Robust Graph Representation Learning for Local Corruption Recovery

Bingxin Zhou*
Shanghai Jiao Tong University
Shanghai, China
The University of Sydney
NSW, Australia

Yuanhong Jiang*
Yu Guang Wang, Jingwei Liang
Shanghai Jiao Tong University
Shanghai, China

Junbin Gao
The University of Sydney
NSW, Australia

Shirui Pan
Griffith University
QLD, Australia

Xiaoqun Zhang
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

The performance of graph representation learning is affected by the quality of graph input. While existing research usually pursues a globally smoothed graph embedding, we believe the rarely observed anomalies are as well harmful to an accurate prediction. This work establishes a graph learning scheme that automatically detects (locally) corrupted feature attributes and recovers robust embedding for prediction tasks. The detection operation leverages a graph autoencoder, which 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 sparsity promoting regularizer. The optimizer approaches a new embedding that is sparse in the framelet domain and conditionally close to input observations. Extensive experiments are provided to validate our proposed model can recover a robust graph representation from black-box poisoning and achieve excellent performance.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Neural networks**.

KEYWORDS

graph neural networks, constrained optimization, spectral transforms, graph denoising

ACM Reference Format:
Bingxin Zhou, Yuanhong Jiang, Yu Guang Wang, Jingwei Liang, Junbin Gao, Shirui Pan, and Xiaoqun Zhang. 2023. Robust Graph Representation Learning for Local Corruption Recovery. In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3543507.3583399

*Both authors contributed equally to this research.

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583399

1 INTRODUCTION

Graph neural networks (GNNs) have received tremendous success in the past few years [7, 52, 59, 63]. Graphs, as the input of GNNs, record useful features and structural information. They exist widely in many fields, such as biomedical science [2], social networks [20], and recommender systems [51].

Similar to other types of real-world data, 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 on their neighborhoods, resulting in a misleading latent feature representation for the predictor.

Existing works are aware of the harmful graph anomalies. i) Graph anomaly detection [16, 37, 44, 64] identifies the small portion of problematic nodes, such as fraudulent users in a social network graph; ii) graph defense[14, 54, 66] refines the learning manner of a classifier to provide promising predictions against potential misleading entities that hurt the model training; iii) optimization-based graph convolutions smooth out global noise, e.g., observation errors in data collection, by special regularization designs [10, 36, 62, 65]. However, the first approach detects the whole irregular node rather than individual node attributes, and it does not amend the flawed representation of the identified outlier. The second approach provides an adequate solution to protect the prediction performance, but most researches focus on graph rewiring, as edges are believed more vulnerable to adversarial attacks. The third choice, although operates on node attributes, only makes implicit optimization against local corruptions. They usually assume that feature corruptions are normally observed in all inputs.

In this paper, we propose an alternative approach by developing a '*detect-and-then-recover*' strategy to protect graph representation learning from a small portion of hidden corruptions in the input node attributes. In particular, an unsupervised encoder module first detects suspicious attributes and assigns a mask matrix to expose their positions. The detector requires no prior knowledge of the distribution of the anomalous attributes. The constructed mask guides a robust reconstruction of the initial input with sparsity promoting regularizers. To guarantee an expressive representation that

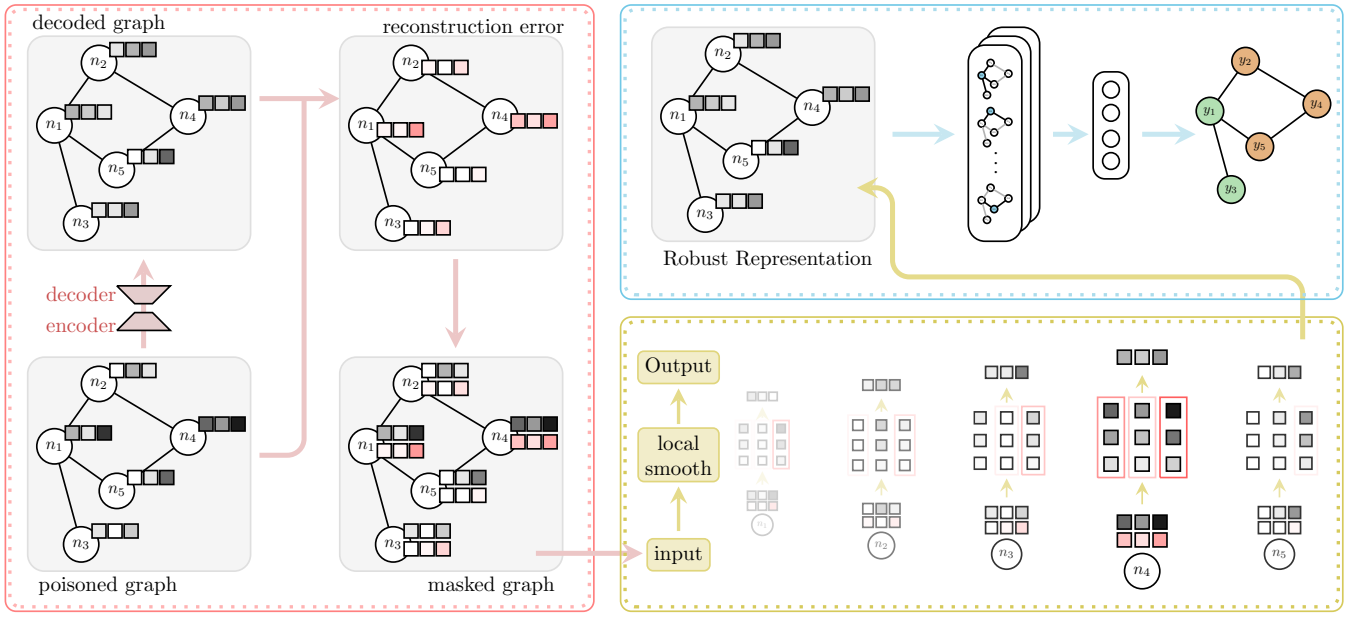


Figure 1: Illustrative architecture of the proposed MAGNET. Input feature attributes (gray) are assumed locally corrupted. The first module constructs a mask matrix (red) as the indices of locally-corrupted input feature attributes (gray). They are sent to an inertia ADMM optimizer (yellow) to iterate a robust feature representation, which is functioned as hidden embeddings by typical graph convolutional layers. It can be encoded by other convolutions or sent to a predictor (blue).

simultaneously reconstructs irregular attributes, preserves small-energy patterns, and eliminates global noise, the regularization is operated on multi-scale framelet coefficients [17, 60]. Meanwhile, the posterior mask guarantees that the signal recovery is predominantly conducted at the essential (anomalous) spots. The optimized embedding is updated iteratively, which acts similarly to a graph convolutional layer that constantly smooths the hidden feature representation for GNN prediction tasks.

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 of unsupervised mask construction, localized robust optimization, and graph representation learning. MAGNET detects local corruptions in an unsupervised manner and approximates a robust graph representation. As indicated above, the alternating direction method of multipliers (ADMM) algorithm [22] is adopted in our optimization procedure.

2 PROBLEM FORMULATION

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ to denote an undirected attributed graph with $n = |\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges, where the latter is usually 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 $\bar{\boldsymbol{U}}$, which reads $\boldsymbol{X} = \bar{\boldsymbol{U}} + \boldsymbol{E}_1 + \boldsymbol{E}_2$ 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 takes a small chance of existence so it would not be observed from a large set of nodes or node attributes. We hereby make the following assumptions on the properties of $\bar{\boldsymbol{U}}$, $\boldsymbol{E}_1$ and $\boldsymbol{E}_2$:

(a) The global noise $\boldsymbol{E}_1$ follows some distribution $\mathcal{D}$ with an expectation of 0;
(b) The outliers take a small portion of the entire graph, or the $\boldsymbol{E}_2$ is a sparse matrix; and
(c) The observations $\boldsymbol{X}$ are close to $\bar{\boldsymbol{U}}$ at regular locations.

The associated objective function to estimate $\bar{\boldsymbol{U}}$ reads

$$\min_{\boldsymbol{U}} \ \alpha \mathrm{Reg}(\boldsymbol{U}) + \mathrm{Loss}(\boldsymbol{E}_2) \quad \text{s.t. } \boldsymbol{X}|_{\boldsymbol{M}} = (\boldsymbol{U} + \boldsymbol{E}_1 + \boldsymbol{E}_2)|_{\boldsymbol{M}}, \tag{1}$$

where α is tunable, and $\boldsymbol{M}$ is a mask matrix of outliers indices. The $\mathrm{Reg}(\cdot)$ and $\mathrm{Loss}(\cdot)$ denote the regularizer and loss function satisfying assumptions (a) and (b), respectively.

- **Choice of** $\mathrm{Loss}(\boldsymbol{E}_2)$**:** The loss function (1) finds a robust approximation $\boldsymbol{U}$ from the input feature $\boldsymbol{X}$. As required by assumption (b), 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 which counts the number of nonzero entries in $\boldsymbol{E}_2$. Unfortunately, the ℓ_0 constrained problem is NP-hard. In substitute, the ℓ_1-norm allows feasible solution for (1) that promotes a good sparsity measure for $\boldsymbol{E}_2$ [11]. Also [32, 43] implemented residual analysis-based *anomaly detection* methods for labeling suspicious outliers.
- **Choice of** $\mathrm{Reg}(\boldsymbol{U})$**:** The regularization on $\boldsymbol{U}$ is a penalty complementary to the loss function, which controls the noise level of $\boldsymbol{X}$ by the smoothness of $\boldsymbol{U}$, and it is usually quantified by some energy measurements. For instance, GCN [29] 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 smooth solution to $\boldsymbol{U}$ [40]. Such a *graph smoothing effect* [10, 36, 65] has been implemented in many spatial-based graph convolution designs [30, 50, 55]. Alternatively, spectral-based methods regularize the graph signal in a transformed domain by $\boldsymbol{L}$. For example, Dong [17] and Zhou *et al.* [62] minimized the ℓ_1-norm total variation of framelet coefficients, and Mahmood *et al.* [38] considered a similar regularization in the wavelet domain. As the total variation reflects redundant local fluctuations, a solution to minimize it is believed to remove noise and preserve local patterns at the same time.

- **Restriction on $\boldsymbol{E}_1 + \boldsymbol{E}_2$:** Compared to the main ingredients of the objective function, the treatment of the fidelity constraint is rather trivial when the outliers $\boldsymbol{E}_2$ do not exist. Usually a regularizer is adopted to minimize the difference between $\boldsymbol{X}$ and $\bar{\boldsymbol{U}} + \boldsymbol{E}_1$, i.e., $\boldsymbol{X} - \boldsymbol{U}$. However, when $\boldsymbol{E}_2$ is assumed to exist at a small proportion, it is undesired to force $\boldsymbol{U}$ to approximate $\boldsymbol{X}$ especially on anomalous locations. Instead, a conditional approximation is placed with the index matrix $\boldsymbol{M}$, in which case only attributes at regular positions are required aligned. The target is then $\min \mathrm{Reg}\left(\boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{U})\right)$ with some penalty $\mathrm{Reg}(\cdot)$, and it can be appended to the main objective by

$$\min_{\boldsymbol{U}} \alpha \mathrm{Reg}(\boldsymbol{U}) + \mathrm{Loss}\left(\boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{U})\right). \tag{2}$$

The above optimization for graph signal recovery is well-defined. Nevertheless, it has three issues in practice. First, a direct minimization of the Dirichlet energy in the spatial domain usually falls into the pitfall of over-smoothing, where the recovered graph loses expressivity drastically [4]. On the other hand, spectral transforms can be sophisticated and time-consuming, which is generally circumvented by the majority of studies. Second, fixing the ℓ_1 or ℓ_2 norm can be too restricted for an arbitrary dataset or application, which could recover a less robust graph representation that affects the prediction task later on. Last but not least, attaining the mask matrix $\boldsymbol{M}$ in (2) can be nasty in practice, as the prior knowledge of $\boldsymbol{M}$ is generally inaccessible.

In order to best handle the identified problems, we propose an objective function design that combines masked ℓ_q $(1 \le q \le 2)$ reconstruction error term and ℓ_p $(0 \le p \le 1)$ regularization, i.e.,

$$\min_{\boldsymbol{U}} \|\boldsymbol{\nu}\boldsymbol{\mathcal{W}}\boldsymbol{U}\|_{p,G} + \frac{1}{2}\|\boldsymbol{M} \odot (\boldsymbol{U} - \boldsymbol{X})\|_{q,G}^{q}, \tag{3}$$

which is convex at $p = 1$ and non-convex when $0 \le p < 1$, regardless of q. Equation (3) enforces the sparse representation by a *graph framelet system* [17, 49, 60, 61]. Through framelet transforms, an input graph signal is decomposed into a low-pass coefficient matrix and multiple high-pass coefficient matrices. The former includes general global information about the graph, and the latter portrays the local properties of the graph at different degrees of detail.

Specifically, the decomposition is realized by a set of multi-scale and multi-level framelet decomposition operators $\boldsymbol{\mathcal{W}}$ under the framelet system, with each $\boldsymbol{\mathcal{W}}_{k,l}$ be an orthonormal basis at $(k,l) \in \{(0,J)\} \cup \{(1,1),\ldots,(1,J),\ldots(K,1),\ldots,(K,J)\}$. This decomposition operator forms the spectral representation of $\boldsymbol{U}$, where its global trend is reflected in the low-pass coefficients $\boldsymbol{\mathcal{W}}_{0,J}\boldsymbol{U}$. Meanwhile, the high-pass coefficients $\boldsymbol{\mathcal{W}}_{k,l}\boldsymbol{U}$ records detailed information at scale k and level l $(1 \le k \le K, 1 \le l \le J)$. They reflect local patterns or noises of the signal. The coefficients at a higher level l contain more localized information with smaller energy.

In addition, we replace α in (2) with a set of hyper-parameters $\boldsymbol{\nu}$ to normalize the high-frequency framelet coefficients $\boldsymbol{\mathcal{W}}\boldsymbol{U}$ of the approximated $\boldsymbol{U}$ in different scales. For example, an initialization ν_0 defines $\boldsymbol{\nu}_{0,J} = 0$ for the low-pass coefficients and $\boldsymbol{\nu}_{k,l} = 4^{-l-1}\nu_0$ for high-pass coefficients.

The *signal reconstruction error* $\|\boldsymbol{M} \odot (\boldsymbol{U} - \boldsymbol{X})\|_q$ guarantees the optimal representation $\boldsymbol{U}$ to be conditionally close to the masked feature matrix $\boldsymbol{X}$ of the given graph. We generalize the conventional ℓ_2 error to a ℓ_q penalty $(1 \le q \le 2)$. Meanwhile, the ℓ_p regularization measures the graph total variation [46] in the framelet domain, where pursuing a small total variation recovers the graph signal from corrupted attribute noises. When p approaches 1 from 2, the framelet representation $\boldsymbol{\mathcal{W}}\boldsymbol{U}$ gains increasing sparsity.

We treat p, q as tunable parameters to allow sufficient freedom in our design for selecting the most qualified optimization model that fits the data and training requirements. Furthermore, we replace the ordinary Euclidean ℓ_k-norm with a graph ℓ_k-norm, denoted as $\ell_{k,G}$, to assign higher penalties to influential high-degree 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 in (1), (3) made three adjustments to tackle the identified issues. First, minimizing the first component smooths out $\boldsymbol{E}_1$ under the assumption (a) 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 the values of p, q to adaptively restrict the sparsity of recovered graph with $p \in [1,2], q \in [0,1]$. 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 optimizing (3), which introduces a two-stage optimization problem. However, this approach blows up the difficulty of solving the existing problem and the mask region could be too complicated to reveal. Instead, we consider an unsupervised GNN scheme to automate the discovery of anomalous positions. We approximate the anomaly score of $\boldsymbol{X}$ with a classic graph anomaly detection scheme that looks for the reconstruction error between the observed and reconstructed matrices from neural networks.

3 MASK MATRIX GENERATION WITH GRAPH ANOMALY DETECTION

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* [37], which are 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 between each node from the representation of its local community.

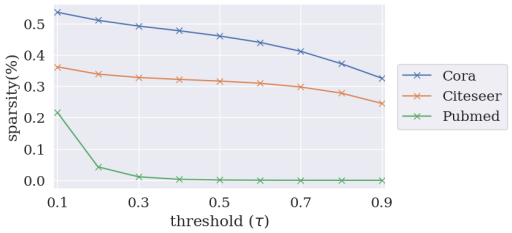


Figure 2: The sparsity of GAE-approximated $1-\boldsymbol{M}$ is directly controlled by the threshold τ.

3.1 Graph Autoencoder

Autoencoder is a powerful tool for reconstructing corrupted objects [1, 31]. GAE [28] is a GCN-based autoencoder, which has become a frequent solution in graph anomaly detection [16, 44, 64]. 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 $\boldsymbol{Z} = \mathrm{GCN}(\boldsymbol{X}, \boldsymbol{A})$. When recovering the graph attributes from its hidden embedding, a feed-forward network is trained to find $\boldsymbol{X}' = \mathrm{FFN}(\boldsymbol{Z})$, a smoothed version of $\boldsymbol{X}$ that is believed to remove essential noise or minor corruptions of the raw input $\boldsymbol{X}$. In [16, 43] for graph anomaly detection, the learning objective of the neural network is to best reconstruct the graph attribute $\boldsymbol{X}'$, which loss function is the weighted average of the squared *approximation errors*, i.e., $\mathcal{L} = \|\boldsymbol{X} - \boldsymbol{X}'\|_2$.

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 approximation error is thus an appropriate indicator for diagnosing such divergence. For a particular node v_i, we calculate $\mathrm{score}(v_i) = \|\boldsymbol{x}_i - \boldsymbol{x}'_i\|_2$.

While the outliers exist rarely, we adopt ℓ_2 penalty for approximating the approximation error (and anomalous scores) for two reasons. First, reconstructing the noisy signal by an autoencoder aims at finding an overall optimal representation without extreme errors. In this sense, the MSE loss is more sensitive to large errors than the MAE loss. Meanwhile, as we do not assume prior knowledge of the distribution of data noise, Gaussian noise (in alignment with ℓ_2 penalty) is more general among other choices (such as the Laplacian distribution with ℓ_1 penalty). On the other hand, the sparsity requirement is satisfied in the later stages of our implementation. As introduced in the next section, the mask matrix construction is instructed by an MAE-loss, which favors a sparse distribution of the local noise in the attribute matrix.

3.2 Mask Matrix Generation

Our primary focus is on graph recovery without explicitly considering the disordered node connections or the adjacency matrix. Also, unlike conventional anomaly detection tasks that locate the problematic nodes, we aim at generating a mask matrix of size $n\times d$ that identifies the corrupted node attributes, and it is defined as

$$\boldsymbol{M} = 1 - \mathrm{threshold}(\|\boldsymbol{X} - \boldsymbol{X}'\|_1, \tau). \tag{4}$$

The established $\boldsymbol{M}$ approximates the element-wise errors by differentiating the raw and the recovered feature matrices from a trained graph autoencoder. Optionally, $\boldsymbol{M}$ can be binarized with a tunable threshold $\tau \in (0, 1)$. Consequently, $\boldsymbol{M}_{ij} = 1$ suggests assigning a new value to the jth feature of node i to avoid potential corruption.

Under assumption (b) that $\boldsymbol{E}_2$ exists rarely in node attributes, the majority of the mask matrix should be 1, i.e., $1-\boldsymbol{M}$ is a sparse matrix. This requirement is guaranteed by both the autoencoder algorithm and the thresholding function. Because the encoder smoothes $\boldsymbol{U}$ to approach $\boldsymbol{X}$ and the magnitude of $\boldsymbol{E}_1$ is assumed smaller than $\boldsymbol{E}_2$'s, the difference between $\boldsymbol{X}$ and $\boldsymbol{X}'$ mainly comes from $\boldsymbol{E}_2$. On top of that, the tunable τ controls the sparsity of $\boldsymbol{M}$. Figure 2 provides an empirical example with three citation networks that are locally perturbed by the attribute injection [16]. A large τ reduces the number of non-zero elements in $1-\boldsymbol{M}$ from a considerably sparse level to a lower degree.

4 OPTIMIZATION ALGORITHM

This section describes the optimization algorithm to solve (3). As described in (6), it is an inertial version of the celebrated alternating direction method of multipliers (ADMM) [22].

4.1 An inertial ADMM

Denote $\boldsymbol{Z} = \boldsymbol{\curlyvee}\boldsymbol{W}\boldsymbol{U}$, then we can rewrite (3) as

$$\min_{\boldsymbol{U},\boldsymbol{Z}} \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{1}{2}\|\boldsymbol{M}\odot(\boldsymbol{U}-\boldsymbol{X})\|_{q,G}^q, \quad \text{s.t.} \quad \boldsymbol{Z} = \boldsymbol{\curlyvee}\boldsymbol{W}\boldsymbol{U}. \tag{5}$$

This forms a standard formulation of problems that can be solved by ADMM. The associated augmented Lagrangian to (5) reads

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

where $\gamma > 0$. We consider the inertial ADMM motivated by Alvarez & Attouch [3] to secure an efficient solver, which defines

$$\begin{aligned}\boldsymbol{Z}_{k+1} &= \arg\min_{\boldsymbol{Z}} \ \|\boldsymbol{\nu}\boldsymbol{Z}\|_{p,G} + \tfrac{\gamma}{2}\|\boldsymbol{Z} - (2\boldsymbol{Y}_k - \widetilde{\boldsymbol{V}}_k)/\gamma\|^2, \\ \boldsymbol{V}_{k+1} &= \boldsymbol{Y}_k - \gamma\boldsymbol{Z}_{k+1}, \\ \widetilde{\boldsymbol{V}}_{k+1} &= \boldsymbol{V}_{k+1} + a_k(\boldsymbol{V}_{k+1} - \boldsymbol{V}_k), \\ \boldsymbol{U}_{k+1} &= \arg\min_{\boldsymbol{U}} \ \tfrac{1}{2}\|\boldsymbol{M}\odot(\boldsymbol{U}-\boldsymbol{X})\|_{q,G}^q + \tfrac{\gamma}{2}\|\boldsymbol{\curlyvee}\boldsymbol{W}\boldsymbol{U} + \widetilde{\boldsymbol{V}}_{k+1}/\gamma\|^2, \\ \boldsymbol{Y}_{k+1} &= \widetilde{\boldsymbol{V}}_{k+1} + \gamma\boldsymbol{\curlyvee}\boldsymbol{W}\boldsymbol{U}_{k+1},\end{aligned} \tag{6}$$

where $\widetilde{\boldsymbol{V}}_{k+1}$ is the inertial term and a_k is the inertial parameter. We refer to [5, 6], and the references therein for the convergence analysis of the above inertial ADMM scheme.

4.2 Subproblems

The solutions to the subproblems of $\boldsymbol{Z}_{k+1}$ and $\boldsymbol{U}_{k+1}$ in (6) depend on p and q, respectively. Here we enumerate the special cases of $p = \{0, 1\}$ and $q = \{1, 2\}$. Other choices of p, such as $p = \frac{1}{2}$, are discussed in [56] and the references therein.

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

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

When $p = 0$, hard-thresholding is applied with $H_\alpha(x) = \begin{cases} x & |x| > \alpha \\ 0 & |x| \le \alpha \end{cases}$.

Table 1: Average performance for node classification over 10 repetitions.

Module	attribute injection								meta attack		
	Cora	Citeseer	PubMed	Coauthor-CS	Wiki-CS	Wisconsin	Texas	OGB-arxiv	Cora	Citeseer	PubMed
clean	81.26±0.65	71.77±0.29	79.01±0.44	90.19±0.48	77.62±0.26	56.47±5.26	65.14±1.46	71.10±0.21	81.26±0.65	71.77±0.29	79.01±0.44
GCN	69.06±0.74	57.58±0.71	67.69±0.40	82.41±0.23	65.44±0.23	48.24±3.19	58.92±2.02	68.42±0.15	75.07±0.64	55.32±2.22	72.88±0.30
APPNP	68.46±0.81	60.04±0.59	68.70±0.47	71.14±0.54	56.53±0.72	61.76±5.21	59.46±0.43	OOM	73.49±0.59	55.67±0.28	70.63±1.07
GNNGuard	61.96±0.30	54.94±1.00	68.50±0.38	80.67±0.88	65.69±0.32	46.86±1.06	59.19±0.81	65.75±0.32	72.02±0.61	57.64±1.31	71.10±0.32
ElasticGNN	77.74±0.79	64.61±0.85	71.23±0.21	79.91±1.39	64.18±0.53	53.33±2.45	59.77±3.24	41.34±0.38	**79.25±0.50**	67.29±1.17	71.95±0.52
AirGNN	76.22±3.75	62.14±0.82	74.73±0.43	80.18±0.31	71.36±0.20	61.56±0.72	59.46±1.24	52.32±0.58	78.94±0.45	65.58±0.63	78.58±0.71
MAGNET$_{one}$	75.88±0.42	59.22±0.34	68.97±0.21	84.04±0.56	70.83±0.29	55.49±1.53	60.27±1.73	68.24±0.30	77.11±0.45	62.49±1.70	75.83±2.05
MAGNET$_{gae}$	**79.07±0.56**	**64.79±0.73**	**75.41±0.35**	**86.50±0.37**	**72.40±0.21**	**64.31±2.60**	**60.81±2.18**	**68.68±0.03**	79.04±0.50	**67.40±0.73**	**78.63±0.32**
MAGNET$_{true}$	78.48±0.67	68.55±0.74	75.63±0.56	89.23±0.40	75.50±0.20	65.69±1.57	60.54±2.16	69.57±0.23	80.88±0.37	67.46±0.95	79.16±0.41

Compared to Z_{k+1}, the update of U_{k+1} in (6) is more complicated as it involves more regularization terms. When $q = 2$, the objective is a quadratic problem. With $\mathcal{W}^\top \mathcal{W} = \mathrm{Id}$, we directly have

$$U_{k+1} = \frac{M \odot X - \mathcal{W}^\top \widetilde{V}_{k+1}}{M + \gamma},$$

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

$$U^{(i+1)} = U^{(i)} - \alpha\big(M \odot (U^{(i)} - X) + \gamma \mathcal{W}^\top (\mathcal{W} U^{(i)} + \widetilde{V}_{k+1}/\gamma)\big)$$

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

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

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

Let Q_{k+1} be the soft-thresholding of $-X - \mathcal{W}^\top \widetilde{V}_{k+1}/\gamma$ and we have

$$U_{k+1} = Q_{k+1} + X. \tag{7}$$

5 NUMERICAL EXPERIMENTS

This section validates MAGNET by a variety of experiments. We start by comparing our methods to three denoising methods on node classification tasks with two types of attribute corruptions. The next two ablation studies digest the influence of regularizers in mask generation and ADMM denoising towards the graph recovery and property prediction performance of the proposed model. To establish an intuitive understanding of the two learning modules, i.e., mask matrix approximation and ADMM optimization, their effects are demonstrated with visualizations. The implementations at https://github.com/bzho3923/MAGnet are programmed with `PyTorch-Geometric` (version 2.0.1) and `PyTorch` (version 1.7.0) and run on NVIDIA® Tesla A100 GPU with 6,912 CUDA cores and 80GB HBM2 mounted on an HPC cluster.

5.1 Experimental Protocol

5.1.1 Benchmark Preparation. We examine MAGNET on eight publicly available benchmark datasets: **Cora**, **Citeseer** and **PubMed** of the citation networks [57]; **Wiki-CS** [39] that classifies articles from Wikipedia database; **Coauthor-CS** [47] that labels the most active field of authors; **OGB-arxiv** from open graph benchmark [26] that represents the citation network between all arXiv papers in Computer Science; and **Wisconsin** and **Texas** [23] that records the web pages from computer science departments of different universities and their mutual links. In particular, the first three datasets are the most classic citation networks that are used for node-level property prediction tasks, **Wiki-CS** employs dense edge connection, **Coauthor-CS** elaborates (relatively) high dimension of feature attributes, **OGB-arxiv** is a large-scale dataset, and the last two web-page datasets are heterophilic graphs.

While 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:

- An *attribute injection* method perturbs attributes through swapping attributes of the most distinct samples in a random subgraph. The noise is added similarly to [16]. A certain number of targeted nodes are randomly selected from the input graph. For each selected node v_i, we randomly pick another k nodes from the graph and select the node v_j whose attributes deviate the most from node v_i among the k nodes by maximizing the Euclidean distance of node attributes, i.e., $\|x_i - x_j\|_2$. Then, we substitute the attributes x_i of node v_i with x_j. Specifically, we set the size of candidates $k = 100$ for small datasets, i.e., **Cora** and **Citeseer**, and $k = 500$ for relatively larger datasets, i.e., **PubMed**, **Coauthor-CS** and **Wiki-CS**.
- Secondly, *meta attack* perturbs node attributes leverages meta-learning in graph adversarial attack to poison node attributes with the meta-gradient of the loss function [66] [1]. We use the technique to create local corruptions on the underlying graph by corrupting a small portion of attributes in the graph's feature matrix, where the specific amount of perturbation varies for different datasets to obtain a noticeable attack effect. Instead of unifying the preprocessing procedures (i.e., conduct the same attack step for all the baseline methods), we unify the input graph (i.e., use the same attacked graph input) for baseline methods by fixing the surrogate model to a 2-layer GCN. The justification comes in two-fold. First, we aim to demonstrate each model's ability to recover a robust representation from given corrupted inputs. For this purpose, it is not required to make baseline-specific

[1] implemented by `DeepRobust` [34] at https://github.com/DSE-MSU/DeepRobust

Table 2: Average performance with different choices on p and q for node classification.

Module			attribute injection					meta attack		
mask	reg (p)	loss (q)	**Cora**	**Citeseer**	**PubMed**	**Coauthor-CS**	**Wiki-CS**	**Cora**	**Citeseer**	**PubMed**
N/A	$L_2^\star$	L_2	69.06±0.74	57.58±0.71	67.69±0.23	82.41±0.40	65.44±0.23	75.07±0.64	55.32±0.64	72.88±0.30
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	71.21±1.12	55.38±2.12	71.52±0.43
		L_2	68.74±0.22	54.07±1.23	58.52±1.02	80.63±0.59	63.63±0.27	74.46±0.60	57.75±2.23	59.76±1.91
	L_1	L_1	69.12±0.57	57.58±0.71	67.69±0.40	82.17±0.41	64.63±0.18	75.07±0.64	55.32±2.22	72.88±0.65
		L_2	**75.88±0.42**	**59.22±0.34**	**68.97±0.21**	**84.04±0.56**	**70.83±0.29**	**77.11±0.45**	**62.49±1.70**	**75.83±0.35**
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	77.42±1.08	56.13±0.47	78.42±0.65
		L_2	66.34±0.81	56.29±1.18	59.15±0.85	83.88±0.55	64.67±0.29	77.03±0.78	55.84±1.37	70.82±0.38
	L_1	L_1	72.76±0.40	63.60±0.66	**75.41±0.35**	86.02±0.59	**72.40±0.21**	78.18±0.56	63.22±1.56	**78.63±0.32**
		L_2	**76.81±0.98**	**64.79±0.14**	71.13±0.25	**86.50±0.37**	60.08±0.39	**79.04±0.50**	**67.40±0.73**	74.47±0.30
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	**80.88±0.37**	**67.46±0.95**	**79.16±0.41**
		L_2	76.44±0.59	65.02±0.97	68.12±1.24	87.01±0.24	71.51±0.20	80.57±0.51	67.21±1.63	71.90±0.41
	L_1	L_1	**78.48±0.67**	**68.55±0.74**	**75.63±0.56**	**89.23±0.37**	**75.50±0.20**	79.99±0.45	65.50±0.81	79.14±0.32
		L_2	77.57±0.92	64.29±0.19	75.19±0.18	86.72±0.31	74.44±0.23	77.16±0.66	67.33±0.49	75.04±0.32

attacks to maximize the effect of an adversarial attack. Meanwhile, as we do not assume prior knowledge such as "where were the perturbations from" or "how were the corruptions designed", pre-determining a corrupted graph for all baseline models corresponds to a fair comparison.

5.1.2 Training Setup. We construct the mask matrix by binary approximation errors from a GCN-based autoencoder constituting 2 encoding layers following 2 decoding layers. During the optimization, an inertial ADMM is iterated for 15 times to fastly recover the corrupted graph. The first six datasets follow the standard public split and processing rules in `PyTorch-Geometric` [21] or `OGB` [26]. For the two heterophilic datasets, we split the training, validation, and test sets following [42]. The reported average test accuracy (or AUROC for **OGB-arxiv**) is evaluated over 10 repetitions.

The models are fine-tuned within the hyper-parameter searching space defined in Table 3, where the first five hyper-parameters are universally applicable to all the models, and the last ν_0 is exclusive to MAGNET. As for model architecture, we construct two convolution layers for GCN (except for **OGB-arxiv**, where we train a 3-layer GCN), and three blocks for ELASTICGNN and APPNP where each block contains a fully-connected layer and a graph convolution. For GNNGUARD and MAGNET, we run the main layers to find the denoised representation $\boldsymbol{U}$ and send it to a 2-layer GCN for node classification. The prediction results are activated by a softmax function for label assignment.

Table 3: Hyperparameter searching space.

Hyperparameters	Searching Space
learning rate	10^{-3}, 5×10^{-3}, 10^{-4}
weight decay (L_2)	10^{-3}, 5×10^{-3}, 10^{-4}
hidden size	64, 128
dropout ratio	0.5
epochs	200, 500
ν_0	10, 100, 500
τ	0.1

5.1.3 Baseline Comparison. A complete investigation on MAGNET is conducted with three different types of mask matrix: all-ones (MAGNET$_{\text{one}}$), GAE-approximated (MAGNET$_{\text{gae}}$), and ground truth (MAGNET$_{\text{true}}$) matrices. The last case advises the upper limit of our model with the 'perfectly' approximated mask matrix. We compare our model to three popular graph smoothing models with different design philosophies: APPNP [30] [2] avoids global smoothness with residual connections; GNNGUARD [58] [3] modifies the neighbor relevance in message passing to mitigate local corruption; ELASTICGNN [36] [4] and AIRGNN [35] [5] pursues local smoothness with a mixture of ℓ_1 and ℓ_2 regularizers. We also train a 2-layer GCN [29] as the baseline method, which smooths graph signals by minimizing the Dirichlet energy in the spatial domain.

5.2 Node Classification with Graph Recovery

Table 1 compares model performance under two types of local corruptions, i.e., attribute injection and graph meta attack. We highlight each score in red or green by their relative improvement or retrogression over GCN. The opacity is determined by the relative percentage change.

We observe that MAGNET$_{\text{gae}}$ outperforms its competitors and recovers at most 94% test accuracy from the perturbed attributes for homophilic graphs and over 180% for heterophilic graphs. In particular, the results on **Wisconsin** by MAGNET is even higher than on the non-corrupted clean graph. It implies the potential of our learning scheme to handle heterophilic graphs. We will leave further investigations to the future regarding our denoising model's representation power in handling heterophilic graphs.

On the other hand, the four baseline methods fail to recover a robust embedding from the locally-corrupted 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

[2] https://github.com/klicperajo/ppnp
[3] https://github.com/mims-harvard/GNNGuard
[4] https://github.com/lxiaorui/ElasticGNN
[5] https://github.com/lxiaorui/AirGNN

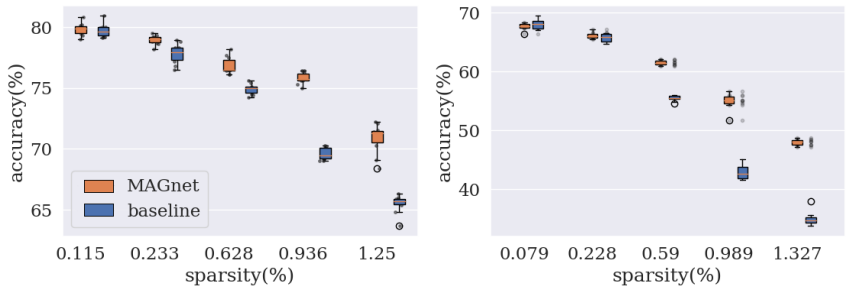


Figure 3: The performance comparison of MAGNET to baseline GCN (corrupted) with different levels of attribute injection to Cora (left) and Citeseer (right).

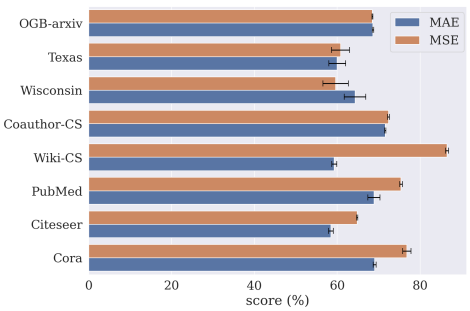


Figure 4: ℓ_1/ℓ_2 approximation error in GAE affects the prediction performance.

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.

Not only the MAGNET achieves promising performance in graph recovery, but also it requires comparable computational cost. Table 4 displays the performance versus running time of the entire program on **Cora** (with 200 epochs) and **OGB-arxiv** (with 500 epochs). We train GAE with 1,000 epochs and 10,000 epochs on the two datasets, respectively. On the small dataset **Cora**, the running speed of GNNGUARD is significantly faster (at 7.35 seconds) than its counterparts (at around 46 seconds) at the cost of the lowest accuracy score. Alternatively, MAGNET$_{\text{gae}}$ achieves the best performance with a comparable speed of 45.65 seconds. On the large-scale dataset **OGB-arxiv**, the advantage of GNNGUARD on the training speed is suppressed by ELASTICGNN and AIRGNN, but it achieves a higher score than the two baselines. Still, it is 6%'s lower than our MAGNET$_{\text{gae}}$ at the cost of a slightly longer training time. As for APPNP, it unfortunately exceeds the memory limit of 100GB, and thus fails to present any results. In addition to directly making the comparison of our method to baseline models, we also report the training time for MAGNET$_{\text{true}}$, which is essentially the training cost for denoising the graph with a given mask, i.e., exclude training GAE. Our MAGNET can be potentially more efficient by employing a fast mask approximation tool. Otherwise, a smaller number of training epochs in GAE could also accelerate the overall speed.

Table 4: Comparison of the computational cost.

	Cora		OGB-arxiv	
	score (%)	time (sec)	score (%)	time (min)
APPNP	68.46	47.15	OOM	OOM
GNNGUARD	61.96	7.35	65.75	35.27
ELASTICGNN	77.74	45.22	41.34	21.13
AIRGNN	76.22	43.92	52.32	11.84
MAGNET$_{\text{gae}}$	79.07	45.65	68.68	36.30
MAGNET$_{\text{true}}$	78.48	33.58	69.57	30.88

5.3 Ablation 1: Reconstruction Error in GAE

The choices on the regularizers of the mask matrix approximation affect the approximation results. We make comparisons to $\mathcal{L} = \|X - X'\|_1$ (MAE loss) and $\mathcal{L} = \|X - X'\|_2$ (MSE loss) in GAE. The experiment is conducted on the seven datasets with injection perturbations, where the performance of prediction accuracy is reported in Figure 4.

While the empirical performance disagrees with an arbitrary choice of one over another, it supports our analysis in Section 3 that MSE loss is generally a more safe choice. More specifically, we observe that ℓ_2-norm is more likely to outperform when the feature scale is relatively small (e.g., between -1 and 1, which gives a higher punishment to considerably small errors than ℓ_1's). As a reference, the attribute values are $\{0, 1\}$ (discrete values) for **Cora**, **Citeseer**, and **Coauthor-CS**, [0,1.226] (continuous values) for **PubMed**, and [-3,3] (continuous values) for **Wiki-CS**. On the other hand, ℓ_1-norm tends to perform better in heterophilic graphs, such as **Wisconsin**. In addition, ℓ_1-norm is less sensitive to the choice of threshold in (4). The true positive and false negative rate for the ℓ_1-based mask matrix does not change drastically among different τs. In the contrast, tuning the threshold τ is critical for the ℓ_2-based scoring function to perform a higher true positive rate and lower false negative rate. When the threshold is properly chosen, its approximation is more reliable than the ℓ_1-based counterparts.

5.4 Ablation 2: Regularizers in Graph Recovery

Table 2 delivers the individual accuracy score for a combination of $p \in \{0, 1\}$ and $q \in \{1, 2\}$ with all-ones, GAE-oriented, and ground truth mask matrices to digest the preference of p, q's choice in different scenarios for two types of corruptions, respectively. We also include the baseline (GCN) at the top, which is equivalent to optimizing a $p = 2$, $q = 2$ regularization in the spatial domain.

Overall, $p = 1$, $q = 2$ is a robust choice when the quality of mask matrix approximation is concerned. The next best choice is $p = 1$, $q = 1$, which is a frequently-adopted choice in conventional denoising tasks. When the mask is believed reliable, $p \in \{0, 1\}$ and $q = 1$ both achieve satisfactory performance. Generally, $p = 0$ fits better to concentrated perturbations in a few entities, while $p = 1$ fits the case when the corruptions are rather diverse.

Figure 3 reports the effect of corruption level, i.e., the sparsity of the ground truth mask, on the performance with **Cora** and **Citeseer**. We visualize the results of MAGNET$_{\text{gae}}$ (orange) with the best-tuned p, q. The performance gain of MAGNET over GCN on both datasets keeps rising along with an increasing level of corruption.

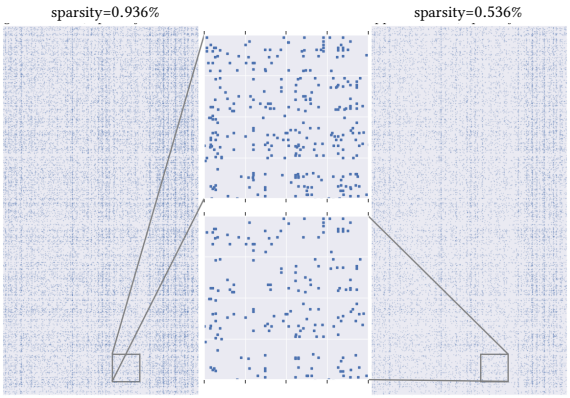


Figure 5: Distribution of the ground truth mask (left) and conditional GAE-approximated mask (right) at threshold $\tau = 0.1$. A 200×200 sub-region is amplified in the middle.

Figure 6: Image recovery with local additive white noise at $\sigma = 50$. Three images display the noisy raw input (left), and the inpainting result by BM3D (middle) and MAGnet (right).

5.5 Investigation on Mask Matrix and Local Regularization

The next experiment digests the effect of the GAE-oriented mask approximation and ADMM optimization with visualizations.

The quality of GAE is verified by the recall of the mask approximation. Figure 5 pictures the conditional mask matrix from model reconstruction error on **Cora** with attribute injection. The sparsity of both matrices can be identified clearly in the amplified subfigures. The approximated mask matrix succeeds in seizing 60% of anomalies, which provides a reliable foundation for subsequent recovering work.

Figure 6 visually exhibits the local effect of ADMM optimization with an example of graph inpainting, where a 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. A lattice-like graph adjacency matrix is prepared with patch neighbors connected with edges. We randomly select 9 nodes to assign white noise of $\mathcal{N}(0, 1)$ on their attributes. We assume a given mask matrix and focus on the performance of the ADMM optimizer. MAGnet restricts major smoothing effects within the masked region. The rest 'clean' area maintains a sharp detail. In comparison, the classic denoising model BM3D [13] blurs the entire scale of the picture.

6 RELATED WORK

The last few years have witnessed a roaring success of graph neural networks, the essential design of which is a graph convolution operation. Depending on whether an adjacency matrix or a graph Laplacian is explicitly employed, a convolution operation falls into two genres of either spectral [8, 15, 53, 60] or spatial-based [25, 29, 33, 40, 48] methods. The widespread message passing scheme [24] encourages continuous development on adjacency matrix-based feature aggregation rules [25, 29, 33, 40, 48]. These spatial methods pursue global feature smoothness by minimizing Dirichlet energy, which establishes a low-pass filter that denoises under ℓ_2 penalty [18, 41, 65]. When multiple convolution layers are stacked in a neural network, it is inevitable to lose small-energy patterns and learn an over-smoothing representation. In contrast, residual connections [9, 30, 45, 55] maintain feature proximity and concurrently keep odd examples, i.e., anomalies, that potentially disturb pattern characterization.

The harmful graph anomalies are getting noticed in the literature. Robust graph optimization refines the learning manner of a classifier to provide promising predictions against potential threats [14, 54, 66]. They are essentially designed to defend against graph adversarial attacks, which usually influence model performance by corrupting node connections. A robust node embedding is indirectly approximated by a refined adjacency matrix [12, 19, 27, 58]. Graph anomaly detection [16, 37, 44, 64] identifies rare problematic nodes, but the related methods merely recover robust representation of anomalies. On top of that, the justification is based on node entities rather than feature values. A few existing works modify the loss function to force local smoothness of the feature space [10, 35, 36], but they limit the regularization in the spatial domain, and the penalties are restricted to ℓ_1 or ℓ_2.

7 CONCLUSION

This research develops a GNN-based framework to recover robust graph data representations from locally corrupted node attributes. We propose a multi-scale sparse regularizer to optimize the hidden representation that guarantees conditional closeness to the disrupted input. The outliers in graph node attributes are explicitly positioned by a learnable mask matrix, which is approximated by an unsupervised graph autoencoder that requires no prior knowledge of the distribution of anomalies. We define the optimization problem by adaptive l_p and l_q errors, where the tunable factors p, q stimulate the maximum ability of the regularizer, and the corresponding optimization can be solved by an efficient inertial ADMM. Incorporating sparsity in multi-level framelet coefficients not only removes global noises and regularizes local anomalous attributes at the same time but also preserves local patterns. Our model achieves competitive performance in recovering a robust graph embedding from local corruptions, whereas graph smoothing and defense baseline methods fail to provide a decent solution.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No. 62172370) and Shanghai Artificial Intelligence Laboratory (No. P22KN00524).

REFERENCES

[1] Charu C Aggarwal. 2017. Linear models for outlier detection. In *Outlier Analysis*. Springer, 65–110.
[2] David Ahmedt-Aristizabal, Mohammad Ali Armin, Simon Denman, Clinton Fookes, and Lars Petersson. 2021. Graph-based deep learning for medical diagnosis and analysis: past, present and future. *Sensors* 21, 14 (2021), 4758.
[3] Felipe Alvarez and Hedy Attouch. 2001. An inertial proximal method for maximal monotone operators via discretization of a nonlinear oscillator with damping. *Set-Valued Analysis* 9, 1 (2001), 3–11.
[4] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. 2020. Analyzing the expressive power of graph neural networks in a spectral perspective. In *ICLR*.
[5] R. I. Boţ and E. R. Csetnek. 2016. An inertial alternating direction method of multipliers. *Minimax Theory and its Applications* 1 (2016), 029–049. Issue 1.
[6] R. I. Boţ, E. R. Csetnek, and C. Hendrich. 2015. Inertial Douglas–Rachford splitting for monotone inclusion problems. *Appl. Math. Comput.* 256 (2015), 472–487.
[7] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.
[8] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *ICLR*.
[9] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *ICML*. 1725–1735.
[10] Siheng Chen, Yonina C Eldar, and Lingxiao Zhao. 2021. Graph unrolling networks: Interpretable neural networks for graph signal denoising. *IEEE Transactions on Signal Processing* 69 (2021), 3699–3713.
[11] Siheng Chen, Aliaksei Sandryhaila, José MF Moura, and Jelena Kovačević. 2015. Signal recovery on graphs: Variation minimization. *IEEE Transactions on Signal Processing* 63, 17 (2015), 4609–4624.
[12] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *NeurIPS*.
[13] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. 2007. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing* 16, 8 (2007), 2080–2095.
[14] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *ICML*. 1115–1124.
[15] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*.
[16] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep Anomaly Detection on Attributed Networks. In *SIAM International Conference on Data Mining (SDM)*.
[17] Bin Dong. 2017. Sparse representation on graphs by tight wavelet frames and applications. *Applied and Computational Harmonic Analysis* 42, 3 (2017), 452–479.
[18] Yushun Dong, Kaize Ding, Brian Jalaian, Shuiwang Ji, and Jundong Li. 2021. AdaGNN: Graph Neural Networks with Adaptive Frequency Response Filter. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.
[19] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. 2020. 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*. 169–177.
[20] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *WWW*. 417–426.
[21] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
[22] Daniel Gabay and Bertrand Mercier. 1976. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications* 2, 1 (1976), 17–40.
[23] Alberto P García-Plaza, Víctor Fresno, Raquel Martínez Unanue, and Arkaitz Zubiaga. 2016. Using fuzzy logic to leverage HTML markup for web page representation. *IEEE Transactions on Fuzzy Systems* 25, 4 (2016), 919–933.
[24] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*.
[25] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.
[26] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*.
[27] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *KDD*. 66–74.
[28] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. In *NeurIPS Workshop on Bayesian Deep Learning*.
[29] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[30] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.
[31] Volodymyr Kovenko and Ilona Bogach. 2020. A Comprehensive Study of Autoencoders' Applications Related to Images.. In *IT&I Workshops*. 43–54.
[32] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. 2017. Radar: Residual Analysis for Anomaly Detection in Attributed Networks.. In *IJCAI*. 2152–2158.
[33] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.
[34] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. 2021. DeepRobust: a Platform for Adversarial Attacks and Defenses. In *AAAI*, Vol. 35.
[35] Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Graph Neural Networks with Adaptive Residual. In *NeurIPS*.
[36] Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. 2021. Elastic graph neural networks. In *ICML*.
[37] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. 2021. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).
[38] Faisal Mahmood, Nauman Shahid, Ulf Skoglund, and Pierre Vandergheynst. 2018. Adaptive graph-based total variation for tomographic reconstructions. *IEEE Signal Processing Letters* 25, 5 (2018), 700–704.
[39] Péter Mernyei and Cătălina Cangea. 2020. Wiki-cs: a wikipedia-based benchmark for graph neural networks. In *ICML Workshop on Graph Representation Learning and Beyond*.
[40] Federico Monti, Michael M Bronstein, and Xavier Bresson. 2017. Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks. In *NeurIPS*.
[41] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv:1905.09550* (2019).
[42] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*.
[43] Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, and Qinghua Zheng. 2018. ANOMALOUS: A Joint Modeling Approach for Anomaly Detection on Attributed Networks.. In *IJCAI*. 3513–3519.
[44] Zhen Peng, Minnan Luo, Jundong Li, Luguo Xue, and Qinghua Zheng. 2020. A Deep Multi-View Framework for Anomaly Detection on Attributed Networks. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[45] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*.
[46] Aliaksei Sandryhaila and Jose MF Moura. 2014. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing* 62, 12 (2014), 3042–3054.
[47] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of Graph Neural Network Evaluation. In *NeurIPS Workshop on Relational Representation Learning Workshop*.
[48] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.
[49] Yu Guang Wang and Xiaosheng Zhuang. 2019. Tight framelets on graphs for multiscale data analysis. In *Wavelets and Sparsity XVIII*, Vol. 11138. International Society for Optics and Photonics, 111380B.
[50] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *ICML*.
[51] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)* (2020).
[52] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.
[53] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. 2018. Graph Wavelet Neural Network. In *ICLR*.
[54] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing* 17, 2 (2020), 151–178.
[55] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*.
[56] Zongben Xu, Xiangyu Chang, Fengmin Xu, and Hai Zhang. 2012. $L_{1/2}$ regularization: A thresholding representation theory and a fast solver. *IEEE Transactions on Neural Networks and Learning Systems* 23, 7 (2012), 1013–1027.
[57] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*.

[58] Xiang Zhang and Marinka Zitnik. 2020. GNNGuard: Defending Graph Neural Networks against Adversarial Attacks. In *NeurIPS*, Vol. 33.
[59] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep learning on graphs: a survey. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[60] Xuebin Zheng, Bingxin Zhou, Junbin Gao, Yu Guang Wang, Pietro Lio, Ming Li, and Guido Montúfar. 2021. How framelets enhance graph neural networks. In *ICML*.
[61] Xuebin Zheng, Bingxin Zhou, Yu Guang Wang, and Xiaosheng Zhuang. 2022. Decimated framelet system on graphs and fast G-framelet transforms. *Journal of Machine Learning Research* 23, 18 (2022), 1–68.
[62] Bingxin Zhou, Ruikun Li, Xuebin Zheng, Yu Guang Wang, and Junbin Gao. 2021. Graph Denoising with Framelet Regularizer. *arXiv:2111.03264* (2021).
[63] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: a review of methods and applications. *AI Open* 1 (2020), 57–81.
[64] Dali Zhu, Yuchen Ma, and Yinlong Liu. 2020. Anomaly Detection with Deep Graph Autoencoders on Attributed Networks. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. 1–6.
[65] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and unifying graph neural networks with an optimization framework. In *WWW*.
[66] Daniel Zügner and Stephan Günnemann. 2019. Adversarial attacks on graph neural networks via meta learning. In *ICLR*.

A GRAPH FRAMELET TRANSFORM

This section briefs the fast computation of undecimated framelet transform on graphs. The initial idea was proposed by Dong [17], which is then developed into a graph convolution [60]. Here we give a brief introduction to the wavelet frame (framelet) system for a graph signal, using the fast approximation on the framelet decomposition and reconstruction operators by Chebyshev approximation, which is the key to an efficient graph convolution.

A.1 Framelet System

A framelet is defined by two key elements: a *filter bank* $\boldsymbol{\eta} := \{a; b^{(1)}, \ldots, b^{(K)}\}$ and a set of *scaling functions* $\Psi = \{\alpha; \beta^{(1)}, \ldots, \beta^{(K)}\}$. We name a the low-pass filter and $b^{(k)}$ the kth high-pass filter with $k = 1, \ldots, K$. The two sets of filters respectively extract the approximated and detailed information of the input graph signal in a transformed domain, i.e., the framelet domain.

There are different choices of filter masks, which results in different tight framelet systems (See [17] for some examples). In general, the selected filter bank and its associated scaling function satisfy the relationship that

$$\widehat{\alpha}(2\xi) = \widehat{a}(\xi)\widehat{\alpha}(\xi), \quad \widehat{\beta^{(k)}}(2\xi) = \widehat{b^k}(\xi)\widehat{\alpha}(\xi), \tag{8}$$

for $k = 1, \ldots, K$ and $\xi \in \mathbb{R}$. We implement the *Haar-type* filter with one high pass, i.e., $K = 1$. For $x \in \mathbb{R}$, it defines

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

With the defined filter bank and the scaling functions, the undecimated framelet basis can be defined for a graph signal with the eigenpairs $\{(\lambda_\ell, \boldsymbol{u}_\ell)\}_{\ell=1}^n$ of the graph Laplacian $\mathcal{L}$. At level $l = 1, \ldots, J$, we define the undecimated framelets for node p by

$$\begin{aligned} \boldsymbol{\varphi}_{l,p}(v) &:= \textstyle\sum_{\ell=1}^n \widehat{\alpha}\left(\frac{\lambda_\ell}{2^l}\right) \overline{\boldsymbol{u}_\ell(p)} \boldsymbol{u}_\ell(v), \\ \boldsymbol{\psi}_{l,p}^{(k)}(v) &:= \textstyle\sum_{\ell=1}^n \widehat{\beta^{(k)}}\left(\frac{\lambda_\ell}{2^l}\right) \overline{\boldsymbol{u}_\ell(p)} \boldsymbol{u}_\ell(v), \quad k = 1, \ldots, K. \end{aligned} \tag{9}$$

We name $\boldsymbol{\varphi}_{l,p}(v), \boldsymbol{\psi}_{l,p}^{(k)}(v)$ with $v \in \mathcal{V}$ the low-pass and the kth high-pass framelet basis, respectively. These bases are called the *undecimated tight framelets* on $\mathcal{G}$. They define an *undecimated framelet system* $\mathrm{UFS}_{J_1}^J(\Psi, \boldsymbol{\eta})$ $(J > J_1)$ for $l_2(\mathcal{G})$ from J_1, which reads

$$\begin{aligned} \mathrm{UFS}_{J_1}^J(\Psi, \boldsymbol{\eta}) &:= \mathrm{UFS}_{J_1}^J(\Psi, \boldsymbol{\eta}; \mathcal{G}) \\ &:= \left\{\varphi_{J_1,p} : p \in \mathcal{V}\right\} \cup \{\psi_{l,p}^{(k)} : p \in \mathcal{V}, l = J_1, \ldots, J\}_{k=1}^K. \end{aligned}$$

A.2 Framelet Decomposition

We now introduce the *framelet decomposition operator* of the defined tight framelet system, i.e., $\mathcal{W}$ that we used in equation 3 to construct the spectral coefficients in the framelet domain.

The framelet decomposition operator $\mathcal{W}_{k,l}$ contains a set of orthonormal bases at $(k, l) \in \{(0, J)\} \cup \{(1, 1), \ldots, (1, J), \ldots, (K, J)\}$. It transforms a given graph signal $\boldsymbol{X}$ to a set of multi-scale and multi-level *framelet coefficients*, i.e., the spectral representation of the graph signal $\boldsymbol{X}$ in the transformed domain. In particular, $\mathcal{W}_{0,J}$ contains $\boldsymbol{\varphi}_{J,p}, p \in \mathcal{V}$ that constructs the low-pass framelet coefficients $\mathcal{W}_{0,J}\boldsymbol{X}$ that preserve approximate information in $\boldsymbol{X}$. They are the smooth representation that reflects the global trend of $\boldsymbol{X}$. Meanwhile, the high-pass coefficients $\mathcal{W}_{k,l}\boldsymbol{X}$ with $\mathcal{W}_{k,l} = \{\boldsymbol{\psi}_{l,p}^{(k)}, p \in \mathcal{V}\}$ records detailed information at scale k and level l. They reflect local patterns or noises of the signal. A larger scale k contains more localized information with smaller energy.

The framelet coefficients can be directly projected by $\langle \boldsymbol{\varphi}_{l,p}, \boldsymbol{X} \rangle$ and $\langle \boldsymbol{\psi}_{l,p}^{(k)}, \boldsymbol{X} \rangle$ for node p at scale level l. For instance, we take eigendecomposition on a graph Laplacian $\mathcal{L}$ and obtain $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n] \in \mathbb{R}^{n \times n}$ be the eigenvectors and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ be the eigenvalues. According to the definition above, the respective filtered diagonal matrices with low-pass and high-pass filters are

$$\begin{aligned} \widehat{\alpha}\left(\frac{\Lambda}{2}\right) &= \mathrm{diag}\left(\widehat{\alpha}\left(\frac{\lambda_1}{2}\right), \ldots, \widehat{\alpha}\left(\frac{\lambda_n}{2}\right)\right), \\ \widehat{\beta^{(k)}}\left(\frac{\Lambda}{2^l}\right) &= \mathrm{diag}\left(\widehat{\beta^{(k)}}\left(\frac{\lambda_1}{2^l}\right), \ldots, \widehat{\beta^{(k)}}\left(\frac{\lambda_n}{2^l}\right)\right). \end{aligned}$$

The associated framelet coefficients at the low pass and the kth high pass are

$$\begin{aligned} \mathcal{W}_{0,J}\boldsymbol{X} &= \boldsymbol{U}\widehat{\alpha}\left(\frac{\Lambda}{2}\right)\boldsymbol{U}^\top \boldsymbol{X}, \\ \mathcal{W}_{k,l}\boldsymbol{X} &= \boldsymbol{U}\widehat{\beta^{(k)}}\left(\frac{\Lambda}{2^{l+1}}\right)\boldsymbol{U}^\top \boldsymbol{X} \quad \forall l = 0, \ldots, J. \end{aligned} \tag{10}$$

A.3 Fast Tight Framelet Transform

To formulate an efficient framelet decomposition, two strategies are considered, including a recursive formulation on filter matrices, and Chebyshev-approximated eigenvectors.

With a filter bank that satisfies (8), the above decomposition can be implemented recursively by

$$\mathcal{W}_{k,1}\boldsymbol{X} = \boldsymbol{U}\widehat{\beta^{(k)}}\left(2^{-R}\Lambda\right)\boldsymbol{U}^\top \boldsymbol{X}$$

for the first level ($l = 1$), and

$$\begin{aligned} \mathcal{W}_{k,l}\boldsymbol{X} = \boldsymbol{U}\widehat{\beta^{(k)}}\left(2^{R+l-1}\Lambda\right)\widehat{\alpha}\left(2^{R+l-2}\Lambda\right)\ldots\widehat{\alpha}\left(2^{-R}\Lambda\right)\boldsymbol{U}^\top \boldsymbol{X} \\ \forall l = 2, \ldots, J, \end{aligned}$$

where the real-value dilation scale R satisfies $\lambda_{\max} \le 2^R\pi$.

Furthermore, we employ an m-order Chebyshev polynomials approximation for efficient framelet decomposition. It avoids eigen-decomposition on graph Laplacian, which can be considerably slow on a large graph.

Denote the m-order approximation of α and $\{\beta^{(k)}\}_{k=1}^{K}$ by $\mathcal{T}_0$ and $\{\mathcal{T}_k\}_{k=1}^{K}$, respectively. The framelet decomposition operator $\mathcal{W}_{r,j}$ is approximated by

$$\mathcal{W}_{k,l} = \begin{cases} \mathcal{T}_0\left(2^{-R}\mathcal{L}\right), & l = 1, \\ \mathcal{T}_r\left(2^{R+l-1}\mathcal{L}\right)\mathcal{T}_0\left(2^{R+l-2}\mathcal{L}\right)\dots\mathcal{T}_0\left(2^{-R}\mathcal{L}\right), & l = 2,\dots,J. \end{cases}$$

We apply the fast-approximated $\mathcal{W}$ in the l_p penalty term of the objective function equation 3 formulated in Section 2.

B INERTIAL ADMM ALGORITHM

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

B.1 Inertial ADMM

We denote $Z = \mathcal{W}U$ and rewrite equation 3 as

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

This forms a standard formulation of problems that can be solved by Alternating Direction Method of Multipliers (ADMM [22]). The associated augmented Lagrangian reads

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

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

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

The above iteration can be equivalently written as

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

If we further define

$$V_{k+1} = Y_k - \gamma Z_{k+1},$$

the above iteration can be reformulated as

$$\begin{aligned} Z_{k+1} &= \arg\min_{Z} \ \|\nu Z\|_{p,G} + \tfrac{\gamma}{2}\|Z - \mathcal{W}U_k - Y_k/\gamma\|^2, \\ &= \arg\min_{Z} \ \|\nu Z\|_{p,G} + \tfrac{\gamma}{2}\|Z - (2Y_k - V_k)/\gamma\|^2, \\ V_{k+1} &= Y_k - \gamma Z_{k+1}, \\ U_{k+1} &= \arg\min_{U} \ \tfrac{1}{2}\|M \odot (U - X)\|_{q,G}^{q} + \tfrac{\gamma}{2}\|\mathcal{W}U + V_{k+1}/\gamma\|^2, \\ Y_{k+1} &= Y_k + \gamma(\mathcal{W}U_{k+1} - Z_{k+1}) \\ &= V_{k+1} + \gamma\mathcal{W}U_{k+1}. \end{aligned}$$

In this paper, we consider the inertial ADMM motivated by [3], whose iteration is provided below:

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

In general, we have $a_k \in [0, 1]$. When the problem is convex, the convergence can be guaranteed by choosing $a_k \in [0, 1/3]$ [5, 6].

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

When $q = 1$, let $Q = U - X$, and consider

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

The optimality condition yields

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

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

$$U_{k+1} = Q_{k+1} + X.$$

C SUPPLEMENTARY MATERIAL

We prepare additional experimental results, including data description, details for percentage improvements we colored in Table 1, as well as additional visualizations for the mask approximation at https://github.com/bzho3923/MAGnet/blob/main/sm.pdf.

Table 5: Summary of the datasets for node classification tasks.

	Cora	Citeseer	PubMed	Wiki-CS	Coauthor-CS	Wisconsin	Texas	OGB-arxiv
# Nodes	2, 708	3, 327	19, 717	11, 701	18, 333	251	183	169, 343
# Edges	5, 429	4, 732	44, 338	216, 123	100, 227	499	309	1, 166, 243
# Features	1, 433	3, 703	500	300	6, 805	1, 7033	1, 703	128
# Classes	7	6	3	10	15	5	5	40
# Training Nodes	140	120	60	580	300	120	87	90, 941
# Validation Nodes	500	500	500	1769	200	80	59	29, 799
# Test Nodes	1, 000	1, 000	1, 000	5847	1000	51	37	48, 603
Label Rate	0.052	0.036	0.003	0.050	0.016	0.478	0.475	0.537
Feature Scale	{0, 1}	{0, 1}	[0, 1.263]	[−3, 3]	{0, 1}	{0, 1}	{0, 1}	[−1.389, 1.639]

Table 6: Improvement percentage of average performance for node classification with injection.

	Cora		Citeseer		PubMed		Coauthor-CS		Wiki-CS		Wisconsin		Texas		OGB-arxiv	
Module	absolute	relative	absolute	relative	absolute	relative	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%	28.03%	159.06%	0.92%	8.68%	-	-
GNNGuard	−10.36%	−58.98%	−4.58%	−17.38%	−1.20%	7.16%	−2.11%	−22.37%	0.00%	0.00%	−2.86%	−16.24%	0.46%	4.34%	−3.90%	−78.07%
ElasticGNN	12.47%	71.00%	12.21%	49.54%	5.23%	31.27%	−3.03%	−32.89%	−1.93%	−10.34%	10.55%	59.88%	1.44%	13.67%	−39.58%	−791.81%
AirGNN	10.37%	58.69%	7.92%	32.14%	10.40%	62.19%	−2.71%	−28.66%	9.05%	48.60%	27.61%	161.85%	0.92%	8.68%	−23.53%	−600.75%
MAGNET-one	9.78%	55.68%	2.85%	11.31%	1.89%	10.80%	1.98%	20.95%	8.24%	44.25%	15.03%	85.29%	2.29%	21.70%	−0.26%	−5.26%
MAGNET-gae	14.49%	82.05%	12.52%	50.81%	11.40%	68.20%	4.96%	52.57%	10.64%	57.14%	33.31%	189.06%	3.21%	30.39%	0.38%	7.60%
MAGNET-true	13.50%	76.85%	19.05%	72.22%	11.73%	70.14%	8.28%	87.66%	15.37%	82.59%	36.17%	205.29%	2.75%	26.05%	1.68%	33.63%

Table 7: Improvement percentage of average performance for node classification with meta-attack.

	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%
AirGNN	5.16%	62.52%	18.55%	62.37%	7.82%	92.99%
MAGNETone (ours)	2.72%	32.96%	12.96%	43.59%	4.05%	48.12%
MAGNETgae (ours)	5.29%	64.14%	21.84%	73.43%	7.89%	93.80%
MAGNETtrue	7.74%	93.86%	21.95%	73.80%	8.62%	102.45%

D DATASET DESCRIPTIONS

Table 5 documents key descriptive statistics of the eight datasets for the node classification tasks. The first five datasets are downloaded from `PyTorch-Gometric` [21] [6], the two heterophilic graphs are from `Geom-GCN` [42] [7], and **OGB-arxiv** is from open graph benchmarks [26] [8]. All the preprocessing, data split, and evaluation are guided by the source codes or standard routines.

E PERCENTAGE PERFORMANCE IMPROVEMENT IN NODE CLASSIFICATION

In the main paper, we use the relative scores of improvements to color the performance in Table 1, which 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 also report the absolute score of improvement for a clear comparison, which is defined by

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

Both scores are reported in Table 6 below. In Table 1, we color the model performance by red for positive relative scores, and by green for negative relative scores. If the absolute value of the relative score is large, the color would be darker.

F GAE VISUALIZATION

Figure 7-Figure 9 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 meta attack perturbation, as such attack focuses major poisonings on a minority of node entities. When making visualizations on such matrices, they are nothing more than a few horizontal lines. For **PubMed**, **Coauthor-CS** and **Wiki-CS**, we only visualize a subset of the full node attribute matrix, as the full matrix is too large to display. For instance, the size of **Coauthor-CS**'s node attribute matrix is about $\approx 18000 \times 7000$. Different from the other three datasets, the attributes in **PubMed** and **Wiki-CS** has non-binary values. Consequently, we report the mask matrix with a threshold. In other words, in **Cora**, **Citeseer**, and **Coauthor-CS**, M_{ij}=1 if the i, jth value in the ground truth

[6] https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html
[7] https://github.com/graphdml-uiuc-jlu/geom-gcn
[8] https://ogb.stanford.edu/docs/nodeprop/#ogbn-arxiv

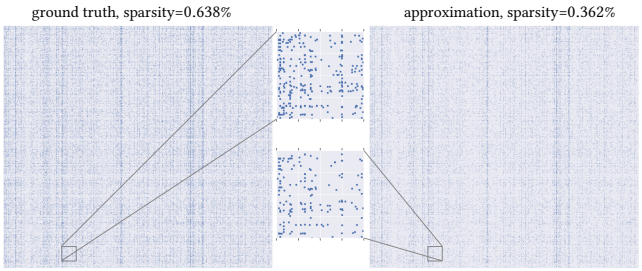


Figure 7: Mask matrix visualization for Citeseer.

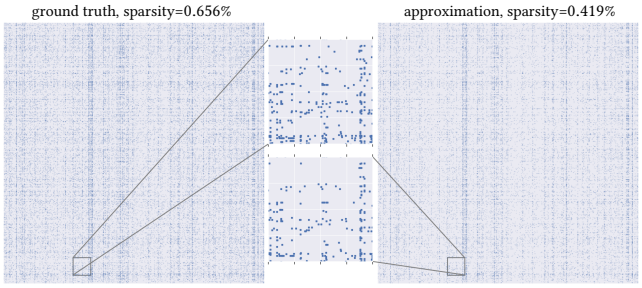


Figure 8: Mask matrix visualization for Coauthor-CS with the first 3000 **nodes and** 3000 **features.**

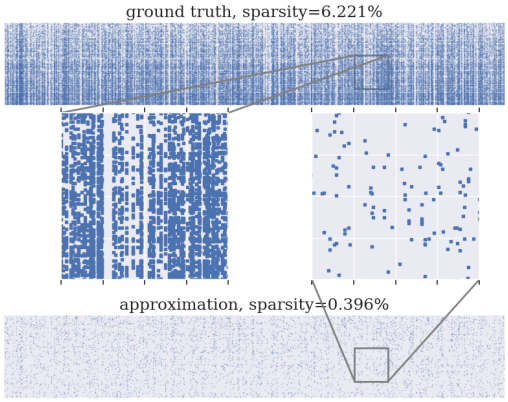


Figure 9: Mask matrix visualization for PubMed with the first 3000 **nodes and** 500 **features at threshold**=0.005**.**

matrix and the perturbed matrix are different. In **PubMed**, M_{ij}=1 if the difference of the jth feature of the ith node in the ground truth matrix and the perturbed matrix are larger than 0.005. The justification is similar to **Wiki-CS**, where the difference threshold is increased to 0.05.

G RELIABILITY OF MASK APPROXIMATION

We investigate the different choices on the regularizes of the mask matrix approximated by the GAE module. In the first ablation study

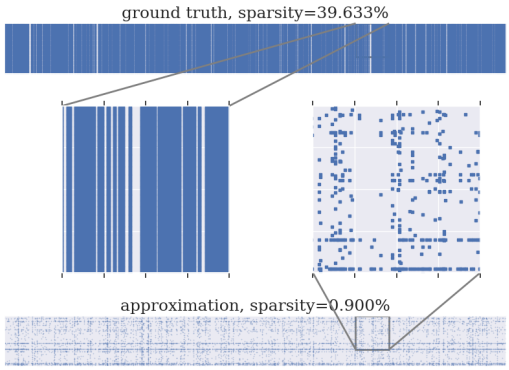
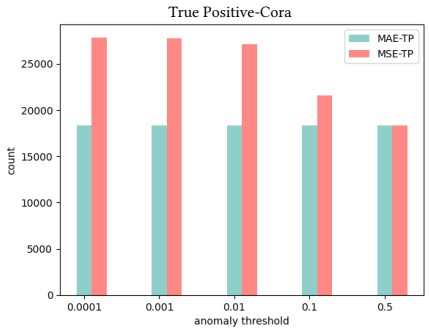


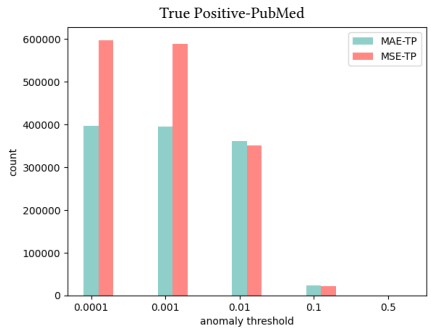
Figure 10: Mask matrix visualization for Wiki-CS with the first 3000 **nodes and** 300 **features at threshold**=0.05**.**

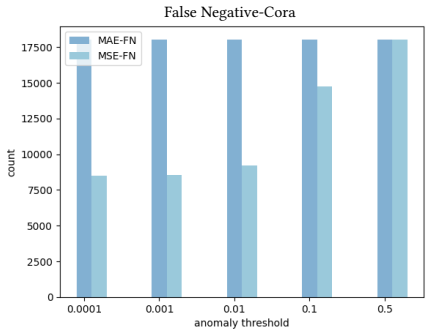
of the main paper we compare the approximation results by ℓ_1 and ℓ_2 normalization in the loss function. Here we visualize the count of true positive and false negative predictions in Figure 11 for the seven (relatively) small datasets. We are more interested in identifying the non-zero positions, or positive approximations, as the majority of the mask matrix is filled with 0.

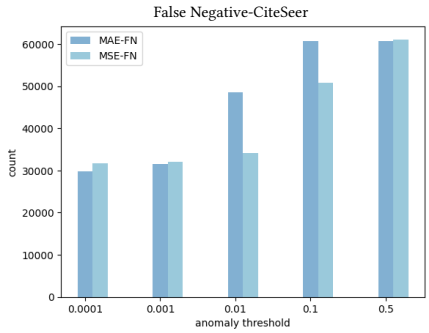
It turns out that the ℓ_1-based mask matrix does not change drastically among different threshold τs. In the contrast, tuning the threshold τ is critical for the ℓ_2-based scoring function to perform a higher true positive rate and lower false negative rate. When the threshold is properly chosen, its approximation is more reliable than the ℓ_1-based counterparts.

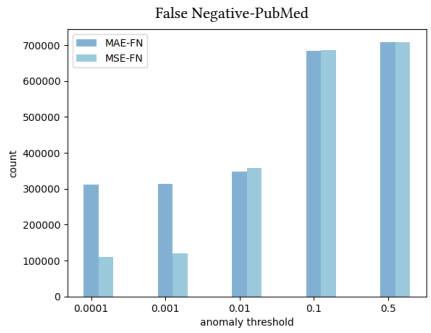

True Positive-Cora
MAE-TP
MSE-TP
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5

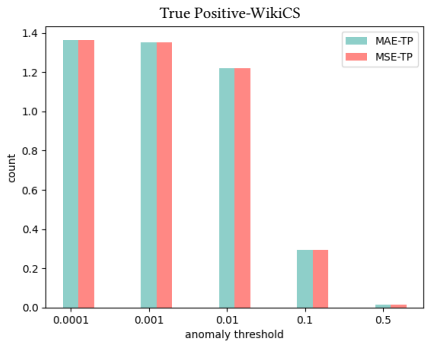

True Positive-CiteSeer
MAE-TP
MSE-TP
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5

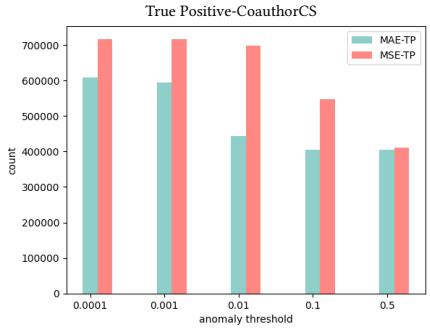

True Positive-PubMed
MAE-TP
MSE-TP
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5


False Negative-Cora
MAE-FN
MSE-FN
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5

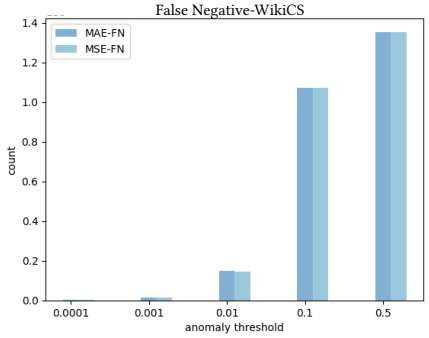

False Negative-CiteSeer
MAE-FN
MSE-FN
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5

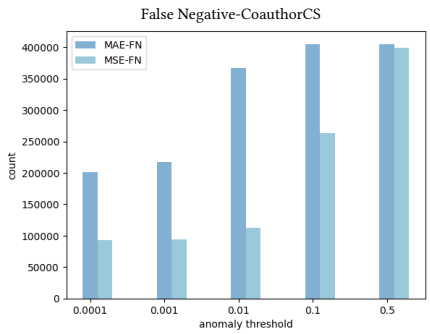

False Negative-PubMed
MAE-FN
MSE-FN
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5

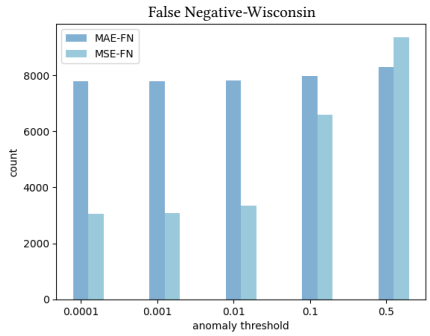

True Positive-WikiCS
MAE-TP
MSE-TP
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5

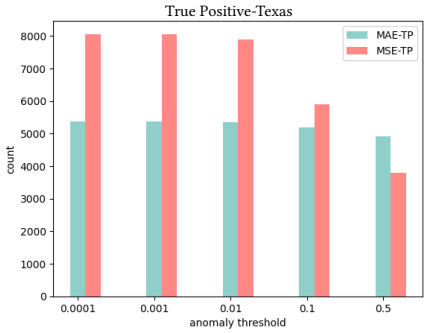

True Positive-CoauthorCS
MAE-TP
MSE-TP
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5

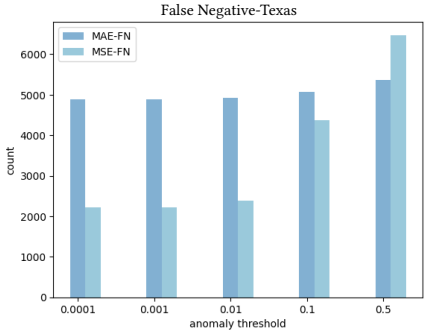

True Positive-Wisconsin
MAE-TP
MSE-TP
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5


False Negative-WikiCS
MAE-FN
MSE-FN
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5


False Negative-CoauthorCS
MAE-FN
MSE-FN
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5


False Negative-Wisconsin
MAE-FN
MSE-FN
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5


True Positive-Texas
MAE-TP
MSE-TP
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5


False Negative-Texas
MAE-FN
MSE-FN
count
anomaly threshold
0.0001
0.001
0.01
0.1
0.5