Beyond Homophily: Robust Graph Anomaly Detection via Neural Sparsification

Zheng Gong[1,2*] , **Guifeng Wang**[3,†] , **Ying Sun**[4] , **Qi Liu**[1,2] ,
Yuting Ning[1,2] , **Hui Xiong**[4] and **Jingyu Peng**[1,2]
[1]School of Computer Science and Technology, University of Science and Technology of China
[2]State Key Laboratory of Cognitive Intelligence
[3]Huawei Technologies Co Ltd
[4]Hong Kong University of Science and Technology (Guangzhou)
{gz70229, ningyt, jypeng28}@mail.ustc.edu.cn, wangguifeng4@huawei.com,
{yings, xionghui}@ust.hk, qiliuql@ustc.edu.cn

Abstract

Recently, graph-based anomaly detection (GAD) has attracted rising attention due to its effectiveness in identifying anomalies in relational and structured data. Unfortunately, the performance of most existing GAD methods suffers from the inherent structural noises of graphs induced by hidden anomalies connected with considerable benign nodes. In this work, we propose SparseGAD, a novel GAD framework that sparsifies the structures of target graphs to effectively reduce noises and collaboratively learns node representations. It then robustly detects anomalies by uncovering the underlying dependency among node pairs in terms of homophily and heterophily, two essential connection properties of GAD. Extensive experiments on real-world datasets of GAD demonstrate that the proposed framework achieves significantly better detection quality compared with the state-of-the-art methods, even when the graph is heavily attacked. Code will be available at https://github.com/KellyGong/SparseGAD.git.

1 Introduction

Graph-based Anomaly Detection (GAD) refers to identifying anomalies that deviate significantly from the majority of objects in relational and structured data. With graph data becoming ubiquitous and ever-growing, graph-based anomaly detection has received increasing attention on account of its vast applications, such as spammer recognition [Ye and Akoglu, 2015], financial fraudster identification [Weber *et al.*, 2019] and sensor fault detection [Gaddam *et al.*, 2020]. Due to the complex interactions between nodes in real-world systems (e.g., partnership in a social network or transactions on a financial platform), detecting anomalies within graph data becomes more challenging than anomaly detection in non-interaction feature space (e.g., image).

Most recently, Graph Neural Networks (GNNs) serve as popular approaches for graph-structure data and are naturally applied in GAD [Dou *et al.*, 2020; Liu *et al.*, 2021]. Driven by the message passing scheme, standard GNN methods usually recursively aggregate and transform the representations of neighbors for each ego node and categorize the abnormal cases. However, many investigations [Dou *et al.*, 2020; Liu *et al.*, 2020] discover that anomalies tend to camouflage themselves and mitigate suspiciousness by connecting with substantial benign entities. Considering a social network as an instance, where nodes represent users and edges indicate relationships, fraudsters establish insincere links to real users, and they can effortlessly inject wrong structure information into the entire network, which leads to poor performance of estimating user credibility via the plain GNNs.

[*]Work done during an internship at Huawei.
[†]Corresponding Author.

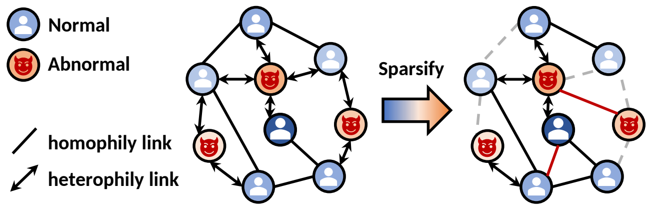

Figure 1: An illustration of our proposed SparseGAD for sparsifying graphs in GAD and detecting anomalies in consideration of homophily and heterophily connections.

To remedy this issue, the existing GAD methods can be generally divided into three categories. That is, (1) applying resampling strategy to selectively aggregate neighborhood information by measuring the label similarity of node pairs [Dou *et al.*, 2020; Liu *et al.*, 2021], (2) designing spectral-based architectures based on adaptive low and high-passing filters [Chai *et al.*, 2022; Tang *et al.*, 2022], (3) adding auxiliary loss to strengthen the expressive power of network [Ding *et al.*, 2019; Zhao *et al.*, 2020].

Although some encouraging progress has been achieved, the capability of anomaly detectors against noisy graph structures remains to be further investigated and enhanced, since the unknowable and ever-changing behaviors of anomalies hinder anomaly detection and lead to poor robustness [Chang and Chang, 2014]. A natural and fundamental question arises here: *how to resist structure noises of GAD for detecting anomalies robustly?* As GNNs are fragile when graphs have disturbing edges, our work is motivated by the intuition that sparsifying graph structures by selecting task-relevant edges not only enhances the detection robustness but also facilitates discovering essential neighbors, for example, an accomplice of fraudsters. Furthermore, there are indeed two disparate and essential connection properties, i.e., *homophily* and *heterophily* in GAD. Specifically, the connected nodes have similar attributes or the same category labels, called homophily, while the connections between dissimilar nodes in terms of attributes or labels are referred to as heterophily. In real-life graphs for anomaly detection, such as Amazon [McAuley and Leskovec, 2013] and Reddit [Kumar *et al.*, 2019], the connections involving anomalies or benign entities are diverse in homophily and heterophily. How to sparsify graph structures and aggregate neighborhood information under such complex interactions remains to be further explored.

With these in mind, we propose a neural sparsification framework for GAD (short for SparseGAD). As illustrated in Figure 1, under the framework, we can effectively optimize the sparsification of graphs and collaboratively learn node representations in consideration of homophily and heterophily, which, to the best of our knowledge, has not been studied before. In practice, our SparseGAD has a three-stage process for this task. First, we exploit a GNN to infer a novel adjacency matrix where the value of each entry can represent the informativeness and properties of neighbors to ego nodes. Second, we sparsify the graph structure by removing task-irrelevant edges and discovering the closely related nodes as neighbors. Third, we share the parameters of GNN in the first stage, conduct the heterophily-aware aggregation scheme to represent each node, and categorize suspicious cases. Our extensive experiments on real-world datasets of GAD demonstrate the superiority and robustness of SparseGAD.

2 Related Work

Graph Neural Networks (GNNs). GNNs [Kipf and Welling, 2017; Veličković *et al.*, 2018; Hamilton *et al.*, 2017] have demonstrated their capacity in various machine learning tasks built upon graph structure data [Wang *et al.*, 2018; Wang *et al.*, 2019], such as molecular property prediction [Zhang *et al.*, 2021], talent acquisition [Sun *et al.*, 2021] and point of interests retrieval [Huang *et al.*, 2021]. Some extensively-used GNNs, such as Graph Convolution Network (GCN) [Kipf and Welling, 2017] implicitly leverage the homophily assumption, resulting in the over-smooth problem with the increasing depth of GNN layers. The common philosophy behind non-homophilic (a.k.a heterophilic) approaches is to weaken the smooth effect. For instance, APPNP [Gasteiger *et al.*, 2019] and GCNII [Chen *et al.*, 2020a] average the GNN-smoothed node representations with the original linear-transformed node embeddings. FAGCN [Bo *et al.*, 2021] and GPR-GNN [Chien *et al.*, 2021] assign learnable weights that can be positive or negative in the aggregation of neighbors' representations and layers' representations respectively.

Graph Structure Learning (GSL). GSL methods aim at rebuilding or refining graph structure to boost GNN performance by following a general pipeline: the adjacency matrix of the graph is characterized with learnable parameters and then jointly optimized with GNN under the supervision of downstream tasks (e.g., node classification [Luo *et al.*, 2021]) or self-supervised signal (e.g., reconstruction error [Fatemi *et al.*, 2021] and contrastive loss [Liu *et al.*, 2022b]). Due to the discrete inherence of the graph structure, the key of GSL is how to parameterize the adjacency matrix. One type of methods [Fatemi *et al.*, 2021; Jin *et al.*, 2020] directly treats each element in the adjacency matrix as a learnable parameter under the assumption of edge independence, while another type of method selects the top-K similar nodes as the ego node's new neighbors by computing node-pair similarity [Liu *et al.*, 2022b; Yu *et al.*, 2020]. Besides, some methods [Jin *et al.*, 2021; Liu *et al.*, 2023] rebuild the topology of graphs thoroughly considering the homophily. However, most recent GSL methods are evaluated in homophilic scenarios, leaving works on heterophilic applications with abundant structure noise (esp. anomaly detection) unexplored.

GNN-based Anomaly Detection. Considering the trend that anomalies, such as financial fraudsters, connect to abundant normal users, which limits the power of conventional GNN (e.g., GCN) based on the homophilic assumption, various techniques are leveraged to alleviate the negative impact. From the spectral domain, AMNet [Chai *et al.*, 2022] and BWGNN [Tang *et al.*, 2022] both design multi-pass spectral filters to discover high-frequency anomalies. To select important neighbors, CARE-GNN [Dou *et al.*, 2020] and AO-GNN [Huang *et al.*, 2022] both use a Reinforcement Learning module by taking the neighbors' similarity measurements with ego nodes and AUC performance as rewards, respectively. While PC-GNN [Liu *et al.*, 2021] directly measures the anomaly probability gap by training an additional MLP with only the node attributes as input. Nevertheless, few works investigate the sparsification of the noisy graph structures in GAD to enhance the robustness of anomaly detection.

3 Preliminary and Problem Statement

In this section, we first introduce the conceptions of homophily and heterophily. Then we describe the specific definition of our paper objective, i.e., graph-based anomaly detection. After that, we introduce the GNN backbone of our SparseGAD framework, which can also be replaced by other advanced GNN architectures.

Homophily versus Heterophily. In homophilic graphs, connected node pairs tend to have same category labels or similar attributes. In contrast, the edges link two dissimilar nodes in the heterophilic graphs. Remarkably, the graphs in GAD embody homophilic and heterophilic edges together.

Graph-based Anomaly Detection. In this paper, we focus on attribute graphs for anomaly detection. Specifically, given an attribute graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathbf{X})$ as input, $\mathcal{V}$ is the node set with size $n = |\mathcal{V}|$, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the edge set. We denote the adjacency matrix of $\mathcal{G}$ as $\mathcal{A} \in \{0,1\}^{n\times n}$, where $\mathcal{A}_{ij} = 1$ if node v_i connects to node v_j, i.e., $e_{ij} \in \mathcal{E}$ and $\mathcal{A}_{ij} = 0$ otherwise, and $\mathcal{N}(v)$ is the neighbor set of node v. $\mathbf{X} \in \mathbb{R}^{n\times f}$ is the matrix of node attributes where each

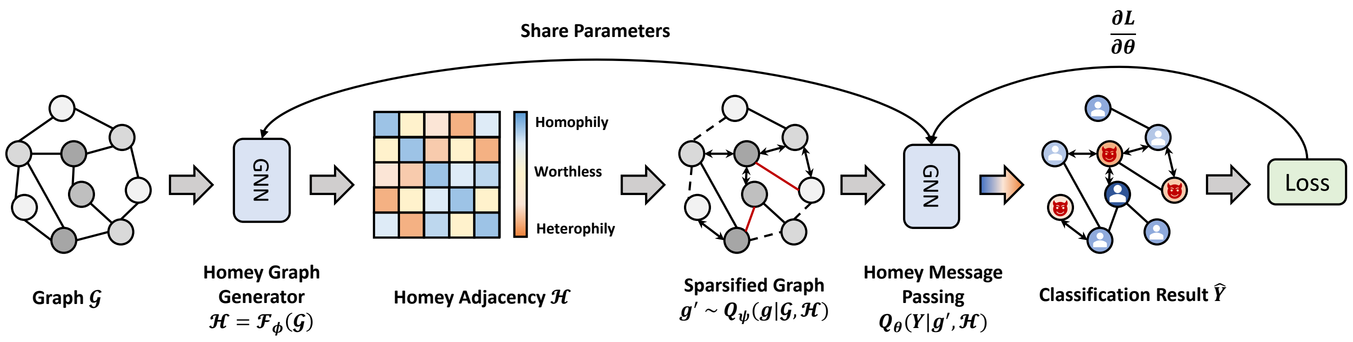
Share Parameters
$\frac{\partial L}{\partial \theta}$
Graph $\mathcal{G}$
GNN
Homey Graph Generator $\mathcal{H} = \mathcal{F}_\phi(\mathcal{G})$
Homophily
Worthless
Heterophily
Homey Adjacency $\mathcal{H}$
Sparsified Graph $g' \sim Q_\psi(g|\mathcal{G},\mathcal{H})$
Homey Message Passing $Q_\theta(Y|g',\mathcal{H})$
Classification Result $\hat{Y}$
Loss

Figure 2: The overview architecture of SparseGAD. Homey Graph Generator $\mathcal{F}$ with a GNN parameterized by ϕ creates *homey* adjacency matrix $\mathcal{H}$ to capture the informativeness (i.e., Assumption 1) and properties (i.e., homophily and heterophily in Assumption 2) of neighbors based on the unsparsified graph $\mathcal{G}$. The structure of $\mathcal{G}$ is then sparsified, and g' is a sparsified sample drawn from $Q_\psi(g|\mathcal{G},\mathcal{H})$. The final anomaly detection results are obtained on the grounds of g' and $\mathcal{H}$ via homey message passing, i.e., $Q_\theta(Y|g',\mathcal{H})$. Note that the parameters of GNN in homey graph generator $\mathcal{F}_\phi$ are shared with the GNN in homey message passing Q_θ, and the different grey levels of nodes in graph $\mathcal{G}$ reflect the diverse node attributes.

row vector $\mathbf{X}_v \in \mathbb{R}^f$ is the corresponding attributes of node v. A part of nodes $\mathcal{V}' \subseteq \mathcal{V}$ has a corresponding binary label $y_v \in \{0,1\}$, where 0 represents *benign entity* and 1 represents *anomaly*, while other nodes remain unknown. Considering notable structure noises arise in the neighbors of anomalies which harms the performance of GNNs [Dou *et al.*, 2020; Yang *et al.*, 2020], the task aims at predicting the suspiciousness of unlabeled nodes by training an anomaly detector on the labeled node information along with the sparsified subgraph g' of graph $\mathcal{G}$. The optimized graph topology of g' is expected to capture the underlying dependence of two nodes and thereby facilitate detecting anomalies robustly.

GNN Backbone. Generally speaking, our GNN is a function $f_{\text{GNN}} : \mathbf{X}, \mathcal{A} \rightarrow \{\mathbf{z}_v\}$, which maps to the representations of each node v with node attribute matrix $\mathbf{X}$ and adjacency matrix $\mathcal{A}$ as input. Specifically, we first transform the node attributes with a linear layer $\mathbf{h}_v^0 = \mathbf{W}_0\mathbf{X}_v + \mathbf{b}_0$, where $\mathbf{h}_v^0, \mathbf{b}_0 \in \mathbb{R}^d$, $\mathbf{W}_0 \in \mathbb{R}^{d\times f}$, and d is the hidden dimension of node representations. Then we utilize a GNN with K layers to aggregate neighbors' information and derive each node's representation. In the k-th layer of GNN, the representation $\mathbf{h}_v^k$ of node v is derived through:

$$\mathbf{h}_v^k = \sigma\Big(\big(\mathbf{h}_v^{k-1} + \sum_{u\in\mathcal{N}(v)} c_{uv}^k \mathbf{h}_u^{k-1}\big)\mathbf{W}^k\Big), \tag{1}$$

where $\mathbf{W}^k \in \mathbb{R}^{d\times d}$ are the learnable parameters, $u \in \mathcal{N}(v)$ is the neighbor of node v in the adjacency matrix $\mathcal{A}$, and σ is the activation function. Please note that the values of c_{uv}^k control the weights of message aggregation from neighbors, and we will illustrate two settings of c_{uv}^k, which play different roles, in subsection 4.3 and subsection 4.5.

Since the anomalies connect to substantial benign entities, over-smoothing neighbors' information limits the power of our model to detect abnormal nodes. Hence, we adopt the aggregation mechanism of Personalized PageRank-based GNN [Gasteiger *et al.*, 2019; Chien *et al.*, 2021] and obtain the aggregated representation $\mathbf{z}_v$ of node v:

$$\mathbf{z}_v = \sum\nolimits_{k=0}^{K} \gamma_k \mathbf{h}_v^k, \tag{2}$$

where $\gamma_k \in \mathbb{R}$ is a learnable parameter, and we initialize the parameters $\{\gamma_k\}_{k=0}^K$ as $\gamma_k = \alpha(1-\alpha)^k$ for $k \in [0, K-1]$, $\gamma_K = (1-\alpha)^K$ with α being a hyperparameter.

4 Method

4.1 Framework Overview

Real-life graphs in anomaly detection usually have complex information in the local neighborhood, where each node builds diverse relationships with dozens of neighbors [McAuley and Leskovec, 2013; Rayana and Akoglu, 2015]. Moreover, anomalies camouflage themselves by establishing false connections with substantial benign entities [Dou *et al.*, 2020; Liu *et al.*, 2020]. Hence, there exist two essential characteristics of GAD. First, neighbors have varying degrees of informativeness to estimate the ego node credibility due to complex relations, such as friendships and partnerships. We can thereby implement sparsified graph by selecting task-relevant edges to enhance the robustness of anomaly detection. Second, the behaviors of anomalies lead to the widespread heterophily in GAD, and thus distinguishing between homophilic and heterophilic neighbors contributes to identifying anomalies based on the message aggregation scheme of GNN.

Accordingly, to characterize the informativeness of neighbors considering **homo**phily and h**e**terophil**y**, we introduce a learnable adjacency matrix $\mathcal{H}$ (hereinafter simplified as *homey* for brevity), which can further facilitate reducing structure noises and aggregating neighborhood information to robustly detect anomalies. Specifically, assuming we have a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathbf{X})$, we derive $\mathcal{H}$ from $\mathcal{G}$ via homey graph generator $\mathcal{F}$ parameterized by ϕ, a deterministic function as $\mathcal{H} = \mathcal{F}_\phi(G)$. We incorporate $\mathcal{H}$ and formulate the GAD task from the perspective of statistical learning:

$$P(Y|\mathcal{G}) = P(Y|\mathcal{G},\mathcal{H}) \approx \sum_{g\in\mathbb{S}_\mathcal{G}} P(Y|g,\mathcal{H})P(g|\mathcal{G},\mathcal{H}), \tag{3}$$

where Y is the prediction target, i.e., anomaly detection, and $\mathbb{S}_\mathcal{G}$ is a set of sparsified graphs of $\mathcal{G}$. However, it is intractable to enumerate all possible $g \in \mathbb{S}_{\mathcal{G}}$, and thus we approximate the distributions $P(Y|g)$ and $P(g|\mathcal{G}, \mathcal{H})$ by tractable functions [Zheng *et al.*, 2020]:

$$\sum_{g \in \mathbb{S}_{\mathcal{G}}} P(Y|g, \mathcal{H}) P(g|\mathcal{G}, \mathcal{H}) \approx \sum_{g \in \mathbb{S}_{\mathcal{G}}} Q_\theta(Y|g, \mathcal{H}) Q_\psi(g|\mathcal{G}, \mathcal{H}), \quad (4)$$

where Q_θ and Q_ψ are approximation functions for $P(Y|g)$ and $P(g|\mathcal{G})$ parameterized by θ and ψ, respectively. Furthermore, to make the above graph sparsification differentiable, we apply reparameterization trick [Kool *et al.*, 2020] to generate the differentiable sample g' as:

$$\sum_{g \in \mathbb{S}_{\mathcal{G}}} Q_\theta(Y|g, \mathcal{H}) Q_\psi(g|\mathcal{G}, \mathcal{H}) \propto \sum_{g' \sim Q_\psi(g|\mathcal{G}, \mathcal{H})} Q_\theta(Y|g', \mathcal{H}), \quad (5)$$

where $g' \sim Q_\psi(g|\mathcal{G}, \mathcal{H})$ denotes $g' = (\mathcal{V}, \mathcal{A}', \mathbf{X})$ is a sparsified graph sampled from $Q_\psi(g|\mathcal{G}, \mathcal{H})$.

4.2 Framework Architecture

In this paper, we propose SparseGAD to implement Equation (5). As shown in Figure 2, SparseGAD consists of three essential components:

- Homey Graph Generator is a GNN with an MLP transformation that implements $\mathcal{H} = \mathcal{F}_\phi(G)$ to generate the homey adjacency matrix which captures the informativeness and properties of neighbors.
- Graph Sparsification is a sampling module that implements $g' \sim Q_\psi(g|\mathcal{G}, \mathcal{H})$ to sample sparsified graph g'.
- Homey Message Passing is a GNN that categorizes suspicious cases $Q_\theta(Y|g', \mathcal{H})$ with the sparsified graph g' and homey adjacency matrix $\mathcal{H}$ as input.

To enable end-to-end and stable training, we share the GNN parameters of homey graph generator with homey message passing and assign different values to c_{uv}^k in Equation (1) depending on the goals of each module. In the following subsections, we will demonstrate these components extensively.

4.3 Homey Graph Generator

The goal of homey graph generator is to generate homey adjacency matrix $\mathcal{H} = \mathcal{F}_\phi(\mathcal{G})$ to facilitate sparsifying graph $\mathcal{G}$ to reduce noises and aggregating messages of neighbors with homophily and heterophily into consideration. Practically, we first obtain node representations $\{\mathbf{z}_v\}$ with node attribute matrix $\mathbf{X}$ and adjacency matrix $\mathcal{A}$ of G as the input of GNN illustrated in Section 3. During this process, we set $c_{uv}^k = 1/\sqrt{(d_u + 1)(d_v + 1)}$ following GCN [Kipf and Welling, 2017], where d_u and d_v are the degrees of node u and v in adjacency matrix $\mathcal{A}$, respectively.

To capture the underlying dependence of node pairs and help discover the underlying neighbors of anomalies and benign entities, we bring in the versatile homey adjacency matrix $\mathcal{H}$, which accommodates the following assumptions:

Assumption 1. *The absolute edge weight $|\mathcal{H}_{uv}|$ shall reflect the informativeness of neighbor u to ego node v. When the neighbor u has no impact on node v, the absolute value $|\mathcal{H}_{uv}| \to 0$.*

Assumption 2. *The value $\mathcal{H}_{uv}$ can help distinguish whether the neighbor u is homophilic or heterophilic.*

Since the heterophilic links make up a significant portion of connections involving anomalies, appropriately capturing the connection properties contributes to the detection quality.

In practice, we use a two-layer MLP to transform the representation $\mathbf{z}_v$ of Equation (2) as $\mathbf{z}'_v = \mathrm{MLP}(\mathbf{z}_v)$ and calculate the cosine similarity of node pairs as the value of each entry of homey adjacency $\mathcal{H}$:

$$\mathcal{H}_{uv} = \frac{\mathbf{z}'_u \cdot \mathbf{z}'_v}{|\mathbf{z}'_u||\mathbf{z}'_v|} \in [-1, 1]. \quad (6)$$

Based on homey adjacency matrix $\mathcal{H}$, we distinguish neighbor u to ego node v as *homophilic*, *worthless* or *heterophilic*:

$$\begin{cases} \mathcal{H}_{uv} \to 1, & \text{if } u \text{ is homophilic to } v, \\ \mathcal{H}_{uv} \to 0, & \text{if } u \text{ is worthless to } v, \\ \mathcal{H}_{uv} \to -1, & \text{if } u \text{ is heterophilic to } v. \end{cases} \quad (7)$$

The value of $\mathcal{H}_{uv}$ is optimized by the learning objective in subsection 4.6, not directly supervised by the edge types [Shi *et al.*, 2022]. Unlike the attention mechanism applied in GAT [Veličković *et al.*, 2018] constraining the learned edge weight to be non-negative via a softmax nonlinear function, cosine similarity allows $\mathcal{H}_{uv}$ to zero-centered real values ranging in [-1, 1]. Note that cosine similarity can be replaced by other functions. In our experiment results, we find cosine similarity effective in the majority of datasets. Besides, cosine similarity is beneficial from two perspectives. That is, (1) keeping symmetry, i.e., $\mathcal{H}_{uv} = \mathcal{H}_{vu}$ without any other adjacency post-processor, like $(\mathcal{H} + \mathcal{H}^T)/2$ in some GSL works [Fatemi *et al.*, 2021; Liu *et al.*, 2022b], (2) containing no extra learnable parameters and effortlessly adapting to our graph sparsification module.

4.4 Graph Sparsification

Due to the camouflage behaviors of anomalies, structure noises inherently exist in unsparsified graph G. Hence, we sparsify structures of G to achieve the following targets. First, we remove unnecessary neighbors which may inject false information into the aggregation mechanism of GNN and decline the anomaly detection performance. Second, the closely related nodes are identified as potential neighbors to obtain profitable signals from other nodes. Third, we regularize the adjacency matrix to keep sparsity in the end-to-end training.

As the absolute value of each entry $|\mathcal{H}_{uv}|$ captures the informativeness of neighbor u to ego node v, we first filter unnecessary neighbors of node v with a threshold δ:

$$\mathcal{N}_1(v) = \{u \in \mathcal{V} | u \in \mathcal{N}(v) \text{ and } |\mathcal{H}_{uv}| > \delta\}. \quad (8)$$

Meanwhile, we discover the potential neighbors of nodes based on the $\mathcal{H}$ through k-nearest neighbors (kNN). As an efficient alternative, we suggest sampling k-nearest neighbors via the Gumbel-Top-k trick [Kool *et al.*, 2020; Kazi *et al.*, 2022]. The sampling strategy can be viewed as a stochastic relaxation of kNN. Specifically, the probability of sampling each node $u \notin \mathcal{N}(v)$ is normalized by $p_{uv} = |\mathcal{H}_{uv}| / \sum_{w \notin \mathcal{N}(v)} |\mathcal{H}_{wv}|$ to follow the categorical distribution [Kool *et al.*, 2020]. Then we select the nodes with top-k α_{uv} values as new neighbors $\mathcal{N}_2(v)$:

$$\begin{aligned} \alpha_{uv} &= \log(p_{uv}) - \log(-\log(\epsilon)), \\ \mathcal{N}_2(v) &= \{u \in \mathcal{V} | u \notin \mathcal{N}(v) \text{ and } \alpha_{uv} \in \text{top-k}(\alpha_{*v})\}, \end{aligned} \quad (9)$$

where ϵ is drawn from Uniform$(0, 1)$ independently for each node u, and top-k(α_{*v}) is the set of top-k values in the column vector α_{*v}. In this paper, we implement kNN on the whole graph. While for large graphs, one can approximate it using locality-sensitive hashing approaches [Halcrow *et al.*, 2020] and reduce the time complexity of kNN from $\mathcal{O}\left(n^2\right)$ to $\mathcal{O}\left(sn(B + \log n)\right)$, where s is the number of hashes per node, and B is the size of buckets. In our work, we resort to brute-force algorithm [Li and Amenta, 2015] to reduce the space complexity of kNN from $\mathcal{O}\left(n^2\right)$ to $\mathcal{O}\left(nk\right)$ and thereby accelerate kNN on GPU devices.

We obtain the neighbors $\mathcal{N}'(v)$ of node v in the sparsified graph g' by assembling the filtered neighbors $\mathcal{N}_1(v)$ and the kNN neighbors $\mathcal{N}_2(v)$ as $\mathcal{N}'(v) = \mathcal{N}_1(v) \cup \mathcal{N}_2(v)$. Accordingly, the each entry $\mathcal{A}'_{uv}$ of sparsified adjacency matrix $\mathcal{A}' \in \{0, 1\}^{n\times n}$ equals to 1 when $u \in \mathcal{N}'(v)$.

Moreover, we regularize the homey adjacency matrix $\mathcal{H}$ to sparsify the graph $\mathcal{G}$. A direct way to solve sparsification is minimizing the ℓ_1-norm [Chen *et al.*, 2020b] and nuclear-norm [Koltchinskii *et al.*, 2011] of the adjacency matrix. However, some investigations [Jin *et al.*, 2020; Zhu *et al.*, 2021] indicate that non-differentiable ℓ_1-norm and nuclear-norm optimize the adjacency matrix even more challenging in the end-to-end training and need unique optimization algorithms, such as Forward-Backward splitting method [Combettes and Pesquet, 2011]. Besides, these methods optimize the adjacency directly, and thereby they are intrinsically transductive and cannot adapt to graphs with unseen nodes [Zhu *et al.*, 2021].

To address these issues, we develop a GAD-oriented regularization for sparsifying graph $\mathcal{G}$. Since excessive sparsification of the adjacency matrix may lead to sub-optimal classification results, inspired by Gumbel-Softmax [Jang *et al.*, 2017], we derive the sparsification normalization $\mathcal{L}_r$ to encourage the absolute value $|\mathcal{H}_{uv}|$ of each entry where edge (u, v) in the unsparsified $\mathcal{G}$ to approximate 1 or 0 via entropy:

$$\mathcal{L}_r = -\sum_{\mathcal{A}_{uv}=1} \left(\omega_{uv} \log \omega_{uv} + (1 - \omega_{uv}) \log(1 - \omega_{uv})/\tau\right), \quad (10)$$

where $\omega_{uv} = \max(\delta, \min(|\mathcal{H}_{uv}|, 1 - \delta))$, and $\tau \geq 0$ is a hyperparameter to control the expectation sparsity of $\mathcal{H}$. The smaller τ means entry values of $\mathcal{H}$ below a higher threshold are desirably optimized to 0. In other words, smaller τ means more sparsed g'. We truncate the gradient of $|\mathcal{H}_{uv}|$ via δ in Equation (8) to reduce the computational cost of the edges with the highest probability to be removed or reserved.

4.5 Homey Message Passing

After we obtain the sparsified graph g', we categorize the suspicious nodes through a GNN. Note that we share the parameters of GNN with the inference in subsection 4.3. The majority of GSL methods [Liu *et al.*, 2021; Fatemi *et al.*, 2021; Liu *et al.*, 2022b] focus on learning graph structure and classifying nodes via different architectures, which may cause extra overhead on space and lead to an unstable training process due to the different scales of parameters.

Moreover, we incorporate the homey adjacency matrix $\mathcal{H}$ in the message passing procedure of GNN due to the following reasons. First, a substantial of edges in GAD assist in concealing anomalies, and we shall control the message passing on this part of edges. Since absolute edge weight $|\mathcal{H}_{uv}|$ reveals the informativeness of neighbor u to ego node v, we control the message flows from neighbors with $|\mathcal{H}_{uv}|$ as a message gate. Second, the neighbors of anomalies mostly are not homophilic in terms of attributes, even some other anomalies, since the behaviors of anomalies are most probably diverse and unpredictable. Hence, we take into account homophily and heterophily in the process of message passing via homey adjacency matrix $\mathcal{H}$.

Specifically, we obtain the final node representations $\{\hat{\mathbf{z}}_v\}$ by applying the GNN described in section 3 with the sparsified g' as input. We set the value c^k_{uv} in Equation (1) as:

$$c^k_{uv} = \mathcal{H}_{uv}/\sqrt{\left(d'_u + 1\right)\left(d'_v + 1\right)}, \quad (11)$$

where d'_u and d'_v are the degree of node u and v in adjacency matrix $\mathcal{A}'$ of sparsified graph g', respectively. Intuitively, from the perspective of the spectral GNNs, positive values of $\mathcal{H}_{uv}$ are equivalent to low-pass filters, whereas the negative values of $\mathcal{H}_{uv}$ indeed facilitate the filtering beyond low-frequency [Yang *et al.*, 2021; Chien *et al.*, 2021]. In this vein, we can detect anomalies in both low and high frequency with the aid of flexible $\mathcal{H}_{uv}$.

4.6 Learning Objective

Our training objective includes two parts. First, we classify the final representations $\{\hat{\mathbf{z}}_v\}$ of nodes into two categories, i.e., benign or abnormal, via a linear transformation:

$$\hat{y}_v = \text{sigmoid}(\hat{\mathbf{z}}_v \mathbf{W} + b), \quad (12)$$

where $\mathbf{W} \in \mathbb{R}^{d\times 1}$, $b \in \mathbb{R}$, and $\hat{y}_v \in (0, 1)$ denotes the probability of node v belonging to anomalies. The node classification loss is derived from cross-entropy loss:

$$\mathcal{L}_c = \sum\nolimits_{v\in\mathcal{V}'} y_v \log \hat{y}_v + (1 - y_v) \log(1 - \hat{y}_v). \quad (13)$$

The second objective is the sparsification regularization $\mathcal{L}_r$ derived from Equation (10). The final learning objective of our model is $\mathcal{L} = \mathcal{L}_c + \lambda\mathcal{L}_r$, where λ is a hyperparameter controlling the relative importance of the node classification loss $\mathcal{L}_c$ and sparsification normalization $\mathcal{L}_r$.

5 Evaluation

5.1 Experimental Setup

Dataset. We conduct extensive experiments to evaluate SparseGAD on three real-world anomaly detection datasets:

Dataset	Amazon	YelpChi	Reddit
# Nodes	11,944	45,954	10,984
# Features	25	32	64
Avg. Degree	800.2	175.2	15.3
Anomaly(%)	9.5	14.5	3.3
$\mathcal{H}_{\text{Edge}}$	0.95	0.77	0.95
$(\mathcal{H}_{\text{Normal}}, \mathcal{H}_{\text{Anomaly}})$	(0.97, 0.10)	(0.87, 0.20)	(0.99, 0.00)

Table 1: The statistics of the real-world datasets.

Dataset	**YelpChi** (1%)		**Amazon** (1%)		**YelpChi** (40%)		**Amazon** (40%)		**Reddit** (40%)	
Method	Average Precision↑ AUC-ROC↑									
GCN	17.35±2.6	53.99±2.9	29.15±7.4	76.11±7.5	19.06±2.4	57.78±1.0	31.26±2.9	80.63±1.2	3.52±0.2	52.34±1.8
GAT	17.14±2.4	53.45±3.1	30.55±6.3	78.45±1.6	21.42±2.2	58.61±1.2	73.69±1.0	91.70±0.5	<u>5.74±0.9</u>	<u>65.79±1.6</u>
GraphSage	36.75±2.8	74.27±1.8	34.27±1.6	82.07±0.9	46.24±2.3	79.60±1.3	64.17±1.6	86.29±0.4	4.95±1.1	61.81±7.1
GIN	22.40±3.9	62.87±4.6	35.09±1.3	82.72±0.5	26.23±2.0	65.20±0.9	45.42±1.2	85.74±1.1	4.20±0.3	57.64±1.2
GPRGNN	34.41±1.8	73.96±0.9	79.95±3.9	89.82±3.0	40.18±0.7	76.95±0.4	82.82±0.8	93.37±0.2	4.72±1.5	57.53±7.1
Neural Sparse	30.03±9.0	67.37±8.4	74.32±7.7	86.42±2.4	34.24±2.1	73.56±1.2	82.87±1.4	92.08±2.1	5.19±0.4	62.73±1.7
IDGL	32.81±4.1	70.51±2.5	78.10±4.0	88.59±2.7	38.82±2.6	78.03±1.3	85.30±2.3	93.51±1.5	5.30±0.3	62.95±1.6
SLAPS	OOM	OOM	75.87±5.2	87.98±3.4	OOM	OOM	84.68±2.1	93.28±0.8	5.49±0.6	63.24±2.5
CARE-GNN	35.30±3.9	74.80±3.2	74.35±5.9	88.67±2.7	38.90±1.1	78.41±1.5	70.46±1.1	90.86±0.8	N/A	N/A
PC-GNN	23.39±1.8	72.93±2.4	79.94±6.0	89.07±1.5	32.77±0.6	80.61±0.7	80.05±1.7	95.75±0.5	N/A	N/A
AMNet	35.15±1.7	73.69±1.3	72.86±5.0	87.53±1.8	57.77±0.9	85.85±1.1	83.72±1.2	94.18±0.4	5.69±0.8	57.69±2.7
BW-GNN	36.17±2.3	76.29±1.5	71.26±3.5	86.71±1.3	62.88±1.2	87.20±0.8	**90.85±1.3**	**97.95±0.3**	5.65±0.9	61.61±2.4
SparseGAD (GCN)	35.82±2.8	75.36±1.8	77.82±3.6	89.02±2.1	53.36±1.8	82.41±1.3	86.40±2.0	95.83±0.6	5.42±0.2	63.17±1.2
SparseGAD-s	<u>38.52±2.1</u>	<u>77.64±1.9</u>	<u>80.29±2.7</u>	<u>91.28±1.6</u>	<u>63.77±1.6</u>	<u>87.80±0.8</u>	<u>89.65±1.3</u>	<u>97.21±0.2</u>	5.47±0.4	64.05±1.2
SparseGAD-h	35.10±2.5	73.57±1.6	78.24±1.5	89.28±0.7	51.49±1.3	81.63±1.0	78.28±1.8	95.42±0.3	5.33±0.2	63.28±1.3
SparseGAD	**40.77±1.8**	**78.73±1.7**	**81.40±1.9**	**93.67±0.9**	**65.82±0.9**	**88.54±0.5**	89.17±1.6	97.03±0.3	**5.98±0.5**	**66.12±1.6**

Table 2: Anomaly detection performance of popular baselines and SparseGAD on the real-world datasets with 1% and 40% training ratios. "OOM" represents the detector is out of memory on the dataset, and "N/A" denotes the detector is not applicable to the dataset.

- **Amazon** [McAuley and Leskovec, 2013] collects the product reviews of the Musical Instrument category on Amazon.com, in which nodes are benign/fraud users.
- **YelpChi** [Rayana and Akoglu, 2015] includes the hotel and restaurant reviews filtered (spam) as anomalies and recommended (legitimate) as normal nodes by Yelp.
- **Reddit** [Kumar *et al.*, 2019] consists of user posts on one month collection of subreddits. The ground-truth labels of banned users are obtained from Reddit.

We process the connections of node pairs in Amazon and YelpChi following [Dou *et al.*, 2020], and Reddit is loaded from PyGod package [Liu *et al.*, 2022a]. The performance of our method is conducted on homogeneous settings, i.e., without the category information of edges, and we leave the adaption of SparseGAD for multiple node types and edge types of heterogeneous graphs in our future works.

The statistics of these datasets are shown in Table 1. We further show the edge homophily [Zhu *et al.*, 2020] and node homophily [Pei *et al.*, 2020] of normal nodes and anomalies for each dataset. The edge homophily $\mathcal{H}_{\text{Edge}}$ is the proportion of edges that link two nodes of the same classes:

$$\mathcal{H}_{\text{Edge}} = \frac{|\{e_{uv} \in \mathcal{E} : y_u = y_v\}|}{|\mathcal{E}|}, \tag{14}$$

while the node homophily of class C is calculated as:

$$\mathcal{H}_C = \frac{1}{|\mathcal{V}_C|} \sum\nolimits_{v \in \mathcal{V}_C} \frac{|\{u \in \mathcal{N}(v) : y_u = y_v\}|}{|\mathcal{N}(v)|}, \tag{15}$$

where C includes normal nodes and anomalies. We discover that the anomaly homophily $\mathcal{H}_{\text{Anomaly}}$ of the datasets in Table 1 are all significantly lower than normal nodes $\mathcal{H}_{\text{Normal}}$, which indicates that heterophilic links are the major part of connections involving anomalies.

Metrics. Following previous works in GAD [Dou *et al.*, 2020; Chai *et al.*, 2022], we evaluate the detection performance with two widely-used and complementary metrics: (1) AUC-ROC reflects the detectors' effectiveness on both normal and abnormal nodes, while (2) Average Precision (AP) focuses more on the anomalies.

Baselines. To verify the effectiveness of our SparseGAD, we compare it with three types of baselines. (1) General GNNs, which consist of GCN [Kipf and Welling, 2017], GAT [Veličković *et al.*, 2018], GraphSage [Hamilton *et al.*, 2017], GIN [Xu *et al.*, 2019] and GPRGNN [Chien *et al.*, 2021]. (2) Graph Structure Learning methods, such as Neural Sparse [Zheng *et al.*, 2020], IDGL [Chen *et al.*, 2020b] and SLAPS [Fatemi *et al.*, 2021]. (3) GNN-based anomaly detectors, including CARE-GNN [Dou *et al.*, 2020], PC-GNN [Liu *et al.*, 2021], AMNET [Chai *et al.*, 2022] and BW-GNN [Tang *et al.*, 2022].

In addition to representative baselines, we also include two variants of SparseGAD to validate the effectiveness of modules: (1) SparseGAD (GCN) utilize Graph Convolution Network [Kipf and Welling, 2017] as the backbone. (2) SparseGAD-s, which removes the sparsification module (or keep $g' = \mathcal{G}$ and $\lambda = 0$). (3) SparseGAD-h ignores heterophily exists in GAD by setting $\mathcal{H}_{uv} = \text{sigmoid}(\mathbf{z}'_u \cdot \mathbf{z}'_v)$.

Experiment Settings. For each experiment, we train all models for 2000 epochs by Adam optimizer. We optimize all models on each dataset by selecting learning rate from {0.01, 0.005, 0.001}, hidden states from {16, 32, 64} and dropout rate from {0, 0.1, 0.2} via grid search, and save the model according to the best AUC in validation. As for SparseGAD, α of Equation (2) and λ are both set to 0.1, and δ is set to 0.05 for all datasets. For each dataset, we search k of kNN from {5, 10, 20} and τ from {0.5, 1.0, 2.0}. We select 40% nodes

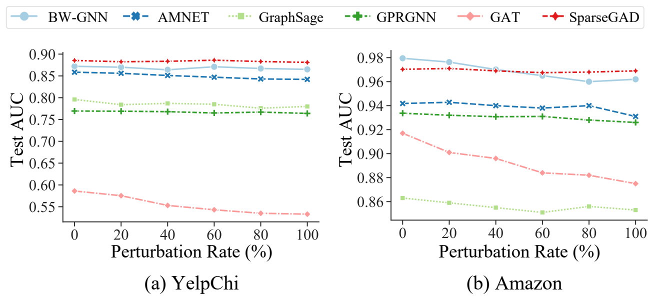

Figure 3: Detection AUC of different models under random attack.

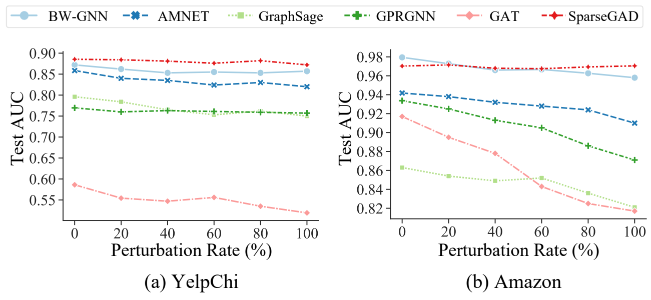

Figure 4: Detection AUC of different models under DICE attack.

for training models in supervised scenarios and 1% in semi-supervised scenarios. The remaining nodes are split by 1:2 for validation and testing. We report the average value and standard deviation of ten independent runs for each dataset. As for the detailed experiment settings, please refer to our code https://github.com/KellyGong/SparseGAD.git.

5.2 Detection Performance

The anomaly detection results of baselines and SparseGAD are reported in Table 2. Overall, SparseGAD achieves the best detection quality in all datasets except Amazon (40%), where BW-GNN obtains the best AUC and AP scores. Under the semi-supervised scenarios (i.e., YelpChi (1%) and Amazon (1%)), SparseGAD outperforms baselines by a large margin. For instance, in YelpChi (1%), which includes only 67 labeled anomalies, SparseGAD has 4.6% and 2.4% absolute improvements in AP and AUC score compared to BW-GNN, which illustrates the superiority of SparseGAD with low-resource anomaly labels. Moreover, the strong heterophily of anomalies (i.e., $\mathcal{H}_{\text{Anomaly}} = 0$) in Reddit leads to the worse performance of advanced graph-based anomaly detectors compared to GAT. By contrast, SparseGAD has consistent improvements in this dataset, proving both the scalability and superiority of our framework.

5.3 Robustness Performance

To compare the robustness of SparseGAD against structure noises with other detectors, we evaluate the detection performance under two types of attacks. **i**). **Random Attack** [Jin *et al.*, 2020] injects fake edges into the graph, which can be viewed as adding random structure noises to the clean graphs. **ii**). **DICE** [Zügner and Günnemann, 2019] aims at fooling GNNs by removing edges between nodes from the same class and inserting edges between nodes from the different classes.

We first poison the graph structure with these two attack methods and then train each model on the poisoned graphs and evaluate the anomaly detection results. We reserve 40% nodes for training models and report the mean AUC of ten independent runs on the test set for each model.

Against Random Attack. In this experiment, we evaluate how SparseGAD behaves under different ratios of random structure noises from 0% to 100% and report the results in Figure 3. Figure 3 shows that the performance of GAT and GraphSage drops significantly with increasing ratios of the random attack, and the performance of AMNET and BW-GNN decline noticeably when the random attack to some

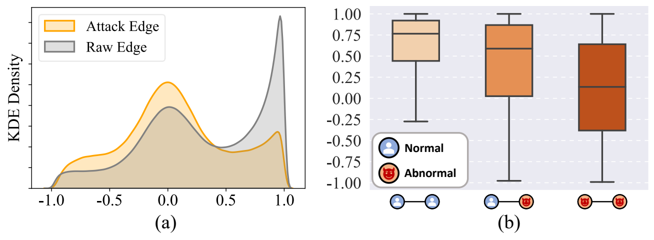

Figure 5: (a) The homey edge weight $\mathcal{H}_{uv}$ KDE density of attack edges and raw edges under DICE attack in YelpChi. (b) $\mathcal{H}_{uv}$ distribution across three different connections of node pairs.

extent. We discover that SparseGAD successfully defends against the random attack and consistently outperforms baselines in most cases.

Against DICE Attack. Since heterophilic links inherently exist in graphs of anomaly detection, we study how SparseGAD performs under different ratios of heterophily links. In practice, we select the anomalies in the test set as target nodes to attack, and the perturbation number of each node is calculated as $\max(10, \text{degree}) \times$ ratio. We vary the ratio of perturbations on every target node with a step size of 20% to remove the existing links and add an equal number of heterophilic links. AUC values of SparseGAD and baselines on target nodes are reported in Figure 4. We discover that when the perturbation ratio increases, GAT and GraphSage perform worse than other models, whereas our SparseGAD is better than baselines on the attacked target nodes for the most part, demonstrating that SparseGAD can resist DICE attack.

5.4 Ablation Study

To validate the effectiveness of SparseGAD framework, we also investigate the anomaly detection performance of SparseGAD with Graph Convolution Network as the backbone, i.e., SparseGAD (GCN) and compare it with other GSL works. As shown in Table 2, we discover that SparseGAD (GCN) outperforms other GSL approaches in most case, which illustrates the superiority and extensibility of our framework in the GAD task. Besides, to better understand how homey adjacency matrix $\mathcal{H}$ and sparsification module help SparseGAD detect anomalies, we conduct ablation studies and compare SparseGAD with other two variants, i.e., SparseGAD-h and SparseGAD-s. As for SparseGAD-h, we only consider the informativeness of neighbors to ego nodes without heterophily via sigmoid activation instead of cosine similarity. As shown in Table 2, we find that SparseGAD achieves coherent improvements in all datasets compared to SparseGAD-h, which suggests that heterophily-aware aggregation is vital in GAD. Besides, in SparseGAD-s, we directly use the raw graph structure without sparsification in the homey message passing. In Table 2, we discover that SparseGAD outperforms SparseGAD-s in most cases, which indicates that the structure noises exists in the graphs of anomaly detection, and selecting the task-relevant edges contributes to more reliable results.

5.5 Parameter Analysis

Before this, we illustrated the effectiveness and robustness of SparseGAD. In this subsection, we attempt to understand the sparsified graph and homey adjacency matrix $\mathcal{H}$ we learned.

Importance of Graph Sparsification. Based on the fact that anomalies tend to connect with plenty of benign entities, if the framework is able to learn a denoised graph structure, the influence of the attack edges shall be eliminated from the poisoned graph. Hence, we investigate the homey edge weight $\mathcal{H}_{uv}$ of attack edges and raw edges in YelpChi under DICE attack with a 20% perturbation ratio. In Figure 5(a), $\mathcal{H}_{uv}$ of attack edges are much smaller than raw edges in the graph and are mainly around 0. The sparsification module will remove most attack edges via threshold δ, which proves that SparseGAD is capable to mitigate the effect of attack edges and learn robust parameters of GNN.

Importance of Homey Adjacency matrix. As heterophilic and homophilic connections exist in GAD, if the framework can detect anomalies precisely, the relationships between different neighbors and ego nodes shall be recognized distinctly. Accordingly, we study whether $\mathcal{H}_{uv}$ can distinguish the connection properties. In Figure 5(b), we visualize $\mathcal{H}_{uv}$ distributions of raw edges across pairs of normal entities, pairs of anomalies, and intersections of anomalies and normal entities in YelpChi (40%). We observe that $\mathcal{H}_{uv}$ among the intersections of anomalies and normal entities are much smaller and more diverse than the other connections between the same types of nodes, which proves that SparseGAD can distinguish the complex relationships of node pairs and thereby facilitate aggregating message from neighbors.

6 Conclusion and Future Work

In this paper, we introduced SparseGAD, a neural sparsification framework for Graph-based Anomaly Detection. Under this framework, we can capture the heterophilic and homophilic reliance of nodes to sparsify graph structure and learn distinguishable node representations for anomalies collaboratively. Our experiments show that our framework outperforms the state-of-the-art baselines in most cases, especially when the structure of graphs is heavily perturbed.

Considering some applications [Wang *et al.*, 2021] of anomaly detection in real-world graphs are accompanied by multiple types of nodes and edges, we would like to investigate how to sparsify graph structure to facilitate detecting anomalies on such heterogeneous graphs in our future works. Besides, some graph compression methods, e.g., [Yin *et al.*, 2022] shed light on how to accelerate graph samplings with GPUs and reduce the graph size for the explosive growth of nodes in large-scale graphs of anomaly detection. We could like to further improve the detection speed with the aid of such techniques.

References

[Bo *et al.*, 2021] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3950–3957, 2021.

[Chai *et al.*, 2022] Ziwei Chai, Siqi You, Yang Yang, Shiliang Pu, Jiarong Xu, Haoyang Cai, and Weihao Jiang. Can abnormality be detected by graph neural networks? In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[Chang and Chang, 2014] Jau-Shien Chang and Wen-Hsi Chang. Analysis of fraudulent behavior strategies in online auctions for detecting latent fraudsters. *Electronic Commerce Research and Applications*, 13(2):79–97, 2014.

[Chen et al., 2020a] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020.

[Chen et al., 2020b] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems*, 33:19314–19326, 2020.

[Chien et al., 2021] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

[Combettes and Pesquet, 2011] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[Ding et al., 2019] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 594–602. SIAM, 2019.

[Dou et al., 2020] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 315–324, 2020.

[Fatemi et al., 2021] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 34:22667–22681, 2021.

[Gaddam *et al.*, 2020] Anuroop Gaddam, Tim Wilkin, Maia Angelova, and Jyotheesh Gaddam. Detecting sensor faults, anomalies and outliers in the internet of things: A survey on the challenges and solutions. *Electronics*, 9(3):511, 2020.

[Gasteiger *et al.*, 2019] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

[Halcrow *et al.*, 2020] Jonathan Halcrow, Alexandru Mosoi, Sam Ruth, and Bryan Perozzi. Grale: Designing networks for graph learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2523–2532, 2020.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[Huang *et al.*, 2021] Jizhou Huang, Haifeng Wang, Yibo Sun, Miao Fan, Zhengjie Huang, Chunyuan Yuan, and Yawen Li. Hgamn: Heterogeneous graph attention matching network for multilingual poi retrieval at baidu maps. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3032–3040, 2021.

[Huang *et al.*, 2022] Mengda Huang, Yang Liu, Xiang Ao, Kuan Li, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Auc-oriented graph neural network for fraud detection. In *Proceedings of the ACM Web Conference 2022*, pages 1311–1321, 2022.

[Jang *et al.*, 2017] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[Jin *et al.*, 2020] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[Jin *et al.*, 2021] Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. Universal graph convolutional networks. *Advances in Neural Information Processing Systems*, 34:10654–10664, 2021.

[Kazi *et al.*, 2022] Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael Bronstein. Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[Koltchinskii *et al.*, 2011] Vladimir Koltchinskii, Karim Lounici, and Alexandre B Tsybakov. Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *The Annals of Statistics*, 39(5):2302–2329, 2011.

[Kool *et al.*, 2020] Wouter Kool, Herke van Hoof, and Max Welling. Estimating gradients for discrete random variables by sampling without replacement. In *International Conference on Learning Representations*, 2020.

[Kumar *et al.*, 2019] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1269–1278, 2019.

[Li and Amenta, 2015] Shengren Li and Nina Amenta. Brute-force k-nearest neighbors search on the gpu. In *Similarity Search and Applications: 8th International Conference, SISAP 2015, Glasgow, UK, October 12-14, 2015, Proceedings 8*, pages 259–270. Springer, 2015.

[Liu *et al.*, 2020] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1569–1572, 2020.

[Liu *et al.*, 2021] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Pick and choose: a gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*, pages 3168–3177, 2021.

[Liu *et al.*, 2022a] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, George H. Chen, Zhihao Jia, and Philip S. Yu. Pygod: A python library for graph outlier detection. *arXiv preprint arXiv:2204.12095*, 2022.

[Liu *et al.*, 2022b] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*, pages 1392–1403, 2022.

[Liu *et al.*, 2023] Yixin Liu, Yizhen Zheng, Daokun Zhang, Vincent Lee, and Shirui Pan. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. In *AAAI*, 2023.

[Luo *et al.*, 2021] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 779–787, 2021.

[McAuley and Leskovec, 2013] Julian McAuley and Jure Leskovec. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. *the web conference*, 2013.

[Pei *et al.*, 2020] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[Rayana and Akoglu, 2015] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. *knowledge discovery and data mining*, 2015.

[Shi *et al.*, 2022] Fengzhao Shi, Yanan Cao, Yanmin Shang, Yuchen Zhou, Chuan Zhou, and Jia Wu. H2-fdetector: A gnn-based fraud detector with homophilic and heterophilic connections. In *Proceedings of the ACM Web Conference 2022*, pages 1486–1494, 2022.

[Sun *et al.*, 2021] Ying Sun, Fuzhen Zhuang, Hengshu Zhu, Qi Zhang, Qing He, and Hui Xiong. Market-oriented job skill valuation with cooperative composition neural network. *Nature communications*, 12(1):1992, 2021.

[Tang *et al.*, 2022] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*, 2022.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[Wang *et al.*, 2018] Hao Wang, Enhong Chen, Qi Liu, Tong Xu, Dongfang Du, Wen Su, and Xiaopeng Zhang. A united approach to learning sparse attributed network embedding. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 557–566. IEEE, 2018.

[Wang *et al.*, 2019] Hao Wang, Tong Xu, Qi Liu, Defu Lian, Enhong Chen, Dongfang Du, Han Wu, and Wen Su. Mcne: An end-to-end framework for learning multiple conditional network representations of social network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1064–1072, 2019.

[Wang *et al.*, 2021] Li Wang, Peipei Li, Kai Xiong, Jiashu Zhao, and Rui Lin. Modeling heterogeneous graph network on fraud detection: A community-based framework with attention mechanism. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1959–1968, 2021.

[Weber *et al.*, 2019] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*, 2019.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[Yang *et al.*, 2020] Xiaoyu Yang, Yuefei Lyu, Tian Tian, Yifei Liu, Yudong Liu, and Xi Zhang. Rumor detection on social media with graph structured adversarial learning. *international joint conference on artificial intelligence*, 2020.

[Yang *et al.*, 2021] Liang Yang, Mengzhe Li, Liyang Liu, bingxin niu, Chuan Wang, Xiaochun Cao, and Yuanfang Guo. Diverse message passing for attribute with heterophily. *neural information processing systems*, 2021.

[Ye and Akoglu, 2015] Junting Ye and Leman Akoglu. Discovering opinion spammer groups by network footprints. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 267–282. Springer, 2015.

[Yin *et al.*, 2022] Hongbo Yin, Yingxia Shao, Xupeng Miao, Yawen Li, and Bin Cui. Scalable graph sampling on gpus with compressed graph. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2383–2392, 2022.

[Yu *et al.*, 2020] Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. Graph-revised convolutional network. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 378–393. Springer, 2020.

[Zhang *et al.*, 2021] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. Motif-based graph self-supervised learning for molecular property prediction. *Advances in Neural Information Processing Systems*, 34:15870–15882, 2021.

[Zhao *et al.*, 2020] Tong Zhao, Chuchen Deng, Kaifeng Yu, Tianwen Jiang, Daheng Wang, and Meng Jiang. Error-bounded graph anomaly loss for gnns. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1873–1882, 2020.

[Zheng *et al.*, 2020] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pages 11458–11468. PMLR, 2020.

[Zhu *et al.*, 2020] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

[Zhu *et al.*, 2021] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*, 2021.

[Zügner and Günnemann, 2019] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations*, 2019.