Graph Neural Network for Local Corruption Recovery

Bingxin Zhou 1,2  Yuanhong Jiang 2  Yu Guang Wang 2,3  Jingwei Liang 2  Junbin Gao 1  Shirui Pan 4  Xiaoqun Zhang 2

Abstract

Graph neural networks (GNNs) have seen a surge of development for exploiting the relational information of input graphs. Nevertheless, messages propagating through a graph contain both interpretable patterns and small perturbations. Despite global noise could be distributed over the entire graph data, it is not uncommon that corruptions appear well-concealed and merely pollute local regions while still having a vital influence on the GNN learning and prediction performance. This work tackles the graph recovery problem from local poisons by a robustness representation learning. Our developed strategy identifies regional graph perturbations and formulates a robust hidden feature representation for GNNs. A mask function pinpointed the anomalies without prior knowledge, and an $\ell_{p,q}$ regularizer defends local poisonings through pursuing sparsity in the framelet domain while maintaining a conditional closeness between the observation and new representation. The proposed robust computational unit alleviates the inertial alternating direction method of multipliers to achieve an efficient solution. Extensive experiments show that our new model recovers graph representations from local pollution and achieves excellent performance.

1. Introduction

Graph neural network (GNN) has been a powerful tool in many fields (Bronstein et al., 2017; Wu et al., 2020; Zhou et al., 2020; Zhang et al., 2020; Atz et al., 2021). Similar to any other real-world dataset, noise is ubiquitous in graphs and it has a noticeable side-effect on graph representation learning, which makes denoising a crucial ingredient in various GNN-based prediction tasks. In addition to global noise that could be distributed over the entire graph, the regional connectivity of a graph brings a vulnerability to small perturbation on the input, where the embedding of a node could easily be misled through rewiring the local topology or injecting features perturbations (Chen et al., 2020a; Jin et al., 2021; Xu et al., 2020). As the latter case does not supply prior information about the noise locality, targeting the contaminant nodes becomes troublesome. Without special operations, a trained model would face significant performance degradation. Consequently, regular graph convolutional layers that seek for global smoothness lose their advantages, no matter they are spectral- (Bruna et al., 2014; Defferrard et al., 2016; Xu et al., 2018a) or spatial-based (Kipf & Welling, 2017; Veličković et al., 2018; Wu et al., 2019), or with special regularization designs (Liu et al., 2021b; Zhou et al., 2021; Zhu et al., 2021).

This paper develops a strategy with graph autoencoder (GAE; Kipf & Welling (2016)) and $\ell_{p,q}$-regularization to handle anomalous node attributes where the perturbations are hidden in a small proportion of nodes. In particular, the GAE module exposes suspicious attributes and assigns a mask matrix to reflect the distribution of the anomalous index over the entire graph. The detection procedure functions in a fully-unsupervised manner that requires no prior. The mask is then submitted to a sparsity regularizer with framelet transforms for a robust approximation of the initial input, where an iteration acts similar to a graph convolutional layer that provides a smoother hidden feature representation for GNN prediction tasks. As if one writes it formally, it takes the form of the following optimization model

$$
\min_U \|\nu W U\|_{p,G} + \frac{1}{2}\|M \odot (U - X)\|_{q,G}^2.
$$

This optimization model is a combination of masked $\ell_q$ (1 $\leq q \leq 2$) reconstruction error term and $\ell_p$ ($0 \leq p \leq 1$) regularization enforcing the sparse representation by the framelet system. The reconstruction error with $q = 2$ is the conventional $\ell_2$ error to guarantee the learned representation $U$ be close to the masked target feature $X$ of the given graph. The $\ell_p$ regularization measures the graph total variation (Sandryhaila & Moura, 2014) in the framelet...
domain, where pursuing a small total variation recovers graph signal from corrupted attribute noises. Framelets establish a wavelet-like system on graphs (Dong, 2017; Wang & Zhuang, 2019; Zheng et al., 2022; 2021). With the transforms of the input signal, the model pumps up the robustness significantly, and the posterior mask guarantees that the denoising is predominantly conducted at the essential spots. When \( p \) approaches 1 from 2, the framelet representation \( WU \) obtains increasing sparsity. Furthermore, when \( p \to 0^+ \), the optimized representation achieves the maximum sparsity. While we treat the \( p, q \) as hyperparameters, our strategy has sufficient freedom for selecting the most qualified optimization model that fits the data and training requirements.

Our model delivers a way of unsupervised local denoising where the mask matrix \( M \) plays a pivotal role in reflecting the anomalous level of the node features. While the fidelity term is conditionally constrained on regular node features, it is required to estimate the mask ahead of solving the main optimization problem (1). As no prior is provided on locating these problematic elements, we employ graph autoencoder, a typical graph reconstruction tool, to generate such a mask matrix from reconstruction errors, which offers a decent match to the ground truth mask (Ding et al., 2019).

Furthermore, we develop an efficient inertial alternating direction method of multipliers (ADMM) algorithm for optimizing (1), which provides a slightly faster algorithm than the plain ADMM with time complexity of \( O(k^{-1}) \). For all \( p \) in \([0, 1] \) and \( q \) in \([1, 2] \), the algorithm is proved convergent.

Combining the three key ingredients of Mask, ADMM, and Graph, we name our model as MAGNET. As illustrated in Figure 1, the three components are in charge of unsupervised mask construction, localized robust optimization, and graph representation learning. The established mask regularization detects local perturbations in an unsupervised manner and approximates robust representation for graph network training. Empirical study shows its good application in robust optimization for black-box graph poisoning. The rest of the paper starts by explaining the context of the research in Section 2. The main operation blocks in Figure 1 are introduced in Section 3 and Section 4. Section 5 validates the empirical performance of MAGNET with various experiments. Previous studies of similar motivation are reviewed in Section 6. We conclude this work in Section 7.

### 2. Problem Formulation

We consider \( G = (V, E, X) \), an undirected attributed graph with \( n = |V| \) nodes and \( |E| \) edges, which is usually described by an adjacency matrix \( A \in \mathbb{R}^{n \times n} \). The observations on the node features \( X \in \mathbb{R}^{n \times d} \) could be a corrupted version of the ground truth data signal \( U \). That is,

\[
X = U + E_1 + E_2, \tag{2}
\]

where \( E_1 \) is noise and \( E_2 \) is outliers. Here, the \( E_2 \) is the main source of the corruption to the graph data and the global noise is relatively marginal. It is different from the
traditional denoising task where the global noise $E_1$ brings the major pollution for input data (Zhou et al., 2021; Zheng et al., 2021). The main difference between $E_1$ and $E_2$ is that noise might universally exist on the entire graph but outliers 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 $U$, $E_1$ and $E_2$:

(a) The global noise $E_1$ follows some distribution $D(0, \sigma)$ where $\sigma$ is considerably small;
(b) The outliers take a small portion of the entire graph, or the $E_2$ is a sparse matrix; and
(c) The observation $X$ is close to $U$ aside from the anomalous locations.

The associated objective function for the optimal $U$ reads

$$
\min_U \alpha \text{Reg}(U) + \beta \text{Loss}(E_2) \quad \text{such that } X|_M = (U + E_1 + E_2)|_M,
$$

where $\alpha$, $\beta$ are tuning parameters, and $M$ is a mask matrix that reveals indices of outliers. The $\text{Reg}(\cdot)$, $\text{Loss}(\cdot)$ denote regularization and loss functions satisfying assumptions (a) and (b), the choice of which we discuss now.

**Choice of Loss($E_2$)** The loss function plays the role of finding the object $U$ approximating the original feature $X$. As required by assumption (b) above, a measure with respect to the sparsity of $E_2$ has to be optimized. Theoretically, the $\ell_0$-norm that counts the number of nonzero entries in $E_1$ best reflects its sparsity. While solving a $\ell_0$ constrained problem is NP-hard, $\ell_1$-norm is usually substituted to allow feasible solution for (3), which still promotes good sparsity measure of $E_1$ (Chen et al., 2015). Such an objective function is practiced in the residual analysis-based anomaly detection methods, which aims at labeling suspicious small outliers, for example, see the work by Li et al. (2017) and Peng et al. (2018).

**Choice of Reg($U$)** The regularization term in (3) is a penalty complementary to the loss, which controls the noise level of $X$ in optimization by the smoothness of $U$, and it is usually quantified by some type of energy. For example, the conventional graph convolutional layer by Kipf & Welling (2017) utilizes normalized Dirichlet energy of $U$ by $\text{tr}(U^\top L U)$, where $L$ denotes the normalized graph Laplacian from $\tilde{L} = I - \frac{1}{2}AD^{-\frac{1}{2}}$ with the adjacency matrix $A$ and its degree matrix $D$. Minimizing such energy encourages message transmission 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 $U$, as stated by Monti et al. (2017). In recent work of graph smoothing effects (Zhu et al., 2021; Liu et al., 2021b), many other spatial-based graph convolutions follow the same spirit to pursue graph smoothness by the Dirichlet energy term, such as the works by Klicpera et al. (2018), Xu et al. (2018b) and Wu et al. (2019). Alternatively, spectral-based methods transform the graph signal to a new domain through the graph Laplacian, and make regularization on it. For example, Dong (2017) and Zhou et al. (2021) minimize the $\ell_1$-norm total variation of framelet coefficients, while Mahmood et al. (2018) take the regularization of the transformed coefficients 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 $E_1 + E_2$** Compared to the main ingredients of the objective function, the treatment to the fidelity constraint is rather trivial if there exists no outliers $E_2$. Usually a regularizer is adopted to minimize the difference between $X$ and $U + E_1$. When $E_2$ is assumed with a small percentage, it becomes irrational to force $U$ to approximate $X$ especially on anomalous locations. Instead, a conditional approximation is placed with the index matrix $M$, in which case only regular locations are required aligned. The target is then $\text{Reg}(M \odot (X - U))$ with some regularizer $\text{Reg}$, and it can be appended to the main objective by

$$
\min_U \alpha \text{Reg}(U) + \beta \text{Loss}(M \odot (X - U)) \quad (4)
$$

The optimization model for graph signal recovery is now well defined. Nevertheless, there are several issues with the existing design that prevents it from being practical. First and foremost, attaining the mask matrix $M$ in (4) can be nasty in practice, as the prior knowledge on $M$ is generally inaccessible, nor the full panorama of it. Also, smoothing a graph could be tricky in the sense that simply minimizing some modified Dirichlet energy could easily fall into the pitfall of over-smoothing, which drastically reduces the expressivity of the recovered graph (Balciar et al., 2020). On the other hand, the spectral transform can be sophisticated and time-consuming, which is generally circumvented by the majority of studies. Moreover, restricting a particular $\ell_1$ or $\ell_2$ norm on an arbitrary dataset or application sacrifices flexibility, which could give rise to the potential precision loss of the recovered graph representation $U$.

We thus propose a new objective function to best handle the identified problems. As introduced in (1), we define

$$
\min_U \|\nu \mathcal{W}U\|_{p,G} + \frac{1}{2} \|M \odot (U - X)\|_{q,G}.
$$

Here $\mathcal{W}$ is a set of framelet decomposition operators at different levels and scales that transform the graph feature.
to the framelet (spectral) domain, and $\nu$ includes the set of tuning parameters for each-level’s framelet transform in $\mathcal{W}$. A brief explanation on the construction of $\mathcal{W}$ is prepared in Appendix A. Note that we replace the ordinary Euclidean $\ell_k$-norm with $\ell_{k,G}$, a graph $\ell_k$-norm, that assigns higher penalties to influential nodes. For a particular node $v_i$ of degree $D_{ii}$, $\|v_i\|_{k,G} := \left(\|v_i\|_k \cdot D_{ii}\right)^{\frac{1}{k}}$.

Compared to the initial design (3), we make three main adjustments to tackle the above-mentioned issues. Instead of minimizing a graph signal’s Dirichlet energy in the spatial domain, we leverage fast graph framelet transform (Dong, 2017; Zheng et al., 2021) to proceed with an efficient multi-level decomposition and smooth out $E_1$ from high-pass information. Through restricting the sparsity of the high-frequency framelet coefficients, the global fluctuation under assumption (a) is removed. Meanwhile, operating multiscale signal processing preserves sufficient detailed information, which prevents the new graph representation from global smoothness. The second new ingredient relaxes the predetermined regularizer to a flexible $\ell_{p,q}$ norm where $p \in [1, 2]$ and $q \in [0, 1]$ are hyperparameters. The optimization can be solved by an inertial version of the alternating direction method of multipliers with promising convergence, which we will introduce in Section 4.

One remaining problem is the unreachable mask matrix $M$ in real-world applications. A potential solution could be adding a sub-problem of optimization to (1) (thus introducing 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 here consider a GNN to automatically discover the anomalous positions. To this end, we approximate the anomaly score of the raw feature matrix with the graph autoencoder, a classic graph anomaly detection scheme that looks for the reconstruction error by differentiating the raw feature matrix and the reconstructed matrix from neural networks. We now present the construction of such an unsupervised mask detector.

Remark 1 Compared to (4), (1) transfers the requirement of sparse $E_2$ to $M$ in the $\ell_q$ norm of the fidelity term. In addition, the level of sparsity for $M$ is controlled by the GAE module and the threshold thereafter. We shall provide more explanations in the next section.

Remark 2 The three hyperparameters $\alpha, \beta, \gamma$ in (4) are deduced to $\nu$, which is a set of tunable parameters adaptive to high-frequency framelet coefficients in different scales.

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 data. A detector identifies 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 $U$. 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). The graph version of autoencoder was first proposed by Kipf & Welling (2016), which revisions have drawn substantial interest for graph anomaly detection (Ding et al., 2019; Peng et al., 2020; Zhu et al., 2020). This section briefs the key components of a GCN-based graph autoencoder network.

Network Training A GCN-based graph autoencoder layer takes the information of $G$ to obtain an embedding $Z \in \mathbb{R}^{n \times h}$ of a hidden size $h$ by $Z = \text{GCN}(X, A)$. The extracted embedding can be used for reconstructing adjacency matrix $A' = \sigma(ZZ^\top)$ for link predictions. Alternatively, one trains a feed-forward network to recover graph attributes by $X' = \text{FFN}(Z)$. The $X'$ is a smoothed version of $X$, 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 $(X', A')$, which loss function is the weighted average of the squared reconstruction errors

$$\mathcal{L} = \alpha \|X - X'\|_2 + (1 - \alpha) \|A - A'\|_2.$$ 

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 $v_i$ reads

$$\text{score}(v_i) = \alpha \|x_i - x'_i\|_2 + (1 - \alpha) \|a_i - a'_i\|_2.$$ 

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

$$M = 1 - \text{threshold}(\|X - X'\|_1, \tau)$$ (5)
4. An Inertia Alternating Direction Method of Multipliers (ADMM)

In this section we introduce the numerical scheme to solve the proposed model (1).

4.1. An inertial ADMM

Denote $Z = \mathbf{WU}$, then we can rewrite (1) as

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

s.t. $Z = \mathbf{WU}.$

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

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

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

$$
Z_{k+1} = \arg\min_Z \|\nu Z\|_{p, G} + \frac{1}{2}\|M \odot (U - X)\|_{q, G}^q,
+ \langle Y_k, \mathbf{WU} - Z \rangle
+ \frac{\gamma}{2}\|\mathbf{WU} - Z\|^2,
$$

$$
U_{k+1} = \arg\min_U \frac{1}{2}\|M \odot (U - X)\|_{q, G}^q,
+ \frac{\gamma}{2}\|\mathbf{WU} - Z_{k+1}\|^2 + \langle Y_k, \mathbf{WU} - Z_{k+1} \rangle,
Y_{k+1} = Y_k + \gamma(\mathbf{WU}_{k+1} - Z_{k+1}).
$$

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

$$
Z_{k+1} = \arg\min_Z \|\nu Z\|_{p, G} + \frac{1}{2}\|Z - (2Y_k - \tilde{V}_k)/\gamma\|^2,
$$

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

$$
\tilde{V}_{k+1} = V_{k+1} + a_k(V_{k+1} - V_k),
$$

$$
U_{k+1} = \arg\min_U \frac{1}{2}\|M \odot (U - X)\|_{q, G}^q
+ \frac{\gamma}{2}\|\mathbf{WU} + \tilde{V}_{k+1}/\gamma\|^2,
Y_{k+1} = \tilde{V}_{k+1} + \gamma\mathbf{WU}_{k+1},
$$

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

4.2. Subproblems

The solutions to the subproblems of $Z_{k+1}$ and $U_{k+1}$ 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 $Z_{k+1}$. For the case of $p = 1$, $\|\nu Z\|_{p, G}$ becomes the $\ell_1$-norm and its update requires a soft-thresholding, that is,

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

When $p = 0$, the following hard-thresholding is applied:

$$
H_\alpha(x) = \begin{cases} x & |x| > \alpha \\ 0 & |x| \leq \alpha. \end{cases}
$$

**Choices of $q$** Compared to $Z_{k+1}$, the update of $U_{k+1}$ in (8) is more complicated as it involves more regularization terms. When $q = 2$, the objective is a quadratic problem. With $\mathbf{W}^\top \mathbf{W} = \text{Id}$, differentiating the function gives

$$
U_{k+1} = \frac{M \odot X - \mathbf{W}^\top \tilde{V}_{k+1}}{M + \gamma},
$$
Table 1. Average performance for node classification with non-targeted local corruption

| 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., 2021b) | **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 |

which requires an element-wise division. Since the fast approximation of \( \mathbf{W} \) is implemented by the Chebyshev approximation, it happens when the approximation degree is considerably small that the approximation has noticeable error, i.e., \( \mathbf{W}^T \mathbf{W} \neq \mathbf{I} \). 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,

\[
U^{(i+1)} = U^{(i)} - \alpha \left( M \odot (U^{(i)} - X) + \gamma \mathbf{W}^T (\mathbf{W}U^{(i)} + \tilde{V}_{k+1}/\gamma) \right), \tag{10}
\]

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

In the case of \( q = 1 \), let \( \mathbf{Q} = U - X \) and consider

\[
\frac{1}{2} \| M \odot \mathbf{Q} \|_{1,G} + \frac{\gamma}{2} \| \mathbf{Q} + \mathbf{X} + \mathbf{W}^T \tilde{V}_{k+1}/\gamma \|^2
\]

under the optimality condition. The \( Q_{k+1} \) is the soft-thresholding of \( -X - \mathbf{W}^T \tilde{V}_{k+1}/\gamma \) and

\[
U_{k+1} = Q_{k+1} + X. \tag{11}
\]

5. Numerical Experiments

This section validates MAGNET on various dimensions. We compare our methods to three baselines on node classification tasks with two types of attribute corruptions. The ablation study reports model performance with different choices of \( p, q \), and at different levels of corruption. The effect of the two learning stages, GAE approximation and ADMM optimization, are discussed then for establishing an intuitive understanding of the two modules.

5.1. Experimental Protocol

Benchmark Preparation We examine MAGNET on five benchmark datasets: **Cora**, **Citeeseer** 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 literature. The first attribute injection method has been considered by Song et al. (2007); Ding et al. (2019), which 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 that leverages meta-learning to pollute node attributes with the meta-gradient of the loss function (Zügner & Günnemann, 2019).

Training Setup For a fast recovering from the corrupted graph, an inertial ADMM is iterated for 15 times. The GAE to approximate binarized 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 A complete investigation on MAGNET is conducted with different forms of the mask matrix. We name them as MAGNET-one, MAGNET-gae, and MAGNET-true to represented all-ones mask, GAE-approximated mask, and ground truth mask. Note that the last case does not participate in the performance comparison. Instead, it indicates the upper limit of our model when establishing the perfect mask matrix. In addition, 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., 2021b) that pursues local smoothness with a mix of \( \ell_1 \) and \( \ell_2 \) regularizers.

5.2. Node Classification with Graph Recovery

The first experiment compares the performance of MAGNET with the baselines on node classification tasks in Table 1.
and Table 2. As the attribute poison of meta-attack is less effective with Wiki-CS and Coauthor-CS, Table 2 only report model performance on the citation networks. To display the direct advantage of each method over the corrupted baseline, we highlight each score’s relative improvement or retrogression in red or green, where darker colors indicate a greater change over the corrupted results. Precise improving percentages are in Table 5 and Table 6 of Appendix C.

It can be observed from the tables that MAGNET-gae outperforms its competitors and recovers at most 94% prediction accuracy from the perturbed attributes. Not to mention it has further potential to make improvements by more accurate mask approximation. As is indicated by the supplementary report model performance on the citation networks. To dis-

Figure 3. The performance comparison of MAGNET to baseline GCN with different levels of attribute injection to Cora.

tribution. The next two experiments visualize the effect of the GAE-oriented mask approximation and ADMM optimization.

To understand the preference of p, q’s choice in different scenarios, we also deliver 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, which is similar to the first experiment. Due to the limit space of the paper, we prepare the results in Table 7 and Table 8 of Appendix C. Generally speaking, $p = 1, q = 2$ 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 frequent-adopted choice in conventional denoising tasks. When the mask is considerably reliable, $p \in \{0, 1\}$ and $q = 1$ both achieve satisfactory performance. To be specific, $p = 0$ is possibly more suitable to the case of perturbations that are concentrated in a few entities, while $p = 1$ fits the case when the corruptions are rather diverse.

5.4. Investigation on the Individual Modules

The next two experiments 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 4 pictures the conditional mask matrix from model reconstruction error on Cora with attribute injection. A $200 \times 200$ 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

| Module     | CORA     | CITEXER  | PUBMED   |
|------------|----------|----------|----------|
| clean      | 72.02±0.75| 71.11±0.95| 70.63±1.07|
| corrupted  | 72.02±0.75| 71.11±0.95| 70.63±1.07|
| APPNP      | 72.02±0.75| 71.11±0.95| 70.63±1.07|
| GNNGUARD   | 72.02±0.75| 71.11±0.95| 70.63±1.07|
| ELASTICGNN | 72.02±0.75| 71.11±0.95| 70.63±1.07|
| MAGNET-gae (ours) | 72.02±0.75| 71.11±0.95| 70.63±1.07|
| MAGNET-gae (ours) | 72.02±0.75| 71.11±0.95| 70.63±1.07|
| MAGNET-true | 80.88±0.37| 67.46±0.95| 79.16±0.41|

Table 2. Average performance for node classification with targeted local corruption

5.3. Ablation Study

The second experiment investigates the performance of MAGNET with two variations. Figure 3 explores the influence of corruption level (and thus sparsity level of the ground truth mask) on the model performance with different choices of regularizers. We conduct our experiment on Cora with a diverse corruption rate from 0.1% to 1.25%. The comparison results to the baseline (two layers of GCN) are reported in Figure 3 with best-tuned $p$ and $q$. With an increasing level of perturbation, the performance gain of MAGNET over baseline keeps raising.

The next best choice is $p = 1, q = 1$, which is a frequent-adopted choice in conventional denoising tasks. When the mask is considerably reliable, $p \in \{0, 1\}$ and $q = 1$ both achieve satisfactory performance. To be specific, $p = 0$ is possibly more suitable to the case of perturbations that are concentrated in a few entities, while $p = 1$ fits the case when the corruptions are rather diverse.
ADMM Optimization Figure 5 visually exhibits the effect of ADMM optimization on local graph inpainting with 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 \( N(0, 1) \) 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. 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 (Bruna et al., 2014; Defferrard et al., 2016; Xu et al., 2018a; Zheng et al., 2021) or spatial-based (Kipf & Welling, 2017; Veličković et al., 2018; Hamilton et al., 2017; Monti et al., 2017; Li et al., 2018) methods. While the widespread message passing scheme (Gilmer et al., 2017) encourages continuous development on the spatial method, the adjacency matrix-based propagation pursues feature smoothness with minimum Dirichlet energy, which establishes a low-pass filter that denoises global under \( l_2 \) penalty (Nt & Maehara, 2019; Zhu et al., 2021; Dong et al., 2021). When multiple convolution layers are stacked in a neural network, it is inevitable to make the learned representation suffer from the issue of over-smoothing. Therefore, residual connections (Xu et al., 2018b; Klicpera et al., 2018; Rong et al., 2019; Chen et al., 2020b) are leveraged to maintain feature proximity but at the cost of concurrently keeping odd examples. Alternatively, robust graph optimization 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). Such methods are essentially designed for defending graph adversarial attacks, which usually influence model performance by corrupting node connections. Consequently, a robust node embedding is usually approximated through refined adjacency matrix (Entezari et al., 2020; Zhang & Zitnik, 2020; Jin et al., 2020; Chen et al., 2020c). In contrast, little research involves explicitly detecting anomalous node features. Rather, they modify the loss function to force local smoothness of the feature space (Liu et al., 2021a; b). On the other hand, graph anomaly detection (Ding et al., 2019; Peng et al., 2020; Zhu et al., 2020; Ma et al., 2021) is designed to identify the small portion of problematic nodes, but the related methods are merely recovering robust representation of anomalies. On top of that, the justification is based on node entities rather than the feature values.

7. Conclusion

We develop MAGNET, a graph neural network for recovering graph data representation from local corruptions on the feature 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 region in graph node attributes. As the anomaly mask merely has prior information, we apply an unsupervised graph autoencoder to approximate it in advance. The optimization problem is decorated with an \( l_{p,q} \) regularization, where \( p, q \) are user tuned to stimulate the maximum ability of the optimizer, which is programmed by an efficient inertial ADMM. In the \( l_p \) regularizer, graph tight framelets are used to represent the feature in various scales. The decomposition of the graph data to the approximate and detailed parts guarantees an excellent recovery tool at locally masked regions. Our proposed model achieves state-of-the-art performance on graph
Graph Neural Network for Local Corruption Recovery

recovery tasks from local corruptions, where the majority of competitors fail to provide a decent solution.

References

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

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.

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

Bot, 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.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *proceedings of International Conference on Learning Representations*, 2014.

Chen, L., Li, J., Peng, J., Xie, T., Cao, Z., Xu, K., He, X., and Zheng, Z. A survey of adversarial learning on graphs. *arXiv:2003.05730*, 2020a.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020b.

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, Y., Wu, L., and Zaki, M. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Advances in Neural Information Processing Systems*, 2020c.

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.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *proceedings of Advances in Neural Information Processing Systems*, 2016.

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.

Dong, Y., Ding, K., Jalaian, B., Ji, S., and Li, J. Adagnn: Graph neural networks with adaptive frequency response filter. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. Association for Computing Machinery, 2021.

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., and Papalexakis, E. E. 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*, pp. 169–177, 2020.

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.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *proceedings of Advances in Neural Information Processing Systems*, 2017.

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

Jin, W., Li, Y., Xu, H., Wang, Y., Ji, S., Aggarwal, C., and Tang, J. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34, 2021.
Graph Neural Network for Local Corruption Recovery

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 Günemann, 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, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In proceedings of AAAI Conference on Artificial Intelligence, 2018.

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., Ding, J., Jin, W., Xu, H., Ma, Y., Liu, Z., and Tang, J. Graph neural networks with adaptive residual. In Advances in Neural Information Processing Systems, 2021a.

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

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 Vanderheynst, 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.

Nt, H. and Maehara, T. Revisiting graph neural networks: All we have is low-pass filters. arXiv:1905.09550, 2019.

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.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In International Conference on Learning Representations, 2019.

Sandryhaila, A. and Moura, J. M. Discrete signal processing on graphs: Frequency analysis. IEEE Transactions on Signal Processing, 62(12):3042–3054, 2014.

Shchur, O., Mumme, M., Bojchevski, A., and Günemann, 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.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In proceedings of International Conference on Learning Representations, 2018.

Wang, Y. G. and Zhuang, X. Tight framelets on graphs for multiscale data analysis. In Wavelets and Sparsity XVIII, volume 11138, pp. 111380B. International Society for Optics and Photonics, 2019.

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, 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, 2020.

Xu, B., Shen, H., Cao, Q., Qiu, Y., and Cheng, X. Graph wavelet neural network. In proceedings of International Conference on Learning Representations, 2018a.

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, 2018b.

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.

Zheng, X., Zhou, B., Wang, Y. G., and Zhuang, X. Decimated framelet system on graphs and fast g-framelet transforms. *Journal of Machine Learning Research*, 23 (18):1–68, 2022.

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 briefly 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 $x$ at scale $j \in \mathbb{Z}$, we define the undecimated framelets $\varphi_{j,p}(g), \psi_{j,p}(g)$ by

$$
\varphi_{j,p}(g) := \sum_{t=1}^{N} \tilde{\alpha} \left( \frac{2^j}{\pi} \right) u_t(p) u_t(v), \\
\psi_{j,p}(g) := \sum_{t=1}^{N} \tilde{\beta}(r) \left( \frac{2^j}{\pi} \right) u_t(p) u_t(v), \quad r = 1, \ldots, n.
$$

(12)

For two integers $J, J_1 (J > J_1)$, an undecimated framelet system from a scale $J_1$ UFS $\Psi$ is an undecimated tight frame for $l_2(G)$, which is a non-homogeneous, stationary affine system:

$$
\text{UFS}_{J_1}(\Psi, \eta) := \text{UFS}_{J_1}(\Psi, \eta; G) := \{ \varphi_{j_1,p} : p \in V \} \cup \{ \psi_{j_1,p} : p \in V, j = J_1, \ldots, J \}_{r=1}^{n}.
$$

(13)

The elements in UFS $\Psi$ are the undecimated tight framelets on $G$.

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

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

The other component that plays a key role for embedding graph topology is the eigenpairs $\{ (\lambda, u) \}_{j=1}^{n}$ 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 $\alpha$ and $\{ \beta^{(1)}, \ldots, \beta^{(K)} \}$ by $T_m^0$ and $\{ T_k^m \}_{k=1}^{K}$, the framelet decomposition operators at $(r, j) \in \{(1, 1), \ldots, (1, J), \ldots, (n, 1), \ldots, (n, J) \}$ is defined by

$$
\mathcal{W}_{k,1} = \begin{cases}
T_r \left( 2^{-K} \mathcal{L} \right), & j = 1 \\
T_r \left( 2^{K-j-1} \mathcal{L} \right) T_0 \left( 2^{K-j-2} \mathcal{L} \right) \ldots T_0 \left( 2^{-K} \mathcal{L} \right), & \text{otherwise,}
\end{cases}
$$

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 $\lambda_{j}/2^{K+j-j} \in (0, \pi)$ for $j = 1, 2, \ldots, n$.

The approximated $\mathcal{W}$ is used in the penalty term of the objective function (1) 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 of the main text.

B.1. Inertial ADMM

We denote $Z = \mathcal{W}U$ and rewrite (1) as

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

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

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

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

$$
Z_{k+1} = \arg \min_{Z} \|\nu Z\|_{p,G} + \frac{1}{2} \|\mathcal{W}U_k - Z\|^2 + \langle Y_k, \mathcal{W}U_k - Z \rangle,
$$

$$
U_{k+1} = \arg \min_{U} \frac{1}{2} \|M \odot (U - X)\|_{q,G}^q + \frac{1}{2}\|\mathcal{W}U - Z_{k+1}\|^2 + \langle Y_k, \mathcal{W}U - Z_{k+1} \rangle,
$$

$$
Y_{k+1} = Y_k + \gamma (\mathcal{W}U_{k+1} - Z_{k+1}).
$$

The above iteration can be equivalently written as

$$
Z_{k+1} = \arg \min_{Z} \|\nu Z\|_{p,G} + \frac{1}{2} \|\mathcal{W}U_k - Z + Y_k / \gamma\|^2,
$$

$$
U_{k+1} = \arg \min_{U} \frac{1}{2} \|M \odot (U - X)\|_{q,G}^q + \frac{1}{2}\|\mathcal{W}U - Z_{k+1} + Y_k / \gamma\|^2,
$$

$$
Y_{k+1} = Y_k + \gamma (\mathcal{W}U_{k+1} - Z_{k+1}).
$$

If we further define

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

The above iteration can be reformulated as

$$
Z_{k+1} = \arg \min_{Z} \|\nu Z\|_{p,G} + \frac{1}{2} \|Z - \mathcal{W}U_k - Y_k / \gamma\|^2,
$$

$$
= \arg \min_{Z} \|\nu Z\|_{p,G} + \frac{1}{2} \|Z - (2Y_k - V_k) / \gamma\|^2,
$$

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

$$
U_{k+1} = \arg \min_{U} \frac{1}{2} \|M \odot (U - X)\|_{q,G}^q + \frac{1}{2}\|\mathcal{W}U + V_{k+1} / \gamma\|^2,
$$

$$
Y_{k+1} = Y_k + \gamma (\mathcal{W}U_{k+1} - Z_{k+1}) = V_{k+1} + \gamma \mathcal{W}U_{k+1}.
$$

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

$$
Z_{k+1} = \arg \min_{Z} \|\nu Z\|_{p,G} + \frac{1}{2} \|Z - (2Y_k - \tilde{V}_k) / \gamma\|^2,
$$

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

$$
\tilde{V}_{k+1} = V_{k+1} + a_k (V_{k+1} - V_k),
$$

$$
U_{k+1} = \arg \min_{U} \frac{1}{2} \|M \odot (U - X)\|_{q,G}^q + \frac{1}{2}\|\mathcal{W}U + \tilde{V}_{k+1} / \gamma\|^2,
$$

$$
Y_{k+1} = Y_k + \gamma (\mathcal{W}U_{k+1} - Z_{k+1}) = \tilde{V}_{k+1} + \gamma \mathcal{W}U_{k+1}.
$$

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 $Q = U - X$, and consider

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

The optimality condition yields

$$\frac{1}{2} \| M \odot \partial \| M \odot Q \|_{1,G} + \gamma \mathcal{W}^T (\mathcal{W}Q + \mathcal{W}X + \tilde{V}_{k+1}/\gamma)$$

$$\iff \frac{1}{2} \| M \odot Q \|_{1,G} + \gamma (Q + X + \mathcal{W}^T \tilde{V}_{k+1}/\gamma)$$

$$\iff \frac{1}{2} \| M \odot Q \|_{1,G} + \frac{\gamma}{2} \| Q + X + \mathcal{W}^T \tilde{V}_{k+1}/\gamma \|^2.$$

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

$$U_{k+1} = Q_{k+1} + X.$$
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, Citess, and Pubmed), a dataset with (relatively) dense edge connection (wiki-cs), and a dataset with (relatively) high dimension of feature attributes (coauthor-cs).

|                  | 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
Graph Neural Network for Local Corruption Recovery

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     |

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

| Module                 | CORA     | CITESEER | PUBMED | COAUTHOR-CS | WIKI-CS |
|------------------------|----------|----------|--------|-------------|---------|
|                        | 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%  |
| 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-core (ours)     | 9.78%   | 55.68%  | 2.85%  | 11.31%      | 1.80%   | 10.80%  | 1.98%    | 20.95%   | 8.24%    | 44.25%   |
| MAGNET-gae (ours)      | 11.13%  | 63.24%  | 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%   |

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:

C.4. Additional Experiment Results: 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 $S_{\text{clean}}$ and $S_{\text{corrupted}}$ are the accuracy score of GCN on the clean dataset and corrupted dataset, respectively. In addition, we report the absolute score of improvement, which is defined by

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

Both scores are reported in Table 5 below. In the main paper, we use the relative score to highlight the performance in Table 1 and Table 2.

C.5. Additional Experiment Results: Ablation Study

This section reports the performance of the ablation models on the node classification tasks. Due to the limit of paper length, we only showcase the line plot of the performance comparison on Cora with various sparsity threshold for the ablation models. While it make sufficient comparison against different selections of $p$ and $q$, as well as a sketchy analysis on the sensitivity to the threshold of the mask matrix, we hereby present more on 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 6. Improvement percentage of average performance for node classification with targeted local corruption

| Module          | CORA     | CITESEER | PUBMED   |
|-----------------|----------|----------|----------|
|                 | absolute | relative | absolute | relative | absolute | relative |
| APPNP           | -2.10%   | 0.63%    | -3.09%   | -36.70%  |
| GNNguard        | -4.06%   | 4.19%    | -2.44%   | -29.04%  |
| ELASTICGNN      | 5.57%    | 21.64%   | -1.28%   | -15.17%  |
| MAGNET-one (ours)| 2.72%    | 12.96%   | 4.05%    | 48.12%   |
| MAGNET-gae (ours)| 5.29%    | 21.84%   | 7.89%    | 93.80%   |
| MAGNET-true     | 7.74%    | 21.95%   | 8.62%    | 102.45%  |

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

| Module | Dataset             | reg (p) | loss (q) | CORA    | CITESEER | PUBMED  | CoAUTHOR-CS | WIKI-CS |
|--------|---------------------|---------|----------|---------|----------|---------|-------------|---------|
| N/A    | L₂                  |         |          |         |          |         |             |         |
| ONE    | L₀ L₁               | L₁ L₂   |          |         |          |         |             |         |
|        | 58.36±0.89          | 57.30±0.74 | 65.77±1.03 | 68.74±0.22 | 54.07±1.23 | 58.52±1.02 | 62.12±0.23 | 59.15±0.85 | 83.88±0.55 | 64.67±0.29 |
| GAE    | L₀ L₁               | L₁ L₂   |          |         |          |         |             |         |
|        | 68.42±1.15          | 54.38±0.54 | 67.74±0.71 | 72.76±0.40 | 63.60±0.66 | 75.41±0.35 | 72.40±0.21 | 68.12±0.40 | 75.40±0.28 | 66.50±0.37 | 60.08±0.39 |
| TRUE   | L₀ L₁               | L₁ L₂   |          |         |          |         |             |         |
|        | 77.15±0.74          | 68.14±0.85 | 74.40±0.56 | 76.44±0.59 | 65.02±0.97 | 71.15±0.25 | 74.20±0.29 | 71.51±0.20 | 75.50±0.20 | 74.44±0.23 | 60.08±0.39 |

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

| Module | Dataset             | reg (p) | loss (q) | CORA    | CITESEER | PUBMED  |
|--------|---------------------|---------|----------|---------|----------|---------|
| N/A    | L₂                  |         |          |         |          |         |
| ONE    | L₀ L₁               | L₁ L₂   |          |         |          |         |
|        | 71.21±1.12          | 55.38±2.12 | 71.52±0.43 | 74.46±0.60 | 57.75±2.23 | 59.76±1.91 | 77.04±0.45 | 62.49±1.70 | 75.89±0.35 |
| GAE    | L₀ L₁               | L₁ L₂   |          |         |          |         |
|        | 74.46±0.60          | 57.75±2.23 | 59.76±1.91 | 78.18±0.56 | 63.22±1.56 | 78.63±0.32 | 78.42±0.65 | 78.42±0.65 | 70.82±0.38 | 78.63±0.32 |
| TRUE   | L₀ L₁               | L₁ L₂   |          |         |          |         |
|        | 77.42±1.08          | 56.13±0.47 | 78.42±0.65 | 77.04±0.50 | 67.40±0.73 | 74.47±0.30 | 78.42±0.65 | 78.42±0.65 | 70.82±0.38 | 78.63±0.32 |

Graph Neural Network for Local Corruption Recovery
### Table 9. Hyperparameter searching space for node classification.

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

#### C.6. Additional Experiment Results: GAE visualization

Figure 6-Figure 8 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.

#### C.7. Additional Experiment Results: 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., $\gamma$ in (8) equals to 0 or not. Below we report the running time of a complete update process under certain convergent condition $\epsilon$, i.e., $\|U_{k+1} - U_k\| < \epsilon$. The speed is counted in seconds.
Graph Neural Network for Local Corruption Recovery

Figure 6. citeseer

Figure 7. first 3000 rows and columns only. The full matrix is too larger to display (18000x7000)

Figure 8. pubmed: first 3000 nodes; 500 features; threshold=0.005

Figure 9. Wikics: first 3000 nodes; 300 features; threshold=0.05