HAGAN: Homophily-Aware Generative Adversarial Network for Graph Anomaly Detection

Wenkai Wang[1], Fan Gao[1], and Meihong Wang[1]✉

School of Informatics, Xiamen University, Xiamen 361102, China
30920231154336@stu.xmu.edu.cn, gaofan@stu.xmu.edu.cn, wangmh@xmu.edu.cn

Abstract. With the increasing prevalence of graph-structured data, graph anomaly detection has emerged as a crucial research domain. Motivated by the realistic challenge that many practical problems are constrained by limited sample data, this study proposes a semi-supervised setting, unlike conventional unsupervised and supervised learning methods, where only a subset of normal samples is available. A key challenge in this context is the absence of anomalous samples, which can lead to model bias and compromise detection performance. To address this issue, we introduce a novel model, Homophily-Aware Generative Adversarial Network (HAGAN), which leverages a generative adversarial network to generate high-quality anomalous nodes. These generated nodes are seamlessly integrated into the real graph using a transformer-based graph autoencoder. Furthermore, the discriminator employs a GNN architecture enhanced with an edge homogeneity identification mechanism to improve anomaly detection. The proposed model is evaluated on four large-scale real-world benchmark datasets, and experimental results demonstrate that HAGAN consistently achieves state-of-the-art performance across multiple evaluation metrics.

Keywords: Machine Learning · Anomaly Detection · GNN.

1 Introduction

In the context of the growing prevalence of graph-structured data, graph anomaly detection (GAD) has emerged as a significant research focus [16]. Unlike traditional anomaly detection [25], graph anomaly detection seeks to identify anomalous patterns within complex relationships and structures. It has been widely applied in various fields, including social networks [18], network security [3], and financial systems [6].

In recent years, a wide array of research has emerged in the field of GAD, primarily focusing on two key approaches: unsupervised and supervised methods. Although these researches have demonstrated promising results, they overlook a critical challenge: the inherent imbalance between normal and anomalous samples in real-world datasets [15]. Unsupervised methods assume that the labels of all nodes are unknown. However, the overwhelming prevalence of normal samples

makes it relatively easy to identify normal nodes, which results in unsupervised methods not fully leveraging these normal samples. Supervised methods typically rely on a subset of nodes labeled as anomalous. However, anomalous patterns in real-world data are both diverse and scarce, making the labeling process costly. Furthermore, this rarity may not offer enough information for the model to effectively learn discriminative features.

Based on the above discussion, we assume a semi-supervised setting in which labels for only a subset of normal samples are available. In this context, an important issue arises from the lack of anomalous samples. Anomalous samples often exhibit significant differences in features or connectivity structures compared to normal samples, and relying solely on normal samples may not provide sufficient information to accurately identify anomalies. In order to address the above problem, this paper proposes **H**omophily-**A**ware **G**enerative **A**dversarial **N**etwork (**HAGAN**) for GAD. Our objective is trying to alleviate the lack of anomalous samples by generating high-quality anomaly nodes.

However, some challenges arise here: **1. How to ensure the high-quality of the generated anomaly nodes?** A common approach is random generation; however, this often leads to suboptimal node quality. To address this limitation, we employ a generative adversarial network (GAN) [14] to generate anomalous nodes, leveraging adversarial training to produce realistic and high-quality results. **2. How to integrate the generated nodes with the original graph ?** Since the generated nodes are isolated, it is imperative to establish structural relationships between the generated nodes and the original nodes. A simple random connection approach may disrupt the inherent characteristics of the original graph structure. Therefore, we utilize a pre-trained graph autoencoder (GAE) to reconstruct the graph structure and seamlessly achieve integration. **3. How to alleviate the problem that generated nodes will introduce noise?** GNN is chosen as the discriminator due to its effectiveness. However, in GAD tasks, GNNs are vulnerable to noise from heterophilic edges [19], where information from normal and anomalous nodes interferes during aggregation. To address this, we integrate an edge homophily identification module that reduces noise from generated nodes and helps the model better capture heterophilic edges inherent in the graph.

To sum up, the main contributions of this paper are as follows:

- We propose HAGAN, a novel framework that accomplishes the semi-supervised GAD task by generating high-quality anomaly nodes.
- We propose a Transformer-based GAE to integrate the generator-produced nodes into the original graph, thereby preserving its inherent structural integrity.
- We integrate an edge homophily identification method into the GNN discriminator to enhance its discriminative ability.
- We evaluate the effectiveness of HAGAN on four real-world benchmark datasets. The experiments demonstrate that our approach delivers state-of-the-art performance.

2 Related Work

Traditional methods such as LOF [2] and isForest [25] struggle to effectively capture both local and global structural information. With the increasing prevalence of graph-structured data, several shallow methods based on graph structures have emerged, such as Radar [23] and ANOMALOUS [30]. However, these shallow techniques typically lack the expressive power that deep learning possesses. In recent years, the rapid advancement of GNN has positioned deep learning methods based on graph structures at the forefront of research.

2.1 Unsupervised Methods

DOMINANT [8] uses GNN to capture both structural and attribute information for anomaly detection in attributed networks. DONE [1] employs deep one-class classification to detect anomalies via distributional deviation. OCNNN [37] optimizes a one-class objective function to distinguish normal and anomalous instances. TAM [31] leverages the stronger affinity among normal nodes compared to anomalous ones. CoCo [36] introduces a method based on the correlation discrepancy between local and global contextual information of nodes.

Besides, Self-supervised methods based on contrastive learning have also emerged as a key research focus in recent years. Notable examples include CoLA [26], ANEMONE [17], SL-GAD [41], Sub-CR [39], CONAD [38] and GRADATE [10]. The core principle of these methods is to enhance the model's discriminative ability by performing contrastive learning at various scales, bringing similar node representations closer together while pushing dissimilar ones further apart.

2.2 Supervised Methods

Current research on supervised methods primarily focuses on edge homophily. Edges between nodes can be classified into two types: homophilic (connecting nodes with the same labels) and heterophilic (connecting nodes with different labels). By identifying edge homophily, these methods optimize the graph structure or design novel aggregation strategies, thereby reducing noise during model training. H^2-FDetector [33] enhances fraud detection by transmitting similar information over homophilic edges and dissimilar information over heterophilic edges. GDN [11] mitigates structural distribution shift in GAD by isolating and constraining key anomalous features. SpareGAD [13] simplifies graph structure through sparsification, removing task-irrelevant edges and employing a heterophilic-aware aggregation scheme. HedGe [40] reduces excessive distributional differences by generating homogeneous edges, modifying the loss function to suppress heterogeneous edge formation.

2.3 Generative Anomaly Detection

Early methods, such as AnoGAN [32] and GAN-AD [22], achieve notable success in their respective domains. However, these methods are not designed for graph

data. AdONE [1] is the first approach to use adversarial training for this task, helping to further minimize the influence of outliers while generating the embeddings. GAAN [5] and AEGIS [7] both employ GAN architectures for graph anomaly detection. GAAN generates fake graph nodes from Gaussian noise and learns latent representations via an encoder, while its discriminator determines whether connected node pairs originate from the real graph. AEGIS, on the other hand, enhances generalization by generating information-rich potential anomalies and training a discriminator to distinguish them from normal data.

3 Preliminaries

3.1 Definition 1. Graph

In this paper, we focus on anomaly detection in attributed graphs with labeled normal nodes, while the number of anomalous labels is extremely sparse. An attributed graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ denotes the node set, $N = |\mathcal{V}|$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ with $e \in \mathcal{E}$ denotes the edge set, $\mathbf{X}_{origin} \in \mathcal{R}^{N \times d}$ denotes the node features with $\mathbf{x}_{origin}(v_i) \in \mathcal{R}^d$ being the attribute vector of v_i and d denotes the attribute dimension in the original data. The adjacency matrix of $\mathcal{G}$ is denoted as $\mathbf{A} \in \{0,1\}^{N \times N}$ and $\mathbf{A}_{ij} = 1$ if and only if $e_{ij} \in \mathcal{E}$.

3.2 Definition 2. Graph Anomaly Detection

Given a graph $\mathcal{G}$ mentioned above, the objective of anomaly detection is to calculate an anomaly score $f(v) \in (0,1)$, where f is an anomaly score function and $v \in \mathcal{V}$. The anomaly score reflects the extent of the abnormality of the node, which means that a higher $f(v)$ indicates a higher probability of anomalous of the node v.

4 Method

In this section, we introduce the details of HAGAN, as presented in Fig. 1. HAGAN consists of two stages: the first stage involves pretraining a Transformer-based Graph Autoencoder, while the second stage focuses on training GAN.

4.1 Transformer-based GAE

GAE demonstrates significant advantages in reconstructing graph structures [24]. However, their effectiveness typically relies on the assumption that the input data inherently exhibit a graph structure, while generated nodes are typically isolated, posing a challenge for integration.

To overcome this limitation, we introduce anchor-based encoding, which plays a role similar to positional encoding [35] by injecting structural information into the generated isolated nodes. The real and generated nodes are treated as a unified sequence. Leveraging the Transformer's powerful sequence modeling capability along with the encoded structural information, we propose a Transformer-based GAE to obtain expressive node embeddings.

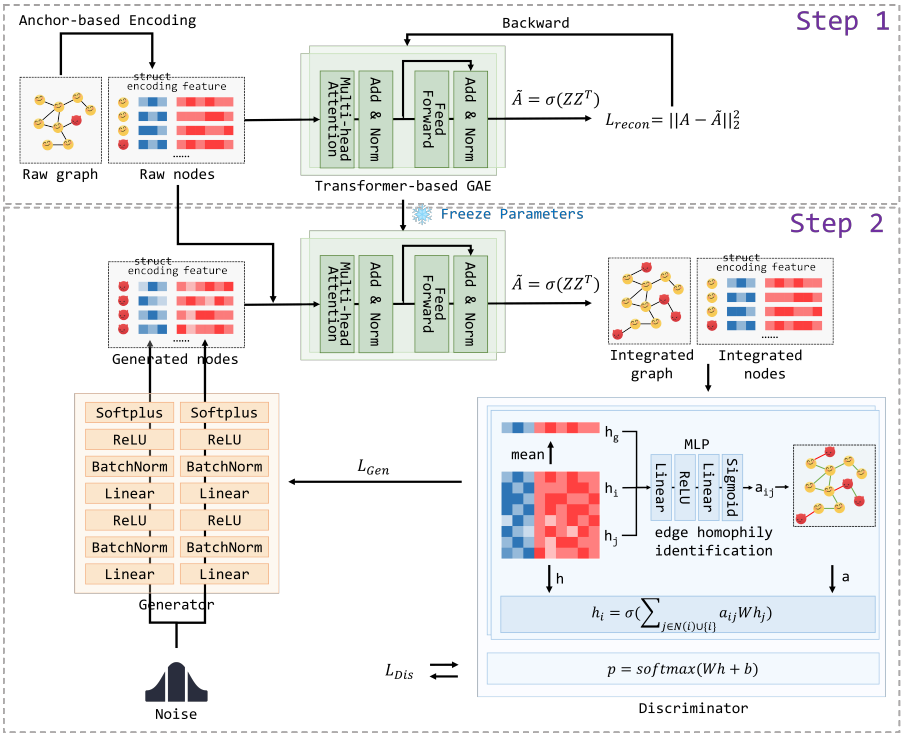


Fig. 1: The overall architecture of HAGAN, which consists of two stages. In Stage 1, a Transformer-based GAE is pretrained on real data. In Stage 2, the generator synthesizes anomalous nodes, which are integrated with the original graph via the pretrained GAE. The integrated graph is then processed by the discriminator, a GNN equipped with an edge homogeneity identification mechanism. Specifically, for each edge (i, j), their features along with the global average feature are fed into an MLP to compute a homophily probability, which is then used to guide the attention coefficient computation.

Anchor-based Encoding Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, we select k anchor points, denoted as $\mathcal{V}_{Anchor} = \{a_1, a_2, ..., a_k\} \subset \mathcal{V}$. For node $v \in \mathcal{V}$, we define its anchor-based encoding (structural features) $\mathbf{X}_{struct}$ as:

$$\mathbf{X}_{struct} = [d(v, a_1), d(v, a_1), ...d(v, a_k)] \tag{1}$$

where $d(v, a_i)$ represents the shortest path distance between node v and anchor point a_i.

Based on the above definition, a crucial aspect of anchor-based encoding is the selection of anchor points. Here, we employ graph diffusion techniques [12]. Formally, given an adjacency matrix $\mathbf{A}$, the graph diffusion matrix $\mathbf{S} \in \mathcal{R}^{N \times N}$

is defined by:

$$\mathbf{S} = \sum_{k=0}^{\infty} \theta_k \mathbf{T}^k \ \ (\theta_k \in [0,1] \ and \ \sum_{k=0}^{\infty} \theta_k = 1) \tag{2}$$

where $\mathbf{T} \in \mathcal{R}^{N \times N}$ denotes the generalized transition matrix, θ_k denotes the weighting coefficient determining the ratio of global-local information. The row $\mathbf{s}_i$ in $\mathbf{S}$ represents the connectivity of node v_i, and its row sum is used to determine each node's connectivity strength, allowing us to select the top k nodes as anchors.

In practice, we use Personalized PageRank (PPR) [28] which choose $\mathbf{T} = \mathbf{A}\mathbf{D}^{-1}$ and $\theta_k = \alpha(1-\alpha)^k$:

$$\mathbf{S}^{PPR} = \alpha(\mathbf{I} - (1-\alpha)\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}) \tag{3}$$

where $\mathbf{I}$ denotes the identity matrix, $\mathbf{D} \in \mathcal{R}^{N \times N}$ denotes the diagonal degree matrix and $\alpha \in (0,1)$ denotes the teleport probability.

Autoencoder We concatenate $\mathbf{X}_{struct}$ and $\mathbf{X}_{origin}$ to leverage their complementary information. Furthermore, linear transformation are applied to obtain the fusion representation $\mathbf{H}_{fusion}$:

$$\mathbf{H}_{fusion} = \mathbf{W}_r[\,\mathbf{W}_s\mathbf{X}_{struct} \,\|\, \mathbf{W}_o\mathbf{X}_{origin}\,] \tag{4}$$

where $||$ is the concatenation operation and $\mathbf{W}_r, \mathbf{W}_s, \mathbf{W}_o$ denotes three distinct learnable parameter matrices.

For encoder, we consider the set of nodes as a sequence and input it into the Transformer's encoder to obtain the node embeddings:

$$\mathbf{H}'^{(l)} = LayerNorm(\mathbf{H}^{(l-1)} + MultiHeadAttention(\mathbf{H}^{(l-1)})) \tag{5}$$

$$\mathbf{H}^{(l)} = LayerNorm(\mathbf{H}'^{(l)} + FFN(\mathbf{H}'^{(l)})) \tag{6}$$

where $\mathbf{H}^{(l)}$ and $\mathbf{H}^{(l-1)}$ respectively denotes the output of the l-th layer and (l-1)-th attention layer. The input $\mathbf{H}^{(0)}$ is $\mathbf{H}_{fusion}$ and the output of the final attention layer $\mathbf{H}^{(L)}$ is denoted as the output node embeddings $\mathbf{Z}$.

For decoder, we choose the widely used inner product method due to its simplicity and effectiveness:

$$\tilde{\mathbf{Z}} = g(\mathbf{Z}) \tag{7}$$

$$\tilde{\mathbf{A}} = \sigma(\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^T) \tag{8}$$

where g denotes a normalization function to prevent gradient explosion and vanishing and $\tilde{\mathbf{A}}$ denotes the reconstructed adjacency matrix.

To facilitate the subsequent use of the generator, we pre-train TGAE using reconstruction loss function:

$$\mathcal{L}_{recon} = \|\mathbf{A} - \tilde{\mathbf{A}}\|_2^2 \tag{9}$$

4.2 GAN Architecture

Fence [27] demonstrated that GAN can achieve superior performance in anomaly detection by modifying the traditional loss function to encourage the generator to produce data at the edges of the normal data distribution. Inspired by this work, we adopt a similar architecture.

Generator In our generator, we utilize gaussian noise as the standard input and employ a dual-branch architecture to produce structural features and original features. We define the model as follows:

$$G_{struct}(z_i) = g(\sigma(f_{struct}(z_i))) \tag{10}$$

$$G_{origin}(z_i) = g(\sigma(f_{origin}(z_i))) \tag{11}$$

where f_{struct} and f_{origin} denote different 3-layer feedforward neural networks, σ denotes the activation function, g denotes a normalization function, $G_{struct}(z_i) \in \mathcal{R}^k$ denotes the generated structural features and $G_{origin}(z_i) \in \mathcal{R}^d$ denotes the generated original features.

The loss function $\mathcal{L}_{gen}$ is designed to comprise three components. $\mathcal{L}_{dist}$ guides the generator to produce samples with a discriminator score of α, positioning them at the boundary of normal nodes to enhance high-quality. $\mathcal{L}_{divr}$ enhances diversity in generated nodes, addressing GANs' susceptibility to mode collapse by maximizing their average distance from the mean. $\mathcal{L}_{kl}$ promotes the dispersion of generated data in a multidimensional space, preventing the generator from concentrating samples in specific dimensions during training. Formally, the loss function is defined as:

$$\mathcal{L}_{gen} = \beta_1 \mathcal{L}_{dist} + \beta_2 \mathcal{L}_{divr} + \mathcal{L}_{kl} \tag{12}$$

$$\mathcal{L}_{dist} = -\frac{1}{M} \sum_{i=1}^{M} log(1 - (|\alpha - D(G(z_i))|)) \tag{13}$$

$$\mathcal{L}_{divr} = \frac{1}{\frac{1}{M} \sum_{i=1}^{M} (\|G(z_i) - \mu\|_2)}, \ \ \mu = \frac{1}{C} \sum_{i=1}^{C} G(z_i) \tag{14}$$

$$\mathcal{L}_{kl} = KL(G_{struct}(z_i) \| \frac{1}{C}) + KL(G_{origin}(z_i) \| \frac{1}{C}) \tag{15}$$

where $\alpha \in [0, 1]$ is a hyperparameter, β_1 and β_2 are weight hyperparameters, $D(\cdot)$ denotes the discriminator, M denotes the number of generated nodes, $G(z_i) = [G_{struct}(z_i) \| G_{origin}(z_i)]$ denotes the output of the generator, $C = k + d$ and $KL(P \| Q)$ denotes the KL divergence.

After obtaining the output from the generator, we employ the pre-trained TGAE to integrate the generated nodes into the original graph:

$$\tilde{\mathbf{X}}_{struct} = \begin{bmatrix} \mathbf{X}_{struct} \\ G_{struct}(z_i) \end{bmatrix}, \ \ \tilde{\mathbf{X}}_{origin} = \begin{bmatrix} \mathbf{X}_{origin} \\ G_{origin}(z_i) \end{bmatrix} \tag{16}$$

$$\tilde{\mathbf{A}} = TGAE(\tilde{\mathbf{X}}_{struct}, \tilde{\mathbf{X}}_{origin}) \tag{17}$$

Next, we select the edges in $\tilde{\mathbf{A}}$ that are related to the generated nodes and add them to the original graph, while preserving the inherent edges. The resulting graph serves as the input to the discriminator.

Notably, we denote the ratio of generated nodes to real nodes as $p_G = \frac{M}{N}$. As a hyperparameter, p_G will be explored further in our experiments.

Discriminator In our discriminator, we utilize a GNN architecture due to its superior capability in processing graph-structured data. However, traditional GNNs indiscriminately aggregate features from all neighboring nodes, which can lead to undesirable feature smoothing: the features of anomalous nodes may be overwhelmed by dominant normal nodes, and normal nodes may also incorporate noisy information from anomalous neighbors.

GraphCAD [4] suggests that normal and anomalous nodes can be effectively distinguished by leveraging the global context of the graph. Inspired by this insight, we introduce an edge homophily identification mechanism based on global context to address this issue.

First, we obtain the fusion representation $\tilde{\mathbf{H}}_{fusion}$:

$$\tilde{\mathbf{H}}_{fusion} = \mathbf{W}_r[\,\mathbf{W}_s\tilde{\mathbf{X}}_{struct} \,\|\, \mathbf{W}_o\tilde{\mathbf{X}}_{origin}\,] \tag{18}$$

where $||$ is the concatenation operation and $\mathbf{W}_r, \mathbf{W}_s, \mathbf{W}_o$ denotes three distinct learnable parameter matrices.

Next, we employ a multilayer GNN to process the graph structure. Specifically, at the l-th layer of GNN, the global context $\mathbf{h}_g^{(l)}$ is computed to aggregate information from all nodes within the graph. In this context, to identify the edge e_{ij}, we leverage nodes v_i, v_j and $\mathbf{h}_g^{(l)}$ to estimate the homophily probability $c_{ij}^{(l)}$:

$$\mathbf{h}_g^{(l)} = \sum_{i=1}^{\tilde{N}} \mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)} \tag{19}$$

$$d_{ij}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)} - \mathbf{W}^{(l)}\mathbf{h}_j^{(l-1)} \tag{20}$$

$$d_{ig}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)} - \mathbf{h}_g^{(l)} \tag{21}$$

$$d_{jg}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}_j^{(l-1)} - \mathbf{h}_g^{(l)} \tag{22}$$

$$c_{ij}^{(l)} = MLP([\,d_{ij}^{(l)} \,\|d_{ig}^{(l)} \,\|d_{jg}^{(l)}\,]) \tag{23}$$

where $\tilde{N} = N+M$, $\mathbf{W}^{(l)}$ is learnable parameter matrices of the l-th and MLP is a two-layer multilayer perceptron with a *sigmoid* activation function, ensuring that $c_{ij}^{(l)}$ is constrained within the range $(0, 1)$.

We incorporate $c_{ij}^{(l)}$ into the computation of attention coefficients, with the intention that $c_{ij}^{(l)}$ approaches 0 for heterogeneous edges. This enables a pruning-like effect during message passing, effectively preventing information exchange

between normal and anomalous nodes. The attention coefficient $a_{ij}^{(l)}$ and the message aggregation process of the GNN are designed as follows:

$$a_{ij}^{(l)} = \frac{c_{ij}^{(l)}}{\sum_{k=1}^{\tilde{N}} c_{ik}^{(l)}} \tag{24}$$

$$\mathbf{h}_i^{(l)} = \sigma(\sum_{j \in \mathcal{N}(i) \cup \{i\}} a_{ij}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l-1)}) \tag{25}$$

where $\mathcal{N}(v)$ denotes the neighbors of i and σ denotes an activation function. The input $\mathbf{H}^{(0)}$ is $\tilde{\mathbf{H}}_{fusion}$ and the output of the final attention layer $\mathbf{H}^{(L)}$ is denoted as the output node embeddings $\mathbf{Z}$.

Finally, we employ a simple detector to obtain the anomaly probability p_v of node v:

$$p_v = sigmoid(\mathbf{W}_p \mathbf{z}_v + \mathbf{b}) \tag{26}$$

where $\mathbf{W}_p$ and $\mathbf{b}$ denote the weight and bias parameters respectively.

The loss function $\mathcal{L}_{dis}$ is designed to comprise two components. $\mathcal{L}_{node}$ is node discrimination loss. $\mathcal{L}_{edge}$ is edge discrimination loss. Formally, the loss function is defined as:

$$\mathcal{L}_{dis} = \mathcal{L}_{node} + \sum_{l=1}^{L} \mathcal{L}_{edge}^{(l)} \tag{27}$$

$$\mathcal{L}_{node} = -\frac{1}{N_t} \sum_{i=1}^{N_t} (\gamma_1 log(1 - D(\mathbf{x}_i)) + \gamma_2 log(D(G(z_i)))) \tag{28}$$

$$\mathcal{L}_{edge}^{(l)} = -\frac{1}{|\mathcal{E}_t|} \sum_{e_{ij} \in \mathcal{E}_t} (y_{ij} log(c_{ij}^{(l)}) + (1 - y_{ij}) log(1 - c_{ij}^{(l)})) \tag{29}$$

where γ_1 and γ_2 are weight hyperparameters, N_t denotes the number of training nodes and $\mathcal{E}_t$ denotes the edges whose both endpoints are within the set of training nodes. For each $e_{ij} \in \mathcal{E}_t$, if node i and node j have the same label (e_{ij} is homophilic), $y_{ij} = 1$, otherwise (e_{ij} is heterophilic) $y_{ij} = 0$.

5 Experiments

5.1 Experiment Setup

Datasets We conduct comprehensive evaluations of HAGAN on four large-scale real-world benchmark datasets, including Amazon [9], Reddit [21], YelpChi [21] and TFinance [34], covering four domains: e-commerce, social media, business reviews, and finance. The key statistics are presented in the appendix.

Dataset	Amazon		Reddit		YelpChi		TFinance	
Metric	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Radar	0.5684	0.1701	0.5826	0.0858	0.5557	0.2906	0.0587	0.0458
ANOMALOUS	0.5205	0.1788	0.5900	0.0883	0.5824	0.3195	0.0579	0.0459
DOMINANT	0.2662	0.0839	0.5722	0.0743	0.4799	0.2549	0.8077	0.4715
DONE	0.7302	0.3653	0.6102	0.0859	0.5188	0.2737	0.8962	0.6910
AdONE	0.7746	0.5003	0.6178	0.0849	0.5199	0.2828	0.8960	0.6035
GAAN	0.6558	0.1672	0.5589	0.0822	0.5319	0.2711	0.6481	0.1579
AEGIS	0.8022	0.4998	0.5474	0.0950	0.4785	0.2557	0.8424	0.6036
CoLA	0.5803	0.1931	0.5488	0.0508	0.4619	0.1718	0.6148	0.2263
OCGNN	0.8602	0.7647	0.5247	0.0655	**0.5973**	0.3131	0.9049	0.7645
CONAD	0.2646	0.0838	0.5714	0.0744	0.4801	0.2545	0.8072	0.4555
TAM	0.8699	0.7454	0.5927	0.0871	0.5819	0.2916	**0.9314**	0.4612
CoCo	0.8620	0.8048	0.5724	0.0747	0.5836	0.3211	0.8790	0.5411
HAGAN(Ours)	**0.9038**	**0.8238**	**0.6223**	**0.0994**	0.5940	**0.3638**	0.9193	**0.7838**

Table 1: The AUC and AP results across four real-world GAD datasets are presented. The best performance in each row is boldfaced, with the second-best underlined.

Baselines HAGAN is compared with two shallow methods, Radar [23] and ANOMALOUS [30], as well as ten GNN-based deep methods, including DOMINANT [8], DONE [1], AdONE [1], GAAN [7], AEGIS [5], CoLA [26], OCGNN [37], TAM [31], and CoCo [36].

Notably, many methods are originally unsupervised. To ensure fairness, we modify these methods according to our semi-supervised setup. For supervised methods, which require anomalous labels, we exclude them from our baseline.

Metrics Following previous studies [31], we employ two widely recognized and complementary evaluation metrics for anomaly detection: Area Under the Receiver Operating Characteristic Curve (AUC) and Area Under the Precision-Recall Curve (AP). Higher AUC/AP indicates better performance.

Implementation Details Our semi-supervised setting use a portion of nodes labeled (50%) as normal for training, while the remaining nodes are used for validation and testing. HAGAN is implemented using pytorch 2.2.0+cu118 with Python 3.10, running on two NVIDIA GeForce RTX 3090 (24GB). All datasets are optimized using the Adam optimizer. The number of anchor nodes k is set to 64, the dimensions of each feature in the hidden layers are set to 64, and the number of layers of GNN is set to 2. Besides, the settings for other hyperparameters are presented in the appendix.

5.2 Performance Comparison

The results are presented in Table 1. HAGAN demonstrates consistently strong performance across all datasets, maintaining stable results, whereas many other

models exhibit significant fluctuations in certain datasets. For instance, OCGNN performs poorly on the Reddit dataset, and DONE struggles with the Amazon dataset. Notably, HAGAN excels in the AP metric, achieving the best performance across all datasets. In particular, on the YelpChi dataset, it outperforms the second-best model, CoCo, by a significant margin of 13.29%. While HAGAN does not achieve the highest AUC scores on the YelpChi and TFinance datasets, it still ranks second, further highlighting its robustness. These results suggest that HAGAN demonstrates strong generalizability and robustness in GAD tasks.

5.3 Transformer-based GAE Analysis

In this analysis, we focus on evaluating the performance of TGAE. Specifically, we implement a GCN-based GAE and a GCN-based VGAE, and modify the TGAE model to separately utilize only the original features (TGAE_{origin}) and only the structural features (TGAE_{struct}) for evaluation. The reconstruction loss $\mathcal{L}_{recon}$ is used as the evaluation metric.

As presented in Table 2, TGAE performs similarly to other common graph autoencoders on most datasets, achieving the best results on the YelpChi dataset. Moreover, TGAE_{struct} outperforms TGAE_{origin}. These experimental results demonstrate the effectiveness of both the anchor-based encoding and TGAE in graph reconstruction tasks.

Model	Amazon	Reddit	YelpChi	TFinance
GAE	**0.2134**	**0.2499**	0.2429	<u>0.2459</u>
VGAE	0.2512	0.2508	0.2515	0.2717
TGAE_{struct}	0.2382	0.2509	<u>0.2192</u>	**0.2257**
TGAE_{origin}	0.2540	0.2509	0.2613	0.2614
TGAE	<u>0.2312</u>	<u>0.2504</u>	**0.2147**	0.2466

Table 2: Reconstruction loss of GAEs.

5.4 Ablation Analysis

In this analysis, we focus on the necessity of three modules. Specifically, we modify HAGAN as follows: 1. **NoGEN** that replace the generator with a random feature generation approach. 2. **NoGAE** that randomly establish connections between generated nodes and real nodes. 3. **NoEHI** that remove the edge homophily identification from the discriminator and replace with GCN.

As presented in Table 3, NoGEN exhibits the poorest performance, which can be attributed to the inability of the random feature generation method to ensure the quality of the generated samples. This further underscores the necessity of our motivation to generate high-quality samples. NoGAE achieves relatively better results, suggesting that the removal of TGAE's influence has

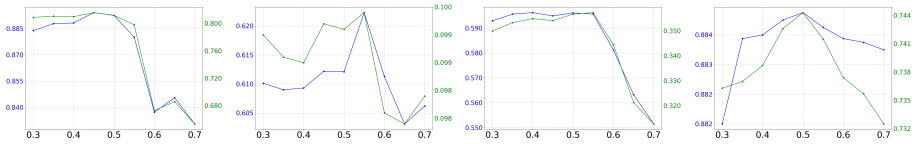
a limited effect. However, it still fails to match the full model, likely due to the disruption of the original graph structure caused by the random connection establishment method. NoEHI performs poorly, which aligns with the Challenge 3 discussed in Section 1, and further validates the rationale behind our proposed edge homogeneity identification mechanism.

Metric	Variants	Dataset			
		Amazon	Reddit	YelpChi	TFinance
AUC	NoGEN	0.6477	0.4275	0.5253	0.8726
	NoGAE	0.7859	0.5935	0.5548	0.9090
	NoEHI	0.7525	0.4345	0.5365	0.8893
	HAGAN	**0.9038**	**0.6223**	**0.5940**	**0.9193**
AP	NoGEN	0.3191	0.0518	0.2675	0.6441
	NoGAE	0.5230	0.0943	0.3096	0.7559
	NoEHI	0.3545	0.0547	0.2775	0.7077
	HAGAN	**0.8238**	**0.0994**	**0.3638**	**0.7838**

Table 3: The AUC and AP of variants.

5.5 Generated Sample Analysis

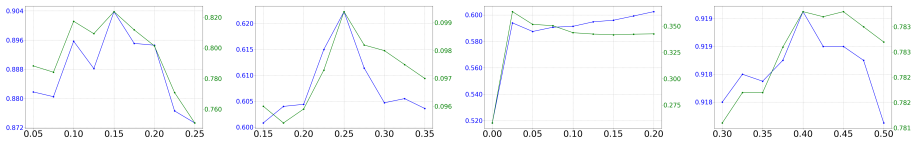
In this analysis, we focus on the performance with respect to α and p_G. We evaluated the performance of HAGAN across various values of α, specifically: 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, and 0.7. As presented in Fig. 2, both metrics exhibit an initial increase followed by a decrease, with the extrema occurring around $\alpha = 0.5$. This is because α represents the probability that a generated sample is classified as anomalous by the discriminator, and a value of 0.5 indicates a state where normal and anomalous instances are difficult to distinguish, aligning with our definition of high-quality anomalies. Furthermore, after reaching the peak, the performance significantly declines, as the generated anomalies become overly distinct, limiting the discriminator's learning efficiency.



(a) Amazon (b) Reddit (c) YelpChi (d) TFinance

Fig. 2: The AUC (blue) and AP (green) with respect to α.

We evaluated the performance of HAGAN across various values of p_G, testing values within the 0% to 50% range and visualizing the values near the extrema. As presented in Fig. 3, both metrics exhibit an initial increase followed by a decrease, and the optimal p_G values vary significantly across different datasets. We believe that an excessively high proportion of anomalous nodes may disrupt the underlying graph structure, deviating from real-world scenarios, while a proportion that is too low may lead the model to overly focus on learning from normal nodes.



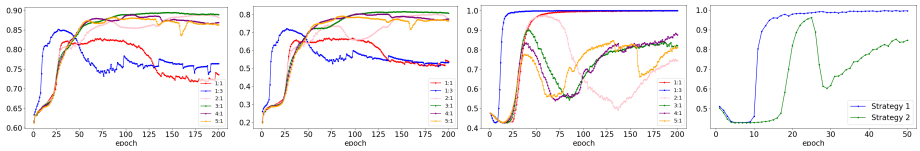
(a) Amazon (b) Reddit (c) YelpChi (d) TFinance

Fig. 3: The AUC (blue) and AP (green) with respect to p_G.

5.6 GAN Adversarial Analysis

In practice, we observe that as training progresses, the generator's performance lags behind the discriminator, leading to a decline in the quality of generated samples. This issue has been addressed in prior research [20, 29], where a common solution is to train the discriminator multiple times within a single epoch. However, while conventional GANs focus on optimizing the generator, our goal is to enhance the discriminator. To this end, we adopt a training strategy **(Strategy 1)** that involves training the generator multiple times within a single epoch. Fig. 4 presents the results of training the generator and discriminator at different ratios on the Amazon dataset, specifically 1:3, 1:1, 2:1, 3:1, 4:1, and 5:1. Fig. 4 (c) presents the variation in the anomaly probability of discriminator outputs for the generated samples at each epoch, under different training ratios. It is evident that when the training ratio is set to 1:3 or 1:1, the anomaly probability rapidly approaches 1.0, indicating that the generator is unable to keep pace with the discriminator, resulting in the generation of low-quality samples. This observation is also reflected in Fig. 4 (a) and (b), where the AUC and AP metrics quickly reach their peaks and then begin to decline. In contrast, when the generator undergoes multiple training iterations, overall performance improves, with the best results observed at a 3:1 training ratio. Furthermore, as illustrated in Fig. 4 (c), the average predicted probability of abnormality for the generated samples by the discriminator hovers around 0.7, with the minimum reaching 0.5. This observation aligns with the objective of the generator in HAGAN to produce samples that are difficult for the discriminator to distinguish from real

anomalies, thereby demonstrating the reliability of the samples generated by the generator.



(a) AUC (b) AP (c) p (d) Yelp

Fig. 4: The results of training the generator and discriminator at ratios 1:3 (blue), 1:1 (red), 2:1 (pink), 3:1 (green), 4:1(purple) and 5:1(orange).

Unfortunately, when we applied Strategy 1 to the YelpChi dataset, the results were suboptimal. We hypothesize that this result can be attributed to the discriminator initially making random predictions that prevented it from providing meaningful feedback to the generator. To address this issue, we propose an alternative training strategy (**Strategy 2**), in which the discriminator is first trained independently, without training the generator, and the generator is subsequently trained multiple times within each epoch. Fig. 4 (d) presents the anomaly probability of generated samples on the YelpChi dataset under different training strategies. As presented, Strategy 2 effectively mitigates the rapid degradation in the quality of generated samples. The performance of HAGAN across four datasets under different training strategies is presented in Table 4.

Metric	Strategy	Dataset			
		Amazon	Reddit	YelpChi	TFinance
AUC	1	**0.9038**	**0.6123**	0.5316	0.9174
	2	0.8533	0.5941	**0.5940**	**0.9188**
AP	1	**0.8238**	**0.0994**	0.2728	0.7784
	2	0.7168	0.0931	**0.3638**	**0.7839**

Table 4: The AUC and AP of Strategies.

5.7 Anchor Nodes Analysis

In this analysis, we focus on the performance with respect to k. Table 5 presents the reconstruction loss under different numbers of anchor points. It can be observed that the performance is relatively poor when the number of anchors k is small. This aligns with our hypothesis, as a limited number of anchors increases the likelihood that many nodes will have identical or similar distances to the

anchors, even if these nodes are structurally dissimilar in the graph. Such cases hinder the ability to effectively capture structural characteristics.

Intuitively, increasing the number of anchors should provide richer and more fine-grained structural representations. However, the experimental results show that the performance does not continuously improve with more anchors. Instead, it begins to decline after a certain point. We speculate that as k increases, newly added anchors may be located close to existing ones or concentrated in densely connected regions, leading to redundant structural information. This redundancy can introduce noise and interfere with the model's ability to make accurate structural distinctions.

k	Amazon	Reddit	YelpChi	TFinance
8	0.2358	0.2507	0.2221	0.2346
16	0.2344	0.2508	0.2201	0.2415
32	0.2322	0.2505	0.2156	**0.2252**
64	0.2312	**0.2504**	0.2147	0.2466
96	**0.2308**	0.2508	0.2134	0.2465
128	0.2329	0.2509	**0.2117**	0.2422

Table 5: Reconstruction loss of k.

5.8 Complexity Analysis

Anchor-based Encoding Complexity The dominant computational cost arises from performing BFS traversals to compute shortest path lengths. Specifically, for each of the k selected anchor nodes, a single-source shortest path is computed. The total complexity of the BFS step across all anchors is $\mathcal{O}(k(n+m))$, where n is the number of nodes and m is the number of edges.

GAE Complexity The computational complexity of the GAE model is mainly determined by the Transformer encoder and the inner product decoder. For a graph with n nodes and embedding dimension d, the encoder requires $\mathcal{O}(Ln^2d)$, and the decoder plus loss computation costs $\mathcal{O}(n^2d)$. Thus, the total complexity per subgraph is $\mathcal{O}(n^2d)$.

GAN Complexity Generator consists of multiple fully connected layers, where the computation primarily involves feature transformation with a time complexity of $\mathcal{O}(N)$, where N is the number of generated nodes. The discriminator employs a GNN, in which each layer models pairwise feature relationships and computes attention weights. The overall time complexity is approximately $\mathcal{O}(N^2d)$, where d is the feature dimension. As the graph size increases, the discriminator tends to become the computational bottleneck.

6 Conclusion

This paper presents a semi-supervised approach for graph anomaly detection and introduces HAGAN, a novel framework that generates high-quality anomaly nodes using a GAN. By incorporating a pre-trained Transformer-based GAE, HAGAN ensures seamless integration of the generated nodes while maintaining structural integrity. The framework also uses a GNN discriminator with edge homophily identification to reduce noise. Experimental results on benchmark datasets show that HAGAN achieves state-of-the-art performance. Future work will explore more effective anchor point selection strategies and investigate the application of generative approaches to dynamic or heterogeneous graphs.

References

1. Bandyopadhyay, S., N, L., Vivek, S.V., Murty, M.N.: Outlier resistant unsupervised deep architectures for attributed network embedding. In: Proceedings of the 13th international conference on web search and data mining. pp. 25–33 (2020)
2. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. pp. 93–104 (2000)
3. Caville, E., Lo, W.W., Layeghy, S., Portmann, M.: Anomal-e: A self-supervised network intrusion detection system based on graph neural networks. Knowledge-Based Systems **258**, 110030 (2022)
4. Chen, B., Zhang, J., Zhang, X., Dong, Y., Song, J., Zhang, P., Xu, K., Kharlamov, E., Tang, J.: Graph contrastive learning for anomaly detection (2021), https://api.semanticscholar.org/CorpusID:251799764
5. Chen, Z., Liu, B., Wang, M., Dai, P., Lv, J., Bo, L.: Generative adversarial attributed network anomaly detection. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 1989–1992 (2020)
6. Cheng, D., Ye, Y., Xiang, S., Ma, Z., Zhang, Y., Jiang, C.: Anti-money laundering by group-aware deep graph learning. IEEE Transactions on Knowledge and Data Engineering **35**(12), 12444–12457 (2023)
7. Ding, K., Li, J., Agarwal, N., Liu, H.: Inductive anomaly detection on attributed networks. In: Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence. pp. 1288–1294 (2021)
8. Ding, K., Li, J., Bhanushali, R., Liu, H.: Deep anomaly detection on attributed networks. In: Proceedings of the 2019 SIAM international conference on data mining. pp. 594–602. SIAM (2019)
9. Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., Yu, P.S.: Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In: Proceedings of the 29th ACM international conference on information & knowledge management. pp. 315–324 (2020)
10. Duan, J., Wang, S., Zhang, P., Zhu, E., Hu, J., Jin, H., Liu, Y., Dong, Z.: Graph anomaly detection via multi-scale contrastive learning networks with augmented view. In: Proceedings of the AAAI conference on artificial intelligence. vol. 37, pp. 7459–7467 (2023)
11. Gao, Y., Wang, X., He, X., Liu, Z., Feng, H., Zhang, Y.: Alleviating structural distribution shift in graph anomaly detection. In: Proceedings of the sixteenth ACM international conference on web search and data mining. pp. 357–365 (2023)

12. Gasteiger, J., Weißenberger, S., Günnemann, S.: Diffusion improves graph learning. Advances in neural information processing systems **32** (2019)
13. Gong, Z., Wang, G., Sun, Y., Liu, Q., Ning, Y., Xiong, H., Peng, J.: Beyond homophily: Robust graph anomaly detection via neural sparsification. In: IJCAI. pp. 2104–2113 (2023)
14. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial networks. Communications of the ACM **63**, 139 – 144 (2014), https://api.semanticscholar.org/CorpusID:1033682
15. Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., Bing, G.: Learning from class-imbalanced data: Review of methods and applications. Expert systems with applications **73**, 220–239 (2017)
16. Huang, M., Liu, Y., Ao, X., Li, K., Chi, J., Feng, J., Yang, H., He, Q.: Auc-oriented graph neural network for fraud detection. In: Proceedings of the ACM web conference 2022. pp. 1311–1321 (2022)
17. Jin, M., Liu, Y., Zheng, Y., Chi, L., Li, Y.F., Pan, S.: Anemone: Graph anomaly detection with multi-scale contrastive learning. In: Proceedings of the 30th ACM international conference on information & knowledge management. pp. 3122–3126 (2021)
18. Khan, W., Haroon, M.: An efficient framework for anomaly detection in attributed social networks. International Journal of Information Technology **14**(6), 3069–3076 (2022)
19. Kim, H., Lee, B.S., Shin, W.Y., Lim, S.: Graph anomaly detection with graph neural networks: Current status and challenges. IEEE Access **10**, 111820–111829 (2022)
20. Kotha, P., Janardhan Babu, V., Ankam, S.: Generative adversarial networks: A comprehensive review. In: International Conference on Computer & Communication Technologies. pp. 105–114. Springer (2023)
21. Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 1269–1278 (2019)
22. Li, D.: Anomaly detection with generative adversarial networks for multivariate time series. arXiv preprint arXiv:1809.04758 (2018)
23. Li, J., Dani, H., Hu, X., Liu, H.: Radar: Residual analysis for anomaly detection in attributed networks. In: IJCAI. vol. 17, pp. 2152–2158 (2017)
24. Lin, M., Wen, K., Zhu, X., Zhao, H., Sun, X.: Graph autoencoder with preserving node attribute similarity. Entropy **25** (2023), https://api.semanticscholar.org/CorpusID:257790352
25. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data (TKDD) **6**(1), 1–39 (2012)
26. Liu, Y., Pan, S., Wang, Y.G., Xiong, F., Wang, L., Chen, Q., Lee, V.C.: Anomaly detection in dynamic graphs via transformer. IEEE Transactions on Knowledge and Data Engineering **35**(12), 12081–12094 (2021)
27. Ngo, P.C., Winarto, A.A., Kou, C.K.L., Park, S., Akram, F., Lee, H.K.: Fence gan: Towards better anomaly detection. In: 2019 IEEE 31St International Conference on tools with artificial intelligence (ICTAI). pp. 141–148. IEEE (2019)
28. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford infolab (1999)
29. Pawar, D.: Advancements and applications of generative adversarial networks: A comprehensive review. International Journal for Research in Applied Science and Engineering Technology (2024), https://api.semanticscholar.org/CorpusID:269903498

30. Peng, Z., Luo, M., Li, J., Liu, H., Zheng, Q., et al.: Anomalous: A joint modeling approach for anomaly detection on attributed networks. In: IJCAI. vol. 18, pp. 3513–3519 (2018)
31. Qiao, H., Pang, G.: Truncated affinity maximization: One-class homophily modeling for graph anomaly detection. Advances in Neural Information Processing Systems **36** (2024)
32. Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U., Langs, G.: Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: International conference on information processing in medical imaging. pp. 146–157. Springer (2017)
33. Shi, F., Cao, Y., Shang, Y., Zhou, Y., Zhou, C., Wu, J.: H2-fdetector: A gnn-based fraud detector with homophilic and heterophilic connections. In: Proceedings of the ACM web conference 2022. pp. 1486–1494 (2022)
34. Tang, J., Li, J., Gao, Z., Li, J.: Rethinking graph neural networks for anomaly detection. In: International Conference on Machine Learning. pp. 21076–21089. PMLR (2022)
35. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
36. Wang, R., Xi, L., Zhang, F., Fan, H., Yu, X., Liu, L., Yu, S., Leung, V.C.M.: Context correlation discrepancy analysis for graph anomaly detection. IEEE Transactions on Knowledge and Data Engineering **37**(1), 174–187 (2025). https://doi.org/10.1109/TKDE.2024.3488375
37. Wang, X., Jin, B., Du, Y., Cui, P., Tan, Y., Yang, Y.: One-class graph neural networks for anomaly detection in attributed networks. Neural computing and applications **33**, 12073–12085 (2021)
38. Xu, Z., Huang, X., Zhao, Y., Dong, Y., Li, J.: Contrastive attributed network anomaly detection with data augmentation. In: Pacific-Asia conference on knowledge discovery and data mining. pp. 444–457. Springer (2022)
39. Zhang, J., Wang, S., Chen, S.: Reconstruction enhanced multi-view contrastive learning for anomaly detection on attributed networks. arXiv preprint arXiv:2205.04816 (2022)
40. Zhang, R., Cheng, D., Liu, X., Yang, J., Ouyang, Y., Wu, X., Zheng, Y.: Generation is better than modification: Combating high class homophily variance in graph anomaly detection. arXiv preprint arXiv:2403.10339 (2024)
41. Zheng, Y., Jin, M., Liu, Y., Chi, L., Phan, K.T., Chen, Y.P.P.: Generative and contrastive self-supervised learning for graph anomaly detection. IEEE Transactions on Knowledge and Data Engineering **35**(12), 12220–12233 (2021)