A Joint Graph Signal and Laplacian Denoising Network Inspired by Majorization-Minimization

Zepeng Zhang* and Ziping Zhao†
* École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
Email: zepeng.zhang@epfl.ch
† ShanghaiTech University, Shanghai, China
Email: zipingzhao@shanghaitech.edu.cn

***Abstract*—Graph neural network (GNN) models have presented astonishing achievements in various application fields. However, they are shown to be vulnerable to adversarial attacks on graph structure and unnoticeable perturbations on the graph structure can cause significant performance drops in GNN models. Based on recent studies that reveal a class of GNN models is performing graph signal denoising (GSD), in this paper, we design a novel robust GNN model from a joint graph signal and Laplacian denoising problem (GSLD), named GSLDN. Specifically, GSLDN is built based on a block majorization-minimization algorithm for solving the GSLD problem. Designed in such a principled way, GSLDN is endowed with the power to fight against adversarial attacks on graph structure. Experiment results demonstrate the effectiveness of GSLDN.**

I. Introduction

Graph neural networks (GNNs) have shown great power in learning representations for graph-structured data and have achieved impressive performance in various graph-related tasks [1]. Typically, a GNN is a neural network consists of consecutive propagation layers, each of which contains two steps, namely, the feature aggregation step and the feature transformation step [2], [3]. The process of passing through the consecutive propagation layers is also referred to as message passing. Recent studies [4]–[6] have proven that the feature aggregation steps in a class of GNNs can be interpreted as performing graph signal denoising (GSD). Specifically, given a graph $\mathcal{G}$ with N nodes, where each node is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^M$, the GSD problem can be defined as follows:

$$\underset{\mathbf{H}}{\text{minimize}} \quad \|\mathbf{H}-\mathbf{X}\|_F^2 + \lambda \text{tr}\left(\mathbf{H}^T\mathbf{L}\mathbf{H}\right), \tag{1}$$

where $\mathbf{H}$ is the output after denoising (or after the message passing process in a GNN), $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N\times M}$ denotes the input feature matrix, λ represents a weight parameter, and $\mathbf{L}$ is the graph Laplacian matrix. From the underlying GSD problem (1) corresponding to a class of GNNs, e.g., graph convolutional network (GCN) [2] and approximate personalized propagation of neural predictions (APPNP) [7], it can be observed that the graph Laplacian matrix is assumed to be credible enough. However, if the graph is noisy (e.g., contains task-irrelevant edges) or under adversarial attacks, the performance of GNNs may drop significantly and even be worse than the performance of a simple baseline that ignores all the relational information among data features such as multi-layer perceptrons [8], [9].

To improve the robustness of GNN models, some studies suggest pre-processing the graph to refine the graph structure before feeding it to a GNN model. For example, the GNN-Jaccard method [10] proposes to remove edges that connect nodes with feature vectors of low "Jaccard similarity". Under the observation that existing adversarial attack methods tend to increase the rank of the graph adjacency matrix, [11] uses a low-rank approximation version of the given graph adjacency matrix as a substitute. Besides pre-processing the graph, there are also some studies designing "graph learners," which are parameterized models to learn the graph structure matrix (e.g., the graph adjacency matrix) that are co-trained with the GNN models [12], [13]. These previous studies, however, all focus on augmenting extra components to the existing GNN models to improve their robustness instead of designing new GNN architectures that are inherently robust.

Inspired by recent studies that build GNNs from an underlying optimization problems and the corresponding iterative algorithms [14]–[16], in this paper, we present GSLDN, which is designed based on a joint graph signal and Laplacian denoising (GSLD) problem. Specifically, the GSLD problem is first reparameterized using the graph Laplacian operator to reduce the parameter dimension. The reparameterized problem is tackled via the block majorization-minimization (BMM) algorithm, which not only gives iterative steps that are friendly to back-propagation training but also guarantees convergence to stationary points. Then a robust and interpretable message passing scheme is induced from the BMM algorithm, based on which we develop GSLDN. Built on a BMM algorithm, the message passing procedure in a trained GSLDN is naturally a parameter-optimized BMM algorithm for solving the GSLD problem. Thus, GSLDN model is endowed with the power to fight against adversarial attacks on the graph structure. The experiment results on real-world datasets demonstrate that the proposed GSLDN model is resistive to adversarial attacks.

II. Graph Signal and Structure Denoising

We consider a positively weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ represents the vertex set containing N nodes, $\mathcal{E}$ denotes the edge set, and $\mathbf{W} \in \mathbb{R}^{N\times N}$ is the weight matrix with w_{ij} indicating the pairwise relationship

between the i-th node and the j-th node. The degree matrix $\mathbf{D} \in \mathbb{R}^{N\times N}$ is a diagonal matrix whose i-th diagonal element $d_{ii} = \sum_{j=1}^{N} w_{ij}$. The positive semi-definite Laplacian matrix is accordingly defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. Assuming each node is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^M$, then the feature matrix is defined as $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N\times M}$. Each column of $\mathbf{X}$ can also be interpreted as a graph signal.

A. Problem Formulation

In this section, we will introduce the GSLD problem, which augments the GSD problem (1) with graph Laplacian learning components to help mitigate the influence of adversarial attacks on graph structure.

For the GSD problem (1), we notice that it only concerns with denoising the graph signal while assumes the Laplacian matrix to be accurate enough. However, when the graph structure is maliciously manipulated by an attacker, such assumption will result in unsatisfactory model performance and may lead to dramatic consequences. Specifically, it has been shown that with unnoticeable and carefully designed modifications, the performance of GNN models significantly drops and may even be worse than the performance of a simple baseline that ignores all relational information [8], [9].

In order to handle potentially attacked graph structures, we propose to augment the objective in Problem (1) with a Laplacian learning term, i.e., $\|\mathbf{L} - \mathbf{L}_n\|_F^2$, where $\mathbf{L}$ is the refined Laplacian matrix that needs to be optimized and $\mathbf{L}_n$ is the given attacked Laplacian matrix. Then the objective of the GSD problem becomes

$$f(\mathbf{H}, \mathbf{L}) = \|\mathbf{H} - \mathbf{X}\|_F^2 + \gamma\|\mathbf{L} - \mathbf{L}_n\|_F^2 + \lambda \mathrm{tr}\left(\mathbf{H}^T\mathbf{L}\mathbf{H}\right). \quad (2)$$

Under the assumption that the denoised feature $\mathbf{H}$ is smooth over the refined graph $\mathbf{L}$, by minimizing $f(\mathbf{H}, \mathbf{L})$, the influence of the adversarial attacks can be mitigated.

It is evident that the Laplacian matrix is symmetric with degrees of freedom equal to $\frac{N(N-1)}{2}$. Thus, the Laplacian matrix $\mathbf{L}$ can be determined by a vector $\mathbf{w} \in \mathbb{R}_+^{\frac{N(N-1)}{2}}$ and the $f(\mathbf{H}, \mathbf{L})$ can be reparameterized accordingly. By reparameterizing $\mathbf{L}$ to $\mathbf{w}$ through the graph Laplacian operator, the dimension of variables is reduced and the positive semi-definiteness of $\mathbf{L}$ is satisfied naturally. In the following, we first introduce the notion of graph Laplacian operator [17].

Definition 1. The graph Laplacian operator $\mathcal{L} : \mathbb{R}^{\frac{N(N-1)}{2}} \to \mathbb{R}^{N\times N}$, $\mathbf{w} \to \mathcal{L}\mathbf{w}$, is defined as

$$[\mathcal{L}\mathbf{w}]_{ij} = \begin{cases} -w_{k_{ij}} & i > j, \\ [\mathcal{L}\mathbf{w}]_{ji} & i < j, \\ -\sum_{j\neq i}[\mathcal{L}\mathbf{w}]_{ij} & i = j, \end{cases}$$

where $k_{ij} = i - j + \frac{j-1}{2}(2n - j)$.

Similarly, the inverse graph Laplacian operator is defined as $\mathcal{L}^{-1} : \mathbb{R}^{N\times N} \to \mathbb{R}^{\frac{N(N-1)}{2}}$ such that $\mathbf{w} = \mathcal{L}^{-1}\mathcal{L}\mathbf{w}$. Based on the graph Laplacian operator, we can derive its adjoint operator $\mathcal{L}^*$, which satisfies $\mathrm{tr}\big((\mathcal{L}\mathbf{w})^T\mathbf{Y}\big) = \mathbf{w}^T\mathcal{L}^*\mathbf{Y}$ [17].

Definition 2. The adjoint operator of the graph Laplacian operator $\mathcal{L}$, i.e., $\mathcal{L}^* : \mathbb{R}^{N\times N} \to \mathbb{R}^{\frac{N(N-1)}{2}}$, $\mathbf{Y} \to \mathcal{L}^*\mathbf{Y}$, is defined as

$$[\mathcal{L}^*\mathbf{Y}]_k = Y_{ii} - Y_{ij} - Y_{ji} + Y_{jj}, \quad k = 1, \ldots, \frac{N(N-1)}{2},$$

where $i, j \in \mathbb{Z}^+$ satisfy $k = i - j + \frac{j-1}{2}(2n - j)$ and $i > j$.

To better understand the graph Laplacian operator and its adjoint operator, more discussions and examples can be found in [17]. Using the notion of Laplacian operator, we rewrite $f(\mathbf{H}, \mathbf{L})$ (where f is reused with a little abuse of the notation) as:

$$f(\mathbf{H}, \mathbf{w}) = \|\mathbf{H} - \mathbf{X}\|_F^2 + \gamma\|\mathcal{L}\mathbf{w} - \mathbf{L}_n\|_F^2 + \lambda \mathrm{tr}\left(\mathbf{H}^T\mathcal{L}\mathbf{w}\mathbf{H}\right).$$

Considering that real-world graphs are generally sparse and low-rank [18], we further consider regularization terms, i.e., $||\mathcal{L}\mathbf{w}||_1$ and $||\mathcal{L}\mathbf{w}||_*$, to promote the sparsity and the low-rankness of the learned Laplacian matrix $\mathcal{L}\mathbf{w}$. It can be proved that $||\mathcal{L}\mathbf{w}||_1 = 4||\mathbf{w}||_1$. Besides, since $\mathcal{L}\mathbf{w}$ is positive semi-definite, we have $||\mathcal{L}\mathbf{w}||_* = \mathrm{tr}(\mathcal{L}\mathbf{w}) = 2||\mathbf{w}||_1$. Thus, the sparsity and the low-rankness promoting property can be achieved with a single regularization term $||\mathbf{w}||_1$. To restrict the scale of the Laplacian matrix, we further consider a regularization term $||\mathbf{w}||_2^2$. Finally, the GSLD problem is defined as follows:

$$\underset{\mathbf{H},\mathbf{w}\geq\mathbf{0}}{\text{minimize}} \quad f(\mathbf{H}, \mathbf{w}) + \alpha\|\mathbf{w}\|_1 + \beta\|\mathbf{w}\|_2^2, \quad (3)$$

where α and β are positive weighting parameters.

B. Solving Problem (3) via BMM

In this section, we will develop an efficient and convergent BMM algorithm for solving Problem (3). Denoting $\mathbf{H}^{(k-1)}$ and $\mathbf{w}^{(k-1)}$ as the value of $\mathbf{H}$ and $\mathbf{w}$ after the $(k-1)$-th iteration, with $\mathbf{H}^{(0)}$ and $\mathbf{w}^{(0)}$ being the initial points, we derive the update steps of $\mathbf{H}$ and $\mathbf{w}$ at the k-th iteration in the following. To proceed with the resolution, we introduce a useful lemma.

Lemma 3 ([19])**.** *For a continuously differentiable function $f(\mathbf{X})$ where ∇f is Lipschitz continuous with Lipschitz constant L, the following result holds for all $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{N\times M}$ and $\xi \geq L$:*

$$f(\mathbf{X}) \leq f(\mathbf{Y}) + \mathrm{tr}\left((\mathbf{X} - \mathbf{Y})^T\nabla f(\mathbf{Y})\right) + \frac{\xi}{2}\|\mathbf{X} - \mathbf{Y}\|_F^2.$$

1) The Resolution of the $\mathbf{H}$-block Subproblem: With $\mathbf{w}$ being held fixed, the subproblem w.r.t. $\mathbf{H}$ becomes

$$\underset{\mathbf{H}}{\text{minimize}} \quad h(\mathbf{H}\,|\,\mathbf{w}^{(k-1)}) = \|\mathbf{H}-\mathbf{X}\|_F^2 + \lambda\mathrm{tr}\big(\mathbf{H}^T\mathcal{L}\mathbf{w}^{(k-1)}\mathbf{H}\big).$$

Let $\lambda_{\max}^{(k-1)}$ be the largest eigenvalue of $\mathcal{L}\mathbf{w}^{(k-1)}$. Then the Lipschitz constant of $\nabla h^{(k-1)}(\mathbf{H})$ is $2 + 2\lambda\lambda_{\max}^{(k-1)}$. With $\eta^{(k)} \geq 2 + 2\lambda\lambda_{\max}^{(k-1)}$, we can construct a majorization function of $h^{(k-1)}(\mathbf{H})$ based on Lemma 3 as follows:

$$\begin{aligned} &p\big(\mathbf{H} \mid \mathbf{w}^{(k-1)}, \mathbf{H}^{(k-1)}\big) \\ =&\|\mathbf{H}^{(k-1)} - \mathbf{X}\|_F^2 + \mathrm{tr}\Big(\lambda(\mathbf{H}^{(k-1)})^T\mathcal{L}\mathbf{w}^{(k-1)}\mathbf{H}^{(k-1)}\Big) \\ &+ \mathrm{tr}\Big((\mathbf{H} - \mathbf{H}^{(k-1)})^T\mathbf{D}^{(k-1)}\Big) + \frac{\eta^{(k)}}{2}\|\mathbf{H} - \mathbf{H}^{(k-1)}\|_F^2, \end{aligned}$$

where $\mathbf{D}^{(k-1)} = 2\mathbf{H}^{(k-1)} - 2\mathbf{X} + 2\lambda\mathcal{L}\mathbf{w}^{(k-1)}\mathbf{H}^{(k-1)}$. Ignoring the constant terms in $p\big(\mathbf{H} \mid \mathbf{w}^{(k-1)}, \mathbf{H}^{(k-1)}\big)$, we obtain a surrogate optimization problem as follows:

$$\underset{\mathbf{H}}{\text{minimize}} \quad \|\mathbf{H} - \mathbf{H}^{(k-1)} + \frac{1}{\eta^{(k)}}\mathbf{D}^{(k-1)}\|_F^2.$$

This surrogate optimization problem has a closed-form solution, which leads to following update rule:

$$\mathbf{H}^{(k)} = \mathbf{H}^{(k-1)} - \frac{1}{\eta^{(k)}}\mathbf{D}^{(k-1)}. \tag{4}$$

Note that this update step can also be interpreted as a gradient step with a stepsize of $\frac{1}{\eta^{(k)}}$.

2) The Resolution of the $\mathbf{w}$*-block Subproblem:* With $\mathbf{H}$ being held fixed, the subproblem with respect to $\mathbf{w}$ becomes

$$\begin{aligned}\underset{\mathbf{w}\geq\mathbf{0}}{\text{minimize}} \;\; & w(\mathbf{w} \mid \mathbf{H}^{(k-1)}) = \|\mathcal{L}\mathbf{w}\|_F^2 - 2\mathrm{tr}\left(\mathbf{L}_n^T\mathcal{L}\mathbf{w}\right) \\ & + \frac{\lambda}{\gamma}\mathrm{tr}\left((\mathbf{H}^{(k-1)})^T\mathcal{L}\mathbf{w}\mathbf{H}^{(k-1)}\right) + \frac{\alpha}{\gamma}\|\mathbf{w}\|_1 + \frac{\beta}{\gamma}\|\mathbf{w}\|_2^2,\end{aligned}$$

where we omit the constant term $\|\mathbf{L}_n\|_F^2$. Using the adjoint operator $\mathcal{L}^*$, we have

$$w(\mathbf{w} \mid \mathbf{H}^{(k-1)}) = \|\mathcal{L}\mathbf{w}\|_F^2 - \mathbf{w}^T\mathbf{c}^{(k-1)} + \frac{\beta}{\gamma}\|\mathbf{w}\|_2^2,$$

where $\mathbf{c}^{(k-1)} = \mathcal{L}^*\big[2\mathbf{L}_n - \frac{\lambda}{\gamma}\mathbf{H}^{(k-1)}(\mathbf{H}^{(k-1)})^T\big] - \frac{\alpha}{\gamma}\mathbf{1}$ with $\mathbf{1} \in \mathbb{R}^{\frac{N(N-1)}{2}}$ being the all-one column vector. Due to the non-negativity constraint, the $\mathbf{w}$-block subproblem does not have a closed-form solution. To obtain closed-form solution, we employ a majorization technique [20].

According to [17, Lemma 1] and [17, Lemma 3], we conclude that the $\|\mathcal{L}\mathbf{w}\|_F^2$ is L-smooth with $L = 2N$. Thus, we can construct a majorization function of $\|\mathcal{L}\mathbf{w}\|_F^2$ with $\xi^{(k)} \geq 2N$ based on Lemma 3, leading to a majorization function of $w(\mathbf{w} \mid \mathbf{H}^{(k-1)})$ as follows:

$$\begin{aligned}& g\big(\mathbf{w} \mid \mathbf{w}^{(k-1)}, \mathbf{H}^{(k-1)}\big) \\ &= \|\mathcal{L}\mathbf{w}^{(k-1)}\|_F^2 + 2\big(\mathbf{w} - \mathbf{w}^{(k-1)}\big)^T\mathcal{L}^*\mathcal{L}\big(\mathbf{w}^{(k-1)}\big) \\ &\quad + \frac{\xi^{(k)}}{2}\big\|\mathbf{w} - \mathbf{w}^{(k-1)}\big\|^2 - \mathbf{w}^T\mathbf{c}^{(k-1)} + \frac{\beta}{\gamma}\|\mathbf{w}\|_2^2.\end{aligned}$$

Note that $\xi^{(1)}, \ldots, \xi^{(K)}$ can be set to be the same or varying in different layers. Ignoring the constant terms in $g\big(\mathbf{w} \mid \mathbf{w}^{(k-1)}, \mathbf{H}^{(k-1)}\big)$, we obtain a surrogate optimization problem as follows:

$$\underset{\mathbf{w}\geq\mathbf{0}}{\text{minimize}} \quad \left(\frac{\xi^{(k)}}{2} + \frac{\beta}{\gamma}\right)\|\mathbf{w}\|_2^2 - \mathbf{w}^T\mathbf{b}^{(k-1)}, \tag{5}$$

where $\mathbf{b}^{(k-1)} = \xi^{(k)}\mathbf{w}^{(k-1)} + 2\mathcal{L}^*\mathcal{L}\big(\mathbf{w}^{(k-1)}\big) - \mathbf{c}^{(k-1)}$. Then using the KKT optimality condition, we have the following update rule:

$$\mathbf{w}^{(k)} = \frac{\gamma}{\gamma\xi^{(k)} + 2\beta}\mathrm{ReLU}\left(\mathbf{b}^{(k-1)}\right). \tag{6}$$

In conclusion, in each iteration, the BMM algorithm update $\mathbf{H}$ as in Eq. (4) and update $\mathbf{w}$ as in Eq. (6). The convergence property of the proposed algorithm is stated in the following theorem.

Theorem 4. *With* $\eta^{(k)} \geq 2 + 2\lambda\lambda_{\max}^{(k-1)}$ *and* $\xi^{(k)} \geq 2N$ *for* $k = 1, \ldots, K$*, the BMM algorithm for solving the GSLD problem* (3) *with update rules as in Eq.* (4) *and Eq.* (6) *converges to the stationary points.*

Proof: With $\eta^{(k)} \geq 2 + 2\lambda\lambda_{\max}^{(k-1)}$ and $\xi^{(k)} \geq 2N$ for $k = 1, \ldots, K$, the proposed algorithm is a standard BMM algorithm. Thus, following the convergence results for general BMM algorithms [21], the proposed algorithm ensures convergence to stationary points. ■

III. The Graph Signal and Laplacian Denoising Network

Based on the BMM algorithm developed in Section II-B, we present the GSLDN below.

GSLDN Architecture

$$\begin{aligned}& \mathbf{H}^{(0)} = \mathrm{ReLU}\left(\mathrm{ReLU}\left(\mathbf{X}\mathbf{\Theta}_1\right)\mathbf{\Theta}_2\right), \quad \mathbf{w}^{(0)} = \mathcal{L}^{-1}\mathbf{L}_n, \\ & \text{for } k = 1, \ldots, K, \\ & \begin{cases}\mathbf{H}^{(k)} = \big(1 - 2\eta^{(k)}\big)\mathbf{H}^{(k-1)} \\ \qquad\qquad -2\eta^{(k)}\lambda\mathcal{L}\mathbf{w}^{(k-1)}\mathbf{H}^{(k-1)} + 2\eta^{(k)}\mathbf{X}, \\ \mathbf{w}^{(k)} = \frac{\gamma}{\gamma\xi^{(k)} + 2\beta}\mathrm{ReLU}\left(\mathbf{b}^{(k-1)}\right)\end{cases} \\ & \mathbf{Z} = \mathrm{softmax}\big(\mathbf{H}^{(K)}\big).\end{aligned}$$

In the above GSLDN, $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ are the learnable weight matrices and $\mathbf{Z}$ is the output probability matrix for prediction. Note that we use the decoupled structure as in APPNP [7], i.e., the feature aggregation steps are not intertwined with the feature transformation steps. Specifically, the feature transformation $\mathrm{ReLU}\left(\mathrm{ReLU}\left(\mathbf{X}\mathbf{\Theta}_1\right)\mathbf{\Theta}_2\right)$ is performed before conducting the feature aggregation steps. To ensure the convergence of the BMM algorithm, i.e., the message passing procedure, we can set $\eta^{(k)}$ and $\xi^{(k)}$ as constants satisfying $\eta^{(k)} \geq 2 + 2\lambda\lambda_{\max}^{(k-1)}$ and $\xi^{(k)} \geq 2N$. To improve the model expressiveness and save the computation of $\lambda_{\max}^{(k-1)}$, we set $\eta^{(k)}$ and $\xi^{(k)}$ to be learnable parameters. Since the GSLDN is induced from the BMM algorithm, the message passing procedure in a trained GSLDN is naturally a parameter-optimized BMM algorithm.

Since GSLDN is a GNN architecture, some extra components used in existing graph structure learning methods, i.e., the pre-processing methods [10], [11] and the graph-learner based methods [12], [13] can be used to further improve the model performance, which we leave for future exploration.

IV. Experimental Results

In this section, we conduct experiments on semi-supervised node classification tasks with two real-world citation graphs, i.e., Cora and Citeseer [22] to validate the effectiveness of the proposed GSLDN.

A. Experiment Settings

To evaluate the effectiveness of GSLDN, we compare it with GCN and several benchmarks that are designed from different perspectives to robustify the GNNs, including GCN-Jaccard [10] that pre-processes the graph by eliminating edges with low Jaccard similarity of node feature vectors, GCN-SVD [11] that uses the low-rank approximation of the given graph adjacency matrix, Pro-GNN [12] that jointly learns a graph structure and a GNN model guided by some predefined structural priors, and Elastic GNN [14] that utilizes trend filtering instead of Laplacian smoothing to promote robustness. For GCN-Jaccard, GCN-SVD, and Pro-GNN, we use the implementation provided in DeepRobust [23]. For Elastic GNN, we follow the implementation provided in the original paper [14].

For each graph, we only consider the largest connected component and randomly select 10%/10%/80% of nodes for training, validation, and testing. The Adam optimizer is used in all experiments. The models' hyperparameters are tuned based on the results of the validation set. The search space of hyperparameters is as follows: 1) learning rate: {0.005, 0.01, 0.05}; 2) weight decay: {0, 5e-5, 5e-4}; 3) dropout rate: {0.1, 0.5, 0.8}; 4) model depth: {2, 4, 8, 16}. For GCN-Jaccard, the threshold of Jaccard similarity for removing dissimilar edges is chosen from {0.01, 0.02, 0.03, 0.04, 0.05, 0.1}. For GCN-SVD, the reduced rank of the graph is tuned from {5, 10, 15, 50, 100, 200}. For Elastic GNN, the regularization coefficients are chosen from {3, 6, 9}. For Pro-GNN, we adopt the hyperparameters provided in their paper [12].

B. Performance Under Adversarial Attack

We evaluate the model performance under the training-time adversarial attacks [9], i.e., the graph is first attacked, and then the GNN models are trained on the perturbed graph. Two types of attacks are considered, namely, the global attack that aims to reduce the overall performance of GNNs [9] and the targeted attack that aims to fool GNNs on some specific nodes [8]. For all the experimental results, we give the average performance and standard variance with 10 independent trials. The best model and the runner-up model are highlighted in bold and wavy underlining, respectively.

1) Model Performance Under Global Attack: For the global attack, we use a representative method called meta-attack [9] and the results at a 20% and a 25% perturbation rate are showcased in Table I. For GCN-Jaccard method, we use the results reported in [12]. From the table, we observe that the proposed GSLDN model achieves better or comparative results compared with other methods. Specifically, the GSLDN outperforms other GNN architectures, i.e., GCN and Elastic GNN, indicating that GSLDN can effectively resist the global attack. For instance, GSLDN improves GCN by 11% on the Cora dataset at a 25% perturbation rate. Note that although Pro-GNN outperforms GSLDN in some cases, it requires training an additional graph learner. Moreover, the graph learner used in Pro-GNN can also be used to boost the performance of GSLDN, which we leave for future work.

TABLE I
CLASSIFICATION PERFORMANCE UNDER GLOBAL ATTACK

Ptb. rate	Cora		Citeseer	
	0.2	0.25	0.2	0.25
GCN	61.5 ± 2.2	56.8 ± 1.4	59.7 ± 0.8	60.0 ± 1.0
GCN-Jaccard	65.7 ± 0.9	60.8 ± 1.1	59.3 ± 1.4	59.9 ± 1.5
GCN-SVD	58.8 ± 2.1	59.1 ± 2.7	65.8 ± 0.7	62.3 ± 0.6
Pro-GNN	**70.1 ± 2.5**	67.0 ± 1.6	**70.1 ± 1.1**	**69.7 ± 0.9**
Elastic GNN	68.3 ± 3.5	65.8 ± 2.5	61.6 ± 1.8	64.0 ± 2.2
GSLDN	69.2 ± 2.0	**67.8 ± 1.1**	67.3 ± 0.4	66.5 ± 0.8

TABLE II
CLASSIFICATION PERFORMANCE UNDER TARGETED ATTACK

Ptb. number	Cora		Citeseer	
	4.0	5.0	4.0	5.0
GCN	61.5 ± 2.2	56.8 ± 1.4	62.5 ± 1.6	52.7 ± 2.0
GCN-Jaccard	61.7 ± 1.1	59.5 ± 1.9	76.3 ± 1.5	72.9 ± 1.7
GCN-SVD	58.8 ± 2.1	59.2 ± 2.7	62.2 ± 3.3	60.2 ± 6.7
Pro-GNN	70.1 ± 2.5	67.0 ± 1.6	75.7 ± 4.9	74.0 ± 7.1
Elastic GNN	68.3 ± 3.5	65.8 ± 2.5	72.1 ± 5.6	74.0 ± 3.9
GSLDN	**70.5 ± 1.5**	**68.3 ± 4.5**	**78.2 ± 2.7**	**75.2 ± 3.5**

2) Model Performance Under Targeted Attack: For the targeted attack, we use a representative method called nettack [8]. The results with 4 and 5 perturbations per targeted node are reported in Table II. Specifically, following [12], we choose the nodes in the test set with degrees larger than 10 as targeted nodes and the classification performance is evaluated on target nodes. From the table, we can see that the proposed GS^{2}DNet attains better performance than other baselines in all cases. For instance, on the Citeseer dataset with 5 perturbations per targeted node, GSLDN improves GCN by 22.5% and outperforms other baselines by 1.2%. Such inspiring results demonstrate that GSLDN can better resist targeted attacks than other baseline methods.

V. CONCLUSION

In this paper, we first introduced graph Laplacian learning components into graph signal denoising. Then we developed a block majorization-minimization algorithm for problem resolution, based on which we proposed the GSLDN model. Designed from an optimization perspective, GSLDN is endowed with the power of performing Laplacian learning during the message passing procedure. Experiments validate the robustness of GSLDN under both the global and targeted attack.

REFERENCES

[1] L. Wu, P. Cui, J. Pei, and L. Zhao, *Graph neural networks: Foundations, frontiers, and applications*. Singapore: Springer Singapore, 2022, p. 725.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[3] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[4] Y. Ma, X. Liu, T. Zhao, Y. Liu, J. Tang, and N. Shah, "A unified view on graph neural networks as graph signal denoising," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1202–1211.

[5] M. Zhu, X. Wang, C. Shi, H. Ji, and P. Cui, "Interpreting and unifying graph neural networks with an optimization framework," in *Proceedings of the Web Conference*, 2021, pp. 1215–1226.

[6] Z. Zhang and Z. Zhao, "Towards understanding graph neural networks: An algorithm unrolling perspective," *arXiv preprint arXiv:2206.04471*, 2022.

[7] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *International Conference on Learning Representations*, 2018.

[8] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2847–2856.

[9] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations*, 2019.

[10] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial examples for graph data: Deep insights into attack and defense," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 4816–4823.

[11] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis, "All you need is low (rank) defending against adversarial attacks on graphs," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 169–177.

[12] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 66–74.

[13] B. Runwal and S. Kumar, "Robustifying GNN via weighted laplacian," in *IEEE International Conference on Signal Processing and Communications*, 2022, pp. 1–5.

[14] X. Liu, W. Jin, Y. Ma, *et al.*, "Elastic graph neural networks," in *International Conference on Machine Learning*, PMLR, 2021, pp. 6837–6849.

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

[16] Z. Zhang, S. Lu, Z. Huang, and Z. Zhao, "ASGNN: Graph neural networks with adaptive structure," *arXiv preprint arXiv:2210.01002*, 2022.

[17] S. Kumar, J. Ying, J. V. de Miranda Cardoso, and D. P. Palomar, "A unified framework for structured graph learning via spectral constraints.," *J. Mach. Learn. Res.*, vol. 21, no. 22, pp. 1–60, 2020.

[18] K. Zhou, H. Zha, and L. Song, "Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes," in *Artificial Intelligence and Statistics*, PMLR, 2013, pp. 641–649.

[19] D. P. Bertsekas, *Nonlinear programming*, 1999.

[20] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 794–816, 2016.

[21] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[22] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[23] Y. Li, W. Jin, H. Xu, and J. Tang, "Deeprobust: A pytorch library for adversarial attacks and defenses," *arXiv preprint arXiv:2005.06149*, 2020.