TIME-VARYING GRAPH SIGNAL INPAINTING VIA UNROLLING NETWORKS

Siheng Chen[1] *, Yonina C. Eldar*[2]

[1] Shanghai Jiao Tong University; [2] Weizmann Institute of Science

ABSTRACT

We propose an interpretable graph neural network based on algorithm unrolling to reconstruct a time-varying graph signal from partial measurements. The proposed *graph unrolling networks* expand algorithm unrolling to the graph-time domain and provide an interpretation of the architecture design from a signal processing perspective. We unroll an iterative inpainting algorithm by mapping each iteration to a single network layer. The feed-forward process is thus equivalent to iteratively reconstructing a time-varying graph signal. We train this network through unsupervised learning, where the input time-varying graph signal is used to supervise the training. By leveraging the learning ability of neural networks, we adaptively capture appropriate priors from input data, instead of manually choosing signal priors. To validate the proposed methods, we conduct experiments on three real-world datasets and demonstrate that our networks achieve smaller reconstruction errors than conventional inpainting algorithms and state-of-the-art graph neural networks.

***Index Terms*—** Graph-temporal data, Graph neural networks, Algorithm unrolling

1. INTRODUCTION

Data today is often generated from a diverse sources, including social, citation, biological, and physical infrastructure [1]. Unlike time-series signals or images, such signals possess complex and irregular structures, which can be modeled as graphs. Analyzing graph signals requires dealing with the underlying irregular relationships. To address this, graph signal processing generalizes classical signal processing tools to the graph domain and provides a series of techniques to process graph signals [1]. In practice, many graph signals are varying across time, which can be further modeled as time-varying graph signals [2]. Those techniques that process time-varying graph signals can be widely used for skeleton-based human action recognition [3], traffic forecasting [4], and multi-agent motion prediction [5].

In this work, we consider time-varying graph signal inpaiting. Signal inpainting is an important task in both signal and image processing [6]. A mainstream approach for inpainting over graphs is to introduce a graph-regularization term that promotes certain properties of the graph signal and then solve a regularized optimization problem to obtain a reconstruction [7]. In image inpainting, the total variation, which captures the integral of the absolute gradient of the image [8], is often used to recover the missing pixels. In time-varying graph signal inpainting, some popular regularizers include low-rank [7] and a quadratic form of the graph Laplacian, which captures the second-order difference of a graph signal, corresponding to a graph smoothness prior [9].

A fundamental challenge here is that we may not know an appropriate prior on an incomplete time-varying graph signal in practice. It is then hard to choose an appropriate graph-regularization term. Furthermore, some graph priors are too complicated to be precisely described in mathematical terms or may lead to computationally intensive algorithms. To solve this issue, it is desirable to learn an appropriate prior from given graph signals; in other words, the inpainting algorithm should have sufficient feasibility to learn from and adapt to arbitrary signal priors.

Deep neural networks have demonstrated strong power in learning abstract, yet effective features from a huge amount of data [10]. As the extension of neural networks to the graph domain, graph neural networks have also received a lot of attention and achieved significant success in social network analysis and geometric data analysis [11]. One mainstream graph neural network architecture is the graph convolutional network (GCN), which relies on a layered architecture that consists of trainable graph convolution operations, followed by pointwise nonlinear functions [12]. These models have shown remarkable success in graph-based semi-supervised learning [11] and graph classification tasks [13]. However, these neural network architectures are typically designed through trial and error. It is thus hard to explain the design rationale and further improve the architectures [14]. Furthermore, most graph neural networks are developed for supervised-learning tasks, such as node classification [11] and graph classification [13]. These tasks require a large number of ground-truth labels, which is expensive to obtain.

In this work, we leverage the powerful learning ability of graph neural networks and combine them with interpretablity based on a signal processing perspective. We further consider an unsupervised-learning setting, where the network has to

The work was done when Siheng Chen was working at Mitsubishi Electric Research Laboratories.

learn from a single incomplete time-varying graph signal while the ground-truth complete signal is unknown. Through unsupervised learning, we demonstrate the generalization ability of the proposed graph neural networks.

We first propose a general iterative algorithm for graph signal inpainting and then transform it to a graph neural network through algorithm unrolling, where each iteration is mapped to a network layer [15]. Compared to conventional inpainting algorithms [7], the proposed graph unrolling network is able to learn a variety of priors from given graph signals by leveraging deep neural networks. Compared to many other graph neural networks [11], the proposed unrolling network is interpretable by following analytical iterative steps. To validate the empirical performance of the proposed method, we conduct experiments on three real-world datasets. We find that graph unrolling networks consistently achieve better inpainting performance than conventional graph signal denoising algorithms and state-of-the-art graph neural networks in the unsupervised setting.

The main contributions of this work include:

• We propose interpretable graph unrolling networks by unrolling a general iterative algorithm for graph signal inpainting in an unsupervised-learning setting; and

• We conduct experiments on three real-world data to validate that the proposed unrolling method outperforms both conventional graph signal inpainting methods and state-of-the-art graph neural networks in the unsupervised setting.

2. INPAINTING NETWORKS VIA UNROLLING

2.1. Problem Formulation

In this section, we mathematically formulate the task of time-varying graph signal inpainting. We consider a graph $G = (\mathcal{V}, \mathcal{E}, \mathrm{A})$, where $\mathcal{V} = \{v_n\}_{n=1}^N$ is the set of *vertices*, $\mathcal{E} = \{e_m\}_{m=1}^M$ is the set of undirected *edges*, and A is a N-by-N graph adjacency matrix, representing connections between vertices. The weight $\mathrm{A}_{i,j}$ of an edge from the ith to the jth vertex characterizes the relation, such as similarity or dependency, between the corresponding signal values. Using the graph representation G, a *time-varying graph signal* is defined as a map that assigns a time series $\mathbf{s}_n$ with length T to the vertex v_n. A time-varying graph signal can be written as a N-by-T matrix defined by

$$\mathrm{X} = \begin{bmatrix} \mathbf{s}_1^T \\ \mathbf{s}_2^T \\ \vdots \\ \mathbf{s}_N^T \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_T \end{bmatrix}, \tag{1}$$

where the column vector $\mathbf{x}_t$ is a graph signal at the tth time stamp and the element $\mathrm{X}_{n,t} = (\mathbf{s}_n)_t = (\mathbf{x}_t)_n$ is the signal coefficient of the nth vertex at the tth time stamp.

Consider a measurement of a graph-time signal modeled as $\mathrm{Y} = \Psi \circ \mathrm{X} + \mathrm{E}$, where $\circ$ is element-wise multiplication, Ψ is a N-by-T sensing matrix with binary elements and E models the noise. Our goal is to reconstruct the original time-varying graph signal X given the measurement Y and the sensing matrix Ψ.

Without any prior information on the original time-varying graph signals, it is impossible to reconstruct the complete data from incomplete measurements. Here we consider a smoothness prior along both temporal and graph domains. Let $\Delta_T = \mathrm{C} - \mathrm{I}$ be the temporal difference operator, where I is the identity matrix and

$$\mathrm{C} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

is a T-by-T cyclic-permutation matrix. For a time series $\mathbf{s} \in \mathbb{R}^T$, its first-order temporal difference is

$$\Delta_T \mathbf{s} = \begin{bmatrix} s_1 - s_T \\ s_2 - s_1 \\ \vdots \\ s_T - s_{T-1} \end{bmatrix}, \tag{2}$$

whose elements are the differences between signal coefficients at two consecutive time stamps. When all elements in $\Delta_T \mathbf{s}$ are small, the time series $\mathbf{s}$ is smooth.

Let Δ_G be the *graph difference operator*, which is the M-by-N oriented incidence matrix of G, whose rows correspond to edges [16, 17]. For example, if e_i is a directed edge that connects the jth node to the kth node ($j < k$), the elements of the ith row of Δ are

$$(\Delta_G)_{i,\ell} = \begin{cases} 1, & \ell = k; \\ -1, & \ell = j; \\ 0, & \text{otherwise.} \end{cases}$$

For a graph signal $\mathbf{x} \in \mathbb{R}^N$, its first-order graph difference is an *edge signal* $\Delta_G \mathbf{x} \in \mathbb{R}^M$, whose ith element $(\Delta_G \mathbf{x})_i = x_k - x_j$, assigns the difference between two adjacent signal coefficients to the ith edge, where the ith edge connects the jth node to the kth node ($j < k$). When all elements in $\Delta_G \mathbf{x}$ are small, the graph signal $\mathbf{x}$ is smooth.

For a time-varying graph signal X, we can combine these two differences and define the first-order graph-time difference as $\Delta_G \mathrm{X} \Delta_T$, which is a M-by-T matrix. When all elements in $\Delta_G \mathrm{X} \Delta_T$ are small, the time-varying graph signal X is smooth. To make it more general, we can consider higher order differences. Let $\Delta_G^p \mathrm{X} \Delta_T^q$ be the pth-order graph and qth-order time difference. We leverage this property to design a smoothness prior.

2.2. Optimization

To recover X, we consider the optimization problem

$$\min_{\mathrm{X}} \; f(\mathrm{X}) = \frac{1}{2} \left\| \Psi \circ \mathrm{X} - \mathrm{Y} \right\|_{\mathrm{F}}^2 + \frac{\lambda}{2} \left\| \Delta_G \mathrm{X} \Delta_T \right\|_{\mathrm{F}}^2, \tag{3}$$

where λ is a hyper-parameter. The first term is a data fitting term and the second term reflects the regularization for a time-varying graph signal.

Since this optimization problem is quadratic, we can solve it using conjugate gradient descent [18]. The update step of the kth iteration is then

$$\tau = \frac{f'(\mathrm{X}^{(k)})^T \mathrm{Z}^{(k)}}{\left(\mathrm{Y} + f'(\mathrm{Z}^{(k)})\right)^T \mathrm{Z}^{(k)}}, \quad (4a)$$

$$\mathrm{X}^{(k+1)} = \mathrm{X}^{(k)} - \tau\, \mathrm{Z}^{(k)}, \quad (4b)$$

$$\gamma = \frac{\left\| f'(\mathrm{X}^{(k+1)}) \right\|_F^2}{\left\| f'(\mathrm{X}^{(k)}) \right\|_F^2}, \quad (4c)$$

$$\mathrm{Z}^{(k+1)} = -f'(\mathrm{X}^{(k+1)}) + \gamma\, \mathrm{Z}^{(k)}, \quad (4d)$$

where the gradient is $f'(\mathrm{X}) = \Psi \circ \mathrm{X} - \mathrm{Y} + \lambda \Delta_G^T \Delta_G \mathrm{X} \Delta_T \Delta_T^T$, $\mathrm{X}^{(0)} = 0$ and $\mathrm{Z}^{(0)} = -f'(\mathrm{X}^{(0)})$. Note that $\mathrm{L}_T = \Delta_T^T \Delta_T$ and $\mathrm{L}_G = \Delta_G^T \Delta_G$ are high-pass filters in the temporal and graph domains, respectively. In (3), we set the difference orders $p = q = 1$; however, various data may fit other orders. To address this, we can leverage the learning ability of deep neural networks, to allow for more flexibility in the prior.

2.3. Algorithm Unrolling

Here we aim to unroll the iteration steps in (4) to one network layer with trainable parameters. When we sequentially stack network layers, the architecture is hypothetically equivalent to running the iteration steps multiple times. The key step is to update the gradient computation, which is leveraged in all four steps in (4). Let $h(\mathrm{L}_G) = \mathrm{L}_G + \sum_{p=2}^{P} h_p \mathrm{L}_G^p$, where h_ps are graph filter coefficients and $g(\mathrm{L}_T) = \mathrm{L}_T + \sum_{q=2}^{Q} g_q \mathrm{L}_T^q$, where g_qs are temporal filter coefficients. We consider a trainable gradient function

$$f'_w(\mathrm{X}) = \Psi \circ \mathrm{X} - \mathrm{Y} + \lambda \cdot h(\mathrm{L}_G)\, \mathrm{X}\, g(\mathrm{L}_T), \quad (5)$$

where λ, h_ps and g_qs are trainable parameters, and the subscript w indicates the gradient function is trainable. We update the iteration steps (4) by replacing the original gradient function $f'(\cdot)$ with (5). Note that we preserve the entire computational structure and only update the gradient function, which is equivalent to considering a data-adaptive regularization term in (3). When $h_p = g_q = 0$, the trainable gradient function degenerates to the original gradient function; otherwise, the trainable gradient function generates higher-order differences in graph and temporal dimensions, which might regularize the reconstructed time-varying graph signal be smoother.

To build a complete network architecture, we initialize $\mathrm{X}^{(0)} = 0, \mathrm{Z}^{(0)} = -g_w(\mathrm{X}^{(0)})$ and sequentially stack K unrolling layers. Finally, we output the reconstruction $\widehat{\mathrm{X}} = \mathrm{X}^{(K+1)}$. The training parameters are shared across all the network layers. To train those parameters, we directly use (3) as the loss function with $\lambda = 0$ and use the stochastic gradient descent to minimize the loss and optimize this network [10].

2.4. Discussion

Here we provide some insights into the proposed unrolling algorithm.

Closed-form solution. The closed-form solution of (3) is

$$\mathrm{vec}(\mathrm{X}) = \left(\Phi + \lambda\, \mathrm{L}_T \otimes \mathrm{L}_G\right)^{-1} \Phi \mathrm{vec}(\mathrm{Y}),$$

where $\Phi = \mathrm{diag}(\mathrm{vec}(\Psi)) \in \mathbb{R}^{NT \times NT}$. When the regularization term in (3) is replaced by $\left\| h(\mathrm{L}_G)\, \mathrm{X}\, g(\mathrm{L}_T)^T \right\|_{\mathrm{F}}^2$ with fixed filters $g(\cdot), h(\cdot)$, the closed-form solution is

$$\mathrm{vec}(\mathrm{X}) = \left(\Phi + \lambda \cdot g(\mathrm{L}_T^T) \otimes h(\mathrm{L}_G)\right)^{-1} \Phi \mathrm{vec}(\mathrm{Y}).$$

When the filter coefficients are trainable, we can optimize them by solving

$$\min_{g(\cdot), h(\cdot)} \left\| \Phi \left(\left(\Phi + \lambda \cdot g(\mathrm{L}_T^T) \otimes h(\mathrm{L}_G) \right)^{-1} \Phi - \mathrm{I} \right) \mathrm{vec}(\mathrm{Y}) \right\|_2^2. \quad (6)$$

The above optimization is clearly enormous. The proposed unrolling algorithm is essentially a fast method to solve (6) through a few unrolling layers.

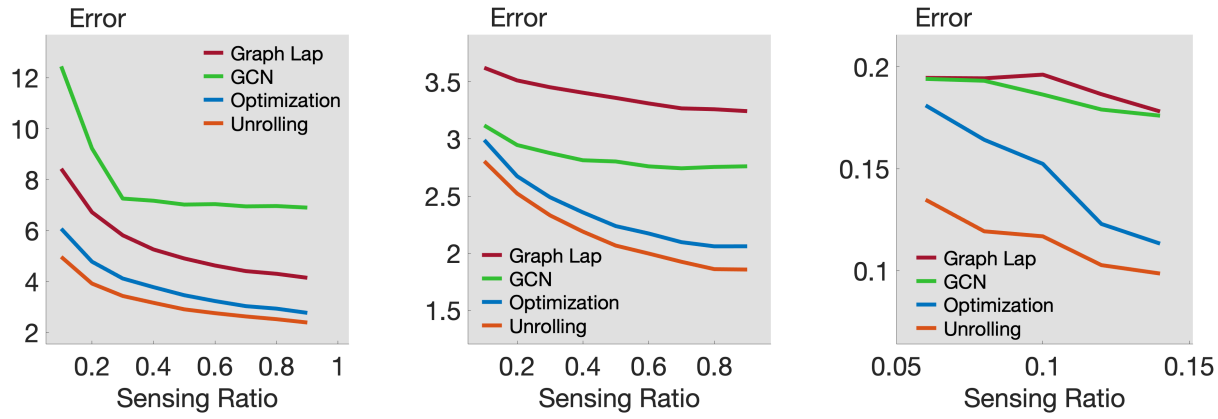
Generalization ability. Here the total number of training parameters is merely $P + Q$, which is much smaller than the number of known entries $\mathbf{1}^T \Psi \mathbf{1}$. It makes the generalization easy. We also try individual trainable parameters in each of the K unrolling layers. Then the total number of training parameters becomes $K(P + Q)$. The experiments show that it produces similar performance with the shared setting. The intuition is that algorithmic structure provides a strong prior, which regularizes the training of parameters.

3. EXPERIMENTAL RESULTS

In this section, we validate the superiority of the proposed method on three real-world datasets.

3.1. Dataset

U.S. temperature data. We consider 150 weather stations in the United States that record their local temperatures [19]. Each weather station has 365 days of recordings (one recording per day), for a total of 54,750 measurements. The graph representing these weather stations is obtained by measuring the geodesic distance between each pair of weather stations. The vertices are represented by an 8-nearest neighbor graph, in which vertices represent weather stations, and each station is connected to the eight closest ones.



(a) U.S. temperature. (b) NYC taxi-pickups. (c) MSR action.

Fig. 1: Reconstruction errors as a function of the sensing ratio in three real-world datasets.

NYC traffic data. We consider the taxi-pickup activity in Manhattan on January 1th, 2014. This is the Manhattan street network with $2,552$ intersections and $3,153$ road segments. We model each intersection as a vertex and each road segment as an edge. Each graph signal is the hourly number of taxi-pickups recorded in each intersection. We consider 24 graph signals for one day in total [20]. Compared to temperature data, the graph signals here are much more complicated. Even two adjacent intersections have correlations, they could have different traffic behaviors.

MSR action data. The dataset is captured by a depth sensor, e.g., Kinect, and the 3D positions of 20 skeletal joints are provided. We pick one action "hand waving", which includes 55 frames. Each graph signal is the 3D coordinates supported on a skeleton graph recorded in each frame [21].

3.2. Experimental Setup

Evaluation Metric. We evaluate the inpainting performance by comparing the difference between the reconstruction and the original time-varying graph signal on the unknown entries; that is,

$$\text{Error} = \left\|(\mathbf{1}\mathbf{1}^T - \Psi) \circ (\widehat{\mathrm{X}} - \mathrm{X})\right\|_{\mathrm{F}} / \left\|\mathbf{1}\mathbf{1}^T - \Psi\right\|_1,$$

where the nominator computes the difference between the reconstruction and the original data on the unknown entries and the denominator counts the number of unknown entries.

Baselines. We compare the proposed method with three baseline methods: graph Laplacian-based inpainting [9], the exact optimization (4) and graph convolutional network [11]. Graph Laplacian-based inpainting is a classical graph signal inpainting method, which uses the quadratic term of graph Laplacian as the regularization term in (3), promoting graph smoothness. The exact optimization uses the conjugate gradient descent to solve (3). Graph convolutional network is a commonly-used graph neural network, which uses trainable graph convolutions to reconstruct a time-varying graph signal from partial measurements with the loss function (3). For graph Laplacian-based inpainting and the exact optimization, we set the number of iterations to be $50,000$ and make sure the solutions achieve convergence. For graph convolutional network, we set the number of network layers to be 3 and the number of iterations to be 100. For the proposed unrolling method, we set the filter orders $P = 2, Q = 5$, the number of network layers to be 50 and the number of iterations to be 100. In graph convolutional network, it is known that deep layers lead to over-smoothing [22]. In our experiments, we do not encounter the over-smoothing issue and deeper layers usually lead to better performances.

3.3. Results

Fig. 1 shows the inpainting performance on three datasets. The x-axis is the ratio between the number of known entries and the number of total entries and the y-axis is the reconstruction error. We see that the proposed unrolling method (in red) outperforms the other competitors. Although the graph convolution network has similar learning ability, it does not include sufficient algorithmic structure and cannot reconstruct well. This reflects the appropriate combination of learning ability and algorithmic structure is important.

4. CONCLUSIONS

This paper proposes a graph unrolling network to reconstruct a time-varying graph signal from partial measurements. This network unrolls an iterative inpainting algorithm by mapping each iteration into a single network layer where the feed-forward process is equivalent to iteratively reconstructing a time-varying graph signal. By leveraging the learning ability of neural networks, we adaptively capture appropriate priors from input incomplete time-varying graph signals, instead of manually choosing signal priors. To validate the proposed methods, we conduct experiments on three real-world datasets and demonstrate that our methods achieve smaller reconstruction errors than conventional inpainting algorithms and state-of-the-art graph neural networks.

5. REFERENCES

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

[2] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Filtering random graph processes over random time-varying graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4406–4421, 2017.

[3] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, June 16-20, 2019*, 2019, pp. 3595–3603.

[4] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *6th International Conference on Learning Representations, Vancouver, BC, Canada, April 30 - May 3, 2018*, 2018.

[5] Y. Hu, S. Chen, Y. Zhang, and X. Gu, "Collaborative motion prediction via neural motion message passing," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. 2020, pp. 6318–6327, IEEE.

[6] M. Vetterli, J. Kovačević, and V. K. Goyal, *Foundations of Signal Processing*, Cambridge University Press, Cambridge, 2014, http://foundationsofsignalprocessing.org.

[7] 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, Sept. 2015.

[8] Y.-W. Wen, R. H. Chan, and A. M. Yip, "A primal-dual method for total-variation-based wavelet domain inpainting," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 106–114, 2012.

[9] 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, pp. 83–98, May 2013.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[12] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2014.

[13] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. 2018, pp. 4438–4445, AAAI Press.

[14] O. Solomon, R. Cohen, Y. Zhang, Y. Yang, Q. He, J. Luo, R. J. G. van Sloun, and Y. C. Eldar, "Deep unfolded robust PCA with application to clutter suppression in ultrasound," *IEEE Transactions on Medical Imaging*, vol. 39, pp. 1051–1063, April 2020.

[15] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. abs/1912.10557, 2019.

[16] J. Sharpnack, A. Rinaldo, and A. Singh, "Sparsistency of the edge lasso over graphs," in *Proc. Int. Conf. Artif. Intell. and Stat.*, La Palma, Apr. 2012, pp. 1028–1036.

[17] Y-X Wang, J. Sharpnack, A. Smola, and R. J. Tibshirani, "Trend filtering on graphs," in *Proc. Int. Conf. Artif. Intell. and Stat.*, San Diego, CA, May 2015.

[18] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, NY, 2004.

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

[20] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal representations on graphs: Tools and applications," arXiv:1512.05406 [cs.AI]., Dec. 2015.

[21] J-Y. Kao, A. Ortega, D. Tian, H. Mansour, and A. Vetro, "Graph based skeleton modeling for human activity analysis," in *2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*. 2019, pp. 2025–2029, IEEE.

[22] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.