in signal processing. The goal of signal restoration is to recover the original signal from observation(s) by solving an inverse problem which includes blind deconvolution [1], denoising [2], and interpolation [3].

Signals often have underlying structures, e.g., sensor, social, transportation and brain networks, power grid, and 3D meshes. Graphs have been utilized to mathematically represent such structures. A graph signal is defined as a discrete-time signal whose domain is nodes of a graph, and the relations between samples are explicitly given by edges. Different from discrete-time signals on a regular grid such as audio and image signals, graph signal processing (GSP) can treat the underlying structure of the signal [4–6]. In this way, GSP is a very powerful tool for compression [7], sampling and restoration [8–10], and analysis of graph signals [11] which could be used in a wide range of applications for irregular-structured data.

In this paper, we focus on denoising of graph signals. Existing approaches can be classified into standard optimization-based approaches such as regularized optimization [12], graph filters and filter banks [13, 14] and deep learning on graphs [15]. As with standard signal processing, a widely-used assumption of ground-truth signals is that the signal is smooth, i.e., it contains higher energy in its lower-frequency components. These methods mainly utilize smoothness prior either on the vertex domain or on the graph frequency domain. In the vertex domain approaches, a minimization problem incorporates the smoothness as a regularization term such as graph total variation (GTV) [16] which basically regularizes the first-order or second-order difference of neighboring signal values. Another approach in the graph frequency domain aims to reduce high graph frequency components via graph Fourier transform. Examples are graph spectral low-pass filters like spectral graph bilateral/trilateral filters [13, 17]. In deep learning-based approaches, a variant of auto-encoder [18] for irregular-structured data has been proposed in [15].

Traditionally, iterative optimization algorithms have been widely used for signal restoration problems to efficiently solve convex optimization problems including regularization [19, 20] as illustrated in Fig. 1(left). The algorithm is iterated with a fixed parameter θ until convergence. The performance and the speed of convergence of conventional optimization algorithms, therefore, mostly depend on the initial choice of parameters (e.g., step size and regularization parameter) whose values are determined manually.

Deep algorithm unrolling (DAU) is a method to build a multilayer network by unrolling the loops of a conventional optimization algorithm and deploying the trainable parameters at each layer [21]. A layer of DAU corresponds to one step of the iterative optimization algorithm. Instead of manually choosing the parameters, those in each layer can be trained from available training data to minimize a loss function. Some variants of DAU have been proposed in the signal processing field [22, 23].

An extension of DAU into the graph setting was recently developed in [22]. It mainly aims to realize graph signal denoising in an unsupervised-learning setting. Specifically, two problems with sparse coding and trend filtering are considered with a new graph convolutional network (GCN) [24, 25]. Furthermore, many GCN implementaions have been proposed so far and used for various applications, e.g., semi-supervised classification [26]. These methods often contain trainable weight matrices in their network that may require a huge number of parameters to be trained. Furthermore, it has been reported in many papers (like [22, 25, 26]) that deeper networks cannot attain good performance for GCNs, in contrast to convolutional neural networks for image and speech signal processing.

In this paper, we leverage the concept of DAU mentioned above to offer an efficient and trainable model-based network for graph signal denoising. On the basis of DAU, we unroll the iterations of Plug-and-Play ADMM (PnP-ADMM) [27, 28], which is an efficient signal restoration method inspired by the alternating direction method of multiplier (ADMM) [29] while being enabled to use an off-the-shelf denoiser in its iterative algorithm. With the DAU structure, we can install trainable model parameters at each layer. Further, the

MN, KY, and YT are supported in part by JST PRESTO under grant JPMJPR1935, JSPS KAKENHI under grants 19K22864 and 20H02145.

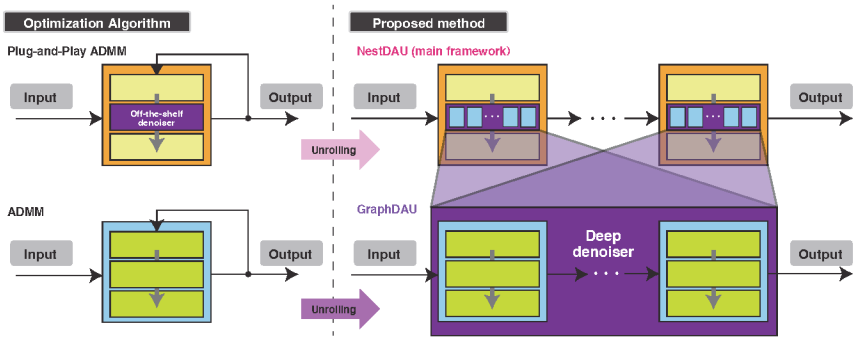

Fig. 1: NestDAU: Our proposed methods are developed based on two conventional iterative optimization algorithms. NestDAU, which is the main framework, is designed based on PnP-ADMM. GraphDAU is a deep denoiser which can be plugged into the NestDAU as submodules.

off-the-shelf denoiser-in-the-loop is realized by DAU. It is obtained by an optimization algorithm including GTV [30]. As a result, the proposed *nested-structured DAU* (called *NestDAU* in this paper) utilizes the concept of DAU both in the main framework and submodule as illustrated in Fig. 1(right). In contrast to the existing work of DAU for graph signals [22], our structure only needs to tune graph-independent parameters. Therefore, the number of parameters keeps small even if the network becomes deeper.

Through some experiments for synthetic and real data, NestDAU achieves better performance than existing optimization- and deep learning-based denoising methods in terms of RMSE.

2. PRELIMINARIES AND PROBLEM FORMULATION

2.1. Graph and Graph Laplacian

Throughout this paper, a vector and a matrix are written in bold and a set is written in a calligraphic letter. An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ consists of a collection of vertices $\mathcal{V} = \{v_i\}_{i=1}^N$ and undirected edges $\mathcal{E} = \{e_{i,j}\}$ with the weighted adjacency matrix $\mathbf{W}$. The numbers of vertices and edges are $|\mathcal{V}| = N$ and $|\mathcal{E}|$, respectively. $w_{i,j} \in \mathbb{R}_{\geq 0}$ denotes the edge weight between v_i and v_j. Then, we define an weighted adjacency matrix of $\mathcal{G}$ represented as an $N \times N$ matrix with $[\mathbf{W}]_{ij} = w_{i,j}$. $[\mathbf{W}]_{ij} = 0$ represents unconnected vertices. In this paper, we consider a graph which has no self-loops, i.e., $[\mathbf{W}]_{ii} = 0$ for all i. The degree matrix is defined as a diagonal matrix with $[\mathbf{D}]_{ii} = \sum_j w_{i,j}$. The combinatorial graph Laplacian matrix of $\mathcal{G}$ is given by $\mathbf{L} = \mathbf{D} - \mathbf{W}$. Since $\mathbf{L}$ is a real symmetric matrix, $\mathbf{L}$ is diagonalizable as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{U}$ is an eigenvector matrix and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$. The weighted graph incident matrix is denoted as $\mathbf{M}$. For notation simplicity, we assign an integer to an edge as $\mathcal{E} = \{e_s\}_{s=1}^{|\mathcal{E}|}$. Then, the s-th row and t-th column of $\mathbf{M}$ corresponding with e_s and v_t,

$$[\mathbf{M}]_{s,t} = \begin{cases} \sqrt{w_{i,j}} & e_s = (v_i, v_j) \text{ and } t = i, \\ -\sqrt{w_{i,j}} & e_s = (v_i, v_j) \text{ and } t = j, \\ 0 & \text{otherwise.} \end{cases}$$

2.2. Problem Formulation

Suppose that an observed *graph signal* $\mathbf{y} \in \mathbb{R}^N$, i.e., the ith element y_i is supposed to be located on v_i of $\mathcal{G}$, is modeled as

$$\mathbf{y} = \mathbf{x} + \mathbf{n}, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^N$ is an unknown graph signal to be restored, and $n_i \sim \mathcal{N}(0, \sigma^2)$ is an i.i.d. additive white Gaussian noise (AWGN). The overall goal of the signal restoration is to recover $\mathbf{x}$ from $\mathbf{y}$ by solving an inverse problem. In this paper, we consider the following minimization problem:

$$\min_{\mathbf{x},\mathbf{s}\in\mathbb{R}^N} f(\mathbf{x}) + \eta g(\mathbf{s}), \quad \text{subject to } \mathbf{x} = \mathbf{s}, \tag{2}$$

where f is some cost function, g is a regularization function, and η is a non-negative parameter which controls influence of regularization.

We follow an approach of PnP-ADMM [27] to (approximately) solve (2). The cost function f is defined as $f(\mathbf{x}) \overset{\text{def}}{=} \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2$ which is the widely-used fidelity term for signal restoration. In the algorithm of PnP-ADMM, the regularization term g is in general implicit. Suppose that two initial variables $\mathbf{s}^{(0)}, \mathbf{t}^{(0)} \in \mathbb{R}^N$ are set, then, the following sequence of subproblems are solved in the pth iteration:

$$\mathbf{x}^{(p+1)} = (1+\rho)^{-1}\left(\mathbf{y} + \rho\left(\mathbf{s}^{(p)} - \mathbf{t}^{(p)}\right)\right), \tag{3}$$

$$\mathbf{s}^{(p+1)} = \mathcal{D}_g(\tilde{\mathbf{s}}^{(p)}), \tag{4}$$

$$\mathbf{t}^{(p+1)} = \mathbf{t}^{(p)} + (\mathbf{x}^{(p+1)} - \mathbf{s}^{(p+1)}), \tag{5}$$

where ρ is step size of the algorithm, $\mathcal{D}_g$ is an arbitrary off-the-shelf denoiser for graph signals and $\tilde{\mathbf{s}}^{(p)} = \mathbf{x}^{(p+1)} + \mathbf{t}^{(p)}$. It is clear that (3) and (5) are independent of the underlying graph, but (4) can be a graph-specific denoiser.

3. PROPOSED METHOD

3.1. Nested-structured DAU for Graph Signal Restoration

To develop a trainable P layer network based on DAU, we take following approaches. Fig. 1(right) is the illustration of the proposed framework.

First, the iterations of the algorithm (3)–(5) are *unrolled* so that it has P sequential layers having the same structure. Then, we set ρ in (3) to be learnable, i.e., $\rho \to \{\rho_p\}_{p=0}^{P-1}$ in which p now indicates the layer number. In other words, the network is basically equivalent to PnP-ADMM with P iterations, but each calculation is conducted with different parameters. Second, a set of off-the-shelf denoisers $\{\mathcal{D}_g^{(p)}\}_{p=0}^{P-1}$ for graph signals are also designed based on DAU.

Hereafter, we refer to this network as NestDAU and show its overall algorithm in Algorithm 1.

Algorithm 1 NestDAU

Input: $\mathbf{y}, \mathbf{s}^{(0)}, \mathbf{t}^{(0)}, \mathcal{D}_g^{(p)}, P$.

for $p = 0, \ldots, P-1$ **do**

$\quad \mathbf{x}^{(p+1)} \leftarrow (1+\rho)^{-1}\left(\mathbf{y} + \rho_p\left(\mathbf{s}^{(p)} - \mathbf{t}^{(p)}\right)\right)$

$\quad \tilde{\mathbf{s}}^{(p)} \leftarrow \mathbf{x}^{(p+1)} + \mathbf{t}^{(p)}$

$\quad \mathbf{s}^{(p+1)} \leftarrow \mathcal{D}_g^{(p)}(\tilde{\mathbf{s}}^{(p)})$ (see Algorithm 2)

$\quad \mathbf{t}^{(p+1)} \leftarrow \mathbf{t}^{(p)} + (\mathbf{x}^{(p+1)} - \mathbf{s}^{(p+1)})$

end for

Output: $\mathbf{x}^{(P)}$.

3.2. Design of Trainable Deep Denoiser

Here, we design a trainable denoiser-in-the-loop $\mathcal{D}_g^{(p)}$ based on DAU. To simplify the notations, the denoising problem here is denoted as $\mathbf{y} = \mathbf{x} + \mathbf{n}$ by setting $\mathbf{s}^{(p+1)} = \mathbf{x}$ and $\tilde{\mathbf{s}}^{(p)} = \mathbf{y}$ in (4).

Indeed, any denoiser can be used as $\mathcal{D}_g^{(p)}$ including GCNs. In this paper, we design $\mathcal{D}_g^{(p)}$ based on a quadratic optimization with ℓ_1 regularization in [22, 31] because it has a small number of parameters leading to computationally-efficient training. It is formulated as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^N, \mathbf{v}\in\mathbb{R}^{|\mathcal{E}|}} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2 + \lambda\|\mathbf{v}\|_1, \quad \text{subject to } \mathbf{v} = \mathbf{M}\mathbf{x}. \tag{6}$$

The second term in (6) can be rewritten as follows:

$$\|\mathbf{M}\mathbf{x}\|_1 = \textstyle\sum_{j\in\mathcal{N}_i} \sqrt{w_{ij}}\,|x_i - x_j|, \tag{7}$$

where $\mathcal{N}_i$ is a set of vertices connecting with v_i.

While [22] utilizes a half-quadratic splitting method [32] as the fundamental iterative algorithm for DAU, we use a different iterative algorithm using ADMM. In fact, ADMM is known as an effective solver of (6). The optimal global solution of (6) can be found by solving a sequence of subproblems shown in (8)–(10) represented as follows:

$$\mathbf{x}^{(\ell+1)} = \left(\mathbf{I} + \frac{1}{\gamma}\mathbf{M}^\top\mathbf{M}\right)^{-1}\left(\mathbf{y} + \frac{1}{\gamma}\mathbf{M}^\top\left(\mathbf{v}^{(\ell)} - \mathbf{u}^{(\ell)}\right)\right), \tag{8}$$

$$\mathbf{v}^{(\ell+1)} = S_{\gamma\lambda}\left(\mathbf{M}\mathbf{x}^{(\ell+1)} + \mathbf{u}^{(\ell)}\right), \tag{9}$$

$$\mathbf{u}^{(\ell+1)} = \mathbf{u}^{(\ell)} + \mathbf{M}\mathbf{x}^{(\ell+1)} - \mathbf{v}^{(\ell+1)}, \tag{10}$$

where γ is the step size of the algorithm and arbitrary initial variables $\mathbf{v}^{(0)}, \mathbf{u}^{(0)} \in \mathbb{R}^{|\mathcal{E}|}$ have to be set. Furthermore, $S_{\gamma\lambda}$ acts as a soft-thresholding operator, i.e., $[S_{\gamma\lambda}(\mathbf{x})]_i = \mathrm{sgn}(x_i)\max\{|x_i| - \gamma\lambda, 0\}$.

To design the trainable denoiser-in-the-loop, we unroll the iteration of (8)–(10). In other words, instead of using the fixed parameters in each iteration, we set γ and $\gamma\lambda$ to be learnable, i.e., $\gamma \to \{\gamma_\ell\}_{\ell=0}^{L-1}$ and $\gamma\lambda \to \{\beta_\ell\}_{\ell=0}^{L-1}$ in which ℓ denotes the layer number of the submodule.

This denoiser for the submodule of NestDAU can also be used as an independent graph signal denoiser. Therefore, we call it GraphDAU hereafter. To avoid the repeated matrix inversion in (8), the eigendecomposition (EVD) of $\mathbf{L}$ is precomputed such that it can reduce the computational complexity. Algorithm 2 shows the overview of GraphDAU.

The numbers of parameters for GraphDAU and NestDAU are $2L$ and $(2L+1)P$, respectively, and they are independent of N.

Algorithm 2 GraphDAU

Input: $\mathbf{y}, \mathbf{M}, \mathbf{L} = \mathbf{M}^\top\mathbf{M}, \mathbf{v}^{(0)}, \mathbf{u}^{(0)}, L$.

Compute the eigendecomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$.

for $\ell = 0, \ldots, L-1$ **do**

$\quad \widetilde{\mathbf{y}}^{(\ell)} \leftarrow \mathbf{y} + \frac{1}{\gamma_\ell}\mathbf{M}^\top\left(\mathbf{v}^{(\ell)} - \mathbf{u}^{(\ell)}\right)$

$\quad \mathbf{x}^{(\ell+1)} \leftarrow \mathbf{U}\left(\mathbf{I} + \frac{1}{\gamma_\ell}\mathbf{\Lambda}\right)^{-1}\mathbf{U}^\top\widetilde{\mathbf{y}}^{(\ell)}$

$\quad \mathbf{v}^{(\ell+1)} \leftarrow S_{\beta_\ell}\left(\mathbf{M}\mathbf{x}^{(\ell+1)} + \mathbf{u}^{(\ell)}\right)$

$\quad \mathbf{u}^{(\ell+1)} \leftarrow \mathbf{u}^{(\ell)} + \mathbf{M}\mathbf{x}^{(\ell+1)} - \mathbf{v}^{(\ell+1)}$

end for

Output: $\mathbf{x}^{(L)}$.

Table 1: Training configuration where BS and LR refer to batch size and learning rate, respectively.

BS	Epoch	Weight decay	Optimizer	LR	Scheduler
1	5	1.0×10^4	Adam [33]	0.02	StepLR

4. EXPERIMENTAL RESULTS

For experiments, we use following graph signals:

- Synthetic signals on a community graph ($N = 250$),
- U.S. temperature data ($N = 614$).

We create piecewise constant graph signals based on cluster labels of a community graph with three communities (see Fig. 2(a)). The graph itself is fixed with all samples, but the cluster labels are different. Each cluster in the graph is assigned an integer value between 0 to 5 randomly as its cluster label. Then, AWGN ($\sigma = \{0.5, 1.0\}$) is added to the ground-truth signals. The dataset is split into 500 training, 50 validation, and 50 testing data.

For real data, we use daily average temperature data in the United States in 2017, provided by QCLCD[1]. The original weather data contain missing values. As a preprocessing, the missing values in the time series are filled by linear interpolation. Then, like the previous example, AWGN ($\sigma = \{3.0, 5.0, 7.0, 9.0\}$) is added. As a result, we obtain 304 training (January 31st to November 30th), 31 validation (December 1st to 31st), and 30 testing (January 1st to 30th) data. The weighted graph is constructed by a 8-nearest neighbor graph based on the geographic coordinates of the stations.

In this paper, we compare the denoising performance with five existing methods listed as follows:

- Diffusion with heat kernel (HD) [34]
- Spectral graph bilateral filter (SGBF) [17]
- ADMM with fixed parameters ((8)–(9)) with 10 iterations
- PnP-ADMM with fixed parameters [28]: Its formulation is given in Section 2.2 and the off-the-shelf denoisers are HD or SGBF with 12 iterations

For a fair comparison, their fixed parameters are tuned by performing a grid search on the validation data to minimize RMSE.

We also include the following deep learning-based methods:

- Multilayer perceptron (MLP)
- Graph convolutional network (GCN) [26]
- Graph unrolling-based trend filtering (GUTF) [22]
- Graph unrolling-based sparse coding (GUSC) [22]

[1] https://www.ncdc.noaa.gov/orders/qclcd/

Table 2: Denoising results (average RMSEs for test data).

	L / P	# of parameters	*Community graph*		*U.S. temperature*			
			$\sigma = 0.5$	$\sigma = 1.0$	$\sigma = 3.0$	$\sigma = 5.0$	$\sigma = 7.0$	$\sigma = 9.0$
Noisy	-	-	0.507	1.004	3.030	5.037	6.999	8.916
HD	-	-	0.307	0.486	1.899	2.374	2.748	3.314
SGBF	-	-	0.285	0.452	1.878	2.346	2.754	3.028
ADMM	10	-	0.196	0.368	1.827	2.320	2.597	2.913
PnP-HD	12	-	0.308	0.482	1.852	2.323	2.583	3.076
PnP-SGBF	12	-	0.285	0.448	1.890	2.345	2.709	2.965
MLP	-	321	0.464	0.832	3.047	4.812	6.432	7.926
GCN	-	321	0.418	0.512	2.175	2.394	2.727	3.098
GUTF	-	19,397	0.179	0.320	2.207	2.348	2.637	2.911
GUSC	-	11,270	0.225	0.368	2.132	2.383	2.738	3.062
GraphDAU	10	20	0.153	0.312	1.744	2.220	2.549	2.822
NestDAU	4	84	0.144	0.300	1.723	**2.206**	2.550	**2.808**
	8	168	0.136	0.295	**1.720**	2.208	2.544	2.834
	12	252	**0.129**	**0.283**	**1.720**	**2.206**	**2.536**	2.867

MLP and GCN are set to have two layers. GUTF and GUSC are set to have one layer, and other specifications are the same as those provided in [22].

The training configuration of our proposed method is determined by our preliminary experiments as shown in Table 1. We set the layer of GraphDAU to $L = 10$ and it is also used as submodules of NestDAU. We compare NestDAU having different numbers of layers, i.e., $P \in \{4, 8, 12\}$, referred to as NestDAU$_P$.

Experimental results are summarized in Table 2 in addition with the number of trainable parameters for the learning-based methods. Clearly, GraphDAU and NestDAU have fewer parameters than MLP, GCN, and the existing DAU for graphs because our proposed method does not contain weight matrices to be trained.

For both signals, the proposed method shows consistently better RMSEs than the existing methods including the deep learning-based methods. Basically, NestDAU is slightly better than GraphDAU because NestDAU includes GraphDAU as its submodules. The number of layers in NestDAU affects the restoration performance: Interestingly, deeper layers usually result in better RMSEs (in contrast to the other GCNs) with an exception for the case with $\sigma = 9.0$ for the U.S. temperature data. GCN is comparable to or slightly worse than the optimization-based approaches while it has a large number of trainable parameters. MLP is much worse than GCN in this experiment because clearly MLP does not consider the underlying graph structure. GUTF and GUSC are comparable to the ADMM-based denoising while they have a huge number of parameters. Note that all methods in this experiment only use a single feature on a node. In [22], it is suggested that using multiple features improves denoising performance especially for deep learning-based approaches: Our future work includes such comparisons.

The visualizations of the denoising results are shown in Fig. 2. As can be seen in the figures, our proposed method suppresses errors compared to the alternative methods.

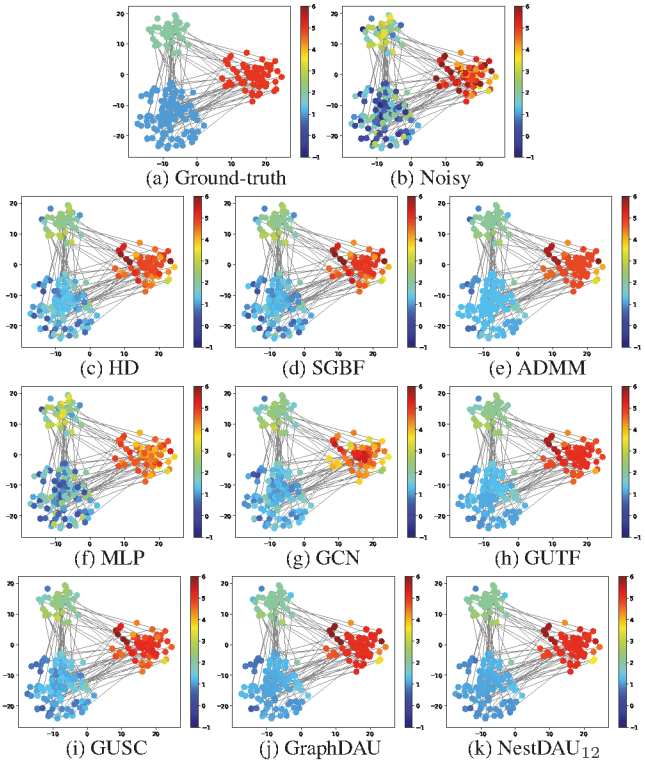

Fig. 2: Restoration results (community graph, $\sigma = 1.0$).

5. CONCLUSIONS

In this paper, we proposed a signal denoising method on graphs with a trainable network. The proposed method, called NestDAU, is designed based on PnP-ADMM as the main framework and ADMM as a submodule. The nested DAU structure outperforms alternative methods including graph low-pass filter, convex optimization-based denoising and deep learning-based approaches. Our future work includes to study the effectiveness of NestDAU for different noise models and more general graph signal restoration problems.

6. ACKNOWLEDGMENTS

We thank Dr. Siheng Chen for providing the code of [22].

7. REFERENCES

[1] D. Ramírez, A. G. Marques, and S. Segarra, "Graph-signal reconstruction and blind deconvolution for diffused sparse in-

puts," *in Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, pp. 4104–4108, 2017.

[2] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *in Proc. IEEE Int. Conf. Comput. Vis.*, pp. 839–846, 1998.

[3] X. Zhang and X. Wu, "Image interpolation by adaptive 2-d autoregressive modeling and soft-decision estimation," *IEEE Trans. Image Process.*, vol. 17, no. 6, pp. 887–896, 2008.

[4] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[5] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: overview, challenges, and applications," *in Proc. IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[6] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, 2013.

[7] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multiresolution graph fourier transform for compression of piecewise smooth images," *IEEE Trans. on Image Process.*, vol. 24, no. 1, pp. 419–433, 2015.

[8] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: sampling theory," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6510–6523, 2015.

[9] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3775–3789, 2016.

[10] Y. Tanaka, Y. C. Eldar, A. Ortega, and G. Cheung, "Sampling signals on graphs: From theory to applications," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 14–30, Nov. 2020.

[11] A. Sakiyama, K. Watanabe, Y. Tanaka, and A. Ortega, "Two-channel critically-sampled graph filter banks with spectral domain sampling," *IEEE Trans. Signal Process.*, vol. 67, no. 6, pp. 1447–1460, Mar. 2019.

[12] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovačević, "Signal recovery on graphs: variation minimization," *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4609–4624, 2015.

[13] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, "Graph signal denoising via trilateral filter on graph spectral domain," *IEEE Trans. on Signal and Inf. Process. over Networks*, vol. 2, no. 2, pp. 137–148, 2016.

[14] Y. Tanaka and A. Sakiyama, "M-channel oversampled graph filter banks," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3578–3590, 2014.

[15] T. H. Do, D. Minh Nguyen, and N. Deligiannis, "Graph autoencoder for graph signal denoising," *in Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, pp. 3322–3326, 2020.

[16] S. Ono, I. Yamada, and I. Kumazawa, "Total generalized variation for graph signals," *in Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, pp. 5456–5460, 2015.

[17] A. Gadde, S. K. Narang, and A. Ortega, "Bilateral filter: graph spectral interpretation and extensions," *in Proc. IEEE Int. Conf. Image Process.*, 2013.

[18] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," *in Proc. Int. Conf. Neural Inf. Process. Syst.*, pp. 341–349, 2012.

[19] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, 2004.

[20] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[21] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," *in Proc. Int. Conf. Mach. Learn*, pp. 399–406, 2010.

[22] S. Chen, Y. C. Eldar, and L. Zhao, "Graph unrolling networks: Interpretable neural networks for graph signal denoising," *IEEE Trans. Signal Process.*, Jun. 2020.

[23] Y. Li, M. Tofighi, J. Geng, V. Monga, and Y. C. Eldar, "Efficient and interpretable deep blind image deblurring via algorithm unrolling," *IEEE Trans. Comput. Imaging*, vol. 6, pp. 666–681, 2020.

[24] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *ArXiv150605163 Cs*, 2015.

[25] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *in Proc. Int. Conf. Neural Inf. Process. Syst.*, pp. 3844–3852, 2016.

[26] T. N. Kipf and M. Welling, "Semi-Supervised classification with graph convolutional networks," in *Int. Conf. on Learn. Representations*, 2017.

[27] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: fixed-point convergence and applications," *IEEE Trans. Comput. Imaging*, vol. 3, no. 1, pp. 84–98, 2017.

[28] Y. Yazaki, Y. Tanaka, and S. H. Chan, "Interpolation and denoising of graph signals using plug-and-play ADMM," *in Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, pp. 5431–5435, 2019.

[29] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *FNT in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[30] K. Bredies, K. Kunisch, and T. Pock, "Total generalized variation," *SIAM J. Imaging Sci.*, vol. 3, no. 3, pp. 492–526, 2010.

[31] Y.-X. Wang, J. Sharpnack, A. J. Smola, and R. J. Tibshirani, "Trend filtering on graphs," *J. Mach. Learn. Res.*, vol. 17, no. 105, pp. 1–41, 2016.

[32] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM J. Imaging Sci.*, vol. 1, no. 3, pp. 248–272, 2008.

[33] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," *ArXiv14126980 Cs*, 2017.

[34] F. Zhang and E. R. Hancock, "Graph spectral image smoothing using the heat kernel," *Pattern Recognition*, vol. 41, no. 11, pp. 3328–3342, 2008.