BDoG-Net: Algorithm Unrolling for Blind Deconvolution on Graphs

Chang Ye, *Member, IEEE*, and Gonzalo Mateos, *Senior Member, IEEE*

***Abstract*—Starting from first graph signal processing (GSP) principles, we present a novel model-based deep learning approach to blind deconvolution of sparse graph signals. Despite the bilinear nature of the observations, by requiring invertibility of the unknown (diffusion graph filter) forward operator we can formulate a convex optimization problem and solve it using the alternating-direction method of multipliers (ADMM). We then unroll and truncate the novel ADMM iterations to arrive at a parameterized neural network architecture for blind deconvolution on graphs (BDoG-Net), which we train in an end-to-end fashion using labeled data. This supervised learning approach offers several advantages, such as interpretability, parameter efficiency, and controllable complexity during inference. Our reproducible numerical experiments corroborate that BDoG-Net exhibits performance on par with the iterative ADMM baseline, but with markedly faster inference times and without the need to manually adjust the step-size or penalty parameters. The application of BDoG-Net to a simplified instance of source localization over networks is also discussed. Overall, our approach combines the best of both worlds by incorporating the inductive biases of a GSP model-based solution within a data-driven, trainable deep learning architecture for blind deconvolution on graphs.**

***Index Terms*—Graph signal processing, network diffusion, deep learning, blind deconvolution, algorithm unrolling.**

I. Introduction

WE CONSIDER a *blind* deconvolution task involving graph signals [37], [47], [48]. In this bilinear inverse problem, we observe P i.i.d. graph signals $\{\mathbf{y}_i\}_{i=1}^P$ that we model as outputs of some unknown diffusion graph filter, i.e., a polynomial in the graph-shift operator of a given graph [9], [19], [29], [35]. The goal is to jointly identify the forward filter coefficients $\mathbf{h}$ and the input signals $\{\mathbf{x}_i\}_{i=1}^P$ that generated the observations. We assume that the inputs are sparse, implying that only a few source nodes inject a signal that spreads through the network; see Fig. 1. This problem can be viewed as an admittedly simplified mathematical abstraction of source localization over networks, with envisioned applications including sensor-based environmental monitoring, opinion formation in social networks, neural signal processing, epidemiology, or disinformation campaigns. We discuss state-of-the-art graph source localization models and learning algorithms in Section V-E, but those can be quite specialized and different than ours. For inspiring prior related model-based blind deconvolution approaches of (non-graph) signals, see e.g., [1], [22], [42].

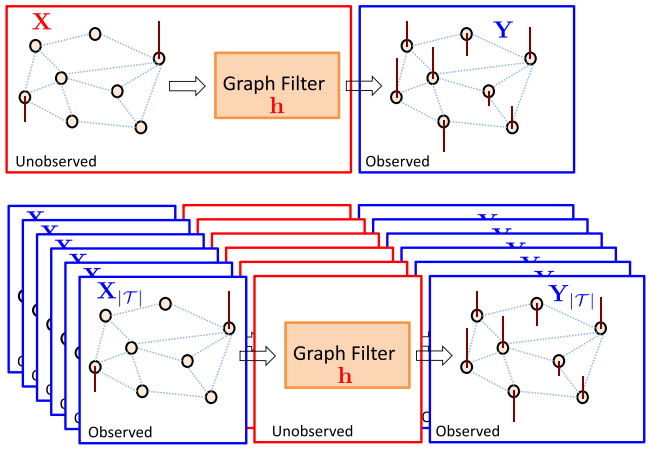


Fig. 1. (Top) Model-based blind deconvolution task [37], [48]. Given P graph signals in $\mathbf{Y}$ modeled as the output of a diffusion graph filter, the goal is to recover the filter coefficients $\mathbf{h}$, and the input signals $\mathbf{X}$ that are assumed to be sparse. Blue panels indicate what is given and red panels represent unobserved quantities. (Bottom) When formulated as a supervised learning problem (an innovation of this work), we rely on a training set $\mathcal{T} := \{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^{|\mathcal{T}|}$ to learn the parameters $\boldsymbol{\Theta}$ of the model $\hat{\mathbf{X}} = \Phi(\mathbf{Y}; \boldsymbol{\Theta})$ by minimizing (5). During training, what is observed and what is not differs from the model-based setting – and the latter is what we encounter during testing or inference.

A. Proposed Approach in Context

To solve this inverse problem in a supervised setting, we propose a novel data-driven machine learning model that blends graph signal processing (GSP) principles with deep learning (DL). Our algorithm unrolling [27] approach leverages the interpretability and parameter efficiency of model-based GSP methods, and also utilizes the power of DL to achieve satisfactory recovery performance and controllable inference complexity for small- to moderate-sized graphs; see [6], [28] for related ideas applied to graph signal denoising and [32], [39], [43], [44] for network topology inference.

Different from most early attempts to localizing sources on networks (e.g., [31], [36], [52]), here we leverage the GSP

Received 31 December 2024; revised 30 July 2025; accepted 4 September 2025. Date of publication 11 September 2025; date of current version 22 September 2025. This work was supported by the Center of Excellence in Data Science, an Empire State Development-designated Center of Excellence. An earlier version of this paper was presented in part at the 2022 European Signal Processing Conference [DOI: 10.23919/EUSIPCO55093.2022.9909689]. The associate editor coordinating the review of this article and approving it for publication was Dr. Giulia Fracastoro. *(Corresponding author: Gonzalo Mateos.)*

The authors are with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627 USA (e-mail: cye7@ur.rochester.edu; gmateosb@ece.rochester.edu).

Digital Object Identifier 10.1109/TSIPN.2025.3608959

toolbox [29] inspired by [30], [37], [50]. The idea in [37] is to cast the (bilinear) blind graph filter identification task as a *linear* inverse problem in the "lifted" rank-one, row-sparse matrix $\mathbf{x}\mathbf{h}^\top$; see also [1], [25] for seminal blind deconvolution work via convex programming. While the rank and sparsity minimization algorithms in [33], [37] can successfully recover sparse inputs along with low-order graph filters (even from a single observation, i.e., with $P = 1$), reliance on matrix lifting can hinder applicability beyond small graphs. Beyond this computational consideration, the overarching assumption of [37] is that the inputs $\{\mathbf{x}_i\}_{i=1}^P$ share a common support when $P > 1$. Moreover, iterative solvers in [33], [37], [50] require carefully tuning step-sizes, as well as carrying out costly grid searches to select regularization parameters in the inverse problems. Instead, the algorithm unrolling-based DL model of this paper learns these parameters (and others) in a end-to-end fashion. Relative to [18], [30], [31], the proposed framework relies on a *convex* estimator of the sparse sources of diffusion, which here we favorably exploit to design a DL architecture as well as to generate training examples.

B. Contributions and Paper Outline

In this context, our starting point is the blind deconvolution formulation in [48]. After reviewing the necessary GSP background, in Section II we reexamine and state the problem in the novel supervised learning setting dealt with here. The *model-based* approach in [48] imposes a mild requirement on invertibility of the graph filter, which facilitates a convex reformulation for the multi-signal case with arbitrary supports (Section III); see also [42] for a time-domain precursor that inspired our line of work in graph settings. While [48] focused on fundamental exact recovery and noise stability guarantees (see also [40], [47] for robustness to graph perturbations), here we shift gears to algorithmic issues and first develop a novel solver based on the alternating-directions method of multipliers (ADMM) [4, Ch. 3.4.4]. The ADMM algorithm in Section III-B is of independent interest as an effective model-based solution to the problem of blind deconvolution on graphs (we reiterate that algorithms were only tangentially treated in e.g., [37], [48], [50]). However, the ADMM's main upshot here is in leveraging its primal and dual variable updates as the blueprint for (sub-)layers of a trainable parametric DL model with prescribed depth. This way we seek to overcome the burden of manually tuning step-size and penalty parameters, plus the time as well as computational overhead that comes with running hundreds or thousands of iterations to attain convergence each time a new problem instance is presented.

To this end, in Section IV we unroll and truncate the ADMM iterations [27], [28], [46], to arrive at a parameterized nonlinear DL architecture for **B**lind **D**econvolution **o**n **G**raphs (BDoG-Net), that we train in an end-to-end fashion using labeled data. This way we leverage inductive biases of a GSP model-based solution in a data-driven trainable deep network, which is interpretable, parameter efficient, and offers controllable complexity during inference [27]. To increase the model's expressive power we explore several customizations to the vanilla BDoG-Net architecture, such as different parameters across layers and learned linear constraints on the inverse filter response. Experiments with both simulated and real network data in Section V demonstrate that BDoG-Net achieves performance on par with the iterative ADMM baseline, while achieving significant post-training speedups. We also show that the model refinements are indeed effective and that BDoG-Net is robust to noise corrupting the observations. All in all, our findings show promise for blind deconvolution on graphs, while they also support the broader prospect of adopting algorithm unrolling as a versatile data-driven tool to tackle network inverse problems; see also [6], [28], [32], [39], [44]. Conclusions are laid out in Section VI, with a discussion of limitations of our approach and potential follow-up work in this space. Some non-essential algorithm construction steps, mathematical arguments, and DL model implementation details are deferred to the appendices. In support of current reproducible research practices, we share the code used to generate the results reported in this paper.

Relationship to the conference precursor [49]: Here we offer full-blown technical details along with extended discussions, schematic diagrams, and appendices. Noteworthy additions over [49] include: (i) BDoG-Net architectural refinements such as learnable constraints and decoupled parameters across layers; (ii) efficient matrix inversion schemes for the model-based ADMM algorithm and the BDoG-Net filter sub-layer; (iii) computational complexity analyses; and (iv) a comprehensive and reproducible performance evaluation protocol. The latter offers comparisons with the iterative ADMM as well as graph neural network (GNN) baselines [9], [41]; a study of inverse filter recovery and source localization performance as a function of the type of random graph ensemble and the number of nodes N; robustness to observation noise and the number of observations P; as well as real data source localization experiments using a version of the Digg 2009 dataset [14].

II. Preliminaries and Problem Statement

We start by introducing the basic graph theoretical background required to formally state the inverse problem dealt with here. Let $G(\mathcal{V}, \mathbf{A})$ denote a weighted and undirected network graph, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of vertices and $\mathbf{A} \in \mathbb{R}_+^{N\times N}$ is the symmetric adjacency matrix. Entry $A_{ij} = A_{ji} \geq 0$ denotes the weight of the edge (i, j).

Graph signals and shift operators: A graph signal $\mathbf{x} : \mathcal{V} \mapsto \mathbb{R}^N$ is an N-dimensional vector, where entry x_i represents the signal value at node $i \in \mathcal{V}$; see [29] for examples in sensor networks, social media, transportation systems, or network neuroscience. As a general algebraic descriptor of network structure relating the entries of $\mathbf{x}$, one can define a *graph-shift operator* $\mathbf{S} \in \mathbb{R}^{N\times N}$ which is a matrix with the same sparsity pattern as G [35]. Accordingly, $\mathbf{S}$ can be viewed as a local, meaning one-hop, diffusion (or aggregation) operator acting on graph signals. See [11], [29] for typical choices including normalized variations of adjacency and Laplacian matrices. Next, we introduce more general operators – graph filters – that linearly combine multi-hop aggregations of graph signals obtained via self compositions of $\mathbf{S}$. Our particular focus will be on simple generative mechanisms behind network diffusion.

A. Graph Filter Models of Linear Network Diffusion

Consider a graph signal $\mathbf{y}$ that is supported on a graph G with shift operator $\mathbf{S}$, and is generated from an input state $\mathbf{x}$ through *linear network diffusion*. Formally, we can write

$$\mathbf{y} = a_0 \prod_{l=1}^{\infty} (\mathbf{I}_N - a_l \mathbf{S})\mathbf{x} = \sum_{l=0}^{\infty} \bar{a}_l \mathbf{S}^l \mathbf{x}, \tag{1}$$

where $\mathbf{S}$ captures one-hop localized interactions among network nodes, and each successive application of the shift in (1) diffuses $\mathbf{x}$ over G. The signal mapping in (1) encompasses several existing models, including heat diffusion, consensus, the DeGroot model of opinion dynamics [8], and mean-field approximations to some workhorse epidemic models [37]. This is a tractable simplification of network phenomena, where input-output relationships can be dynamic and nonlinear.

Despite the infinite degree of the polynomial expressions in (1), the Cayley-Hamilton theorem ensures that they are equivalent to polynomials of degree upper bounded by N [15, pp. 109-110]. Defining the vector of coefficients $\mathbf{h} := [h_0, \ldots, h_{L-1}]^\top$ and the (convolutional) graph filter

$$\mathbf{H} := h_0 \mathbf{I}_N + h_1 \mathbf{S} + \ldots + h_{L-1} \mathbf{S}^{L-1} = \sum_{l=0}^{L-1} h_l \mathbf{S}^l, \tag{2}$$

the signal model in (1) can be rewritten as $\mathbf{y} = \mathbf{H}\mathbf{x}$ for some specific $\mathbf{h}$ and $L \leq N$. As formalized in the ensuing section, in this paper we adopt $\mathbf{y} = \mathbf{H}\mathbf{x}$ as a forward model for the measurements $\mathbf{y}$. We want to recover $\mathbf{x}$ when $\mathbf{h}$ is also unknown. For an up to date and comprehensive survey of graph filters; the interested reader is referred to [19].

Frequency domain representation: Since $\mathbf{S}$ is symmetric, it is diagonalizable as $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, with $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$ collecting the eigenvalues. This spectral decomposition of $\mathbf{S}$ is used in GSP to represent graph filters and signals in the graph frequency domain. Specifically, let us use the eigenvalues of $\mathbf{S}$ to define the Vandermonde matrix $\mathbf{\Psi}_L \in \mathbb{R}^{N \times L}$, where $\Psi_{ij} := \lambda_i^{j-1}$. The frequency representations of a signal $\mathbf{x}$ and filter $\mathbf{h}$ are defined as $\tilde{\mathbf{x}} := \mathbf{V}^\top \mathbf{x}$ and $\tilde{\mathbf{h}} := \mathbf{\Psi}_L \mathbf{h}$, respectively. The former is by definition of the graph Fourier transform (GFT) [29], [34] and the latter follows since the filter output $\mathbf{y} = \mathbf{H}\mathbf{x}$ in the frequency domain can be written as

$$\tilde{\mathbf{y}} = \text{diag}\left(\mathbf{\Psi}_L \mathbf{h}\right) \mathbf{V}^\top \mathbf{x} = \text{diag}\left(\tilde{\mathbf{h}}\right) \tilde{\mathbf{x}} = \tilde{\mathbf{h}} \circ \tilde{\mathbf{x}}. \tag{3}$$

This identity is analogous to the convolution theorem for temporal signals, where we find $\tilde{\mathbf{y}}$ is given by the elementwise product $(\circ)$ of $\tilde{\mathbf{x}}$ and the filter's frequency response $\tilde{\mathbf{h}}$.

B. Problem Statement

For given graph G with shift operator $\mathbf{S}$, consider a diffusion filter $\mathbf{H} = \sum_{l=0}^{L-1} h_l \mathbf{S}^l$ whose coefficients $\mathbf{h}$ are *unknown*. Like the graph topology, the filter order $L \leq N$ is assumed to be given. Suppose we observe P diffused signals that we arrange in a matrix $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_P] \in \mathbb{R}^{N \times P}$, where $\mathbf{Y} = \mathbf{H}\mathbf{X}$ and the latent inputs are $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_P] \in \mathbb{R}^{N \times P}$. We make the following assumption on the input signals.

Assumption 1 (Input sparsity): Signals $\mathbf{X} \in \mathbb{R}^{N \times P}$ are *sparse* with at most $S \ll N$ non-zero entries per column.

In this context, blind deconvolution amounts to jointly estimating sparse $\mathbf{X}$ and the filter coefficients $\mathbf{h}$ up to scaling and (possibly) permutation ambiguities [50]; see also Fig. 1 (top) for a depiction of this task first studied in [37]. Assumption 1 is well justified when the signals in $\mathbf{Y}$ represent diffused versions of a *few* localized sources in G, here indexed by $\text{supp}(\mathbf{X}) := \{(i,j) \mid X_{ij} \neq 0\}$ (columns of $\mathbf{X}$ may have different supports). Also, without such structural constraints the problem is ill-posed, because the number of unknowns $NP + L$ in $\{\mathbf{X}, \mathbf{h}\}$ exceeds the NP observations in $\mathbf{Y}$.

All in all, using (3) we formulate the feasibility problem

$$\text{find } \{\mathbf{X}, \mathbf{h}\} \text{ s. to } \mathbf{Y} = \mathbf{V}\text{diag}\left(\mathbf{\Psi}_L \mathbf{h}\right) \mathbf{V}^\top \mathbf{X}, \ \|\mathbf{X}\|_0 \leq PS, \tag{4}$$

where the ℓ_0-(pseudo) norm $\|\mathbf{X}\|_0 := |\text{supp}(\mathbf{X})|$ counts the non-zero entries in $\mathbf{X}$. In words, we are after the solution to a system of bilinear equations subject to a sparsity constraint in $\mathbf{X}$; a hard problem due to the non-convex ℓ_0-norm as well as the bilinear constraints. To deal with the latter, similar to [42], [50] we will henceforth assume that the filter $\mathbf{H}$ is invertible.

Blind deconvolution as a supervised learning problem: Suppose that $\mathbf{X}$ is drawn from some distribution of sparse matrices, say the Bernoulli-Gaussian model for which one can establish (4) is identifiable [50, Remark 1]. Likewise, suppose the filter taps $\mathbf{h}$ are drawn from a distribution such that $\mathbf{H}$ is invertible with high probability. Then given independent training samples $\mathcal{T} := \{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^{|\mathcal{T}|}$ adhering to (1), our goal in this paper is to learn a judicious parametric mapping that predicts $\hat{\mathbf{X}} = \Phi(\mathbf{Y}; \mathbf{\Theta})$ by minimizing a loss function

$$L(\mathbf{\Theta}) := \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \ell(\mathbf{X}_i, \Phi(\mathbf{Y}_i; \mathbf{\Theta})), \tag{5}$$

where $\mathbf{\Theta}$ are learnable parameters; see Fig. 1 (bottom) for a schematic depiction of the training setting. Depending on the application, a training set may be available from historical data, or for instance it may be generated using a simulator of the diffusion process. Alternatively, given observations $\mathbf{Y}_i$ one can obtain source labels $\mathbf{X}_i$ by solving a convex optimization problem as discussed in the ensuing section; see also [37], [50]. This way, the perspective is to *learn to approximate the minimizers* of a relaxation to (4). While admittedly the data-generation and training phases can be time consuming, they are offline and need to be performed once. In turn, the upshot is a faster method during testing or inference.

We will design the deep network $\Phi(\cdot; \mathbf{\Theta})$ in Section IV, using iterations of a model-based solution we develop in Section III as a layer by layer blueprint. The particular choice of the loss ℓ will be discussed in Section V. While not made explicit in our notation, $\Phi(\cdot; \mathbf{\Theta})$ makes internal predictions of the diffusion filter from which $\hat{\mathbf{X}}$ is obtained at the output.

III. Model-Based Blind Deconvolution on Graphs

Here we review the model-based solution to the blind deconvolution problem proposed in [48], [50], which relies on a convex

relaxation of (4) when the diffusion filter is invertible. Then we develop novel ADMM iterations to solve said relaxation, which we unroll in Section IV to obtain the BDoG-Net model that we train using data by minimizing (5).

A. Convex Relaxation for Invertible Graph Filters

Conditions for the invertibility of a graph filter can be readily obtained in the graph spectral domain. Indeed, the filter's frequency response $\tilde{h}_i$ should not vanish at any of the discrete frequencies λ_i, $i = 1, \ldots, N$, otherwise the input information in the nulled frequency modes is lost; see (3).

Assumption 2 (Filter invertibility): The graph filter $\mathbf{H} = \mathbf{V}\text{diag}(\tilde{\mathbf{h}})\mathbf{V}^\top$ is invertible, meaning $\tilde{h}_i = \sum_{l=0}^{L-1} h_l \lambda_i^l \neq 0$, for all $i = 1, \ldots, N$.

Under Assumption 2, one can show that the inverse operator $\mathbf{G} := \mathbf{H}^{-1}$ is also a polynomial graph filter on G, of degree at most $N-1$ [35, Theorem 4]. To be more specific, let $\mathbf{g} \in \mathbb{R}^N$ be the vector of inverse-filter coefficients, i.e., $\mathbf{H}^{-1} := \mathbf{G} = \sum_{l=0}^{N-1} g_l \mathbf{S}^l$. Then one can equivalently rewrite the forward model $\mathbf{Y} = \mathbf{H}\mathbf{X}$ for the observations as

$$\mathbf{X} = \mathbf{G}\mathbf{Y} = \mathbf{V}\text{diag}(\tilde{\mathbf{g}})\mathbf{V}^\top\mathbf{Y}, \tag{6}$$

where $\tilde{\mathbf{g}} := \mathbf{\Psi}_N \mathbf{g} \in \mathbb{R}^N$ is the inverse filter's frequency response and $\mathbf{\Psi}_N \in \mathbb{R}^{N\times N}$ is Vandermonde. Naturally, $\mathbf{G} = \mathbf{H}^{-1}$ implies the condition $\tilde{\mathbf{g}} \circ \tilde{\mathbf{h}} = \mathbf{1}_N$ on the frequency responses. Leveraging (6), one can recast (4) as a *linear* inverse problem

$$\min_{\{\mathbf{X},\tilde{\mathbf{g}}\}} \|\mathbf{X}\|_0, \ \text{s. to } \mathbf{X} = \mathbf{V}\text{diag}(\tilde{\mathbf{g}})\mathbf{V}^\top\mathbf{Y}, \ \mathbf{X} \neq \mathbf{0}. \tag{7}$$

The ℓ_0 norm in (7) makes the problem NP-hard to optimize. Over the last two decades or so, convex-relaxation approaches to tackle sparsity-minimization problems have enjoyed remarkable success, since they often entail no loss of optimality; see Remark 1. Accordingly, we instead: (i) seek to minimize the ℓ_1-norm convex surrogate of the cardinality function, that is $\|\mathbf{X}\|_{1,1} = \sum_{i,j} |X_{ij}|$; and (ii) express the filter in the graph spectral domain as in (6) to obtain the cost function

$$\begin{aligned}\|\mathbf{X}\|_{1,1} &= \|\mathbf{G}\mathbf{Y}\|_{1,1} \\ &= \|\mathbf{V}\text{diag}(\tilde{\mathbf{g}})\mathbf{V}^\top\mathbf{Y}\|_{1,1} \\ &= \|(\mathbf{Y}^\top\mathbf{V} \odot \mathbf{V})\tilde{\mathbf{g}}\|_1.\end{aligned}$$

In arriving at the last equality we used that $\|\mathbf{X}\|_{1,1} = \|\text{vec}[\mathbf{X}]\|_1$ and invoked properties of the vectorization operator, where $\odot$ denotes the Khatri-Rao (i.e., columnwise Kronecker) product. This suggests solving the convex ℓ_1-synthesis problem (a linear program), e.g., [51], namely

$$\widehat{\tilde{\mathbf{g}}} = \underset{\tilde{\mathbf{g}}\in\mathbb{R}^N}{\arg\min} \ \|(\mathbf{Y}^\top\mathbf{V} \odot \mathbf{V})\tilde{\mathbf{g}}\|_1, \quad \text{s. to } \mathbf{1}_N^\top\tilde{\mathbf{g}} = 1. \tag{8}$$

While the linear constraint in (8) avoids the trivial solution $\widehat{\tilde{\mathbf{g}}} = 0$, it also serves to fix the (arbitrary) scale of the estimated filter. Once the frequency response $\widehat{\tilde{\mathbf{g}}}$ of the inverse filter is recovered, the input signals can be reconstructed via $\text{vec}[\hat{\mathbf{X}}] = (\mathbf{Y}^\top\mathbf{V} \odot \mathbf{V})\widehat{\tilde{\mathbf{g}}}$. Because $\mathbf{S}$ and L are known, one can also recover the filter $\mathbf{H}$ as in system identification, if so desired.

Remark 1 (Recovery and stability guarantees): Sufficient conditions were derived in [48], under which (8) succeeds in exactly recovering the true inverse filter response with high probability. This result holds for Bernoulli-Gaussian distributed $\mathbf{X}$ [23], [42]. Stability to additive noise corrupting the observations $\mathbf{Y}$ [48], or, perturbations in the graph-shift operator eigenbasis $\mathbf{V}$ [47], has also been established.

All in all, under Assumption 2 one can readily use e.g., an off-the-shelf interior-point method to solve (8) efficiently [48], [50]. Next, we propose a specialized sparsity-minimization algorithm that exploits the problem's unique structure.

B. ADMM Algorithm

Problem (8) can be solved using the ADMM. Let $\mathbf{x} = \text{vec}[\mathbf{X}] \in \mathbb{R}^{NP}$ and denote $\mathbf{Z} := \mathbf{Y}^\top\mathbf{V} \odot \mathbf{V} \in \mathbb{R}^{NP\times N}$. Using variable splitting, problem (8) can be equivalently written as

$$\min_{\{\mathbf{x},\tilde{\mathbf{g}}\}} \|\mathbf{x}\|_1, \ \text{s. to } \mathbf{Z}\tilde{\mathbf{g}} - \mathbf{x} = \mathbf{0}_{NP}, \ \mathbf{1}_N^\top\tilde{\mathbf{g}} = c, \tag{9}$$

where $c = 1$, but we will henceforth treat it as a generic nonzero constant in case we want to adjust the scale of $\tilde{\mathbf{g}}$. Associating dual variables $\boldsymbol{\lambda} \in \mathbb{R}^{NP}$ and $\mu \in \mathbb{R}$ to the equality constraints in (9), the augmented Lagrangian function becomes

$$\begin{aligned}\mathcal{L}_\rho(\mathbf{x}, \tilde{\mathbf{g}}, \boldsymbol{\lambda}, \mu) = \|\mathbf{x}\|_1 + \frac{\rho_\lambda}{2}\|\mathbf{Z}\tilde{\mathbf{g}} - \mathbf{x} + \boldsymbol{\lambda}/\rho_\lambda\|_2^2 \\ + \frac{\rho_\mu}{2}(\mathbf{1}_N^\top\tilde{\mathbf{g}} - c + \mu/\rho_\mu)^2,\end{aligned} \tag{10}$$

after completing the squares, where ρ_λ and ρ_μ are non-negative penalty coefficients. Letting $\mathbf{\Gamma} := \rho_\lambda\mathbf{Z}^\top\mathbf{Z} + \rho_\mu\mathbf{1}_N\mathbf{1}_N^\top$ for notational convenience, then the ADMM [4], [5] updates are given by ($k = 0, 1, 2, \ldots$ will henceforth denote iterations)

$$\tilde{\mathbf{g}}[k+1] = \mathbf{\Gamma}^{-1}\left[\mathbf{Z}^\top(\rho_\lambda\mathbf{x}[k] - \boldsymbol{\lambda}[k]) + (\rho_\mu c - \mu[k])\mathbf{1}_N\right], \tag{11}$$

$$\mathbf{x}[k+1] = \mathcal{S}_{\rho_\lambda^{-1}}(\mathbf{Z}\tilde{\mathbf{g}}[k+1] + \boldsymbol{\lambda}[k]/\rho_\lambda), \tag{12}$$

$$\boldsymbol{\lambda}[k+1] = \boldsymbol{\lambda}[k] + \rho_\lambda(\mathbf{Z}\tilde{\mathbf{g}}[k+1] - \mathbf{x}[k+1]), \tag{13}$$

$$\mu[k+1] = \mu[k] + \rho_\mu(\mathbf{1}_N^\top\tilde{\mathbf{g}}[k+1] - c). \tag{14}$$

For completeness, (11)–(14) are derived in Appendix A. The soft-thresholding operator $\mathcal{S}_{\rho_\lambda^{-1}}(\cdot) = \text{sign}(\cdot)\max(|\cdot| - \rho_\lambda^{-1}, 0)$ in (12) acts component-wise on the entries of its vector argument. The initialization $\{\mathbf{x}[0], \boldsymbol{\lambda}[0], \mu[0]\}$ can be arbitrary, and we typically let the initial conditions be equal to zero.

Different from the solvers in [33], [37], the provably convergent ADMM updates are free of expensive singular-value decompositions per iteration. The inversion of the $N \times N$ matrix $\mathbf{\Gamma}$ is done once, offline, and $\mathbf{\Gamma}^{-1}\mathbf{Z}^\top$, $\mathbf{\Gamma}^{-1}\mathbf{1}_N$ are cached to run the iterations. Even more, we show in Appendix B that the matrix $\mathbf{Z}\mathbf{Z}^\top$ is diagonal. Thus, $\mathbf{\Gamma}$ is a rank-one correction of a diagonal matrix, which can be computed efficiently using the matrix inversion lemma; see Appendix C for the details and associated computational complexity analysis.

In the next section, we unroll the ADMM iterations (11)–(14) to arrive at the trainable parametric model $\Phi(\mathbf{Y}; \mathbf{\Theta})$.

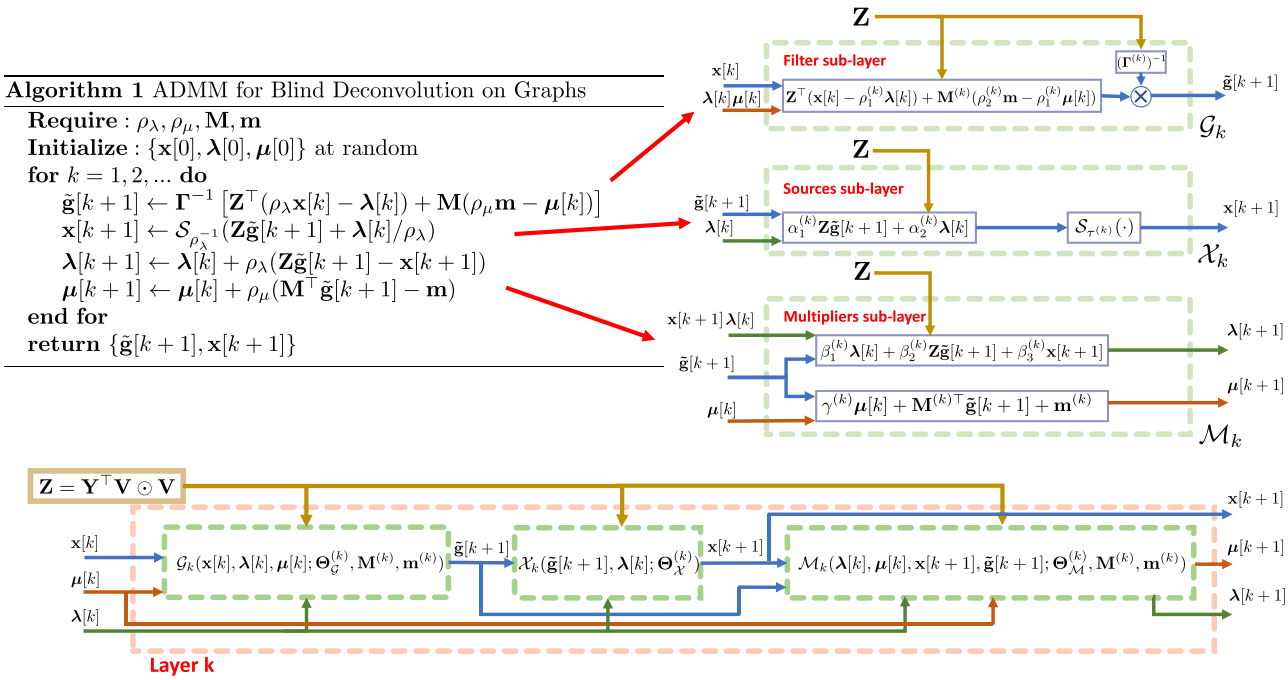


Fig. 2. The layerwise structure of BDoG-Net, a DL model $\hat{\mathbf{X}} = \Phi(\mathbf{Y}; \boldsymbol{\Theta})$ consisting of K layers. Layer k is a composition of three sub-layers: (i) a filter sub-layer $\mathcal{G}_k$; followed by (ii) a sources sub-layer $\mathcal{X}_k$; followed by (iii) a multipliers sub-layer $\mathcal{M}_k$. Each of these sub-layers is a direct mapping of a primal or dual variable update in the ADMM algorithm of Section III-B, tabulated as Algorithm 1 for convenience. Learnable parameters in $\boldsymbol{\Theta}$ are $\boldsymbol{\Theta}_{\mathcal{G}}^{(k)} = \{\rho_1^{(k)}, \rho_2^{(k)}\}$, $\boldsymbol{\Theta}_{\mathcal{X}}^{(k)} = \{\alpha_1^{(k)}, \alpha_2^{(k)}, \alpha_3^{(k)}, \tau^{(k)}\}$, $\boldsymbol{\Theta}_{\mathcal{M}}^{(k)} = \{\beta_1^{(k)}, \beta_2^{(k)}, \beta_3^{(k)}, \gamma^{(k)}\}$, $\mathbf{M}^{(k)}, \mathbf{m}^{(k)}$, $k = 1, \ldots, K$.

IV. Blind Deconvolution Via Unrolling

The algorithm unrolling (or deep unfolding) principle was pioneered in [13] for the problem of sparse coding natural images using overparameterized dictionaries. The technical approach in [13] was to truncate and map *iterations* of the iterative shrinkage-thresholding algorithm (ISTA) [2] to *layers* in a deep network that can be trained from data. Learnable weights are often optimization step-sizes, regularization and penalty parameters, or other matrices when additional expressive power is needed. One of the greatest DL challenges has been architectural search, and unrolling offers a principled approach to model design by using tested algorithms as architectural templates. The perspective is to *learn to approximate* solutions with substantial computational savings during inference, relative to the optimization algorithm. While the former process entails a forward pass through a feedforward neural network (NN) with few layers, the latter could entail running hundreds (pr thousands) of iterations until convergence.

Beyond parsimonious signal modeling, there has been a surge in popularity of unrolled deep networks for a wide variety of applications in signal and image processing; see e.g., [27] for a recent review. Most relevant to our approach is the unrolling of ADMM iterations for undersampled image reconstruction [46], and recent advances to learn from graph data [6], [28], [32], [39], [43], [44]. However, none of these works has dealt with the blind deconvolution task over networks via the formulation in Section III (which has broader applicability; see the opening paragraph in Section I).

A. BDoG-Net: ADMM as Architectural Blueprint

We construct the BDoG-Net architecture by unrolling the ADMM iterations (11)–(14) into a DL model. To this end, we map individual primal and dual variable updates as sub-layers within a layer; see Fig. 2 for a schematic depiction of this process. We then compose a prescribed number K of layers to constitute the parametric mapping $\Phi(\mathbf{Y}; \boldsymbol{\Theta})$. ADMM penalty coefficients $\{\rho_\lambda, \rho_\mu\}$ will be treated as learnable parameters in $\boldsymbol{\Theta}$. Observations $\mathbf{Y}$ are inputs to the NN. Just like in the model-based approach in Section III, the architecture leverages graph structure information through the eigenvectors $\mathbf{V}$. The K-th layer output is used to form the source predictions $\hat{\mathbf{X}}$.

Architectural refinements: In designing BDoG-Net's sub-layers, we will deviate slightly from a strict ADMM unrolling of (11)–(14) in order to enhance overall predictive performance. For instance, in each sub-layer we introduce several additional parameters to increase the model's expressive power.

In the original formulation (8), we included the linear constraint $\mathbf{1}^\top\tilde{\mathbf{g}} = 1$ as a simple mechanism to prevent an undesirable all-zero solution and fix the (otherwise arbitrary) scale of the solution. However, this rigid constraint might limit the method's recovery potential in some cases. In addition, the scale fixing parameter $c = 1$ is no longer needed in the supervised learning setting dealt with here. Scale information will be implicitly conveyed through examples $\mathcal{T} := \{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^{|\mathcal{T}|}$, and can thus be learned during training. These considerations motivate replacing the constraint $\mathbf{1}^\top\tilde{\mathbf{g}} = c$ in (9) with $\mathbf{M}^\top\tilde{\mathbf{g}} = \mathbf{m}$, where

$\mathbf{M} \in \mathbb{R}^{N\times d}$ and $\mathbf{m} \in \mathbb{R}^d$ are learnable parameters. The associated modifications to the ADMM algorithm in Section III-B are straightforward, and we will not spell them out here in the interest of brevity.

We will also forgo the parameter sharing constraint imposed by the unrolled ADMM iterations. This is standard practice [27], and we have experimentally found that a model with different parameters per layer performs better and leads to more stable training. We would be remiss by not mentioning there is a trade-off, since a NN with coupled parameters offers the flexibility to train with K_{train} layers and then perform inference over a deeper architecture with $K_{\text{test}} > K_{\text{train}}$ layers. We leave this exploration as future work. Next, we describe the design of each sub-layer in detail.

Filter sub-layer: This sub-layer $\mathcal{G}_k$ refines the inverse filter coefficient estimate $\tilde{\mathbf{g}}[k+1]$ at layer k, based on the source estimates $\mathbf{x}[k]$ and the dual variables $\{\boldsymbol{\lambda}[k], \boldsymbol{\mu}[k]\}$ from the previous layer. We mimic the $\tilde{\mathbf{g}}$ update in (11), and introduce some minor tweaks. To avoid problems with the inversion of $\boldsymbol{\Gamma}$ in the eventuality $\rho_\lambda = \rho_\mu = 0$ during training, we opt for the reparameterization $\rho_1 := 1/\rho_\lambda$, $\rho_2 := \rho_\mu/\rho_\lambda$ and impose non-negativity constraints on both parameters. Moreover, we consider decoupled parameters $\boldsymbol{\Theta}_{\mathcal{G}}^{(k)} := \{\rho_1^{(k)}, \rho_2^{(k)}\}$ across layers $k = 1, \ldots, K$, to increase the network capacity. Finally, the constraint's constant vector $\mathbf{1}_N$ and the scale-normalization constant c are replaced by learnable parameters $\{\mathbf{M}^{(k)}, \mathbf{m}^{(k)}\}_{k=1}^K$, thus obtaining [cf. (11)]

$$\begin{aligned}\tilde{\mathbf{g}}[k+1] = &\left(\mathbf{Z}^\top\mathbf{Z} + \rho_2^{(k)}\mathbf{M}^{(k)}\mathbf{M}^{(k)\top}\right)^{-1}\Big[\mathbf{Z}^\top\Big(\mathbf{x}[k]\\ &-\rho_1^{(k)}\boldsymbol{\lambda}[k]\Big) + \mathbf{M}^{(k)}\left(\rho_2^{(k)}\mathbf{m}^{(k)} - \rho_1^{(k)}\boldsymbol{\mu}[k]\right)\Big]\\ &:= \mathcal{G}_k\left(\mathbf{x}[k], \boldsymbol{\lambda}[k], \boldsymbol{\mu}[k]; \boldsymbol{\Theta}_{\mathcal{G}}^{(k)}, \mathbf{M}^{(k)}, \mathbf{m}^{(k)}\right), \quad (15)\end{aligned}$$

where $\rho_1^{(k)}, \rho_2^{(k)} \geq 0$, for $k = 1, \ldots, K$.

Recall that $\mathbf{Z} := \mathbf{Y}^\top\mathbf{V} \odot \mathbf{V}$, so every time a new data mini-batch $\mathbf{Y}$ is to be processed one needs to (re)invert matrices $\boldsymbol{\Gamma}^{(k)}$; see Appendices C and D for a computationally-efficient implementation, especially when $d = 1$.

Sources sub-layer: Here we update the source estimates $\mathbf{x}[k]$ based on $\tilde{\mathbf{g}}[k+1]$ in (15) and the multiplier $\boldsymbol{\lambda}[k]$. The sub-layer $\mathcal{X}_k$ imitates (13), but instead of a single tunable parameter ρ_λ we introduce learnable combination weights $\{\alpha_1^{(k)}, \alpha_2^{(k)}\}_{k=1}^K$ and thresholds $\{\tau^{(k)}\}_{k=1}^K$; all collected in $\boldsymbol{\Theta}_{\mathcal{X}}^{(k)}$, for each sub-layer $k = 1, \ldots, K$. We propose [cf. (12)]

$$\begin{aligned}\mathbf{x}[k+1] &= \mathcal{S}_{\tau^{(k)}}\left(\alpha_1^{(k)}\mathbf{Z}\tilde{\mathbf{g}}[k+1] + \alpha_2^{(k)}\boldsymbol{\lambda}[k]\right)\\ &:= \mathcal{X}_k\left(\tilde{\mathbf{g}}[k+1], \boldsymbol{\lambda}[k]; \boldsymbol{\Theta}_{\mathcal{X}}^{(k)}\right), \quad (16)\end{aligned}$$

where the sparsifying thresholds are naturally constrained as $\tau^{(k)} \geq 0$, for $k = 1, \ldots, K$. Notice how (16) implements a simple linear filter on the sub-layer inputs $\{\tilde{\mathbf{g}}[k+1], \boldsymbol{\lambda}[k1]\}$, followed by a point-wise nonlinear activation, which is reminiscent of vanilla NN layers.

Multipliers sub-layer. In this simple linear sub-layer $\mathcal{M}_k$, we perform parallel updates of the Lagrange multipliers $\{\boldsymbol{\lambda}[k+1], \boldsymbol{\mu}[k+1]\}$ by combining $\{\boldsymbol{\lambda}[k], \boldsymbol{\mu}[k]\}$ from layer $k-1$ and the primal variables $\{\tilde{\mathbf{g}}[k+1], \mathbf{x}[k+1]\}$. Layer-$k$ combination weights $\boldsymbol{\Theta}_{\mathcal{M}}^{(k)}$ are learnable parameters $\{\beta_1^{(k)}, \beta_2^{(k)}, \beta_3^{(k)}\}_{k=1}^K$ and $\{\gamma^{(k)}\}_{k=1}^K$, leading to [cf. (13)–(14)]

$$\boldsymbol{\lambda}[k] = \beta_1^{(k)}\boldsymbol{\lambda}[k-1] + \beta_2^{(k)}\mathbf{Z}\tilde{\mathbf{g}}[k] + \beta_3^{(k)}\mathbf{x}[k], \quad (17)$$

$$\boldsymbol{\mu}[k] = \gamma^{(k)}\boldsymbol{\mu}[k-1] + \mathbf{M}^{(k)\top}\tilde{\mathbf{g}}[k] + \mathbf{m}^{(k)}. \quad (18)$$

Each BDoG-Net layer is thus a sequential composition of these three sub-layers, in the order we have introduced them: first the filter sub-layer $\mathcal{G}_k$, then the sources $\mathcal{X}_k$, and finally the multipliers sub-layer $\mathcal{M}_k$. Notice how the data embedded in $\mathbf{Z}$ is not only fed to the first layer $K = 1$, but to all subsequent layers in a way akin to residual neural networks (ResNets).

In closing, we note that the intial states $\{\mathbf{x}[0], \boldsymbol{\lambda}[0], \mu[0]\}$ can be: (i) used as a means to incorporate prior information (especially on the source locations $\mathbf{x}$); (ii) randomly initialized as we do in the ensuing experiments; or (iii) learned from data along with $\boldsymbol{\Theta}$ as it is customary with recurrent neural networks (RNNs). Going all the way to layer K, source predictions are generated as $\hat{\mathbf{X}} = \Phi(\mathbf{Y}; \boldsymbol{\Theta}) = \text{unvec}[(\mathbf{Y}^\top\mathbf{V} \odot \mathbf{V})\tilde{\mathbf{g}}[K]]$. Given a training set $\mathcal{T} := \{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^{|\mathcal{T}|}$ of e.g., syntethic data, or, real signals $\mathbf{Y}_i$ and input estimates obtained using ADMM, learning is accomplished by using mini-batch stochastic gradient descent to minimize the loss $L(\boldsymbol{\Theta})$ in (5). Parameter efficiency is a well-documented feature of unrolled architectures [27]. Further training details, including the specification of the loss and hyperparameter choices, are outlined in the numerical evaluation Section V and in Appendix E.

V. Numerical Evaluation

We perform a comprehensive numerical evaluation to assess the recovery performance and computational efficiency of the proposed BDoG-Net architecture. We test the model in various instances of the blind deconvolution task described in Section II-B. First we run simulated tests (see Section V-A for a general description of the experimental setting) and compare BDoG-Net against the iterative ADMM algorithm in Section V-B, across different types of graphs (Section V-C), and against a selection GNN [10] in Section V-D. Finally, in Section V-E we examine a real data scenario by leveraging the Digg 2009 data set [14]. For this challenging task, we compare BDoG-Net against the Invertible Validity-aware Graph Diffusion (IVGD) NN approach for source localization in [41] and the Source Localization Variational AutoEncoder (SL-VAE) in [24]. The Python notebook with the code used to obtain the experimental results reported here is publicly available at https://hajim.rochester.edu/ece/sites/gmateos/code/BDoG-Net.zip.

A. General Experimental Settings

Synthetic data generation: The graph shift operator is selected as the degree-normalized adjacency matrix $\mathbf{S} = \text{diag}^{-\frac{1}{2}}(\mathbf{A}\mathbf{1}_N) \cdot \mathbf{A} \cdot \text{diag}^{-\frac{1}{2}}(\mathbf{A}\mathbf{1}_N)$. With $\mathcal{T}$ denoting the training set, the sparse inputs $\mathbf{X} \in \mathbb{R}^{N\times|\mathcal{T}|}$ are drawn from the Bernoulli-Gaussian model; e.g., [48]. Specifically, $\mathbf{X} = \boldsymbol{\Omega} \circ \mathbf{R}$, where $\boldsymbol{\Omega} \in \mathbb{R}^{N\times|\mathcal{T}|}$

has i.i.d. entries $\Omega_{ij} \sim \text{Bernoulli}(\theta)$, with sparsity level $\theta = 0.15$; and $\mathbf{R} \in \mathbb{R}^{N\times|\mathcal{T}|}$ is a random matrix (independent of $\mathbf{\Omega}$) with i.i.d. entries $R_{ij} \sim \text{Normal}(0,1)$. Unless otherwise stated, realizations of filter coefficients are generated as $\mathbf{h} = (\mathbf{e}_1 + \varphi\mathbf{b})/\|\mathbf{e}_1 + \varphi\mathbf{b}\|_1$, where $\mathbf{e}_1 = [1, 0, \ldots, 0]^\top \in \mathbb{R}^L$ is the first canonical basis vector and $\mathbf{b} \sim \text{Normal}(\mathbf{0}_L, \mathbf{I}_L)$. The analysis in [48] suggests that recovery is harder for "less-impulsive" filters, so we henceforth stick to a challenging instance where $\varphi = 1$. The filter order is chosen to be $L = 5$.

For each training epoch, the training samples in $\mathcal{T}$ are randomly split into Q mini-batches of $P = |\mathcal{T}|/Q$ signals, namely $\{\mathbf{X}_q\}_{q=1}^Q \in \mathbb{R}^{N\times P}$. We sample Q graph filter coefficients $\{\mathbf{h}_q\}_{q=1}^Q$ ($L = 5$, $\varphi = 1$) and randomly assign them to the input signal mini-batches to generate the observations $\mathbf{Y}_q = \mathbf{V}\text{diag}(\mathbf{\Psi}_L\mathbf{h}_q)\mathbf{V}^\top\mathbf{X}_q, q = 1, \ldots, Q$. We use a validation set $\mathbf{X}_{\text{val}}$ (Bernoulli-Gaussian with $\theta = 0.15$) of size $P_{\text{val}} = P$, with observations $\mathbf{Y}_{\text{val}} = \mathbf{V}\text{diag}(\mathbf{\Psi}_L\mathbf{h}_{\text{val}})\mathbf{V}^\top\mathbf{X}_{\text{val}}$, where $\mathbf{h}_{\text{val}}$ is drawn from the same distribution as $\{\mathbf{h}_q\}_{q=1}^Q$.

Graphs: In the following experiments, we implement BDoG-Net and compare it with other approaches using various undirected and unweighted random graphs, as well as real-world networks. In Section V-B, we use Erdős-Rényi (ER) random graphs with $N = 20$ nodes and edge formation probability $p = 0.3$. In Section V-C, we examine BDoG-Net's recovery performance across various graph ensembles, including: (i) ER ($N = 20$, $p = 0.3$); (ii) stochastic block model (SBM) with $N = 20$ nodes and $N_C = 3$ communities (with edge probabilities $p_{\text{within}} = 0.8$, $p_{\text{between}} = 0.2$); (iii) Barabási–Albert (BA) with $N = 20$ nodes; (iv) random geometric (RG) with $N = 20$ nodes and critical distance $r_{\text{cri}} = 0.2$; and (v) real-world social networks such as dolphins ($N = 62$) and Zachary's karate club ($N = 34$). In Section V-D, we evaluate BDoG-Net on the same SBM graph used in Section V-C. In Section V-E, we use sub-graphs with $N = 20$ nodes randomly sampled from the Digg friendship network [14]. These sub-graphs and their construction are discussed in further detail in Section V-E.

Training method: We train BDoG-Net with $K = 5$ layers and use the relative root mean square error (RE) of $\mathbf{X}$ as loss function. Notice that if $\{\hat{\mathbf{X}}, \hat{\mathbf{h}}\}$ is a solution to the bilinear problem, then so is $\{-\hat{\mathbf{X}}, -\hat{\mathbf{h}}\}$ and accordingly we minimize

$$L(\mathbf{\Theta}) = \sum_{q=1}^{Q} \min\left(\frac{\|\Phi(\mathbf{Y}_q;\mathbf{\Theta}) - \mathbf{X}_q\|_F}{\|\mathbf{X}_q\|_F}, \frac{\|\Phi(\mathbf{Y}_q;\mathbf{\Theta}) + \mathbf{X}_q\|_F}{\|\mathbf{X}_q\|_F}\right)$$

using the Adam optimizer [21] implemented in PyTorch.

We initialize $\{\rho_1^{(k)}, \rho_2^{(k)}, \tau^{(k)}\}_{k=1}^K$ as i.i.d. samples from the uniform distribution in [0,1], since these parameters are constrained to be non-negative. All other parameters in $\mathbf{\Theta}$ are randomly drawn from a standard Gaussian distribution. We consider 30 epochs for training. In each epoch, we estimate the sparse sources $\{\Phi(\mathbf{Y}_q; \hat{\mathbf{\Theta}}_q)\}_{q=1}^Q$ using the training batches $\{\mathbf{Y}_q; \mathbf{X}_q\}_{q=1}^Q$. We choose one batch out of every 20 batches to compute the loss on the validation set $\{\mathbf{Y}_{\text{val}}; \mathbf{X}_{\text{val}}\}$ and record both the value of the loss and the network parameters. In the end, we select the model $\hat{\mathbf{\Theta}}$ that has minimum validation loss across the entire training process.

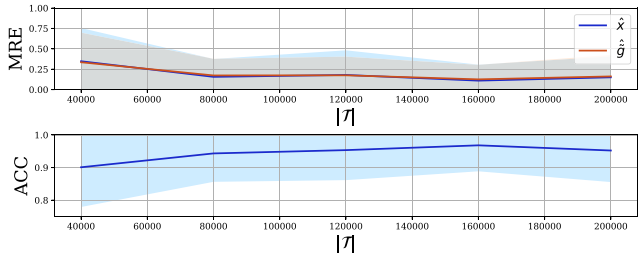


Fig. 3. Recovery performance vs. training set size $|\mathcal{T}|$. (top) Mean test relative error (MRE) of the recovered source signal $\hat{\mathbf{X}}$ (blue) and estimated inverse filter frequency response $\hat{\tilde{\mathbf{g}}}$ (red), respectively. The shaded region represents the estimated standard error, after averaging over 10 realizations. (bottom) Mean accuracy (ACC) of source support estimation and estimated standard deviation. The best performance is attained for $|\mathcal{T}| \geq 160$k, but gains are marginal beyond 80k signals.

Testing protocol and figures of merit: For testing, we sample an independent test set $\{\mathbf{X}_{\text{test}}, \mathbf{h}_{\text{test}}\}$, where $\mathbf{X}_{\text{test}} \in \mathbb{R}^{N\times P_{\text{test}}}$, $P_{\text{test}} = P$. We generate diffused signals $\mathbf{Y}_{\text{test}} = \mathbf{V}\text{diag}(\mathbf{\Psi}_L\mathbf{h}_{\text{test}})\mathbf{V}^\top\mathbf{X}_{\text{test}} + \eta\mathbf{N}$, where $\mathbf{N} \sim \text{Uniform}(-1,1)^{N\times P_{\text{test}}}$ and η is the noise level. We do a forward inference pass through the trained BDoG-Net model to generate predictions $\hat{\mathbf{X}} = \Phi(\mathbf{Y}_{\text{test}}; \hat{\mathbf{\Theta}})$, and use the ground-truth sources $\mathbf{X}_{\text{test}}$ to assess recovery performance. Naturally, the quality of the estimated inverse filter frequency response $\hat{\tilde{\mathbf{g}}}_{\text{test}}$ can be evaluated as well.

For performance assessment, we consider two figures of merit. Firstly, we evaluate the test error $\text{RE} = \|\Phi(\mathbf{Y}_{\text{test}}; \hat{\mathbf{\Theta}}) - \mathbf{X}_{\text{test}}\|_F / \|\mathbf{X}_{\text{test}}\|_F$. We also compute the accuracy (ACC) in recovering the support of $\mathbf{X}_{\text{test}}$, i.e., the source locations. To identify the support, we introduce a thresholding approach with threshold $\kappa = 10^{-1}$. If a predicted entry satisfies $|[\Phi(\mathbf{Y}_{\text{test}}; \hat{\mathbf{\Theta}})]_{ij}| \geq \kappa$, the index pair (i, j) will be considered a member of the estimated support $\text{supp}_\kappa(\cdot)$. Accordingly, the recovered sources are $\hat{\mathcal{I}}_{\text{test}} := \text{supp}_\kappa(\Phi(\mathbf{Y}_{\text{test}}; \hat{\mathbf{\Theta}}))$. We also apply the threshold to the ground-truth sources, so the sought support set is $\mathcal{I}_{\text{test}} := \text{supp}_\kappa(\mathbf{X}_{\text{test}})$.

Determining the training set size: To explore the relation between recovery performance and and the size of training set $|\mathcal{T}|$, we train BDoG-Net on a ER graph ($N = 20, p = 0.3$) with different $|\mathcal{T}|$, and compute RE and ACC on the test set. We try $|\mathcal{T}| \in \{40\text{ k}, \ldots, 200\text{ k}\}$, with fixed minibatch size $P = 400$. For each $|\mathcal{T}|$, the experiment is repeated 10 times. As illustrated in Fig. 3 (top), we find that the mean RE initially decreases when $|\mathcal{T}|$ increases, and then it becomes stable when $|\mathcal{T}| \geq 160$k. This is consisent with Fig. 3 (bottom), which shows ACC reaches a maximum value when $|\mathcal{T}| \approx 160$k. At least in this setting, the gains are marginal beyond $|\mathcal{T}| \approx 80$k. In order to make sure that the BDoG-Net is trained well, we henceforth use $|\mathcal{T}| = 200$k as default for the rest experiments.

The minibatch size P also affects the training process. As discussed in [48], P naturally drives the successful recovery rate of the convex relaxation (8). Generally, a larger network (N) and/or denser sources (θ) require larger P. But when the total number of training samples $|\mathcal{T}|$ is fixed, a larger P might challenge the training process as the number Q of minibatches drops (increasing variability). Our results show that $P = 400$ is

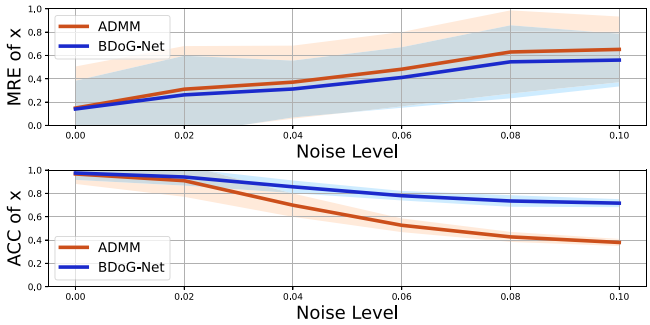


Fig. 4. Recovery performance of BDoG-Net vs. ADMM for different noise levels. (top) Test MRE of the recovered source signal $\hat{\mathbf{X}}$ estimated via iterative ADMM (red) and BDoG-Net (blue), as a function of η. (bottom)) Mean ACC of support estimation for both methods. The shaded region represents the estimated standard error, after averaging over 10 realizations. Performance degrades gracefully for both approaches, but BDoG-Net exhibits better robustness.

sufficient for BDoG-Net to train stably on graphs with $N = 20$ nodes, yielding satisfactory test performance; see Fig. 3.

Hyperparameter selection: The BDoG-Net architecture has two hyperparameters: the number of layers K and the number of columns in $\mathbf{M}$, i.e., d. To balance network complexity and recovery performance, we found that $K = 5$ is the optimal number of layers after experimenting with different values through a grid search. For d, a larger value implies $\tilde{\mathbf{g}}$ is constrained to a smaller subspace with more parameters to learn. Our numerical tests have shown that performance improves markedly when going from $d = 1$ to 2, with diminishing returns for $d \geq 2$ given the increase in complexity. As a result, we use $d = 2$ for all subsequent experiments.

B. Comparisons With the ADMM Algorithm

We compare BDoG-Net with the iterative ADMM algorithm in Section III-B. We examine their recovery performance and inference times during testing, studying the effect of different noise levels η, number of signals P, and number of nodes N.

Noise level η: To explore the robustness of BDoG-Net in the presence of additive noise corruptin $\mathbf{Y}$, we first train the model with noise-free data ($\eta = 0$) as outlined in Section V-A. Then we generate test sets with different noise level η and evaluate the recovery performance. We also run ADMM (Algorithm 1) on the same test sets and we compare the mean RE and ACC of both approaches (averaged over 10 independent realizations). Fig. 4 shows BDoG-Net achieves lower RE than ADMM as the noise level increases. We also find BDoG-Net still attains a high ACC even when $\eta > 0.06$. While the performance of both approaches degrades gracefully (see [48] for noise stability results of the convex relaxation we solve via ADMM), BDoG-Net exhibits higher tolerance to noise in this setting. In addition, the mean wall-clock inference time for BDoG-Net, averaged over 10 realizations, is around 0.009 s, uniformly across different noise levels. On the other hand, the mean elapsed time for ADMM, averaged over 10 realizations, is 1.990 s at $\eta = 0$ and 7.420 s at $\eta = 0.1$. We find ADMM requires more iterations to attain convergence when the noise added to the observations increases.

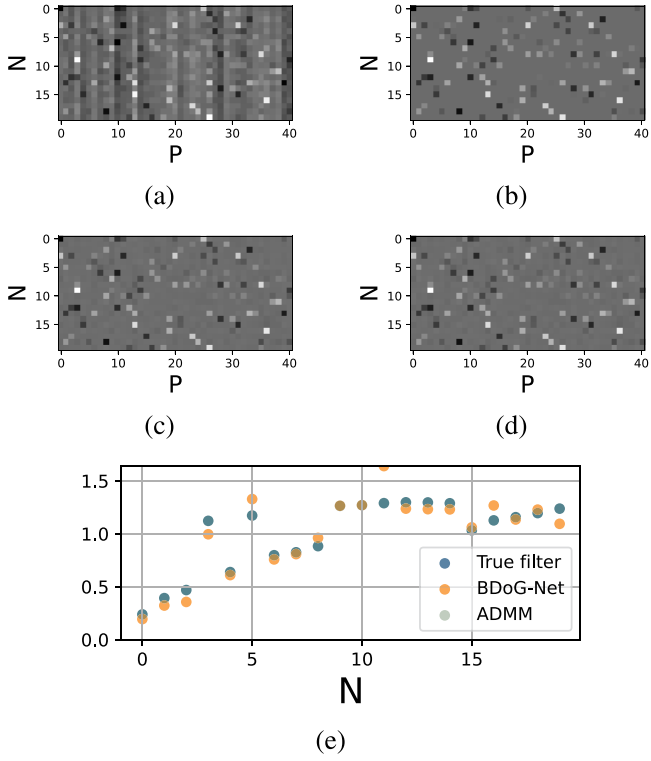


Fig. 5. A visual comparison of BDoG-Net and ADMM for a representative test realization in the blind deconvolution task. (a) Observations $\mathbf{Y}$; (b) ground-truth sources $\mathbf{X}_{\text{test}}$; (c) BDoG-Net source estimates; (d) ADMM source estimates. (e) Recovered inverse filter $\hat{\tilde{\mathbf{g}}}$: ground truth (blue), BDoG-Net (orange), ADMM (green). For (a)-(d), only $P_1 = 41$ columns are shown out of a total $P = 400$. BDoG-Net's ability to generate accurate predictions (approximating ADMM's model-based solution) is apparent.

For a qualitative assessment, estimation results for a representative test realization when $\eta = 0$ are shown in Fig. 5. In the interest of space, we depict the first $P_1 = 41$ (out of $P = 400$) columns of the observation matrix $\mathbf{Y}$, ground-truth sources $\mathbf{X}_{\text{test}}$, as well as BDoG-Net and ADMM source estimates $\hat{\mathbf{X}}$ in Fig. 5(a)–(d), respectively. In Fig. 5(e), the true inverse filter frequency response (blue), and the corresponding estimates obtained via BDoG-Net (orange) and ADMM (green) are presented. We find BDoG-Net recovers the input signals and inverse filter fairly accurately in the absence of noise, offering reasonably good approximations to the ADMM solutions.

Number of nodes N: We conducted some timing experiments for ER graphs with increasing number of nodes N. To this end, we trained BDoG-Net models for $N \in \{20, 40, \ldots, 100\}$, with fixed source sparsity level $\theta = 0.15$ and training set size $|\mathcal{T}| = 200$ k. For testing, we let $P_{\text{test}} = P = 400$ in all cases. While the recovery error attained by BDoG-Net naturally increases with graph size N (the problem becomes more challenging and we do not add more training data or increasing the test batch size P_{test}), the timing results in Table I – especially when compared to ADMM – are telling. Indeed, notice how BDoG-Net's mean inference time remains fairly invariant and around 10^{-2}s. On the other hand, ADMM scales worse with N and is typically a couple of orders of mangnitude slower when it comes to obtaining a solution. Granted, BDoG-Net's extra training time is

TABLE I
MEAN INFERENCE TIME (SEC.). COMPARISON BETWEEN BDOG-NET AND THE ADMM SOLVER FOR DIFFERENT N, WITH $P = 400$.

N	BDoG-Net	ADMM
20	0.95×10^{-2}	2.04
40	1.09×10^{-2}	5.70
60	1.27×10^{-2}	9.41
80	1.42×10^{-2}	12.29
100	1.64×10^{-2}	14.62

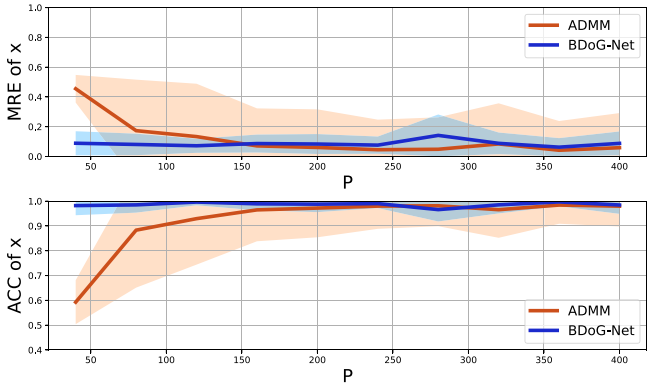


Fig. 6. Recovery performance of BDoG-Net vs. ADMM for different number of signals. (top) Test MRE of the recovered source signal $\hat{\mathbf{X}}$ estimated via iterative ADMM (red) and BDoG-Net (blue), as a function of P. (bottom) Mean ACC of support support estimation for both methods. The shaded region represents the estimated standard error, after averaging over 10 realizations. BDoG-Net can outperform ADDM, especially when the number of signals is smaller because it benefits from training, and exhibits reduced variability.

not accounted for here – we wish to highlight the computational efficiency of an unrolling during inference.

Observation size P: Finally, we compare the recovery performance of BDoG-Net and ADMM as a function of the number of observations P. The training minibatch size remains equal to P and other experiment parameters are kept fixed, e.g., $N = 20$, $S = \theta N = 3$. We tested $P \in \{40, 80, \ldots, 400\}$ and the results are shown in Fig. 6. For both the mean RE and the ACC, BDoG-Net outperforms ADMM when $P < 160$. When $P \geq 160$, the iterative ADMM achieves similar (or marginally better) mean RE and ACC than BDoG-Net, but the latter exhibits reduced variability across realizations. In terms of timing, the mean inference time for BDoG-Net is around 0.009s across the range of P values; while for ADMM it is 8.412s at $P = 40$, it decreases to 0.838s at $P = 160$, and then it increases to 1.851s at $P = 400$. When P is too small, it is harder to obtain a good solution via the relaxation (8), and ADMM may struggle to converge. However, BDoG-Net has stable recovery performance for different P because it benefits from an additional training phase.

C. Recovery Performance on Different Graphs

We also study the efficacy of BDoG-Net in identifying the sources across various graph ensembles, including random graphs ($N = 20$) such as ER, SBM, RG and BA, as well as the real karate club graph ($N = 34$), and the dolphins social network ($N = 62$). We use the settings and methods described in Section V-A. In Table II we report the mean RE of $\hat{\mathbf{X}}$ and $\hat{\tilde{\mathbf{g}}}$, as well as the support recovery ACC. Despite the relatively high RE (> 0.3) for the source signal $\hat{\mathbf{X}}$ or the inverse filter $\hat{\tilde{\mathbf{g}}}$ for some of the graphs (especially the more structured and larger ones), the support estimate ACC remains high (> 0.8). The result has a twofold interpretation. First, BDoG-Net's subpar RE performance on certain graphs may be due to $\tilde{\mathbf{g}}$ having a significant component that is orthogonal to span$(\mathbf{1}_N)$. Indeed, results in [48] show that $\|\mathbf{P}_1^{\perp}\tilde{\mathbf{g}}\|_2$ (where $\mathbf{P}_1^{\perp} := \mathbf{I}_N - \frac{1}{N}\mathbf{1}\mathbf{1}^{\top}$ is the projector onto span$^{\perp}(\mathbf{1}_N)$) can be viewed as a condition number of the problem (8)). Hence, the higher variability (the eigenvalue distribution of different graph types affects the Vandermonde matrix $\mathbf{\Psi}_L$) of randomly generated filters $\tilde{\mathbf{g}}$ for some graphs, may increase the problem difficulty that manifests through higher REs. But BDoG-Net is designed to optimize a more flexible version of (8), with a learnable constraint. Using $\mathbf{M}$ instead of $\mathbf{1}_N$ affects the problem's conditioning, (we conjecture) likely contributing to keep ACC rates at satisfactory levels. A more in-depth analysis is certainly of interest, but beyond the scope of this paper.

TABLE II
BDOG-NET RECOVERY PERFORMANCE FOR DIFFERENT GRAPHS

Graph	N	$\|\mathbf{P}_1^{\perp}\tilde{\mathbf{g}}\|_2$	MRE of $\hat{\mathbf{X}}$	MRE of $\hat{\tilde{\mathbf{g}}}$	ACC
ER	20	6.885	0.149	0.164	0.953
SBM	20	7.806	0.219	0.215	0.914
RG	20	7.591	0.383	0.377	0.869
BA	20	14.547	0.579	0.537	0.772
karate	34	23.996	0.454	0.452	0.958
dolphins	62	39.254	0.719	0.578	0.841

D. Comparison With a GNN Approach

Here we explore the feasibility of using BDoG-Net for community detection in an SBM with N_c communities. Our goal is not to demonstrate state-of-the-art performance in this well-investigated task, but rather to find an application domain where comparison with the GNN models in [10] is feasible. So far, the support of each $\mathbf{x}_i$ was specified via i.i.d Bernoulli random variables, in community detection all of the sources (the S non-zero entries of $\mathbf{x}_i$) are drawn randomly within a single subset of indices from $\{1, \ldots, N\}$, representing the members of the community to be identified. So we cast this simple version of community detection as *structured* source localization as in [10, Sec. V-A], where active sources can be located in only one of the N_c communities.

To illustrate this further, we consider an SBM graph with $N = 20$ nodes and $N_c = 3$ communities. Each of the input signals $\mathbf{x}_i$ are generated as follows: (i) select a community by drawing an integer c_i uniformly at random from $\{1, \ldots, N_c\}$; (ii) randomly select $S = \theta N = 3$ nodes among the members of the selected community c_i – the active sources which we encode in the support vector $\boldsymbol{\omega}_i \in \{0, 1\}^N$; (ii) compute $\mathbf{x}_i = \mathbf{r}_i \circ \boldsymbol{\omega}_i$, where $\mathbf{r}_i \sim \text{Normal}(\mathbf{0}_N, \mathbf{I}_N)$. All in all, $\mathbf{x}_i$ are samples from a variation of a Bernoulli-Gaussian model, whose support is constrained to a randomly-chosen subset of nodes (the members of the chosen community c_i).

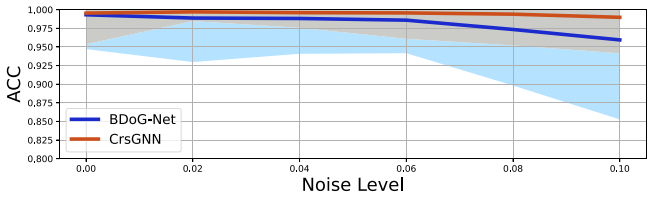


Fig. 7. Recovery performance of BDoG-Net vs. GNN for different noise levels. Mean ACC of estimated communities estimated via GNN [10] (red) and BDoG-Net (blue) as a function of η. The shaded region represents the estimated standard deviation, after averaging over 10 realizations. BDoG-Net is trained on a harder (higher-resolution) source localization task, without community label supervision – which the GNN model in [10] struggles to solve. Still, BDoG-Net attains competitive community detection ACC rates.

Experimental details: For training, we generate $|\mathcal{T}| = 200$k input signals $\mathbf{X} \in \mathbf{R}^{N\times|\mathcal{T}|}$ and source community labels $\mathbf{K} \in \{0,1\}^{N_c\times|\mathcal{T}|}$, via one-hot encoding of each c_i. We train BDoG-Net as described in V-A. For the baseline selection GNN model [10], we generate $|\mathcal{T}|$ graph filters $\{\mathbf{h}_i\}_{i=1}^{|\mathcal{T}|}$ and assign them one to one to the input signal $\{\mathbf{x}_i\}_{i=1}^{|\mathcal{T}|}$ to generate the observations $\{\mathbf{y}_i\}_{i=1}^{|\mathcal{T}|}$, mimicking the setting in [10, Sec. V-A]. The GNN is trained with supervised data $\{c_i, \mathbf{y}_i\}_{i=1}^{|\mathcal{T}|}$. For testing, we generate one test set of $P_{\text{test}} = P = 400$ samples, i.e., $\mathbf{X}_{\text{test}} \in \mathbf{R}^{N\times P}$, a graph filer $\mathbf{h}_{\text{test}}$, and observations $\mathbf{Y}_{\text{test}} = \mathbf{V}\text{diag}(\mathbf{\Psi}_L\mathbf{h}_{\text{test}})\mathbf{V}^\top\mathbf{X}_{\text{test}} + \eta\mathbf{N}$, for different noise levels $\eta \in \{0, 0.02, \ldots, 0.1\}$. Notice that our goal is to estimate the source community as in [10, Sec. V-A], not the source nodes. But since BDoG-Net recovers the input signal $\hat{\mathbf{X}}$, we apply a strategy that considers the community, where the entry with highest magnitude of the estimated source $\hat{\mathbf{x}}_i$ resides, as BDoG-Net's community estimate.

Results and discussion: We compare the mean community detection ACC for both BDoG-Net and the GNN; the results are reported in Fig. 7. Apparently, the GNN attains very high ACC rates for the entire noise level spectrum. Remarkably, BDoG-Net achieves competitive performance that is robust across noise levels. Notice that BDoG-Net is trained on a harder source localization task (node identification) and without community label supervision, which the GNN model in [10] struggles to solve. To carry out the comparison, we settled on community detection, a lower resolution source localization variant that favors the GNN approach.

E. Real-Data Source Localization Experiment

In this last section, we test BDoG-Net on a simplified source localization problem we set up using the Digg 2009 data set [14]. For broader context, it is prudent to mention that highly-performant and scalable DL approaches are available for network source localization based on more general (e.g., dynamic, nonlinear, epidemic) forward models than the one in Section II-A; see e.g., [7], [16], [17], [24], [41], [45] and [20], [38] for tutorial treatments. This growing literature is valuable and relevant, but in this paper we tackle a different blind deconvolution problem on graphs, which can be linked to an admittedly simplistic rendition of source localization. Building on the solid theoretical properties of the convex relaxation for this inverse problem (Section III-A), our goal has been to leverage GSP insights for the principled design of BDoG-Net used in a new supervised learning setting (cf. Fig. 1). We are not after a state-of-the-art source localization approach, but it is still worthwhile to illustrate (and project) BDoG-Net's practical value.

Data preprocessing: The Digg 2009 data set consists of vote records and a social network of users. The vote records contain 3 M votes from 139 k users on 3553 popular stories, along with the voting timestamps. The social network of users includes 1.7 M friendship links between 71 k unique users. To assess BDoG-Net's source localization capabilities in a real-world setting, we treat the voting history of each story as a signal diffused over the social network graph G . However, processing a graph with 139k or 71k users is infeasible for BDoG-Net as there are only at most 3553 training and testing samples. To ensure that the graph is connected and not too large so that it can be processed for effective learning, we randomly selected $N = 20$ users with the following criteria: (i) they must have cast at least 100 votes; (ii) all of their friendship links are mutual; and (iii) their friendship subgraph is connected. Following these guidelines, we randomly sampled 5 subgraphs with $N = 20$ nodes from Digg, with mean degree 16.0 ± 0.85 and $P_{\text{story}} = 3299.0 \pm 35.8$ stories voted by each user. To generate the graph signals, we considered each story as a sample, using the 10% earliest votes as the binary sources $\mathbf{x}_p$ and all votes as the observations $\mathbf{y}_p$. Because the sources $\mathbf{X} \in \{0,1\}^{N\times P_{\text{story}}}$ are a subset of the observations $\mathbf{Y} \in \{0,1\}^{N\times P_{\text{story}}}$, we only consider identifying the sources from the support set of the observations $\text{supp}(\mathbf{Y})$.

Experimental details: The challenge of localizing sources using BDoG-Net on the sampled Digg data is twofold. First, the sample size is limited, i.e., $|\mathcal{T}| \approx 3.5$k, challenging effective training; secondly, both the sources $\mathbf{X}$ and observations $\mathbf{Y}$ are binary. However, if we focus on recovering $\text{supp}(\mathbf{X})$, BDoG-Net may still yield useful results given the binary observations $\mathbf{Y}$. To address these challenges, we adopted several strategies.

First, we found that selecting a proper mini-batch size P, along with an appropriate observation size P_{test}, was crucial for both training and testing. After several attempts, we determined that while a larger training batch size P would improve the performance of BDoG-Net, it would also result in a smaller training set size $|\mathcal{T}|$, as we wanted to use $P_{\text{test}} = P$ for consistency. Therefore, we opted for $P = 400$, which set $P_{\text{test}} = 400$ and $|\mathcal{T}| = P_{\text{story}} - P_{\text{test}}$, so $|\mathcal{T}| \approx 2.9$k on average.

To address the second challenge of binary inputs, we aim to find some calibration operator $\Phi_c : \mathbf{Y} \mapsto \mathbf{Y}'$ that maps the binary observations $\mathbf{Y}$ to real-valued graph signals $\mathbf{Y}'$, which can be viewed as the result of diffusing the sparse binary sources $\mathbf{X}$. Then we can apply BDoG-Net to predict $\hat{\mathbf{X}} = \Phi(\mathbf{Y}'; \mathbf{\Theta})$, as depicted in Fig. 8 (bottom). We were inspired by [41], where an invertible graph diffusion network (IVGD) is proposed based on the invertible residual network (i-ResNet) [3]. We adopt an i-ResNet structure to construct the *invertible* mapping Φ_c, which can be approximated via fix-point iterations from its inverse $\Phi_c^{-1} : \mathbf{Y}' \mapsto \mathbf{Y}$; see Algorithm 2. Specifically, we consider $\Phi_c^{-1}(\mathbf{Y}'; \mathbf{\Theta}_c) = \frac{1}{2}(\mathbf{Y}' + \text{MLP}(\mathbf{Y}'))$, where $\text{MLP}(\cdot; \mathbf{\Theta}_c)$ is

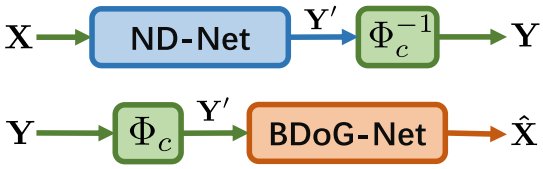


Fig. 8. (top) Pre-training architecture and (bottom) calibrated BDoG-Net.

Algorithm 2: Calibration Mapping Φ_c via Inversion.

Require Φ_c^{-1} , $\mathbf{Y}$, $K_{\max}$ (number of iterations).
Initialize $\mathbf{Y}'[0] = \mathbf{Y}$.
for $k = 1, 2, \ldots, K_{\max}$ **do**
  $\mathbf{Y}'[k] \leftarrow 2\mathbf{Y} - \Phi_c^{-1}(\mathbf{Y}'[k-1])$.
end
Output $\mathbf{Y}'[K_{\max}] = \Phi_c(\mathbf{Y})$.

a three-layer, 1000-hidden unit multi-layer perceptron (MLP) with learnable parameters $\mathbf{\Theta}_c$. We train Φ_c^{-1} along with a network diffusion NN (ND-Net), as shown in Fig. 8 (top). Details of this pre-training process are presented in Appendix E.

BDoG-Net is trained using $\mathcal{T} = \{\mathbf{X}, \Phi_c^{-1}(\mathbf{Y})\}$; see Fig. 8 (bottom). Because the complement $\mathcal{I}_Y^c$ of $\mathcal{I}_Y := \text{supp}(\mathbf{Y})$ cannot include the sources, we want to penalize $\mathcal{I}_Y$ more. To this end, we use the weighted loss function

$$L_\phi(\mathbf{\Theta}) = \sum_{q=1}^{Q} \min\left(L^+_{q,\phi}(\mathbf{\Theta}), L^-_{q,\phi}(\mathbf{\Theta})\right),$$

where $L^\pm_{q,\phi}(\mathbf{\Theta}) := \frac{\|[\Phi(\mathbf{Y}_q;\mathbf{\Theta})\pm\mathbf{X}_q]_{\mathcal{I}_Y}\|_F + \phi\|[\Phi(\mathbf{Y}_q;\mathbf{\Theta})\pm\mathbf{X}_q]_{\mathcal{I}_Y^c}\|_F}{\|\mathbf{X}_q\|_F}$ and $\phi = 0.01$. With these adjustments, training proceeds as described in Section V-A.

For the IVGD baseline, we run the algorithm provided in [41]. The number of hidden units is chosen to be 50, consistent with the experimental setting described in [41].

We also implement two source localization baselines that can be run on the Digg data set, IVGD and SL-VAE with settings consistent with the works [41] and [24], respectively.

Results and discussion: To evaluate the source localization performance, we compute the ROC curve and AUC of $\{\hat{\mathbf{X}}_{\mathcal{I}_Y}, [\mathbf{X}_{\text{test}}]_{\mathcal{I}_Y}\}$, where $\hat{\mathbf{X}}$ are the predicted sources, $\mathbf{X}_{\text{test}}$ is the ground-truth of the test set and $\mathcal{I}_Y = \text{supp}(\mathbf{Y}_{\text{test}})$. The experiment is repeated twice for each of the 5 sampled subgraphs, each time the training and test sets are randomly split from all the samples $\{\mathbf{X}, \mathbf{Y}\}$. An ROC generated for one representative sampled subgraph is depicted in Fig. 9. The mean AUC averaged over 10 realizations is 0.57, 0.53, and 0.52 for BDoG-Net, SL-VAE, and IVGD, respectively. Besides, the execution time per realization, including training and testing in Google Colab without GPU, is about 20 minutes, 5 hours, and 1 h for BDoG-Net, SL-VAE, and IVGD, respectively. We find that although none of the methods perform admirably in this hard problem, BDoG-Net is more efficient and marginally better at learning representations that are predictive of the sources, at least in the test case.

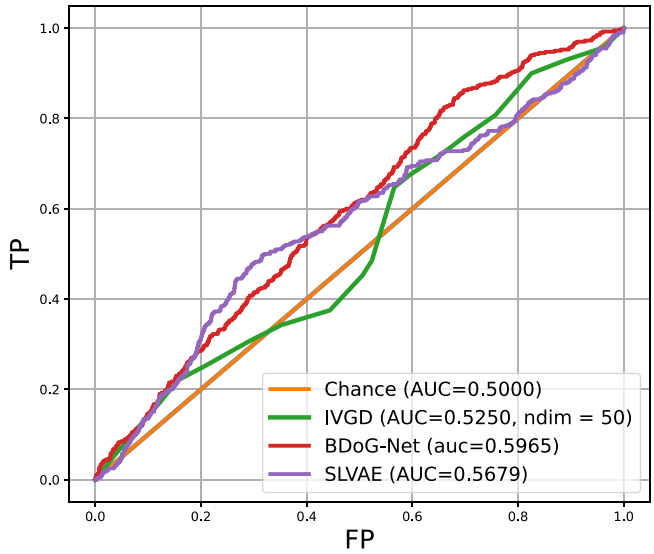


Fig. 9. The ROC curve of for one representative sampled subgraph. The mean AUC over 10 different training/testing set realizations are 0.57, 0.53 and 0.52 for BDoG-Net, SL-VAE and IVGD, respectively.

VI. Conclusions, Limitations, and Future Work

We developed BDoG-Net, a novel DL approach for blind deconvolution of graph signals. The unrolled architecture fruitfully leverages inductive biases stemming from model-based ADMM iterations we also developed, is parameter efficient, fairly robust to noise, and offers controllable complexity after training. Our experimental results with simulated and real network data demonstrate that BDoG-Net exhibits performance on par with the iterative ADMM baseline it is trained to approximate, while attaining order-of-magnitude speedups to generate source location predictions during inference for simple problem instances. Admittedly, there is still work to be done to arrive at a truly scalable solution that is compatible with large-scale problems and also expressive enough to capture the intricacies of e.g., dynamic and non-linear diffusion models for state-of-the-art source localization over networks. Importantly, BDoG-Net opens the door for further architectural refinements by leveraging advances in optimization, DL, and machine learning on graphs, which we intend to pursue as future work. Exciting ideas include designing a model that fully operates in the vertex domain as well as expanding our performance evaluation protocol to study generalization and transfer to larger graphs, possibly establishing stability and transferability properties of the resulting unrolled (G)NNs. We also look forward to exploring additional application domains in network neuroscience, seismology, and epidemiology.

Appendix

A. Derivation of the ADMM Updates

The ADMM algorithm can be viewed as blending block coordinate-descent (BCD) updates for the primal variables $\{\tilde{\mathbf{g}}[k], \mathbf{x}[k]\}$, with dual gradient-ascent iterations for the

Lagrange multipliers $\{\boldsymbol{\lambda}[k], \boldsymbol{\mu}[k]\}$; see e.g., [4], [5], [12]. Accordingly, when used to solve problem (9) it entails the following three steps per iteration $k = 0, 1, \ldots$:

[S1] Filter updates:

$$\tilde{\mathbf{g}}[k+1] = \arg\min_{\tilde{\mathbf{g}}} \mathcal{L}_\rho(\mathbf{x}[k], \tilde{\mathbf{g}}, \boldsymbol{\lambda}[k], \mu[k]). \tag{19}$$

[S2] Sources' updates:

$$\mathbf{x}[k+1] = \arg\min_{\mathbf{x}} \mathcal{L}_\rho(\mathbf{x}, \tilde{\mathbf{g}}[k+1], \boldsymbol{\lambda}[k], \mu[k]). \tag{20}$$

[S3] Lagrange multiplier updates:

$$\boldsymbol{\lambda}[k+1] = \boldsymbol{\lambda}[k] + \rho_\lambda(\mathbf{Z}\tilde{\mathbf{g}}[k+1] - \mathbf{x}[k+1]), \tag{21}$$

$$\mu[k+1] = \mu[k] + \rho_\mu(\mathbf{1}_N^\top \tilde{\mathbf{g}}[k+1] - c). \tag{22}$$

The Lagrange multiplier updates in [S3] coincide with (13)–(14). These correspond to gradient-ascent iterations, since the gradients of the dual function are equal to the respective constraint violations in (9) [5].

What remains is to show that [S1]-[S2] can be simplified to (11)–(12). Starting with [S1], note that the augmented Lagrangian is a strictly-convex, smooth quadratic function with respect to $\tilde{\mathbf{g}}$. The gradient of (10) is

$$\nabla_{\tilde{\mathbf{g}}} \mathcal{L}_\rho(\mathbf{x}, \tilde{\mathbf{g}}, \boldsymbol{\lambda}, \mu) = \rho_\lambda \mathbf{Z}^\top(\mathbf{Z}\tilde{\mathbf{g}} - \mathbf{x} + \boldsymbol{\lambda}/\rho_\lambda) + \rho_\mu \mathbf{1}_N(\mathbf{1}_N^\top \tilde{\mathbf{g}} - c + \mu/\rho_\mu).$$

The unique minimizer of (19) satisfies the first-order optimality condition $\nabla_{\tilde{\mathbf{g}}} \mathcal{L}_\rho(\mathbf{x}[k], \tilde{\mathbf{g}}, \boldsymbol{\lambda}[k], \mu[k]) = \mathbf{0}_N$. Solving the linear system of equations immediately yields (11), where $\boldsymbol{\Gamma} := \rho_\lambda \mathbf{Z}^\top \mathbf{Z} + \rho_\mu \mathbf{1}_N \mathbf{1}_N^\top$.

Shifting our focus to [S2], one recognizes (20) as the proximal operator of the function $\rho_\lambda^{-1}\|\mathbf{x}\|_1$ evaluated at $\mathbf{Z}\tilde{\mathbf{g}}[k+1] + \boldsymbol{\lambda}[k]/\rho_\lambda$. Said proximal operator is a soft-thresholding operator; e.g. [26], and the update rule (12) follows.

To derive the iterations in Fig. 2, which tabulates the ADMM algorithm for the modified formulation with the general constraint $\mathbf{M}^\top \tilde{\mathbf{g}} = \mathbf{m}$, one simply mimics [S1]-[S3] to instead minimize the updated augmented Lagrangian

$$\mathcal{L}_\rho(\mathbf{x}, \tilde{\mathbf{g}}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \|\mathbf{x}\|_1 + \frac{\rho_\lambda}{2}\|\mathbf{Z}\tilde{\mathbf{g}} - \mathbf{x} + \boldsymbol{\lambda}/\rho_\lambda\|_2^2 + \frac{\rho_\mu}{2}\|\mathbf{M}^\top \tilde{\mathbf{g}} - \mathbf{m} + \boldsymbol{\mu}/\rho_\mu\|_2^2.$$

B. Diagonal Structure of $\mathbf{Z}^\top\mathbf{Z}$

Recall $\mathbf{Z} := \mathbf{Y}^\top\mathbf{V} \odot \mathbf{V} \in \mathbb{R}^{NP\times N}$, where $\mathbf{Y} \in \mathbb{R}^{N\times P}$ and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_N] \in \mathbb{R}^{N\times N}$. Letting $\tilde{\mathbf{Y}} = \mathbf{V}^\top\mathbf{Y} \in \mathbb{R}^{N\times P}$, we have

$$\mathbf{Z} = \tilde{\mathbf{Y}}^\top \odot \mathbf{V} = \begin{bmatrix} [\tilde{\mathbf{Y}}^\top]_{11}\mathbf{v}_1 & [\tilde{\mathbf{Y}}^\top]_{12}\mathbf{v}_2 & \ldots & [\tilde{\mathbf{Y}}^\top]_{1N}\mathbf{v}_N \\ [\tilde{\mathbf{Y}}^\top]_{21}\mathbf{v}_1 & [\tilde{\mathbf{Y}}^\top]_{22}\mathbf{v}_2 & \ldots & [\tilde{\mathbf{Y}}^\top]_{2N}\mathbf{v}_N \\ & \vdots & & \\ [\tilde{\mathbf{Y}}^\top]_{P1}\mathbf{v}_1 & [\tilde{\mathbf{Y}}^\top]_{P2}\mathbf{v}_2 & \ldots & [\tilde{\mathbf{Y}}^\top]_{PN}\mathbf{v}_N \end{bmatrix}$$

from where it follows that

$$[\mathbf{Z}^\top\mathbf{Z}]_{ij} = \begin{bmatrix} [\tilde{\mathbf{Y}}^\top]_{1i}\mathbf{v}_i \\ [\tilde{\mathbf{Y}}^\top]_{2i}\mathbf{v}_i \\ \vdots \\ [\tilde{\mathbf{Y}}^\top]_{Pi}\mathbf{v}_i \end{bmatrix}^\top \cdot \begin{bmatrix} [\tilde{\mathbf{Y}}^\top]_{1j}\mathbf{v}_j \\ [\tilde{\mathbf{Y}}^\top]_{2j}\mathbf{v}_j \\ \vdots \\ [\tilde{\mathbf{Y}}^\top]_{Pj}\mathbf{v}_j \end{bmatrix} = \sum_{p=1}^{P} [\tilde{\mathbf{Y}}^\top]_{pi}[\tilde{\mathbf{Y}}^\top]_{pj}\mathbf{v}_i^\top\mathbf{v}_j = [\tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^\top]_{ij}\delta_{ij}, \tag{23}$$

where $\delta_{ij} = \mathbb{I}\{i = j\}$ is the Kronecker delta. Notice that (23) follows since the graph-shift operator eigenvectors are orthogonal. All in all, we have shown that $\mathbf{Z}^\top\mathbf{Z}$ is an $N \times N$ diagonal matrix with diagonal elements $\{\|\mathbf{v}_i^\top\mathbf{Y}\|_2^2\}_{i=1}^N$. We can thus write $\mathbf{Z}^\top\mathbf{Z} = \text{diag}(\|\mathbf{v}_1^\top\mathbf{Y}\|_2^2, \ldots, \|\mathbf{v}_N^\top\mathbf{Y}\|_2^2)$.

C. Inverting $\mathbf{Z}^\top\mathbf{Z} + \rho\mathbf{1}_N\mathbf{1}_N^\top$ Via the Matrix Inversion Lemma

The Sherman–Morrison–Woodbury formula states

$$(\mathbf{U} + \mathbf{BCD})^{-1} = \mathbf{U}^{-1} - \mathbf{U}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{DU}^{-1}\mathbf{B})^{-1}\mathbf{DU}^{-1}. \tag{24}$$

To apply this identity to invert $\boldsymbol{\Gamma} \propto \mathbf{ZZ}^\top + \rho\mathbf{1}_N\mathbf{1}_N^\top$, define $\mathbf{z} := [\|\mathbf{v}_1^\top\mathbf{Y}\|_2^2, \ldots, \|\mathbf{v}_N^\top\mathbf{Y}\|_2^2]^\top \in \mathbb{R}^N$, and then let $\mathbf{U} := \mathbf{Z}^\top\mathbf{Z} = \text{diag}(\mathbf{z})$. Comparing $\mathbf{Z}^\top\mathbf{Z} + \rho\mathbf{1}_N\mathbf{1}_N^\top$ and (24), we let $\mathbf{B} := \mathbf{1}_N$, $\mathbf{D} = \mathbf{1}_N^\top$, and $\mathbf{C} = \rho$. The matrix sum that is to be inverted in the right-hand-side of (24) is a scalar, namely $\mathbf{C}^{-1} + \mathbf{DU}^{-1}\mathbf{B} = \rho^{-1} + \mathbf{1}_N^\top(\mathbf{z}^{-1} \circ \mathbf{1}_N) := \zeta$, where with an abuse of notation we let $\mathbf{z}^{-1} := [\|\mathbf{v}_1^\top\mathbf{Y}\|_2^{-2}, \ldots, 1/\|\mathbf{v}_N^\top\mathbf{Y}\|_2^{-2}]^\top \in \mathbb{R}^N$ be the (entry-wise) vector reciprocal of $\mathbf{z}$. Applying (24), we obtain

$$\begin{aligned}(\mathbf{Z}^\top\mathbf{Z} + \rho\mathbf{1}_N\mathbf{1}_N^\top)^{-1} &= (\mathbf{Z}^\top\mathbf{Z})^{-1} - \frac{(\mathbf{Z}^\top\mathbf{Z})^{-1}\mathbf{1}_N\mathbf{1}_N^\top(\mathbf{Z}^\top\mathbf{Z})^{-1}}{\zeta} \\ &= \text{diag}(\mathbf{z}^{-1}) - \frac{\text{diag}(\mathbf{z}^{-1})\mathbf{1}_N\mathbf{1}_N^\top\text{diag}(\mathbf{z}^{-1})}{\zeta} \\ &= \text{diag}(\mathbf{z}^{-1}) - \frac{\mathbf{z}^{-1}(\mathbf{z}^{-1})^\top}{\zeta}. \end{aligned} \tag{25}$$

While (25) is simple when the correction to $\mathbf{Z}^\top\mathbf{Z}$ is a scaled version of $\mathbf{1}_N\mathbf{1}_N^\top$, a general rank-one correction of the form $\rho\mathbf{mm}^\top$ is almost identical. Indeed, one just needs to re-evaluate $\zeta = \rho^{-1} + \mathbf{m}^\top(\mathbf{z}^{-1} \circ \mathbf{m})$ and the right-most summand in the third line of (25) becomes $\zeta^{-1}(\mathbf{m} \circ \mathbf{z}^{-1})(\mathbf{m} \circ \mathbf{z}^{-1})^\top$.

The computational complexity of (25) includes: i) computing $\mathbf{z}^{-1}$, the entrywise reciprocal of $\mathbf{z}$, $O(N)$ assuming $\mathbf{z}$ is given; ii) computing $\zeta = \rho^{-1} + \mathbf{m}^\top(\mathbf{z}^{-1} \circ \mathbf{m})$, or the sum $\mathbf{m}^\top(\mathbf{z}^{-1} \circ \mathbf{m}) = \sum_i m_i^2/z_i$, $O(N)$; iii) computing the outer product $(\mathbf{m} \circ \mathbf{z}^{-1})(\mathbf{m} \circ \mathbf{z}^{-1})^\top$, $O(N^2)$; iv) normalizing by ζ, $O(1)$; and updating the diagonal entries by adding $\text{diag}(\mathbf{z}^{-1})$, an extra $O(N)$. As a result, the overall computational complexity of inverting $\mathbf{Z}^\top\mathbf{Z} + \rho\mathbf{mm}^\top$ is $O(N^2)$.

There are no order-wise savings when $\mathbf{m} = \mathbf{1}_N$, which is what we require to invert $\boldsymbol{\Gamma} := \rho_\lambda\mathbf{Z}^\top\mathbf{Z} + \rho_\mu\mathbf{1}_N\mathbf{1}_N^\top$ in the ADMM update (11), or, $\boldsymbol{\Gamma}^{(k)} = \mathbf{Z}^\top\mathbf{Z} + \rho_2^{(k)}\mathbf{1}_N\mathbf{1}_N^\top$ in BDoG-Net's filter sub-layer when the constraint parameters $\mathbf{M}^{(k)}$ and $\mathbf{m}^{(k)}$ are not learnt [49]. When $d = 1$, the formula (25) can also be applied to the refined filter sub-layer (11); see Appendix D for the general

case. To appreciate the overall savings, recall that the computational complexity of inverting a general $N \times N$ matrix is $O(N^\omega)$, where $\omega \in \{2.376, 2.807, 3\}$ for three different kind of algorithms; namely, the Coppersmith–Winograd algorithm, the Strassen algorithm, and Gauss–Jordan elimination, respectively.

D. Inverting $\mathbf{Z}^\top\mathbf{Z} + \rho\mathbf{M}\mathbf{M}^\top$

Let $\mathbf{U} = \mathbf{Z}^\top\mathbf{Z} = \text{diag}(\mathbf{z})$, $\mathbf{C} = \rho\mathbf{I}_d$, and $\mathbf{B} = \mathbf{D}^\top = \mathbf{M} \in \mathbb{R}^{N\times d}$. From the matrix inversion lemma (24) we have,

$$(\mathbf{Z}^\top\mathbf{Z} + \rho\mathbf{M}\mathbf{M}^\top)^{-1} = \text{diag}(\mathbf{z}^{-1}) - \text{diag}(\mathbf{z}^{-1})\mathbf{M}\bar{\mathbf{M}}^{-1}\mathbf{M}^\top\text{diag}(\mathbf{z}^{-1}),$$

where $\bar{\mathbf{M}} = \rho^{-1}\mathbf{I}_d + \mathbf{M}^\top\text{diag}(\mathbf{z}^{-1})\mathbf{M} \in \mathbb{R}^{d\times d}$.

E. Pre-Training Process

To learn the inverse calibration mapping $\Phi_c^{-1} : \mathbf{Y}' \mapsto \mathbf{Y}$ directly, we would need a training set $\{\mathbf{Y}, \mathbf{Y}'\}$. We model the latent $\mathbf{Y}'$ as the output of network diffusion process driven by sources $\mathbf{X}$, with some unknown graph filter $\mathbf{H}' = \mathbf{V}\text{diag}(\tilde{\mathbf{h}}')\mathbf{V}^\top$; i.e., $\mathbf{Y}' = \mathbf{H}'\mathbf{X}$. To predict and generate $\mathbf{Y}'$, we design a learnable parametric function $\hat{\mathbf{Y}}' = \Upsilon(\mathbf{X}; \mathbf{\Theta}_\Upsilon)$ via unrolling, similar to BDoG-Net. Estimating both $\mathbf{Y}'$ and $\tilde{\mathbf{h}}'$ from the input signal $\mathbf{X}$ is an ill-posed problem, hence we assume the diffused signal $\mathbf{Y}'$ is close to $\mathbf{X}$. Then we consider the following constrained optimization problem to predict (and thus generate) network diffusion outputs,

$$\min_{\tilde{\mathbf{h}}, \mathbf{Y}'} \|\mathbf{Y}' - \mathbf{V}\text{diag}(\tilde{\mathbf{h}})\mathbf{V}^\top\mathbf{X}\|_\text{F}^2 + \rho_1\|\mathbf{Y}' - \mathbf{X}\|_\text{F}^2 \quad \text{s. to } \bar{\mathbf{M}}^\top\tilde{\mathbf{h}} = \bar{\mathbf{m}}, \tag{26}$$

where $\bar{\mathbf{M}} \in \mathbb{R}^{N\times d}, \bar{\mathbf{m}} \in \mathbb{R}^d$ will be learned.

Similar to BDoG-Net, we derive ADMM iterations to solve (26) and use the unrolling principle to construct the sub-layers of the network diffusion NN (ND-Net) $\hat{\mathbf{Y}}' = \Upsilon(\mathbf{X}; \mathbf{\Theta}_d)$; details are omitted to avoid repetition. We also consider $K = 5$ layers and generate predictions as $\hat{\mathbf{Y}}' = (\mathbf{X}^\top\mathbf{V} \odot \mathbf{V})\tilde{\mathbf{h}}[K] = \Upsilon(\mathbf{X}; \mathbf{\Theta}_d)$. Given this architecture, we compose ND-Net Υ and the inverse calibration mapping Φ_c^{-1} to obtain the pre-training model $\mathbf{Y} = \Phi_c^{-1}(\Upsilon(\mathbf{X}; \mathbf{\Theta}_\Upsilon); \mathbf{\Theta}_c)$; see Fig. 8 (top). During the pre-training process, the learnable parameters $\{\mathbf{\Theta}_c, \mathbf{\Theta}_\Upsilon\}$ are learned from binary data $\mathcal{T} = \{\mathbf{Y}_q, \mathbf{X}_q\}_q^Q$.

REFERENCES

[1] A. Ahmed, B. Recht, and J. Romberg, "Blind deconvolution using convex programming," *IEEE Trans. Inf. Theory*, vol. 60, no. 3, pp. 1711–1732, Mar. 2014.

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

[3] J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen, "Invertible residual networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 573–582.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, 2nd ed. Nashua, NH, USA: Athena-Scientific, 1999.

[5] S. Boyd et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[6] S. Chen, Y. C. Eldar, and L. Zhao, "Graph unrolling networks: Interpretable neural networks for graph signal denoising," *IEEE Trans. Signal Process.*, vol. 69, pp. 3699–3713, 2021.

[7] L. Cheng, P. Zhu, K. Tang, C. Gao, and Z. Wang, "GIN-SD: Source detection in graphs with incomplete nodes via positional encoding and attentive fusion," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 55–63.

[8] M. H. DeGroot, "Reaching a consensus," *J. Amer. Stat. Assoc*, vol. 69, pp. 118–121, 1974.

[9] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 128–138, Nov. 2020.

[10] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 1034–1049, Feb. 2019.

[11] A. Gavili and X.-P. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Trans. Signal Process.*, vol. 65, no. 23, pp. 6303–6318, Dec. 2017.

[12] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas, and H. Zhu, "Decentralized learning for wireless communications and networking," in *Splitting Methods in Communication, Imaging, Science, and Engineering*, R. Glowinski, S. J. Osher, and W. Yin, Eds. Berlin, Germany: Springer, 2016, pp. 461–497.

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

[14] T. Hogg and K. Lerman, "Social dynamics of digg," *EPJ Data Sci.*, vol. 1, pp. 1–26, 2012.

[15] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[16] D. Hou, C. Gao, Z. Wang, X. Li, and X. Li, "Random full-order-coverage based rapid source localization with limited observations for large-scale networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 5, pp. 4213–4226, Sep./Oct. 2024.

[17] D. Hou, C. Gao, Z. Wang, and X. Li, "FGSSI: A feature-enhanced framework with transferability for sequential source identification," *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 4, pp. 4300–4314, Jul./Aug. 2025.

[18] C. Hu, X. Hua, J. Ying, P. M. Thompson, G. E. Fakhri, and Q. Li, "Localizing sources of brain disease progression with network diffusion model," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 7, pp. 1214–1225, Oct. 2016.

[19] E. Isufi, F. Gama, D. I. Shuman, and S. Segarra, "Graph filters for signal processing and machine learning on graphs," *IEEE Trans. Signal Process.*, vol. 72, pp. 4745–4781, 2024.

[20] J. Jiang, S. Wen, S. Yu, Y. Xiang, and W. Zhou, "Identifying propagation sources in networks: State-of-the-art and comparative studies," *IEEE Commun. Surveys Tut.*, vol. 19, no. 1, pp. 465–481, Firstquarter 2017.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[22] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding blind deconvolution algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 12, pp. 2354–2367, Dec. 2011.

[23] Y. Li, K. Lee, and Y. Bresler, "Identifiability in bilinear inverse problems with applications to subspace or sparsity-constrained blind gain and phase calibration," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, pp. 822–842, Feb. 2017.

[24] C. Ling, J. Jiang, J. Wang, and Z. Liang, "Source localization of graph diffusion via variational autoencoders for graph inverse problems," in *Proc. Conf. Knowl. Discov. Data Mining*, 2022, pp. 1010–1020.

[25] S. Ling and T. Strohmer, "Self-calibration and biconvex compressive sensing," *Inverse Problems*, vol. 31, no. 115002, pp. 1–31, 2015.

[26] M. Mardani, G. Mateos, and G. B. Giannakis, "In-network sparsity-regularized rank minimization: Algorithms and applications," Mar. 2012, *arXiv:1203.1570*.

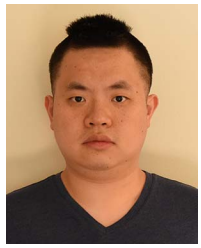
[27] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, Mar. 2021.

[28] M. Nagahama, K. Yamada, Y. Tanaka, S. H. Chan, and Y. C. Eldar, "Graph signal restoration using nested deep algorithm unrolling," *IEEE Trans. Signal Process.*, vol. 70, pp. 3296–3311, 2022.

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

[30] R. Pena, X. Bresson, and P. Vandergheynst, "Source localization on graphs via ℓ_1 recovery and spectral graph theory," in *Proc. IEEE Image, Video, Multidimensional Signal Process. Workshop*, 2016, pp. 1–5.

[31] P. C. Pinto, P. Thiran, and M. Vetterli, "Locating the source of diffusion in large-scale networks," *Phys. Rev. Lett.*, vol. 109, no. 068702, pp. 1–5, 2012.
[32] X. Pu, T. Cao, X. Zhang, X. Dong, and S. Chen, "Learning to learn graph topologies," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 1–14.
[33] D. Ramírez, A. G. Marques, and S. Segarra, "Graph-signal reconstruction and blind deconvolution for structured inputs," *Signal Process.*, vol. 188, 2021, Art. no. 108180.
[34] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, Jun. 2014.
[35] 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.
[36] E. Sefer and C. Kingsford, "Diffusion archeology for diffusion progression history reconstruction," in *Proc. IEEE Conf. Data Mining*, 2014, pp. 530–539.
[37] S. Segarra, G. Mateos, A. G. Marques, and A. Ribeiro, "Blind identification of graph filters," *IEEE Trans. Signal Process.*, vol. 65, no. 5, pp. 1146–1159, Mar. 2017.
[38] S. Shelke and V. Attar, "Source detection of rumor in social network–A review," *Online Social Netw. Media*, vol. 9, pp. 30–42, 2019.
[39] H. Shrivastava, X. Chen, B. Chen, G. Lan, S. Aluru, and L. Song, "GLAD: Learning sparse graph recovery," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–22.
[40] V. M. Tenorio, S. Rey, and A. G. Marques, "Blind deconvolution of sparse graph signals in the presence of perturbations," in *Proc. Int. Conf. Acoust., Speech, Signal Process*., 2024, pp. 9406–9410.
[41] J. Wang, J. Jiang, and L. Zhao, "An invertible graph diffusion neural network for source localization," in *Proc. ACM Web Conf.*, 2022, pp. 1058–1069.
[42] L. Wang and Y. Chi, "Blind deconvolution from multiple sparse inputs," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1384–1388, Oct. 2016.
[43] M. Wasserman and G. Mateos, "Graph structure learning with interpretable Bayesian neural networks," *Trans. Mach. Learn. Res.*, pp. 1–27, 2024.
[44] M. Wasserman, S. Sihag, G. Mateos, and A. Ribeiro, "Learning graph structure from convolutional mixtures," *Trans. Mach. Learn. Res.*, pp. 1–29, 2023.
[45] X. Yan, H. Fang, and Q. He, "Diffusion model for graph inverse problems: Towards effective source localization on complex networks," in *Proc. Adv. Neural. Inf. Process. Syst.*, 2023, pp. 22326–22 350.
[46] Y. Yang, J. Sun, H. Li, and Z. Xu, "ADMM-CSNet: A deep learning approach for image compressive sensing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 521–538, Mar. 2020.
[47] C. Ye and G. Mateos, "Blind deconvolution of graph signals: Robustness to graph perturbations," *IEEE Signal Process. Lett.*, vol. 32, pp. 1381–1385, 2025.
[48] C. Ye and G. Mateos, "Blind deconvolution on graphs: Exact and stable recovery," *Signal Process.*, vol. 230, May 2025, Art. no. 109864.
[49] C. Ye and G. Mateos, "Learning to identify sources of network diffusion," in *Proc. Eur. Signal Process. Conf.*, 2022, pp. 727–731.
[50] C. Ye, R. Shafipour, and G. Mateos, "Blind identification of invertible graph filters with multiple sparse inputs," in *Proc. Eur. Signal Process. Conf.*, 2018, pp. 121–125.
[51] H. Zhang, M. Yan, and W. Yin, "One condition for solution uniqueness and robustness of both l1-synthesis and l1-analysis minimizations," *Adv. Comput. Math.*, vol. 42, no. 6, pp. 1381–1399, 2016.
[52] P. Zhang, J. He, G. Long, G. Huang, and C. Zhang, "Towards anomalous diffusion sources detection in a large network," *ACM Trans. Internet Technol.*, vol. 16, no. 1, pp. 1–24, 2016.



Chang Ye (Member, IEEE) received the B.Sc. degree in physics from Sun Yat-Sen University, Guangzhou, China, in 2014, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Rochester, Rochester, NY, USA, in 2017 and 2025, respectively. Since 2025, he has been working toward the postgraduate studies in financial engineering with the Department of Economics and Finance, City University of Hong Kong, Hong Kong. From May 2020 to August 2020, he was an applied Scientist Intern with Amazon. His research interests include graph signal processing, deep learning, time series analysis, and generative models in finance.

Gonzalo Mateos (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the Universidad de la República, Montevideo, Uruguay, in 2005, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 2009 and 2011, respectively. From 2004 to 2006, he was a Systems Engineer with Asea Brown Boveri, Uruguay. In 2013, he was a Visiting Scholar with Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA. In 2014, he joined the University of Rochester, Rochester, NY, USA, where he is currently a Professor with the Department of Electrical and Computer Engineering, Department of Computer Science (secondary appointment), and the Associate Director for Research with the Goergen Institute for Data Science and Artificial Intelligence, University of Rochester. His research interests include statistical learning from complex data, network science, decentralized optimization, and graph signal processing. He was the Asaro Biggar Family Fellow in Data Science from 2020 to 2023.