Robust Graph Representation Learning for Local Corruption Recovery

Bingxin Zhou [1 2] **Yuanhong Jiang** [3] **Yu Guang Wang** [3 4] **Jingwei Liang** [3] **Junbin Gao** [1] **Shirui Pan** [5]
Xiaoqun Zhang [3]

Abstract

Real-world graph observations may contain local corruptions by abnormal behaviors. While existing research usually pursues global smoothness in graph embedding, these rarely observed anomalies are harmful to an accurate prediction. This work establishes a graph learning scheme that automatically detects corrupted node attributes and recovers robust embedding for prediction tasks. The detection operation does not make any assumptions about the distribution of the local corruptions. It pinpoints the positions of the anomalous node attributes in an unbiased mask matrix, where robust estimations are recovered with an $\ell_{p,q}$ regularizer. We alleviate an inertial alternating direction method of multipliers to approach a new embedding that is sparse in the framelet domain and conditionally close to input observations. Extensive experiments validate the model recovers robust graph representations from black-box poisoning and achieves excellent performance.

1. Introduction

Graph neural networks (GNNs; Bronstein et al. (2017); Wu et al. (2020b); Zhou et al. (2020); Zhang et al. (2020); Atz et al. (2021)) have received tremendous success in the past few years. Graphs, as the input of GNNs, record useful features and structural information, and they exist widely in many fields such as biomedical science (Ahmedt-Aristizabal et al., 2021), social networks (Fan et al., 2019), and recommendation systems (Wu et al., 2020a).

[1]The University of Sydney Business School, The University of Sydney, Camperdown, Sudney NSW 2006, Australia. [2]Zhangjiang Institute for Advanced Study, Shanghai Jiao Tong University, Shanghai, 200240, China. [3]Institute of Natural Sciences, School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai, 200240, China. [4]School of Mathematics and Statistics, University of New South Wales, Sydney NSW 2052, Australia [5]Department of Data Science and AI, Faculty of IT, Monash University, Melbourne VIC 3800, Australia.. Correspondence to: Bingxin Zhou <bzho3923@uni.sydney.edu.au>.

Presented at the ICML 2022 Workshop on Topology, Algebra, and Geometry in Machine Learning.

Similar to other real-world observations, inaccurate observations are ubiquitous in graphs with a noticeable side-effect for graph representation learning. For instance, fraudulent users in social media tend to fake user avatars or online activities. A recommender might be dysfunctional by mislabeled items or users. Such disruptive observations hinder the model fitting and prediction. The feature aggregation in graph representation learning accentuates the negative influence of irregular entities to their neighborhoods, resulting in a misleading latent feature representation for the predictor. To this end, this research proposes to protect the graph prediction performance by recovering a robust representation for the irregular entities.

Existing works are aware of the harmful graph anomalies. Graph anomaly detection (Ding et al., 2019; Peng et al., 2020; Zhu et al., 2020; Ma et al., 2021) identifies the small portion of problematic nodes; graph defense refines the learning manner of a classifier to provide promising predictions against potential threats (Dai et al., 2018; Zügner & Günnemann, 2019; Xu et al., 2020); optimization-based graph convolutions smooth out global noise by special regularization designs (Liu et al., 2021; Zhou et al., 2021; Zhu et al., 2021; Chen et al., 2021). However, the first candidate detects irregular nodes rather than node attributes, and it does not amend the flawed representation of the identified outlier. The second approach provides an adequate solution to amend the prediction performance, but most researches focus on graph rewiring, as edges are believed more vulnerable to adversarial attacks. Last but not least, the third choice usually makes assumptions on the distribution of feature corruptions that they are normally observed in all inputs.

Instead, this paper develops a '*detect-and-then-recovery*' strategy to save graph representation learning from a small portion of hidden corruptions in input node attributes. In particular, an unsupervised encoder module first exposes suspicious attributes and assigns a mask matrix to record their positions. The detector requires no prior on the distribution of the anomalous attributes. The constructed mask is submitted to a sparsity regularizer with graph framelet transforms (Dong, 2017; Zheng et al., 2021) to find a robust approximation of the initial input, where the iterative optimizing scheme acts similar to graph convolutional layers that constantly smooth the hidden feature representation

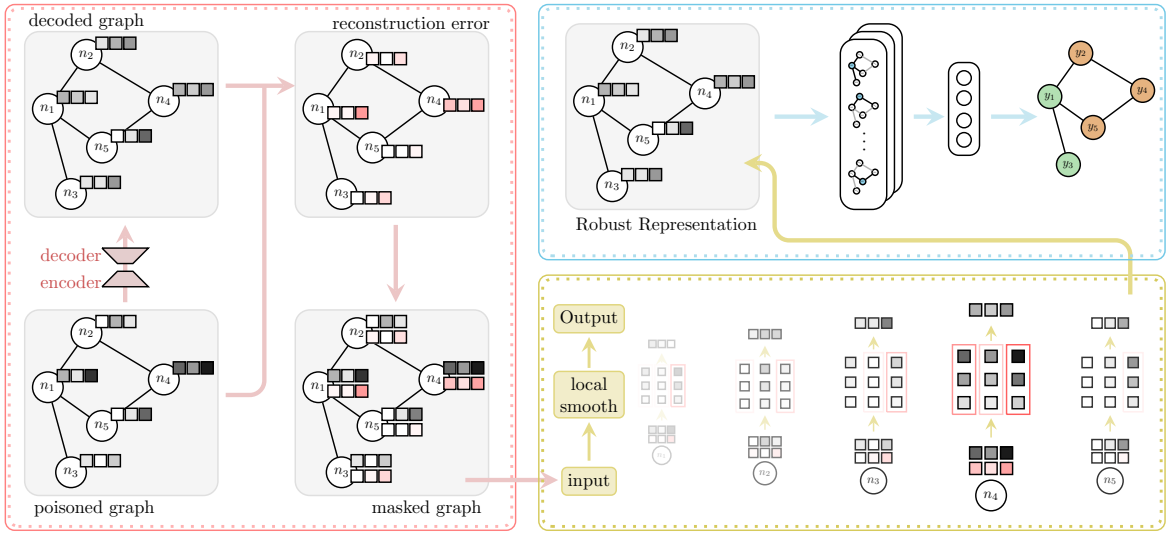

Figure 1. A brief grasp of the proposed MAGNET's architecture. The input feature attributes (gray) are assumed locally corrupted. The first module (red) constructs a mask matrix with graph autoencoder, which is then sent to an ADMM-oriented optimizer together with the perturbed graph input to iterate a robust feature representation (in yellow). The new representation is functioned as hidden embeddings by typical graph convolutional layers, which can be further encoded by other convolutions, or be sent to a predictor (in blue).

for GNN prediction tasks. With the transforms of the input signal, the posterior mask guarantees that the denoising is predominantly conducted at the essential spots, and the representation pumps up the robustness significantly.

Combining the three key ingredients of **M**ask, **A**DMM, and **G**raph, we name our model as MAGNET. Figure 1 illustrates the three components for unsupervised mask construction, localized robust optimization, and graph representation learning. The proposed MAGNET detects local corruptions in an unsupervised manner and approximates a robust representation for graph network training. We also develop an efficient inertial alternating direction method of multipliers (ADMM) algorithm, which provides a faster convergence than the plain ADMM with time complexity of $\mathcal{O}(k^{-1})$.

2. Problem Formulation

An undirected attributed $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ has $n = |\mathcal{V}|$ nodes with the interactions described by an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. The observed d-dimensional node features $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ is a noised version of the ground truth signal $\boldsymbol{U}$, i.e.,

$$\boldsymbol{X} = \boldsymbol{U} + \boldsymbol{E}_1 + \boldsymbol{E}_2, \tag{1}$$

where we call $\boldsymbol{E}_1$ global noise and $\boldsymbol{E}_2$ outliers or local corruptions. The main difference between $\boldsymbol{E}_1$ and $\boldsymbol{E}_2$ is that the former might universally exist on the entire graph, and the latter generally take a small chance of existence so it cannot be observed from a large set of nodes. We hereby make three assumptions on the properties of $\boldsymbol{U}, \boldsymbol{E}_1$ and $\boldsymbol{E}_2$: (a) $\boldsymbol{E}_1 \sim \mathcal{D}(0, \sigma)$ where σ is considerably small; (b) $\boldsymbol{E}_2$ take a small portion of the entire graph; and (c) $\boldsymbol{X}$ is close to $\boldsymbol{U}$ besides the anomalous locations. The associated objective function to estimate $\boldsymbol{U}$ reads

$$\begin{aligned} &\min_{\boldsymbol{U}} \alpha \mathrm{Reg}(\boldsymbol{U}) + \beta \mathrm{Loss}(\boldsymbol{E}_2) \\ &\text{such that } \boldsymbol{X}|_{\boldsymbol{M}} = (\boldsymbol{U} + \boldsymbol{E}_1 + \boldsymbol{E}_2)|_{\boldsymbol{M}}, \end{aligned} \tag{2}$$

where α, β are tuning parameters, and $\boldsymbol{M}$ is a mask matrix of outliers indices. The $\mathrm{Reg}(\cdot), \mathrm{Loss}(\cdot)$ denote regularizers and loss functions satisfying assumptions (a) and (b).

Choice of $\mathrm{Loss}(\boldsymbol{E}_2)$. The loss function plays the role of finding an object $\boldsymbol{U}$ that approximates the input feature $\boldsymbol{X}$. Assumption (b) requires a measure with respect to the sparsity of $\boldsymbol{E}_2$ has to be optimized. The best choice for sparsity, in theory, is ℓ_0-norm that counts the number of nonzero entries in $\boldsymbol{E}_2$. While solving a ℓ_0 constrained problem is NP-hard, ℓ_1-norm allows feasible solution for (2) that promotes good sparsity measure of $\boldsymbol{E}_2$ (Chen et al., 2015). Such a design has been practiced in residual analysis-based *anomaly detection* methods to label suspicious small outliers (Li et al., 2017; Peng et al., 2018).

Choice of $\mathrm{Reg}(\boldsymbol{U})$. The regularization term is a penalty complementary to the loss, which controls the noise level of $\boldsymbol{X}$ by the smoothness of $\boldsymbol{U}$, and it is usually quantified by some type of energy. For instance, GCN (Kipf & Welling, 2017) utilizes normalized Dirichlet energy of $\boldsymbol{U}$ by $\mathrm{tr}(\boldsymbol{U}^\top \tilde{\boldsymbol{L}} \boldsymbol{U})$, where $\tilde{\boldsymbol{L}}$ denotes the normalized graph Laplacian from $\tilde{\boldsymbol{L}} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}$ with the adjacency matrix $\boldsymbol{A}$ and its degree matrix $\boldsymbol{D}$. Minimizing such energy encourages message transmissions among connected nodes. By extracting the summary of neighborhood space, unnoticeable noise is likely to be smoothed out. In addition, minimizing the Dirichlet energy implies a low-rank solution to $\boldsymbol{U}$ (Monti et al., 2017). Such a *graph smoothing effect*

(Zhu et al., 2021; Liu et al., 2021) that minimizes Dirichlet energy can be found in many spatial-based graph convolutions (Klicpera et al., 2018; Xu et al., 2018; Wu et al., 2019). Alternatively, spectral-based methods regularizes the graph signal in a transformed domain by $\boldsymbol{L}$. For example, Dong (2017) and Zhou et al. (2021) minimize the ℓ_1-norm total variation of framelet coefficients, and Mahmood et al. (2018) take a similar regularization in the wavelet domain. As the total variation reflects redundant local fluctuations, a solution to minimize it is believed to remove noisy details while simultaneously preserving essential patterns.

Restriction on $\boldsymbol{E}_1 + \boldsymbol{E}_2$. Compared to the main ingredients of the objective function, the treatment to the fidelity constraint is rather trivial if the outlier $\boldsymbol{E}_2$ does not exist, where a regularizer can be adopted to minimize the difference between $\boldsymbol{X}$ and $\boldsymbol{U} + \boldsymbol{E}_1$. When $\boldsymbol{E}_2$ is assumed with a small percentage, it becomes irrational to force $\boldsymbol{U}$ to approximate $\boldsymbol{X}$ especially on anomalous locations. Instead, a conditional approximation should be placed with the index matrix $\boldsymbol{M}$, in which case only attributes at regular positions are required aligned, i.e., $\mathrm{Reg}\left(\boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{U})\right)$. The regularization can be appended to the main objective by

$$\min_{\boldsymbol{U}} \alpha \mathrm{Reg}(\boldsymbol{U}) + \beta \mathrm{Loss}\left(\boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{U})\right). \tag{3}$$

The above optimization for graph signal recovery is well-defined. Nevertheless, it has three issues in application. First, a direct minimization of the Dirichlet energy in spatial domain usually falls into the pitfall of over-smoothing, where the recovered graph loss expressivity drastically (Balcilar et al., 2020). On the other hand, spectral transforms can be sophisticated and time-consuming, which is generally circumvented by the majority of studies. Second, restricting a ℓ_1 or ℓ_2 norm does not adapt to a specific dataset or application, which could give rise to the precision loss of the recovered graph representation. Last but not least, attaining the mask matrix $\boldsymbol{M}$ in (3) can be nasty in practice, as the prior knowledge of $\boldsymbol{M}$ is generally inaccessible.

We hereby formulate a new objective function

$$\min_{\boldsymbol{U}} \|\boldsymbol{\nu}\boldsymbol{\mathcal{W}}\boldsymbol{U}\|_{p,G} + \frac{1}{2}\|\boldsymbol{M} \odot (\boldsymbol{U} - \boldsymbol{X})\|_{q,G}^{q}, \tag{4}$$

where $\boldsymbol{\mathcal{W}}$ is a set of multi-scale and multi-level framelet decomposition operators. See Appendix A for a brief explanation on the fast approximation of $\boldsymbol{\mathcal{W}}$. These decomposition operators transform the input feature $\boldsymbol{U}$ to low-pass and high-pass framelet coefficients in the framelet (spectral) domain. The two hyperparameters α, β in (3) are deduced to $\boldsymbol{\nu}$, which is a set of tunable parameters adaptive to high-frequency framelet coefficients in different scales. We also replace the ordinary Euclidean ℓ_k-norm with a graph ℓ_k-norm, denoted as $\ell_{k,G}$, to assign higher penalties to influential nodes. For an arbitrary node $\boldsymbol{v}_i$ of degree $\boldsymbol{D}_{ii}$, $\|\boldsymbol{v}_i\|_{k,G} := \left(\|\boldsymbol{v}_i\|^k \cdot \boldsymbol{D}_{ii}\right)^{\frac{1}{k}}$.

Compared to the initial design (2), (4) made three adjustments to tackle the identified issues. First, minimizing the first component smooths out $\boldsymbol{E}_1$ from high-pass framelet coefficients, which avoids information loss by spatial Dirichlet energy minimization. In other words, the global noise $\boldsymbol{E}_1$ is removed without sacrificing small-energy features. Secondly, we adopt $\ell_{p,q}$-norm to adaptively restrict the sparsity of recovered graph with tunable $p \in [1, 2], q \in [0, 1]$. As introduced in Section 4, the optimization can be solved by an inertial version of the alternating direction method of multipliers with promising convergence.

A potential solution to the unreachable mask matrix $\boldsymbol{M}$ is to add a sub-problem of optimization to (4), which introduces a two-stage optimization problem. However, this approach blows up the difficulty of solving the existing problem and the mask could be a very complicated region to reveal. Instead, we consider an unsupervised GNN scheme to automate the anomalous positions discovery. We approximate the anomaly score of the raw feature matrix with a classic graph anomaly detection scheme that looks for the reconstruction error between the raw feature matrix and the reconstructed matrix from neural networks.

3. Graph Anomaly Detection for Mask Matrix

Graph anomaly detection is an important subproblem of the general graph signal recovery where outliers are assumed to exist in the input. A detector identifies nodes with $\boldsymbol{E}_2$ as *community anomalies* (Ma et al., 2021), which is defined as nodes that have distinct attribute values compared to their neighbors of the same community. The underlying assumption here is that the outlier is sparse and their coefficients are antipathetic from the smooth graph signal $\boldsymbol{U}$. We hereby consider GNN-based algorithms to investigate the difference of each node from the representation of its local community.

3.1. Graph Autoencoder

Autoencoder is a powerful tool for reconstructing corrupted objects (Aggarwal, 2017; Kovenko & Bogach, 2020). GAE (Kipf & Welling, 2016) is a GCN-based classic graph autoencoder network, which revisions have drawn substantial interest for graph anomaly detection (Ding et al., 2019; Peng et al., 2020; Zhu et al., 2020).

Network Training. A GAE layer takes the information of $\mathcal{G}$ to obtain an embedding $\boldsymbol{Z} \in \mathbb{R}^{n \times h}$ of a hidden size h by $\boldsymbol{Z} = \mathrm{GCN}(\boldsymbol{X}, \boldsymbol{A})$. Alternatively, one trains a feed-forward network to recover graph attributes by $\boldsymbol{X}' = \mathrm{FFN}(\boldsymbol{Z})$. The $\boldsymbol{X}'$ is a smoothed version of $\boldsymbol{X}$, which is believed removing essential noise or minor corruptions of the raw input. In the design by Ding et al. (2019) and Peng et al. (2018) for graph anomaly detection, the learning objective of the neural network is to best reconstruct the graph data

$(\boldsymbol{X}', \boldsymbol{A}')$, which loss function is the weighted average of the squared reconstruction errors

$$\mathcal{L} = \alpha \|\boldsymbol{X} - \boldsymbol{X}'\|_2.$$

Anomaly Score. Once the graph is reconstructed, it is straightforward to find outliers by comparing the input and output representation. For node entities, a corrupted object has different patterns from its local community, so its initial representation should be distinct from the smoothed representation. The reconstruction error is thus an appropriate indicator for diagnosing such divergence. The anomaly score for a particular node $\boldsymbol{v}_i$ reads

$$\text{score}(\boldsymbol{v}_i) = \|\boldsymbol{x}_i - \boldsymbol{x}_i'\|_2.$$

3.2. Mask Matrix Generation

Unlike conventional anomaly detection tasks, we aim at generating a mask matrix of size $n \times d$ that identifies the anomalous attributes, instead of corrupted node entities. In addition, our primary focus is on graph recovery which does not explicitly consider the disordered node connections or the adjacency matrix. We thus define the mask matrix by tweaking the anomaly score function

$$\boldsymbol{M} = 1 - \text{threshold}(\|\boldsymbol{X} - \boldsymbol{X}'\|_1, \tau) \tag{5}$$

By differentiating the raw feature matrix and the reconstructed matrix from a graph autoencoder, we establish a mask matrix that deduces element-wise reconstruction errors. It is then binarized with the threshold value $\tau \in (0, 1)$. For the jth feature of node i, $\boldsymbol{M}_{ij} = 0$ indicates the value is trustworthy and $\boldsymbol{M}_{ij} = 1$ suggests assigning a new value to avoid potential corruption.

Under assumption (b) that $\boldsymbol{E}_2$ rarely exist, the vast majority of the mask matrix should be 1, i.e., $1 - \boldsymbol{M}$ should be a sparse matrix. This requirement is double-assured by the autoencoder algorithm and the thresholding function. As the autoencoder does simply blurs $\boldsymbol{X}$ to reach the smooth $\boldsymbol{U}$ and the magnitude of $\boldsymbol{E}_1$ is assumed smaller than $\boldsymbol{E}_2$'s, the difference of $\boldsymbol{X}$ and $\boldsymbol{X}'$ mainly comes from $\boldsymbol{E}_2$. On top of that, the tunable τ makes further adjustments on the sparsity of $\boldsymbol{M}$. Figure 2 supplements empirical evidence for the effect of selecting different τ on the sparsity of $1 - \boldsymbol{M}$ with perturbed citation networks. An increasing τ is illustrated to reduce the number of non-zero elements in $1 - \boldsymbol{M}$ from a considerably sparse level to a lower degree.

4. An Inertia Alternating Direction Method of Multipliers (ADMM)

This section introduces the numerical scheme to solve (4).

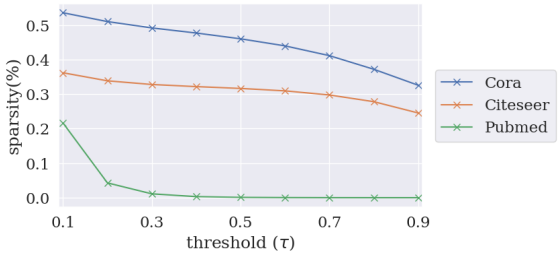

Figure 2. Demonstration on the effect of τ to the sparsity level of the mask matrix on citation networks with attribute injection.

4.1. An inertial ADMM

Denote $\boldsymbol{Z} = \boldsymbol{\mathcal{W}}\boldsymbol{U}$, then we can rewrite (4) as

$$\begin{aligned} &\min_{\boldsymbol{U},\boldsymbol{Z}} \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{1}{2}\|\boldsymbol{M} \odot (\boldsymbol{U} - \boldsymbol{X})\|_{q,G}^q, \\ &\text{s.t. } \boldsymbol{Z} = \boldsymbol{\mathcal{W}}\boldsymbol{U}. \end{aligned} \tag{6}$$

This forms a standard formulation of problems that can be solved by ADMM (Gabay & Mercier, 1976). The associated augmented Lagrangian to (6) reads

$$\begin{aligned} \mathcal{L}(\boldsymbol{U}, \boldsymbol{Z}; \boldsymbol{Y}) := &\|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{1}{2}\|\boldsymbol{M} \odot (\boldsymbol{U} - \boldsymbol{X})\|_{q,G}^q \\ &+ \langle \boldsymbol{Y}, \boldsymbol{\mathcal{W}}\boldsymbol{U} - \boldsymbol{Z}\rangle + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U} - \boldsymbol{Z}\|^2, \end{aligned}$$

where $\gamma > 0$. To find a saddle-point of $\mathcal{L}(\boldsymbol{U}, \boldsymbol{Z}; \boldsymbol{Y})$, ADMM applies the following iteration

$$\begin{aligned} \boldsymbol{Z}_{k+1} =& \arg\min_{\boldsymbol{Z}} \ \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U}_k - \boldsymbol{Z}\|^2 \\ &+ \langle \boldsymbol{Y}_k, \boldsymbol{\mathcal{W}}\boldsymbol{U}_k - \boldsymbol{Z}\rangle, \\ \boldsymbol{U}_{k+1} =& \arg\min_{\boldsymbol{U}} \ \tfrac{1}{2}\|\boldsymbol{M} \odot (\boldsymbol{U} - \boldsymbol{X})\|_{q,G}^q \\ &+ \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U} - \boldsymbol{Z}_{k+1}\|^2 + \langle \boldsymbol{Y}_k, \boldsymbol{\mathcal{W}}\boldsymbol{U} - \boldsymbol{Z}_{k+1}\rangle, \\ \boldsymbol{Y}_{k+1} =& \boldsymbol{Y}_k + \gamma(\boldsymbol{\mathcal{W}}\boldsymbol{U}_{k+1} - \boldsymbol{Z}_{k+1}). \end{aligned} \tag{7}$$

To secure an efficient solver, we consider the inertial ADMM motivated by Alvarez & Attouch (2001). To derive the scheme, we define $\boldsymbol{V}_{k+1} = \boldsymbol{Y}_k - \gamma\boldsymbol{Z}_{k+1}$ and obtain the following inertial scheme for (7)

$$\begin{aligned} \boldsymbol{Z}_{k+1} &= \arg\min_{\boldsymbol{Z}} \ \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{\gamma}{2}\|\boldsymbol{Z} - (2\boldsymbol{Y}_k - \widetilde{\boldsymbol{V}}_k)/\gamma\|^2, \\ \boldsymbol{V}_{k+1} &= \boldsymbol{Y}_k - \gamma\boldsymbol{Z}_{k+1}, \\ \widetilde{\boldsymbol{V}}_{k+1} &= \boldsymbol{V}_{k+1} + a_k(\boldsymbol{V}_{k+1} - \boldsymbol{V}_k), \\ \boldsymbol{U}_{k+1} &= \arg\min_{\boldsymbol{U}} \ \tfrac{1}{2}\|\boldsymbol{M} \odot (\boldsymbol{U} - \boldsymbol{X})\|_{q,G}^q \\ &\quad + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U} + \widetilde{\boldsymbol{V}}_{k+1}/\gamma\|^2, \\ \boldsymbol{Y}_{k+1} &= \widetilde{\boldsymbol{V}}_{k+1} + \gamma\boldsymbol{\mathcal{W}}\boldsymbol{U}_{k+1}, \end{aligned} \tag{8}$$

where $\widetilde{\boldsymbol{V}}_{k+1}$ is called the inertial term and a_k is the inertial parameter. We refer to the work of Boţ & Csetnek (2014) and Boţ et al. (2015), and the references therein for the convergence analysis of the above inertial ADMM scheme.

Table 1. Average performance with **non-targeted** attribute injection

Module	CORA	CITESEER	PUBMED	COAUTHOR-CS	WIKI-CS
clean	81.26±0.65	71.77±0.29	79.01±0.44	90.19±0.48	77.62±0.26
corrupted	69.06±0.74	57.58±0.71	67.69±0.40	82.41±0.23	65.44±0.23
APPNP (Klicpera et al., 2018)	68.46±0.81	60.04±0.59	68.70±0.47	71.14±0.54	56.53±0.72
GNNGUARD (Zhang & Zitnik, 2020)	61.96±0.30	54.94±1.00	68.50±0.38	80.67±0.88	65.69±0.32
ELASTICGNN (Liu et al., 2021)	**77.74±0.79**	64.61±0.85	71.23±0.21	79.91±1.39	64.18±0.53
MAGNET-one (ours)	75.88±0.42	59.22±0.34	68.97±0.21	84.04±0.56	70.83±0.29
MAGNET-gae (ours)	76.81±0.69	**64.79±0.73**	**75.41±0.35**	**86.50±0.37**	**72.40±0.21**
MAGNET-true	78.48±0.67	68.55±0.74	75.63±0.56	89.23±0.40	75.50±0.20

4.2. Subproblems

The solutions to the subproblems of $\boldsymbol{Z}_{k+1}$ and $\boldsymbol{U}_{k+1}$ depend on p and q. Here we enumerate the special cases of $p = \{0, 1\}$, $q = \{1, 2\}$ which covers the maximum and minimum sparsity in regularization.

Choices of p. Different values of p affects the thresholding operator in the update of $\boldsymbol{Z}_{k+1}$. For the case of $p = 1$, $\|\boldsymbol{\nu Z}\|_{p,G}$ becomes the ℓ_1-norm and its update requires a soft-thresholding, that is,

$$S_\alpha(x) = \operatorname{sign}(x) \odot \max\left\{|x| - \alpha, 0\right\}.$$

When $p = 0$, a hard-thresholding defines $H_\alpha(x) = x$ when $|x| > \alpha$, and $H_\alpha(x) = 0$ otherwise.

Choices of q. Compared to $\boldsymbol{Z}_{k+1}$, the update of $\boldsymbol{U}_{k+1}$ in (8) is more complicated as it involves more regularization terms. When $q = 2$, the objective is a quadratic problem.

With $\boldsymbol{\mathcal{W}}^\top \boldsymbol{\mathcal{W}} = \mathrm{Id}$, differentiating the function gives

$$\boldsymbol{U}_{k+1} = \left(\boldsymbol{M} \odot \boldsymbol{X} - \boldsymbol{\mathcal{W}}^\top \widetilde{\boldsymbol{V}}_{k+1}\right) / \left(\boldsymbol{M} + \gamma\right), \tag{9}$$

which requires an element-wise division. Since the fast approximation of $\boldsymbol{\mathcal{W}}$ is implemented by the Chebyshev approximation, it happens when the approximation degree is considerably small that the approximation has noticeable error, i.e., $\boldsymbol{\mathcal{W}}^T \boldsymbol{\mathcal{W}} \neq \mathrm{Id}$. Alternatively, the descent type methods such as gradient descent, conjugate gradient can be applied to inexactly solve the sub-problem with I steps of iteration. At the ith ($i \leq I$) iteration,

$$\begin{aligned} \boldsymbol{U}^{(i+1)} = \boldsymbol{U}^{(i)} - \alpha \Big(\boldsymbol{M} \odot (\boldsymbol{U}^{(i)} - \boldsymbol{X}) \\ + \gamma \boldsymbol{\mathcal{W}}^\top (\boldsymbol{\mathcal{W}} \boldsymbol{U}^{(i)} + \widetilde{\boldsymbol{V}}_{k+1}/\gamma)\Big), \end{aligned} \tag{10}$$

where α is the step-size. The solution is then $\boldsymbol{U}_{k+1} = \boldsymbol{U}^{(I)}$.

In the case of $q = 1$, let $\boldsymbol{Q} = \boldsymbol{U} - \boldsymbol{X}$ and consider

$$\tfrac{1}{2}\|\boldsymbol{M} \odot \boldsymbol{Q}\|_{1,G} + \tfrac{\gamma}{2}\|\boldsymbol{Q} + \boldsymbol{X} + \boldsymbol{\mathcal{W}}^\top \widetilde{\boldsymbol{V}}_{k+1}/\gamma\|^2$$

under the optimality condition. The $\boldsymbol{Q}_{k+1}$ is the soft-thresholding of $-\boldsymbol{X} - \boldsymbol{\mathcal{W}}^\top \widetilde{\boldsymbol{V}}_{k+1}/\gamma$ and

$$\boldsymbol{U}_{k+1} = \boldsymbol{Q}_{k+1} + \boldsymbol{X}. \tag{11}$$

5. Numerical Experiments

5.1. Experimental Protocol

Benchmark Preparation. We examine MAGNET on five benchmark datasets: **Cora**, **Citeseer** and **PubMed** of the citation networks (Yang et al., 2016), **Wiki-CS** (Mernyei & Cangea, 2020) that classifies articles from Wikipedia database, and **Coauthor-CS** (Shchur et al., 2018) that labels the most active field of authors. As the given datasets do not provide ground truth of anomalies, we conduct two types of black-box poisoning methods that have been used in graph anomaly detection and graph defense. A **attribute injection** method (Song et al., 2007; Ding et al., 2019) perturbs attribute through swapping attributes of the most distinct samples in a random subgraph. A set of anomalies can be obtained from different random subgraphs. We also adopt a **graph adversarial attack** (Zügner & Günnemann, 2019) that leverages meta-learning to pollute node attributes with the meta-gradient of the loss function.

Training Setup. For a fast recovering from the corrupted graph, an inertial ADMM is iterated for 15 times. The GAE to approximate binary reconstruction errors for the mask matrix consists of two GCN encoding layers followed with two GCN attribute decoding layers. The hidden representation is sent to GAE attribute reconstruction module for the final prediction. All datasets follow the standard public split and processing rules. The test performance is evaluated by the average accuracy of 10 repetitions.

Baseline Comparison. We investigate three types of $\boldsymbol{M}$: all-ones (MAGNET-one), GAE-approximated (MAGNET-gae), and ground truth (MAGNET-true) matrices, where the last case does not participate in the performance comparison. Instead, it indicates the upper limit of MAGNET with the perfect mask matrix. We compare our model to three popular baseline models for graph smoothing but with different design philosophy: APPNP (Klicpera et al., 2018) that avoids global smoothness with residual connections; GNNGUARD (Zhang & Zitnik, 2020) that modifies the neighbor relevance in message passing to mitigate local corruption; and ELASTICGNN (Liu et al., 2021) that pursues local smoothness with a mix of ℓ_1 and ℓ_2 regularizers.

Table 2. Average performance with **targeted** adversarial attack

Module	CORA	CITESEER	PUBMED
clean	81.26±0.65	71.77±0.29	79.01±0.44
corrupted	75.07±0.64	55.32±2.22	72.88±0.30
APPNP	73.49±0.59	55.67±0.28	70.63±1.07
GNNGUARD	72.02±0.61	57.64±1.31	71.10±0.32
ELASTICGNN	**79.25±0.50**	67.29±1.17	71.95±0.52
MAGNET-one (ours)	77.11±0.45	62.49±1.70	75.83±2.05
MAGNET-gae (ours)	79.04±0.50	**67.40±0.73**	**78.63±0.32**
MAGNET-true	80.88±0.37	67.46±0.95	79.16±0.41

5.2. Node Classification with Graph Recovery

Table 1-2 compare model performance under two types of local corruptions, where Table 2 excludes **Wiki-CS** and **Coauthor-CS** as the attribute poison of meta-attack is ineffective on them. We highlight each score in red (green) by their relative improvement (retrogression) over the non-smoothed corrupted baseline (row 2).

It can be observed that MAGNET-gae outperforms its competitors and recovers at most 94% prediction accuracy from the perturbed attributes. Furthermore, a more accurate mask approximation could push the prediction performance of graph representation up to MAGNET-true's scores. On the other hand, the three baseline graph smoothing methods fail to denoise local corruption within the input. In some cases they are fooled to make worse predictions, as they are not able to distinguish outliers from regular patterns. APPNP leverages residual connections to avoid global smoothness by reinforcing local anomalies; GNNGUARD makes modifications on the edge connection to indirectly influence node representations; ELASTICGNN, although realizing the importance of local smoothness, designs the optimization with low-pass filters and restricts stringent consistency of the new representation even on anomalous positions.

5.3. Further Investigation

We next visualize the effect of the GAE-oriented mask approximation and ADMM optimization.

Mask Approximation. We first verify the quality of GAE through the recall of the approximation. Figure 3 pictures the conditional mask matrix from model reconstruction error on **Cora** with attribute injection. A 200×200 sub-region is amplified for both the ground truth matrix and the approximated mask matrix at the middle, which indicates clearly the sparsity of both matrices. The approximated mask matrix succeeds in seizing 60% of anomalies, which provides a reliable foundation to the subsequent recovering work. Other visualizations of different datasets are in Appendix C

ADMM Optimization. Figure 4 visually exhibits the effect of ADMM optimization on local graph inpainting with

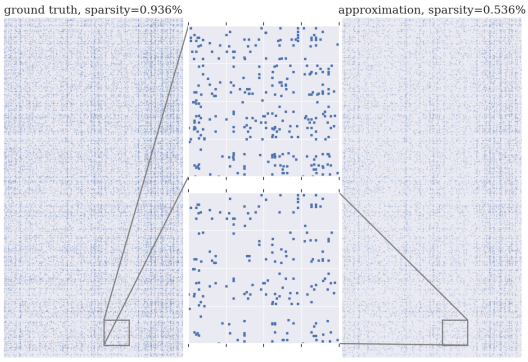

Figure 3. The ground truth mask (left), and conditional GAE-approximated mask (right) at threshold $\tau = 0.1$.

Original image
Noise only in Red Region

Local PSNR=17.43
Global PSNR=27.64

Local PSNR=24.59
Global PSNR=33.81

Figure 4. Image recovery with local additive white noise ($\sigma = 50$). Three images are the noisy raw input (left), the inpainting result by BM3D (middle), and the masked ADMM by MAGNET (right).

an example of image denoising, where the raw picture is chosen from the BSD68 dataset with 480×320 pixels. It is processed to a graph of $2,400$ nodes and 64 feature attributes. Each node is transformed by a patch of 8×8 pixels. We select 9 of the nodes to assign white noise of $\mathcal{N}(0, 1)$ on the attributes. As we are interested in the performance of the ADMM optimizer, we assume a given mask matrix. Compare to the classic denoising model BM3D (Dabov et al., 2007), MAGNET restricts major smoothing effects within the masked region. The rest of the 'clean' area maintains a sharp detail, which is very contradictory to the BM3D's result, which blurs the entire scale of the picture.

6. Conclusion

We develop MAGNET, a graph neural network for recovering a robust graph data representation from locally corrupted node attributes. The key computational unit for coping with regional outliers is based on a sparse and multi-scale regularizer with a mask of the anomalous positions in graph node attributes, which is approximated with an unsupervised graph autoencoder that requires no prior knowledge on the distribution of anomalies. The optimization problem is decorated with an $l_{p,q}$ regularization with an efficient inertial ADMM, where the tunable p, q stimulate the maximum ability of the optimizer. The multi-level framelet coefficients

removes global noises and regularizes local anomalous attributes simultaneously. Our proposed model achieves satisfying performance to recovering a robust graph embedding from local corruptions. In contrast, graph smoothing and defense baseline methods fail to provide a decent solution.

Acknowledgements

YW, JL and XZ acknowledge support from the Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102). YW and JL also acknowledges support from SJTU and Huawei ExploreX Funding (SD6040004/034). We are also grateful to the anonymous reviewers for their feedback.

References

Aggarwal, C. C. Linear models for outlier detection. In *Outlier Analysis*, pp. 65–110. Springer, 2017.

Ahmedt-Aristizabal, D., Armin, M. A., Denman, S., Fookes, C., and Petersson, L. Graph-based deep learning for medical diagnosis and analysis: past, present and future. *Sensors*, 21(14):4758, 2021.

Alvarez, F. and Attouch, H. An inertial proximal method for maximal monotone operators via discretization of a nonlinear oscillator with damping. *Set-Valued Analysis*, 9(1):3–11, 2001.

Atz, K., Grisoni, F., and Schneider, G. Geometric deep learning on molecular representations. *Nature Machine Intelligence*, pp. 1–10, 2021.

Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., and Honeine, P. Analyzing the expressive power of graph neural networks in a spectral perspective. In *International Conference on Learning Representations*, 2020.

Boţ, R. I. and Csetnek, E. R. An inertial alternating direction method of multipliers. *arXiv preprint arXiv:1404.4582*, 2014.

Boţ, R. I., Csetnek, E. R., and Hendrich, C. Inertial douglas–rachford splitting for monotone inclusion problems. *Applied Mathematics and Computation*, 256:472–487, 2015.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Chen, S., Sandryhaila, A., Moura, J. M., and Kovačević, J. Signal recovery on graphs: Variation minimization. *IEEE Transactions on Signal Processing*, 63(17):4609–4624, 2015.

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

Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8): 2080–2095, 2007.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. Adversarial attack on graph structured data. In *International conference on machine learning*, pp. 1115–1124. PMLR, 2018.

Ding, K., Li, J., Bhanushali, R., and Liu, H. Deep anomaly detection on attributed networks. In *SIAM International Conference on Data Mining (SDM)*, 2019.

Dong, B. Sparse representation on graphs by tight wavelet frames and applications. *Applied and Computational Harmonic Analysis*, 42(3):452–479, 2017.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *The world wide web conference*, pp. 417–426, 2019.

Gabay, D. and Mercier, B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *proceedings of International Conference on Learning Representations*, 2017.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2018.

Kovenko, V. and Bogach, I. A comprehensive study of autoencoders' applications related to images. In *IT&I Workshops*, pp. 43–54, 2020.

Li, J., Dani, H., Hu, X., and Liu, H. Radar: Residual analysis for anomaly detection in attributed networks. In *IJCAI*, pp. 2152–2158, 2017.

Li, Y., Jin, W., Xu, H., and Tang, J. Deeprobust: a platform for adversarial attacks and defenses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

Liu, X., Jin, W., Ma, Y., Li, Y., Liu, H., Wang, Y., Yan, M., and Tang, J. Elastic graph neural networks. In *ICML*, 2021.

Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q. Z., Xiong, H., and Akoglu, L. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Mahmood, F., Shahid, N., Skoglund, U., and Vandergheynst, P. Adaptive graph-based total variation for tomographic reconstructions. *IEEE Signal Processing Letters*, 25(5): 700–704, 2018.

Mernyei, P. and Cangea, C. Wiki-cs: a wikipedia-based benchmark for graph neural networks. *arXiv:2007.02901*, 2020.

Monti, F., Bronstein, M. M., and Bresson, X. Geometric matrix completion with recurrent multi-graph neural networks. In *NIPS*, 2017.

Peng, Z., Luo, M., Li, J., Liu, H., and Zheng, Q. Anomalous: A joint modeling approach for anomaly detection on attributed networks. In *IJCAI*, pp. 3513–3519, 2018.

Peng, Z., Luo, M., Li, J., Xue, L., and Zheng, Q. A deep multi-view framework for anomaly detection on attributed networks. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

Song, X., Wu, M., Jermaine, C., and Ranka, S. Conditional anomaly detection. *IEEE Transactions on knowledge and Data Engineering*, 19(5):631–645, 2007.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Wu, S., Sun, F., Zhang, W., Xie, X., and Cui, B. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*, 2020a.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020b.

Xu, H., Ma, Y., Liu, H.-C., Deb, D., Liu, H., Tang, J.-L., and Jain, A. K. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462. PMLR, 2018.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

Zhang, X. and Zitnik, M. Gnnguard: Defending graph neural networks against adversarial attacks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

Zheng, X., Zhou, B., Gao, J., Wang, Y. G., Lio, P., Li, M., and Montúfar, G. How framelets enhance graph neural networks. In *proceedings of International Conference on Machine Learning*, 2021.

Zhou, B., Li, R., Zheng, X., Wang, Y. G., and Gao, J. Graph denoising with framelet regularizer. *arXiv:2111.03264*, 2021.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: a review of methods and applications. *AI Open*, 1:57–81, 2020.

Zhu, D., Ma, Y., and Liu, Y. Anomaly detection with deep graph autoencoders on attributed networks. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6. IEEE, 2020.

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

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2019.

A. Graph Framelet Transform

This section briefs the fast computation of undecimated framelet transform on graphs. The initial idea was proposed by Dong (2017), which is then developed to a graph convolution by Zheng et al. (2021). The key to the implementation is to make fast approximation on the framelet decomposition and reconstruction operators through Chebyshev approximation.

Framelet transform divides an input signal to multiple channels by a set of low-pass and high-passes *framelet operators*. For a specific nodes p of a signal $\boldsymbol{x}$ at *scale* $j \in \mathbb{Z}$, we define the undecimated framelets $\boldsymbol{\varphi}_{j,p}(g), \boldsymbol{\psi}^r_{j,p}(g)$ by

$$\begin{aligned} \boldsymbol{\varphi}_{j,p}(g) &:= \textstyle\sum_{\ell=1}^N \widehat{\alpha}\left(\frac{\lambda_\ell}{2^j}\right) \overline{\boldsymbol{u}_\ell(p)} \boldsymbol{u}_\ell(v), \\ \boldsymbol{\psi}^r_{j,p}(g) &:= \textstyle\sum_{\ell=1}^N \widehat{\beta^{(r)}}\left(\frac{\lambda_\ell}{2^j}\right) \overline{\boldsymbol{u}_\ell(p)} \boldsymbol{u}_\ell(v), \quad r = 1, \ldots, n. \end{aligned} \tag{12}$$

For two integers J, J_1 ($J > J_1$), an *undecimated framelet system* from a scale J_1 $\mathrm{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta})$ is an *undecimated tight frame* for $l_2(\mathcal{G})$, which is a non-homogeneous, stationary affine system:

$$\begin{aligned} \mathrm{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta}) &:= \mathrm{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta}; \mathcal{G}) \\ &:= \{\varphi_{J_1,p} : p \in V\} \cup \left\{\psi^r_{j,p} : p \in V, j = J_1, \ldots, J\right\}_{r=1}^n. \end{aligned} \tag{13}$$

The elements in $\mathrm{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta})$ are the *undecimated tight framelets* on $\mathcal{G}$.

We call $\Psi = \{\alpha; \beta^{(1)}, \ldots, \beta^{(K)}\}$ a set of *scaling functions*, and it is determined by a filter bank $\eta := \{a; b^{(1)}, \ldots, b^{(K)}\}$. In particular, we consider the Haar-type filter with one high pass. For $x \in \mathbb{R}$, it defines

$$\widehat{\alpha}(x) = \cos(x/2) \;\text{ and }\; \widehat{\beta^{(1)}}(x) = \sin(x/2).$$

The other component that plays a key role for embedding graph topology is the eigenpairs $\{(\lambda, \boldsymbol{u})\}_{j=1}^n$ of the graph Laplacian $\mathcal{L}$. The set of framelet decomposition operator projects input signals to a transformed domain as *framelet coefficients*. To allow fast approximation of the filter spectral functions, m-order Chebyshev polynomials is considered. Denote the m-order approximation of α and $\{\beta^{(1)}, \ldots, \beta^{(K)}\}$ by $\mathcal{T}_0^m$ and $\{\mathcal{T}_k^m\}_{k=1}^K$, the framelet decomposition operators at $(r, j) \in \{(1,1), \ldots, (1,J), \ldots (n,1), \ldots, (n,J)\} \cup \{(0,J)\}$ is defined by

$$\mathcal{W}_{k,1} = \begin{cases} \mathcal{T}_r\left(2^{-K}\mathcal{L}\right), & j = 1, \\ \mathcal{T}_r\left(2^{K+j-1}\mathcal{L}\right) \mathcal{T}_0\left(2^{K+j-2}\mathcal{L}\right) \ldots \mathcal{T}_0\left(2^{-K}\mathcal{L}\right), & \text{otherwise}, \end{cases}$$

where the dilation scale K satisfies $\lambda_{\max} \leq 2^K \pi$. In this definition, the finest scale is $1/2^{K+J}$ that guarantees $\lambda_\ell / 2^{K+J-j} \in (0, \pi)$ for $j = 1, 2, \ldots, n$.

The approximated $\mathcal{W}$ is used in the penalty term of the objective function (4) formulated in Section 2. To learn more facts about the undecimated framelet transform on graph, we refer the readers to the work by Dong (2017); Zheng et al. (2021).

B. More details on the ADMM algorithm

This section provides essential details for the inertial ADMM to understand the update rules defined in Section 4.

B.1. Inertial ADMM

We denote $\boldsymbol{Z} = \boldsymbol{\mathcal{W}}\boldsymbol{U}$ and rewrite (4) as

$$\min_{\boldsymbol{U},\boldsymbol{Z}} \ \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{1}{2}\|\boldsymbol{M}\odot(\boldsymbol{U}-\boldsymbol{X})\|_{q,G}^{q}, \ \text{ such that } \boldsymbol{Z} = \boldsymbol{\mathcal{W}}\boldsymbol{U}.$$

This forms a standard formulation of problems that can be solved by Alternating Direction Method of Multipliers (ADMM; Gabay & Mercier (1976)). The associated augmented Lagrangian reads

$$\mathcal{L}(\boldsymbol{U},\boldsymbol{Z};\boldsymbol{Y}) := \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{1}{2}\|\boldsymbol{M}\odot(\boldsymbol{U}-\boldsymbol{X})\|_{q,G}^{q} + \langle \boldsymbol{Y}, \boldsymbol{\mathcal{W}}\boldsymbol{U}-\boldsymbol{Z}\rangle + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U}-\boldsymbol{Z}\|^2.$$

To find a saddle-point of $\mathcal{L}(\boldsymbol{U},\boldsymbol{Z};\boldsymbol{Y})$, ADMM applies the following iteration

$$\begin{aligned}
\boldsymbol{Z}_{k+1} &= \arg\min_{\boldsymbol{Z}} \ \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U}_k - \boldsymbol{Z}\|^2 + \langle \boldsymbol{Y}_k, \boldsymbol{\mathcal{W}}\boldsymbol{U}_k - \boldsymbol{Z}\rangle,\\
\boldsymbol{U}_{k+1} &= \arg\min_{\boldsymbol{U}} \ \tfrac{1}{2}\|\boldsymbol{M}\odot(\boldsymbol{U}-\boldsymbol{X})\|_{q,G}^{q} + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U} - \boldsymbol{Z}_{k+1}\|^2 + \langle \boldsymbol{Y}_k, \boldsymbol{\mathcal{W}}\boldsymbol{U} - \boldsymbol{Z}_{k+1}\rangle,\\
\boldsymbol{Y}_{k+1} &= \boldsymbol{Y}_k + \gamma\big(\boldsymbol{\mathcal{W}}\boldsymbol{U}_{k+1} - \boldsymbol{Z}_{k+1}\big).
\end{aligned}$$

The above iteration can be equivalently written as

$$\begin{aligned}
\boldsymbol{Z}_{k+1} &= \arg\min_{\boldsymbol{Z}} \ \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U}_k - \boldsymbol{Z} + \boldsymbol{Y}_k/\gamma\|^2,\\
\boldsymbol{U}_{k+1} &= \arg\min_{\boldsymbol{U}} \ \tfrac{1}{2}\|\boldsymbol{M}\odot(\boldsymbol{U}-\boldsymbol{X})\|_{q,G}^{q} + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U} - \boldsymbol{Z}_{k+1} + \boldsymbol{Y}_k/\gamma\|^2,\\
\boldsymbol{Y}_{k+1} &= \boldsymbol{Y}_k + \gamma\big(\boldsymbol{\mathcal{W}}\boldsymbol{U}_{k+1} - \boldsymbol{Z}_{k+1}\big).
\end{aligned}$$

If we further define

$$\boldsymbol{V}_{k+1} = \boldsymbol{Y}_k - \gamma\boldsymbol{Z}_{k+1}.$$

The above iteration can be reformulated as

$$\begin{aligned}
\boldsymbol{Z}_{k+1} &= \arg\min_{\boldsymbol{Z}} \ \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{\gamma}{2}\|\boldsymbol{Z} - \boldsymbol{\mathcal{W}}\boldsymbol{U}_k - \boldsymbol{Y}_k/\gamma\|^2,\\
&= \arg\min_{\boldsymbol{Z}} \ \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{\gamma}{2}\|\boldsymbol{Z} - (2\boldsymbol{Y}_k - \boldsymbol{V}_k)/\gamma\|^2,\\
\boldsymbol{V}_{k+1} &= \boldsymbol{Y}_k - \gamma\boldsymbol{Z}_{k+1},\\
\boldsymbol{U}_{k+1} &= \arg\min_{\boldsymbol{U}} \ \tfrac{1}{2}\|\boldsymbol{M}\odot(\boldsymbol{U}-\boldsymbol{X})\|_{q,G}^{q} + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U} + \boldsymbol{V}_{k+1}/\gamma\|^2,\\
\boldsymbol{Y}_{k+1} &= \boldsymbol{Y}_k + \gamma\big(\boldsymbol{\mathcal{W}}\boldsymbol{U}_{k+1} - \boldsymbol{Z}_{k+1}\big)\\
&= \boldsymbol{V}_{k+1} + \gamma\boldsymbol{\mathcal{W}}\boldsymbol{U}_{k+1}.
\end{aligned}$$

In this paper, we consider the inertial ADMM motivated by Alvarez & Attouch (2001), whose iteration is provided below:

$$\begin{aligned}
\boldsymbol{Z}_{k+1} &= \arg\min_{\boldsymbol{Z}} \ \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{\gamma}{2}\|\boldsymbol{Z} - (2\boldsymbol{Y}_k - \widetilde{\boldsymbol{V}}_k)/\gamma\|^2,\\
\boldsymbol{V}_{k+1} &= \boldsymbol{Y}_k - \gamma\boldsymbol{Z}_{k+1},\\
\widetilde{\boldsymbol{V}}_{k+1} &= \boldsymbol{V}_{k+1} + a_k(\boldsymbol{V}_{k+1} - \boldsymbol{V}_k),\\
\boldsymbol{U}_{k+1} &= \arg\min_{\boldsymbol{U}} \ \tfrac{1}{2}\|\boldsymbol{M}\odot(\boldsymbol{U}-\boldsymbol{X})\|_{q,G}^{q} + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{U} + \widetilde{\boldsymbol{V}}_{k+1}/\gamma\|^2,\\
\boldsymbol{Y}_{k+1} &= \boldsymbol{Y}_k + \gamma\big(\boldsymbol{\mathcal{W}}\boldsymbol{U}_{k+1} - \boldsymbol{Z}_{k+1}\big)\\
&= \widetilde{\boldsymbol{V}}_{k+1} + \gamma\boldsymbol{\mathcal{W}}\boldsymbol{U}_{k+1}.
\end{aligned}$$

In general, we have $a_k \in [0,1]$. When the problem is convex, the convergence can be guaranteed choosing $a_k \in [0, 1/3[$ (Boţ & Csetnek, 2014; Boţ et al., 2015).

B.2. Solution to Subproblem of $q = 1$

When $q = 1$, let $\boldsymbol{Q} = \boldsymbol{U} - \boldsymbol{X}$, and consider

$$\min_{\boldsymbol{Y}} \ \tfrac{1}{2}\|\boldsymbol{M} \odot \boldsymbol{Y}\|_{1,G} + \tfrac{\gamma}{2}\|\boldsymbol{\mathcal{W}}\boldsymbol{Y} + \boldsymbol{\mathcal{W}}\boldsymbol{X} + \widetilde{\boldsymbol{V}}_{k+1}/\gamma\|^2.$$

The optimality condition yields

$$\begin{aligned}
&\frac{1}{2}\boldsymbol{M} \odot \partial\|\boldsymbol{M} \odot \boldsymbol{Q}\|_{1,G} + \gamma\boldsymbol{\mathcal{W}}^T(\boldsymbol{\mathcal{W}}\boldsymbol{Q} + \boldsymbol{\mathcal{W}}\boldsymbol{X} + \widetilde{\boldsymbol{V}}_{k+1}/\gamma) \\
\Longleftrightarrow\ &\frac{1}{2}\boldsymbol{M} \odot \partial\|\boldsymbol{M} \odot \boldsymbol{Q}\|_{1,G} + \gamma(\boldsymbol{Q} + \boldsymbol{X} + \boldsymbol{\mathcal{W}}^T\widetilde{\boldsymbol{V}}_{k+1}/\gamma) \\
\Longleftrightarrow\ &\tfrac{1}{2}\|\boldsymbol{M} \odot \boldsymbol{Q}\|_{1,G} + \tfrac{\gamma}{2}\|\boldsymbol{Q} + \boldsymbol{X} + \boldsymbol{\mathcal{W}}^T\widetilde{\boldsymbol{V}}_{k+1}/\gamma\|^2.
\end{aligned}$$

From above we have that $\boldsymbol{Q}_{k+1}$ is the soft-thresholding of $-\boldsymbol{X} - \boldsymbol{\mathcal{W}}^T\widetilde{\boldsymbol{V}}_{k+1}/\gamma$ and

$$\boldsymbol{U}_{k+1} = \boldsymbol{Q}_{k+1} + \boldsymbol{X}.$$

C. Experiments

This section provides more details on the experiments conducted in this work.

C.1. Dataset

Table 3 documents key descriptive statistics of the five datasets for the node classification tasks. We make this particular selection to include the most classic citation network (**Cora, Citesser, and Pubmed** (Yang et al., 2016)), a dataset with (relatively) dense edge connection (**wiki-cs** (Mernyei & Cangea, 2020)), and a dataset with (relatively) high dimension of feature attributes (**coauthor-cs** (Shchur et al., 2018)).

Table 3. Summary of the datasets for node classification tasks.

	Cora	**Citeseer**	**Pubmed**	**wiki-cs**	**coauthor-cs**
# Nodes	2,708	3,327	19,717	11,701	18,333
# Edges	5,429	4,732	44,338	216,123	100,227
# Features	1,433	3,703	500	300	6,805
# Classes	7	6	3	10	15
# Training Nodes	140	120	60	580	300
# Validation Nodes	500	500	500	1769	200
# Test Nodes	1,000	1,000	1,000	5847	1000
Label Rate	0.052	0.036	0.003	0.050	0.016

C.2. Poison Preparation

We detail here the pre-processing we conduct on the two types of graph poison methods.

Injection We add the injection noise following a similar strategy by Ding et al. (2019). A certain number of targeted nodes are randomly selected from the graph and ready to change their attributes without perturbing the edge connectivity. For each selected node i, we randomly pick another k nodes from the graph and select the node j whose attributes deviate the most from node i among the k nodes by maximizing the Euclidean distance $\|x_i - x_j\|_2$. Then, we substitute the attributes x_i of node i with x_j. In this work, we set the value of k to 100 for small datasets such as CORA and CITESEER, and k to 500 for relative larger datasets such as PUBMED, COAUTHOR-CS and WIKI-CS.

Meta-attack The method meta-attack in perturbing the attributes is achieved by (Zügner & Günnemann, 2019). Although meta-attack mainly perturbs the edge connectivity in (Zügner & Günnemann, 2019), we take the technique to create local noise on the graph by corrupting a small amount of attributes in the feature matrix of a graph. In this work, the amount of perturbation varies for different graphs to obtain a noticeable attack effect.

C.3. Model Preparation

We disclose the full details of all the models examined in the experiments, including the access of model implementation, and their tuning space. All the experiments are conducted with PyTorch on NVIDIA ® Tesla A100 GPU with 6,912 CUDA cores and 80GB HBM2 mounted on an HPC cluster. All benchmark datasets are publicly available in the PyTorch Geometry library.

C.3.1. AVAILABILITY OF MODEL IMPLEMENTATION

We have uploaded our model to `https://github.com/bzho3923/MAGnet/tree/main/submission`. In addition, we take the official implementation of the baseline models from the repository:

- GNNGUARD: `https://github.com/mims-harvard/GNNGuard`
- ELASTICGNN: `https://github.com/lxiaorui/ElasticGNN`

- APPNP: `https://github.com/klicperajo/ppnp`

For the graph attack module METTACK, we follow the package implementation of DeepRobust (Li et al., 2021), which is available at `https://github.com/DSE-MSU/DeepRobust`.

C.3.2. TUNING SPACE

We now reveal the tuning space of the implemented models as the guarantee of reproducible results and fair comparison. Both our proposed MAGNET and the baseline models are optimized to their best performance by tuning:

Table 4. Hyperparameter searching space for node classification.

Hyperparameters	CORA	CITESEER	PUBMED	COAUTHOR-CS	WIKI-CS
Learning rate	5e-3	5e-3	5e-3	5e-3	5e-3
Weight decay (L_2)	5e-3	5e-3	1e-3	1e-3	1e-3
Hidden size	128	128	128	128	128
Dropout ratio	0.5	0.5	0.5	0.5	0.5
Epochs	100	100	100	100	100

C.4. Percentage Performance Improvement in Node Classification

This section supplements the percentage improvement of node classification tasks mentioned in the main text and in Table 1 and Table 2. The relative score of improvements is calculated by

$$\text{Relative Score} = \frac{S - S_{\text{corrupted}}}{S_{\text{clean}} - S_{\text{corrupted}}},$$

where S denotes the current score of the performed model, and S_{clean} and $S_{\text{corrupted}}$ are the accuracy score of GCN on the clean dataset and corrupted dataset, respectively. We use the relative score to highlight the performance in Table 1 and Table 2. The precise relative scores are reported in Table 5 and Table 6, respectively.

Table 5. Improvement percentage of average performance for node classification with non-targeted local corruption

	CORA		CITESEER		PUBMED		COAUTHOR-CS		WIKI-CS	
Module	absolute	relative	absolute	relative	absolute	relative	absolute	relative	absolute	relative
APPNP	−0.95%	−5.44%	4.27%	17.34%	1.49%	8.92%	−13.68%	−148.29%	−13.62%	−73.15%
GNNGUARD	−10.36%	−58.98%	−4.58%	−17.38%	−1.20%	7.16%	−2.11%	−22.37%	0.00%	0.00%
ELASTICGNN	12.47%	71.00%	12.21%	49.54%	5.23%	31.27%	−3.03%	−32.89%	−1.93%	−10.34%
MAGNET-one (ours)	9.78%	55.68%	2.85%	11.31%	1.89%	10.80%	1.98%	20.95%	8.24%	44.25%
MAGNET-gae (ours)	11.13%	63.34%	12.52%	50.81%	11.40%	68.20%	4.96%	52.57%	10.64%	57.14%
MAGNET-true	13.50%	76.85%	19.05%	72.22%	11.73%	70.14%	8.28%	87.66%	15.37%	82.59%

Table 6. Improvement percentage of average performance for node classification with targeted local corruption

	CORA		CITESEER		PUBMED	
Module	absolute	relative	absolute	relative	absolute	relative
APPNP	−2.10%	−25.53%	0.63%	2.13%	−3.09%	−36.70%
GNNGUARD	−4.06%	−49.27%	4.19%	14.10%	−2.44%	−29.04%
ELASTICGNN	5.57%	67.53%	21.64%	72.77%	−1.28%	−15.17%
MAGNET-one (ours)	2.72%	32.96%	12.96%	43.59%	4.05%	48.12%
MAGNET-gae (ours)	5.29%	64.14%	21.84%	73.43%	7.89%	93.80%
MAGNET-true	7.74%	93.86%	21.95%	73.80%	8.62%	102.45%

We also report the absolute score of improvement in the two tables above for a direction comparison, which is

$$\text{Absolute Score} = \frac{S - S_{\text{corrupted}}}{S_{\text{corrupted}}}.$$

C.5. Ablation Study

This section reports the performance of the ablation models on the node classification tasks. We investigate the influence of p, q in different datasets of different types of local corruption. The particular performance comparison under anomaly injection and mettack are listed in Table 7 and Table 8, respectively.

Table 7. Average performance for ABLATION study on node classification with two layers of GCN. (injection)

Module			Dataset				
mask	reg (p)	loss (q)	CORA	CITESEER	PUBMED	COAUTHOR-CS	WIKI-CS
N/A	$L_2^\star$	L_2	69.12±0.57	57.58±0.71	67.69±0.23	82.41±0.40	65.44±0.23
ONE	L_0	L_1	58.36±0.89	57.30±0.74	65.77±1.03	82.17±0.41	64.58±0.21
		L_2	68.74±0.22	54.07±1.23	58.52±1.02	80.63±0.59	63.63±0.27
	L_1	L_1	69.12±0.57	57.58±0.71	67.69±0.40	82.17±0.41	64.63±0.18
		L_2	**75.88±0.42**	**59.22±0.34**	**68.97±0.21**	**84.04±0.56**	**70.83±0.29**
GAE	L_0	L_1	68.42±1.15	54.38±0.54	67.74±0.71	83.95±0.52	61.96±0.16
		L_2	66.34±0.81	56.29±1.18	59.15±0.85	83.88±0.55	64.67±0.29
	L_1	L_1	72.76±0.40	63.60±0.66	**75.41±0.35**	86.02±0.59	**72.40±0.21**
		L_2	**76.81±0.98**	**64.79±0.14**	71.13±0.25	**86.50±0.37**	60.08±0.39
TRUE	L_0	L_1	77.15±0.74	68.14±0.85	74.40±0.56	88.14±0.46	72.42±0.29
		L_2	76.44±0.59	65.02±0.97	68.12±1.24	87.01±0.24	71.51±0.20
	L_1	L_1	**78.48±0.67**	**68.55±0.74**	**75.63±0.56**	**89.23±0.37**	**75.50±0.20**
		L_2	77.57±0.92	64.29±0.19	75.19±0.18	86.72±0.31	74.44±0.23

Table 8. Average performance for ABLATION study on node classification with two layers of GCN. (Mettack)

Module			Dataset		
mask	reg (p)	loss (q)	**Cora**	**Citeseer**	**PubMed**
N/A	$L_2^\star$	L_2	75.07±0.64	55.32±0.64	72.88±0.30
ONE	L_0	L_1	71.21±1.12	55.38±2.12	71.52±0.43
		L_2	74.46±0.60	57.75±2.23	59.76±1.91
	L_1	L_1	75.07±0.64	55.32±2.22	72.88±0.65
		L_2	**77.11±0.45**	**62.49±1.70**	**75.83±0.35**
GAE	L_0	L_1	77.42±1.08	56.13±0.47	78.42±0.65
		L_2	77.03±0.78	55.84±1.37	70.82±0.38
	L_1	L_1	78.18±0.56	63.22±1.56	**78.63±0.32**
		L_2	**79.04±0.50**	**67.40±0.73**	74.47±0.30
TRUE	L_0	L_1	**80.88±0.37**	**67.46±0.95**	**79.16±0.41**
		L_2	80.57±0.51	67.21±1.63	71.90±0.41
	L_1	L_1	79.99±0.45	65.50±0.81	79.14±0.32
		L_2	77.16±0.66	67.33±0.49	75.04±0.32

C.6. GAE visualization

Figure 5-Figure 7 visualize the sparse mask matrix of the five datasets with anomaly injection. We are interested in the recall of the mask matrix that exposes the quality of the mask matrix approximation. As a result, we print the conditional mask matrix that marks the positions of approximated index that are at the true anomalous spots. Note that we did not visualize

the mask matrix from the datasets under mettack perturbation, as such attack focuses major poisonings in a minority of node entities. When making visualizations on such matrices, they are nothing more than a few horizontal lines.

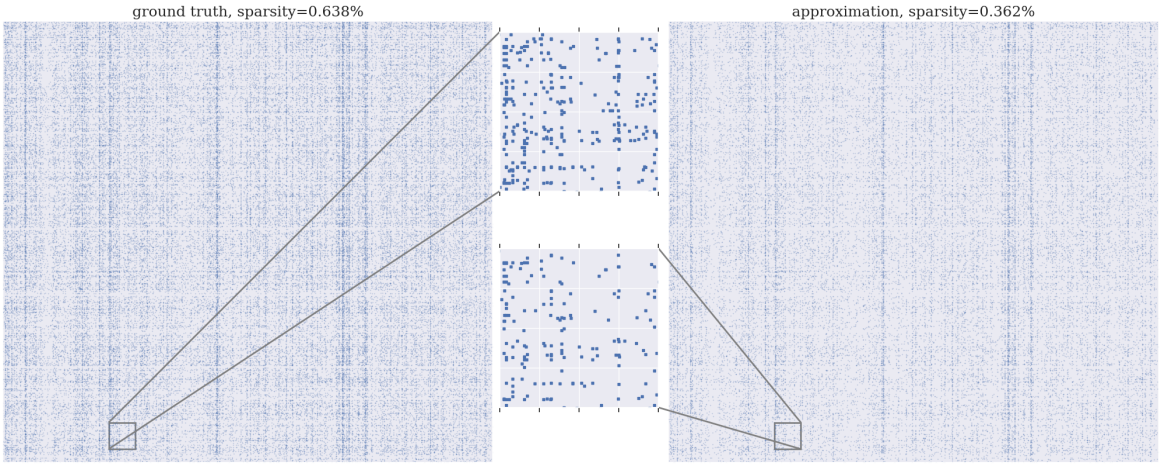

Figure 5. citeseer

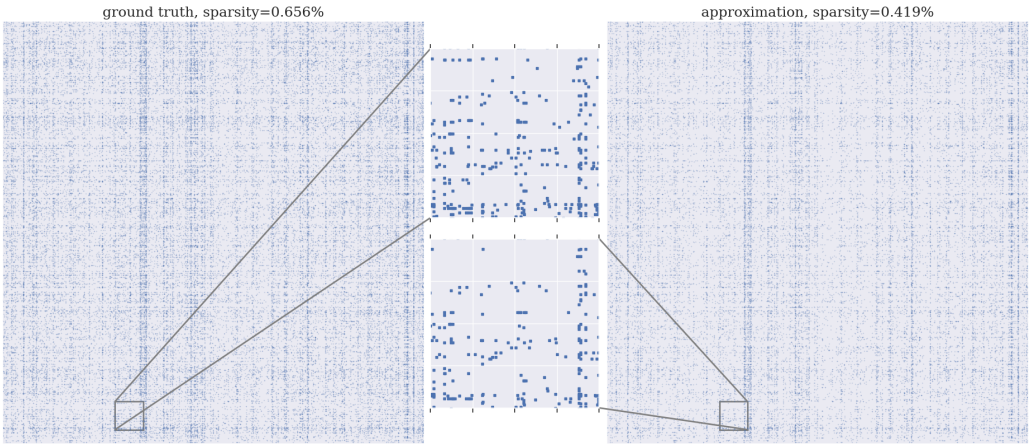

Figure 6. first 3000 rows and columns only. The full matrix is too larger to display (18000x7000)

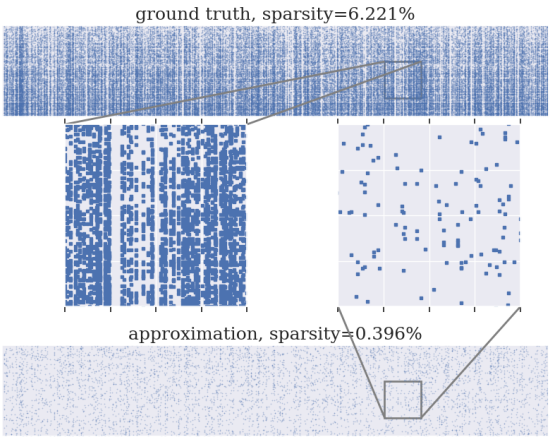

Figure 7. pubmed: first 3000 nodes; 500 features; threshold=0.005

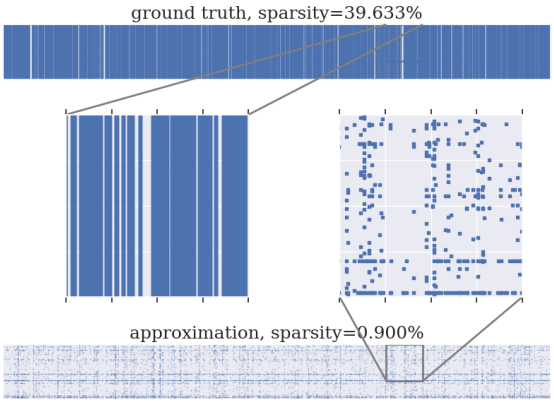

Figure 8. Wikics: first 3000 nodes; 300 features; threshold=0.05

C.7. Computational Speed

We show the inertial effect of ADMM by the counting the computational speed of our model on **Cora**. We specifically consider two settings with and without the boosting effect, i.e., γ in (8) equals to 0 or not. Below we report the running time of a complete update process udnder certain convergent condition ϵ , i.e., $\|\boldsymbol{U}_{k+1} - \boldsymbol{U}_k\| < \epsilon$. The speed is counted in seconds.

Table 9. Hyperparameter searching space for node classification.

	w/ boost(sec)	**w/o boost**(sec)
$p = 1, q = 2$	49.38	54.16
$p = 1, q = 1$	90.44	94.84
$p = 0, q = 2$	12.43	13.05
$p = 0, q = 1$	4.37	5.43